

MS SQL Server

Richard Waymire
Rick Sawtell

Übersetzung:
Frank Langenau



MS SQL Server

Bitte beachten Sie: Der originalen Printversion liegt eine CD-ROM bei.
In der vorliegenden elektronischen Version ist die Lieferung einer CD-ROM nicht enthalten.
Alle Hinweise und alle Verweise auf die CD-ROM sind ungültig.

Markt + Technik Verlag

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Ein Titeldatensatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich.

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen
eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter
Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben
und deren Folgen weder eine juristische Verantwortung noch
irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und
Herausgeber dankbar.

Autorisierte Übersetzung der amerikanischen Originalausgabe:
Teach Yourself Microsoft SQL Server in 21 Days © 2000 by SAMS Publishing

Alle Rechte vorbehalten, auch die der fotomechanischen
Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten
Modelle und Arbeiten ist nicht zulässig.

Fast alle Hardware- und Software-Bezeichnungen, die in diesem Buch
erwähnt werden, sind gleichzeitig auch eingetragene Warenzeichen
oder sollten als solche betrachtet werden.

Umwelthinweis:
Dieses Buch wurde auf chlorfrei gebleichtem Papier gedruckt.
Die Einschumpffolie – zum Schutz vor Verschmutzung – ist aus
umweltverträglichem und recyclingfähigem PE-Material.

10 9 8 7 6 5 4 3 2 1

05 04 03 02 01

ISBN 3-8272-6021-3

© 2001 by Markt+Technik Verlag,
ein Imprint der Pearson Education Deutschland GmbH.
Martin-Kollar-Straße 10–12, D–81829 München/Germany
Alle Rechte vorbehalten
Übersetzung: Frank Langenau
Lektorat: Rainer Fuchs, rfuchs@pearson.de
Herstellung: Claudia Bäurle, cbaeurle@pearson.de
Einbandgestaltung: Grafikdesign Heinz H. Rauner, Gmünd
Satz: reemers publishing services gmbh, Krefeld, (www.reemers.de)
Druck und Verarbeitung: Kösel, Kempten (www.koeselbuch.de)

Printed in Germany

Inhaltsverzeichnis

	Einführung	23
	Über dieses Buch.	24
	Wer sollte dieses Buch lesen?	25
	In diesem Buch verwendete Konventionen	25
	Woche 1 – Vorschau	27
Tag 1	Relationale Datenbanken und Microsoft SQL Server 2000.	29
	1.1 SQL Server auf Erfolgskurs.	30
	1.2 Die Geschichte von SQL Server	31
	1.3 Was ist eine Datenbank?	32
	Objekte relationaler Datenbanken	33
	1.4 Relationale Datenbanken entwerfen	34
	Interview mit dem Kunden.	37
	Objekte organisieren.	38
	Normalisieren der Daten	38
	Datenbank und Tabellen erstellen.	40
	1.5 SQL Server und das Client/Server-Modell	40
	1.6 Zusammenfassung.	41
	1.7 Workshop.	41
	Fragen & Antworten	41
	Kontrollfragen	42
	Übungen	42
Tag 2	SQL Server 2000 installieren	45
	2.1 Die SQL Server-Produktfamilie	46
	Standard Edition	46
	Enterprise Edition.	47
	Personal Edition	47
	Developer Edition.	47
	Enterprise Evaluation Edition	48
	Microsoft SQL Server 2000 Desktop Engine (MSDE)	48

2.2	Voraussetzungen für die Installation von SQL Server	48
	Hardware- und Software-Voraussetzungen.	48
	Windows NT/Windows 2000-Optionen.	50
	Lizenzierung	51
2.3	SQL Server 2000 installieren	52
	Start des Installationsvorganges.	52
	Betriebssystemabhängige Voraussetzungen	53
	SQL Server 2000 (vollständig) installieren.	54
2.4	Weitere optionale SQL Server-Komponenten	76
	Microsoft SQL Server Analysis Services	76
	Microsoft English Query.	76
2.5	Nach der Installation: Bestandsaufnahme	76
	Die SQL Server-Dienste	77
	Installationsordner.	78
	Einträge in der Windows-Registrierung	80
	Die Standarddatenbanken	81
	Vorgegebene Benutzernamen in SQL Server	83
2.6	Benannte und Standardinstanzen	84
	Standardinstanzen.	84
	Benannte Instanzen	85
	Gemeinsame Komponenten.	85
	Eine benannte Instanz installieren.	86
2.7	Aktualisierung von einer vorherigen SQL Server-Version	86
	Aktualisierung von SQL Server 6.5	86
2.8	Problembehebung bei der Installation	87
	sqlstp.log	88
	*.OUT-Dateien	88
	SQL Server-Fehlerprotokoll	88
	Windows NT/2000-Ereignisprotokoll.	88
2.9	SQL Server entfernen	89
2.10	Zusammenfassung.	89
2.11	Workshop.	90
	Fragen & Antworten	90
	Kontrollfragen	91
	Übungen	91
Tag 3	SQL Server 2000 Tools und Utilities	93
3.1	Die Programmgruppe Microsoft SQL Server 2000	94
	Die Datei readme.txt.	95
	Onlinedokumentation.	95

	Dienst-Manager.	101
	SQL Server-Clientkonfiguration.	105
	SQL Server-Netzwerkkonfiguration	109
	Query Analyzer	112
	Enterprise Manager.	124
	Daten importieren und exportieren	131
	XML-Unterstützung von SQL in IIS konfigurieren.	131
	Profiler.	131
3.2	Die Programmgruppe Microsoft SQL Server – Versionsumstellung	131
	SQL Server-Aktualisierungs-Assistent.	131
	Microsoft SQL Server 6.5	132
	SQL Server 6.x deinstallieren	132
3.3	Weitere SQL Server Tools und Utilities	133
	Verbindungswerkzeuge.	133
	Server-Diagnose- und -Wartungswerkzeuge.	139
	BCP – ein eigener Komplex	146
3.4	Zusammenfassung.	146
3.5	Fragen & Antworten	146
3.6	Workshop.	147
	Kontrollfragen	147
	Übungen	148
Tag 4	Datenbanken und Datendateien erstellen und verwalten	149
4.1	Eine Datenbank erstellen	150
4.2	Informationen über Datenbanken	158
4.3	Datenbankoptionen festlegen	161
4.4	Die Größe von Datenbanken ändern.	165
	Eine Datenbank erweitern	167
	Datenbanken verkleinern	171
4.5	Datenbanken umbenennen	174
4.6	Arbeit mit Dateigruppen	175
4.7	Eine Datenbank löschen.	180
4.8	Zusammenfassung.	182
4.9	Workshop.	182
	Fragen & Antworten	182
	Kontrollfragen	184
	Übungen	184

Tag 5	Sicherheit	187
5.1	Das SQL Server-Sicherheitsmodell	188
	Windows-Authentifizierung	189
	SQL Server-Authentifizierung	191
	SQL Server-Datenbankbenutzername	191
	Berechtigungen	191
5.2	Die Sicherheitsmodi von SQL Server	191
	Sicherheit im gemischten Modus	192
	Windows-Authentifizierungsmodus	197
	Sicherheitsmodus einstellen	205
	Benutzernamen grafisch verwalten	207
5.3	Datenbankbenutzer	210
	Einer Datenbank einen Benutzer hinzufügen	210
	Der Benutzername guest	214
	Einen Alias hinzufügen	215
	Den Datenbankbesitzer ändern	216
5.4	Rollen	217
	Die öffentliche Rolle	218
	Serverweite Rollen	218
	Datenbankrollen	221
5.5	Anwendungsrollen	227
5.6	Zusammenfassung	230
5.7	Workshop	230
	Fragen & Antworten	230
	Kontrollfragen	231
	Übungen	231
Tag 6	Berechtigungen	233
6.1	Warum Berechtigungen verwenden?	234
6.2	Implementieren von Datenbankberechtigungen	235
	Berechtigungstypen	235
	Vorrang von Berechtigungen	236
6.3	Spezielle Berechtigungen von SQL Server	236
	Berechtigungen fester Serverrollen	237
	sa	240
	Feste Datenbankrollen	240
	Der Datenbankbesitzer (dbo)	244
	Berechtigungen des Datenbankobjektbesitzers	246
	Benutzerberechtigungen	248
	Die Rolle public	248

6.4	Anweisungsberechtigungen	249
	Die Berechtigung CREATE DATABASE	249
	Die Berechtigungen CREATE TABLE, VIEW, FUNCTION, PROCEDURE, DEFAULT und RULE	250
	Die Anweisungsberechtigungen BACKUP DATABASE und BACKUP LOG	250
	Anweisungsberechtigungen zuweisen	251
6.5	Objektberechtigungen	255
	Objektberechtigungen zuweisen	256
6.6	Berechtigungen für Sichten, gespeicherte Prozeduren und Funktionen	263
	Berechtigungen für Sichten	264
	Berechtigungen für gespeicherte Prozeduren	264
	Berechtigungen für benutzerdefinierte Funktionen	265
6.7	Besitzketten	265
	Einzelbesitzerkette	266
	Unterbrochene Besitzketten	266
6.8	Berechtigungsstrategien entwerfen	268
	Empfehlungen	268
	Was man tun sollte und was nicht	268
	Sicherheitsskripts erstellen	270
6.9	Zusammenfassung	273
6.10	Workshop	273
	Fragen & Antworten	273
	Kontrollfragen	274
Tag 7	Sicherungen von Datenbanken	275
7.1	Daten schützen durch Spiegelung, Duplexing und Striping von Datenträgern	276
	RAID	278
7.2	Wie die Datensicherung funktioniert	281
	Arten der Sicherung	282
	Transaktionen und Prüfpunkte	283
	Alternative Sicherungsstrategien	284
7.3	Terminologie	284
7.4	Überlegungen zur Datensicherung	285
	Wer ist für die Sicherungen verantwortlich?	285
	Wie stellen Sie fest, ob Ihre Datenbanksicherung brauchbar ist?	286
	Welche Datenbanken sollten gesichert werden?	286

	Welche Medien sollten Sie für Ihre Datenbanksicherung verwenden?	286
	Wie oft sollten Sie Ihre Datenbank sichern?	287
	Wie lange sollten Sie die Sicherungskopien aufbewahren?	287
	Wie lange dauert die Wiederherstellung einer Sicherungskopie?	288
	Haben Sie einen Katastrophenplan?	288
	Einige Überlegungen zu Bändern	289
7.5	Benutzerdatenbanken sichern	290
	Datenbank-Wiederherstellungsmodelle	290
7.6	Systemdatenbanken sichern	293
	Die master-Datenbank	294
	Die msdb-Datenbank	294
	Die model-Datenbank	295
	Die tempdb-Datenbank	295
	Die distribution-Datenbank	295
7.7	Vorbereitung der Datenbanksicherung	296
	Erstellung eines Sicherungsmediums	296
	Überprüfung der Datenbankkonsistenz	301
7.8	Sicherung einer Datenbank	306
	Verwendung der Sicherungsmedien	306
	Die Anweisung BACKUP DATABASE (für komplette Datenbanken)	308
	Die Anweisung BACKUP DATABASE (für Dateien und Dateigruppen)	313
	Die Anweisung BACKUP LOG	314
	Sicherungen mit Enterprise Manager	317
7.9	Zusammenfassung	321
7.10	Workshop	322
	Fragen & Antworten	322
	Kontrollfragen	323
	Übungen	323
Woche 2 – Vorschau		325
Tag 8	Wiederherstellen von Datenbanken	329
8.1	Datenbanken wiederherstellen	330
	Automatische Wiederherstellung	331
	Manuelle Wiederherstellung	334
8.2	Wiederherstellungsszenarien	356
	Wiederherstellung nach dem Ausfall einer Festplatte	357

	Wiederherstellung nach dem Verlust der master-Datenbank	359
8.3	Zusammenfassung	363
8.4	Workshop	363
	Fragen & Antworten	364
	Kontrollfragen	364
	Übungen	364
Tag 9	Datentypen und Tabellen	367
9.1	Tabellen	368
9.2	Spalten	368
9.3	SQL Server-Bezeichner	369
	Datentypen	370
	NULL-Zulässigkeit	372
	Sortierung	372
9.4	Zeichenfolgen	372
9.5	Binäre Daten	373
9.6	Ganze Zahlen	374
9.7	Numerische Datentypen	375
	Ungefähre numerische Datentypen	375
	Genaue numerische Datentypen	376
9.8	Besondere Datentypen	377
	bit	377
	Text und Image	378
	RowGUID	378
	sql_variant	379
	table	380
9.9	Die Datentypen Datum und Zeit	380
9.10	Währungsdaten	382
9.11	Der Datentyp rowversion (timestamp)	382
9.12	ANSI und Unicode	383
9.13	Benutzerdefinierte Datentypen	384
9.14	Berechnete Spalten	386
9.15	Tabellen erstellen	387
	Die Anweisung CREATE TABLE	388
	Erstellen von Tabellen über den SQL Server Enterprise Manager	389
	Tabellen löschen	390
9.16	Zusammenfassung	391

9.17	Workshop	392
	Fragen & Antworten	392
	Kontrollfragen	394
	Übungen	395
Tag 10	Daten mit SELECT abrufen	397
10.1	Einfache SELECT-Anweisungen	398
	Spaltenüberschriften ändern	401
	Verwenden von Literalen	402
10.2	Daten manipulieren	403
	Arithmetische Operatoren	404
	Mathematische Funktionen	405
	Zeichenfolgenfunktionen	407
	Datumsfunktionen	410
10.3	Systemfunktionen	412
10.4	Datenumwandlungen	414
10.5	Zeilen auswählen	416
	Vergleichsoperatoren	416
	Bereiche	417
	Listen	418
	Zeichenfolgen	420
	Unbekannte Werte	421
	Abrufen von Zeilen über mehrere Suchkriterien	423
10.6	Duplikate entfernen	425
10.7	Sortieren von Daten mit der Klausel ORDER BY	427
10.8	Zwischenbilanz	428
10.9	Aggregatfunktionen	429
	GROUP BY und HAVING	431
	COMPUTE und COMPUTE BY	433
10.10	Superaggregate (ROLLUP und CUBE)	436
10.11	Datenkorrelation	439
	Implementieren von Verknüpfungen	439
	Innere Verknüpfungen	440
	Natürliche Verknüpfungen	441
	Kreuz- bzw. unbeschränkte Verknüpfungen	442
	Äußere Verknüpfungen	444
	Selbstverknüpfungen	446
10.12	Unterabfragen	447
	Wie die Unterabfrage verwendet wird	447
	Typen von Unterabfragen	448
	Korrelierte Unterabfragen	449

	10.13	SELECT INTO	450
	10.14	Der UNION-Operator	452
	10.15	Zusammenfassung.	454
	10.16	Workshop.	455
		Fragen & Antworten	455
		Kontrollfragen	456
		Übungen	456
Tag 11		Daten bearbeiten	459
	11.1	Transaktionsprotokolle	460
	11.2	Daten einfügen	461
		Die Anweisung INSERT VALUES	463
		Standardwerte	466
		Einfügen von Daten mit SELECT.	466
		Einfügen von Daten mit gespeicherten Prozeduren	468
	11.3	Daten löschen	470
		DELETE über eine Suchtabelle	471
		TRUNCATE TABLE	473
	11.4	Aktualisieren mit UPDATE	474
		UPDATE.	474
		UPDATE über eine Suchtabelle	476
	11.5	Zusammenfassung.	478
	11.6	Workshop.	478
		Fragen & Antworten	478
		Kontrollfragen	479
		Übungen	479
Tag 12		Transact-SQL-Erweiterungen	481
	12.1	Stapel.	482
	12.2	Skripts	483
	12.3	Transaktionen	484
		Transaktionstypen	485
		Verteilte Transaktionen.	495
	12.4	Transact-SQL-Steuerungsstrukturen	498
		Blöcke (BEGIN...END)	498
		Benutzerdefinierte Meldungen (PRINT)	498
		Bedingte Ausführung (IF...ELSE)	499
		Bedingungsliste (CASE)	501
		Die Funktion COALESCE	504
		Die Funktion NULLIF	505
		Schleifen (WHILE)	506

12.5	Sperren	507
	Sperrentypen	508
	Steuern des Sperrverhaltens	509
	Sperren überwachen	511
12.6	Zusammenfassung.	518
12.7	Workshop	519
	Fragen & Antworten	519
	Kontrollfragen	520
	Übung	521
Tag 13	Indizes	523
13.1	Warum eigentlich Indizes?	524
	Indexstrukturen	525
	Indexoptionen	527
13.2	Indizes in SQL Server	531
13.3	Die Indexnutzung	538
	Wann wird mein Index benutzt?	538
	Wie kann ich die Indexnutzung überprüfen?	539
	Den Abfrageoptimierer umgehen.	545
13.4	Indexwartung.	548
	Statistiken	548
	Der Befehl DBCC UPDATEUSAGE	551
	Der Befehl DBCC SHOWCONTIG	552
	Der Befehl DBCC DBREINDEX	554
	Der Befehl DBCC INDEXDEFRAG	556
	Indizierte Sichten	556
13.5	Indizes mit dem Enterprise Manager verwalten.	559
13.6	Indexauswahl.	563
	Gruppierte Indizes	564
	Nicht gruppierte Indizes	565
	Der Indexoptimierungs-Assistent	565
13.7	Volltextindizes	565
13.8	Zusammenfassung.	569
13.9	Workshop	570
	Fragen & Antworten	570
	Kontrollfragen	571
	Übungen	571

Tag 14	Datenintegrität.	573
14.1	Wie die Datenintegrität gewährleistet wird.	574
14.2	Typen der Integrität.	575
	Domänenintegrität	575
	Referenzielle Integrität	575
	Entitätsintegrität	576
14.3	Traditionelle Methoden der Integrität	576
	Benutzerdefinierte Datentypen	576
	Standards	580
	Regeln	586
14.4	ANSI-Einschränkungen und Mechanismen der deklarativen Integrität.	590
	Die Eigenschaft IDENTITY.	590
	ANSI-Einschränkungen	596
	Einschränkungen löschen.	607
14.5	Datenintegrität und Enterprise Manager	608
	Einschränkungen zurückweisen.	614
	Einschränkungen abschalten	615
14.6	Zusammenfassung.	616
14.7	Workshop	617
	Fragen & Antworten	617
	Kontrollfragen	617
	Übungen	618
Woche 3 – Vorschau		619
Tag 15	Sichten, Trigger, gespeicherte Prozeduren und Funktionen	621
15.1	Erstellen und Bearbeiten von Sichten	622
	Sichten erstellen	624
	Informationen über Sichten	627
	Sichttypen	628
	Datenänderungen mit Sichten	631
	Sichten ändern	631
	Entfernen von Sichten	632
15.2	Gespeicherte Prozeduren	632
	Gespeicherte Prozeduren erstellen.	634
	Informationen über gespeicherte Prozeduren	636
	Erstellen einer Prozedurengruppe	637
	Parameter in gespeicherten Prozeduren.	638
	Verwenden der Option WITH RECOMPILE	641
	Remote gespeicherte Prozeduren	643

15.3	Trigger	644
	Die Tabellen inserted und deleted	645
	Erstellen von Triggern mit der Anweisung CREATE TRIGGER	645
	Erzwingen der Datenintegrität mit Triggern	648
	INSTEAD OF-Trigger	653
	Informationen über Trigger	654
15.4	Benutzerdefinierte Funktionen	654
	Skalarfunktionen	655
	Inlinefunktionen mit Tabellenrückgabe	658
	Funktionen mit mehreren Anweisungen und Tabellenrückgabe	658
15.5	Zusammenfassung	660
15.6	Workshop	660
	Fragen & Antworten	661
	Kontrollfragen	662
	Übungen	663
Tag 16	Replikation – Entwurf	665
16.1	Was ist Replikation?	666
16.2	Faktoren bei verteilten Daten	667
16.3	Verteilungsmethoden	668
	Mergereplikation	668
	Snapshotreplikation	670
	Transaktionsreplikation	671
	Aktualisierbare Abonnements	672
	Snapshotreplikation mit aktualisierenden Abonnenten	674
	Transaktionsreplikation mit aktualisierenden Abonnenten	675
	Verteilte Transaktionen	676
16.4	Das Verleger/Abonnent-Modell	677
	Artikel	677
	Publikationen	680
16.5	Wie arbeitet Replikation?	681
	Wie arbeitet Mergereplikation?	682
	Interna der Snapshotreplikation	685
	Interna der Transaktionsreplikation	686
	Funktionsweise beim Aktualisieren über eine Warteschlange	688
16.6	Replikationsszenarios	689
	Zentrale Verleger	689
	Zentraler Verleger mit Remoteverteiler	689

	Neuverleger (Publizierender Abonnent)	690
	Zentraler Abonnent	691
	Mehrere Verleger und Abonnenten	693
	Publizierte Daten transformieren	694
16.7	SQL Server 2000-Datenbanken im Internet publizieren.	694
	Publizieren über ein VPN	694
	Publizieren über Microsoft Proxy Server	695
	Publizieren mithilfe von FTP	695
16.8	Empfehlungen	695
16.9	Replikation in einer heterogenen Umgebung	696
	Publizieren von Daten an Nicht-SQL Server 2000-Datenbanken.	697
	Nicht-SQL Server 2000-Datenbanken an SQL Server 2000 publizieren	697
16.10	Zusammenfassung.	698
16.11	Workshop	698
	Fragen & Antworten	698
	Kontrollfragen	700
	Übung	700
Tag 17	Replikation – Implementierung	701
17.1	Sicherheitsbetrachtungen	702
17.2	Server einrichten	703
17.3	Replikation installieren	704
	Installation des Verteilungsservers	704
	Erstellen einer Publikation	712
	Erstellen eines Abonnements	719
17.4	Überlegungen zur Replikation	728
	Probleme beim Publizieren	728
	Probleme beim Abonnieren	729
	Leistungsprobleme	730
17.5	Überwachen der Replikation	730
	Der Replikationsmonitor	731
	Verteilungsserver verwalten	735
	Replikationsskripts.	739
	Systemmonitor und Replikation	741
	Gespeicherte Prozeduren einsetzen	742
	Überlegungen zur Verwaltung	744
	Fehlerbehebung	745
	Probleme bei der Wiederherstellung der Replikation	746

17.6	Zusammenfassung	748
17.7	Workshop	748
	Fragen & Antworten	748
	Kontrollfragen	749
	Übung	749
Tag 18	Aufträge mit dem SQL Server-Agent planen	751
18.1	Warum SQL Server automatisieren?	752
18.2	Welches Planungstool ist zu empfehlen?	753
	Der Windows 2000-Zeitplandienst	753
	Der SQL Server-Agent-Dienst	755
	SQL Mail-Integration	761
	Konfiguration eines Microsoft Outlook-Nachrichtenprofils	762
	SQL Mail für SQL Server konfigurieren	768
	SQL Mail für den SQL Server-Agent konfigurieren	769
18.3	Das Senden von E-Mail vom SQL Server	769
18.4	Aufträge	771
	Einen neuen Auftrag erstellen	772
	Auftragschritte	772
	Zeitplanung von Aufträgen	775
	Auftragsbenachrichtigungen	778
	Auftragssystem-Optionen	778
	Multiserveraufträge	780
18.5	Warnungen und Operatoren	780
	Konfigurieren von Operatoren	780
	Warnungen erstellen	782
	Warnungssystem-Optionen	786
18.6	Integration des Systemmonitors	787
18.7	Der Datenbank-Wartungsplanungs-Assistent	788
18.8	Zusammenfassung	794
18.9	Workshop	795
	Fragen & Antworten	795
	Kontrollfragen	795
	Übungen	796
Tag 19	Daten zwischen Servern migrieren	797
19.1	Daten verschieben mit Sichern und Wiederherstellen	798
19.2	Daten mit BCP laden	800
	BCP-Berechtigungen	801
	BCP einsetzen	801

	Systemeigener Modus und Zeichenmodus	805
	Formatdateien	806
	Stapelgrößen	808
	Schnelles BCP	809
19.3	Der Befehl BULK INSERT	811
19.4	Verwenden der Data Transformation Services	813
	DTS-Pakete	814
	Paketsicherheit	816
	Paketkomponenten	816
	Daten mithilfe des DTS-Import/Export-Assistenten exportieren.	819
19.5	Zusammenfassung.	826
19.6	Workshop.	827
	Fragen & Antworten	827
	Kontrollfragen	828
	Übungen	828
Tag 20	Leistungsüberwachung	829
20.1	Hardware auswählen und optimieren.	830
	Hauptspeicher	830
	Prozessor	831
	Netzwerk	831
	Festplatten	832
20.2	Die Konfigurationsparameter von Windows 2000.	834
20.3	Leistungsindikatoren des Windows 2000-Systemmonitors.	835
	Speicher.	836
	Prozessor	836
	Netzwerk	836
	Festplatte	836
20.4	SQL Server konfigurieren.	837
	Prozessoroptionen	839
	Speicherkonfigurationsoptionen.	841
	Ein-/Ausgabeoptionen.	843
	Abfrage/Index-Optionen	844
20.5	Überwachen der Leistung von SQL Server	845
	SQL Server-Leistungsindikatoren im Systemmonitor	845
	Gespeicherte Systemprozeduren für die Überwachung der Leistung.	846
20.6	Das Fenster Aktuelle Aktivität im SQL Server Enterprise Manager.	848
	SQL Server Profiler.	849

20.7	Der Indexoptimierungs-Assistent	858
20.8	Zusammenfassung.....	863
20.9	Workshop.....	863
	Fragen & Antworten	863
	Kontrollfragen	864
	Übungen	864
Tag 21	XML und SQL Server 2000	865
21.1	Was ist XML?.....	866
21.2	Auf SQL Server via IIS zugreifen	868
	Virtuelle Verzeichnisse und SQL Server	868
	IIS und das virtuelle Verzeichnis einrichten	869
	SQL Server über URLs ansprechen	875
	Gespeicherte Prozeduren ausführen	879
	Vorlagendateien	882
21.3	XML in SQL Server lesen und schreiben	885
	Gültige SELECT-Anweisungen.....	885
	Die FOR XML-Klausel.....	886
21.4	Zusammenfassung.....	891
21.5	Workshop.....	891
	Fragen und Antworten	891
	Kontrollfragen	892
	Übungen	892
	A. Antworten	893
A.1	Antworten für Tag 1.....	894
	Antworten zu den Kontrollfragen.....	894
	Lösung zu den Übungen	894
A.2	Antworten für Tag 2.....	900
	Antworten zu den Kontrollfragen.....	900
A.3	Antworten für Tag 3.....	900
	Antworten zu den Kontrollfragen.....	900
A.4	Antworten für Tag 4.....	901
	Antworten zu den Kontrollfragen.....	901
	Lösungen zu den Übungen	901
A.5	Antworten für Tag 5.....	903
	Antworten zu den Kontrollfragen.....	903
	Lösungen zu den Übungen	904
A.6	Antworten für Tag 6.....	904
	Antworten zu den Kontrollfragen.....	904

A.7	Antworten für Tag 7.	905
	Antworten zu den Kontrollfragen	905
	Lösungen zu den Übungen	905
A.8	Antworten für Tag 8.	906
	Antworten zu den Kontrollfragen	906
A.9	Antworten für Tag 9.	907
	Antworten zu den Kontrollfragen	907
	Lösungen zu den Übungen	908
A.10	Antworten für Tag 10.	909
	Antworten zu den Kontrollfragen	909
	Lösungen zu den Übungen	909
A.11	Antworten für Tag 11.	911
	Antworten zu den Kontrollfragen	911
	Lösungen zu den Übungen	911
A.12	Antworten für Tag 12.	912
	Antworten zu den Kontrollfragen	912
	Lösung zur Übung	913
A.13	Antworten für Tag 13.	914
	Antworten zu den Kontrollfragen	914
	Lösungen zu den Übungen	914
A.14	Antworten für Tag 14.	915
	Antworten zu den Kontrollfragen	915
A.15	Antworten für Tag 15.	915
	Antworten zu den Kontrollfragen	915
	Lösungen zu den Übungen	916
A.16	Antworten für Tag 16.	917
	Antworten zu den Kontrollfragen	917
	Lösung zur Übung	917
A.17	Antworten für Tag 17.	918
A.18	Antworten für Tag 18.	918
	Antworten zu den Kontrollfragen	918
A.19	Antworten für Tag 19.	918
	Antworten zu den Kontrollfragen	918
A.20	Antworten für Tag 20.	919
	Antworten zu den Kontrollfragen	919
A.21	Antworten für Tag 21.	919
	Antworten zu den Kontrollfragen	919

B. Die CD zum Buch	921
Stichwortverzeichnis	923

Einführung

Willkommen zu *Microsoft SQL Server 2000 in 21 Tagen* von Markt&Technik. Wir hoffen, dass Ihnen dieses Buch nicht nur bei Ihrem Einstieg in SQL Server hilfreich ist, sondern sich auch noch später bei Ihrer täglichen Arbeit mit diesem Produkt als wertvoll erweist.

Um SQL Server erfolgreich installieren, verwalten und konfigurieren zu können sowie für die Fehlerbehebung gewappnet zu sein, brauchen Sie fundierte Kenntnisse. Diese lassen sich zwei allgemeinen Kategorien bzw. Aufgabenbereichen zuordnen:

- SQL Server-Programmierer
- SQL Server-Administrator



In einer kleineren Firma sind Entwickler, Programmierer und Administrator oftmals dieselbe Person, während große Unternehmen die Funktionen gewöhnlich getrennt sind und auch mehrere Personen in jeder Rolle tätig sein können.

Ein SQL Server-Entwickler ist im Allgemeinen dafür verantwortlich, die Datenbank zu entwerfen, zu programmieren und mit den anfänglichen Daten zu füllen. Nachdem der Datenbankentwurf abgeschlossen ist, geht die Verantwortlichkeit für die Datenbank oftmals an den Administrator über, der sich um die laufenden Wartungsaufgaben kümmert.

Ein SQL Server-Administrator ist üblicherweise für die alltägliche Administration der Datenbank verantwortlich. Dieses Buch richtet sich hauptsächlich an Administratoren, viele Lektionen betreffen aber auch die Programmierer. Auch wenn manche Administratoren nie mit der Programmierung in Berührung kommen werden, sind die jeweiligen Lektionen so konzipiert, dass sich die Administratoren bei Bedarf in die entsprechenden Themen einarbeiten können. Die hier behandelten Programmieraufgaben beziehen sich auch auf Fertigkeiten, die gegebenenfalls vom Administrator verlangt werden – zum Beispiel Tabellen und Indizes erstellen oder komplexe Abfragen schreiben.

Über dieses Buch

Dieses Buch ist als Lehrbuch für den angehenden SQL Server-Administrator konzipiert. Es gliedert sich in 21 Lektionen, die Sie in 21 Tagen absolvieren können. Obwohl einige Lektionen länger als andere sind, sollte keine Lektion mehr als vier Stunden in Anspruch nehmen – der Durchschnitt liegt bei etwa zwei Stunden.

Das Buch beginnt mit den Grundlagen von SQL Server und vermittelt im weiteren Verlauf die unterschiedlichsten Kenntnisse und Verfahren, die ein Administrator für seine tägliche Arbeit braucht. Da sich das Buch auch an den Einsteiger richtet, wird nicht nur erklärt, was zu tun ist, sondern auch, warum es zu tun ist.

Die Lektionen dieses 21-Tage-Kurses bieten jeweils ein ungefähr gleiches Tagespensum. Weil die Lektionen aufeinander aufbauen, ist es wichtig, dass Sie sie in der richtigen Reihenfolge durcharbeiten oder mindestens verstehen, was in der Lektion eines gegebenen Tages vorkommt, falls Sie ihn überspringen. Die Themen und Inhalte der einzelnen Kapitel entnehmen Sie bitte dem eingelegten Lehrplan im Buch oder dem Inhaltsverzeichnis.

Wer sollte dieses Buch lesen?

Dieses Buch setzt keine Vorkenntnisse zu SQL Server voraus. Wenn Sie bereits etwas Erfahrung haben – um so besser.

Erfahrungen mit Windows NT oder Windows 2000 sind zwar nicht notwendig, helfen Ihnen aber bei Ihrer Arbeit mit SQL Server, weil viele der optionalen Komponenten von SQL Server eine Interaktion mit Windows NT/2000 erfordern.

In diesem Buch verwendete Konventionen

SQL Server lässt sich so installieren, dass bei der Eingabe von Anweisungen zwischen Groß- und Kleinschreibung unterschieden wird. Die Standardinstallation – und auch die für das Buch gewählte Beispielinstallation – sieht keine Berücksichtigung der Groß-/Kleinschreibung vor. Sämtliche Befehle an den Computer und insbesondere die Transact-SQL-Anweisungen sind in Schreibmaschinenschrift gedruckt. In vielen Beispielen sind die in SQL Server reservierten Wörter durchgängig in Großbuchstaben geschrieben:

```
CREATE DATABASE
```

Allerdings ist das nicht unbedingt erforderlich und man kann auch Kleinbuchstaben oder eine gemischte Schreibweise verwenden.



Abschnitte mit neuen Begriffen sind mit dem Symbol *Neuer Begriff* gekennzeichnet, damit Sie den gesuchten Begriff schnell finden können – vor allem, wenn Sie ihn anhand der Seitenzahl aus dem Index suchen.



In den Übungsbeispielen sind alle von Ihnen einzugebenden Anweisungen mit dem Symbol für Eingabe gekennzeichnet und alle Ergebnisse, die SQL Server liefert, mit dem Symbol für Ausgabe. Manchmal mussten lange Zeilen auf mehrere Druckzeilen aufgeteilt werden, damit sie sich vollständig darstellen lassen. SQL Server behandelt derartige Zeilen wie eine einzige Zeile. In den meisten Fällen ist es ohnehin nicht erforderlich, die Anweisungen genau Zeile für Zeile wie im Buch gezeigt einzugeben, da es sich aus der Syntax ergibt, wann eine logische Zeile beginnt und endet. Lediglich mitten in einer (in Anführungszeichen eingeschlossenen) Zeichenfolge sollten Sie keinen Zeilenumbruch vorsehen, da die zusätzlich eingefügten Zeichen als zur Zeichenfolge gehörend zählen würden und somit das Ergebnis beeinflussen.



Dieses Symbol kennzeichnet einen Abschnitt, der das jeweils voranstehende Codebeispiel erläutert.



Hier finden Sie interessante oder wichtige Punkte, die Ihnen das Verständnis der Konzepte und Verfahren näher bringen.



Warnungen liefern Informationen über ungeeignete Konstruktionen oder gefährliche Fehler.



Das sind kleine Info-Häppchen für Ihre Orientierung in praktischen Situationen.

Neueste Produktinformationen zu SQL Server finden Sie unter <http://www.microsoft.com/germany/backoffice/sql/> oder auf der englischen Site unter <http://www.microsoft.com/sql>.

Tag 1	Relationale Datenbanken und Microsoft SQL Server 2000	31	W O C H E 1
Tag 2	SQL Server 2000 installieren	45	
Tag 3	SQL Server 2000 Tools und Utilities	95	
Tag 4	Datenbanken und Datendateien erstellen und verwalten	151	
Tag 5	Sicherheit	187	
Tag 6	Berechtigungen	233	
Tag 7	Sicherungen von Datenbanken	275	

Tag 8	Wiederherstellen von Datenbanken	333	W O C H E 2
Tag 9	Datentypen und Tabellen	367	
Tag 10	Daten mit SELECT abrufen	397	
Tag 11	Daten bearbeiten	459	
Tag 12	Transact-SQL-Erweiterungen	481	
Tag 13	Indizes	523	
Tag 14	Datenintegrität	573	

Tag 15	Sichten, Trigger, gespeicherte Prozeduren und Funktionen	627	W O C H E 3
Tag 16	Replikation – Entwurf	665	
Tag 17	Replikation – Implementierung	701	
Tag 18	Aufträge mit dem SQL Server-Agent planen	759	
Tag 19	Daten zwischen Servern migrieren	805	
Tag 20	Leistungsüberwachung	829	
Tag 21	XML und SQL Server 2000	865	

Woche 1 – Vorschau

In dieser ersten Woche erhalten Sie genügend Informationen über Microsoft SQL Server 2000, damit Sie mit der Verwaltung und dem Entwurf von Datenbanken beginnen können.

Tag 1 gibt eine Einführung in SQL Server und einige Hintergrundinformationen über relationale Datenbanken und deren Entwurf.

Am Tag 2 installieren Sie SQL Server und untersuchen einige Setuptools.

Tag 3 betrifft die Tools und Utilities, die zum Lieferumfang von SQL Server gehören. Diese Tools setzen Sie auf Ihrem weiteren Weg mit SQL Server ausgiebig ein.

Tag 4 betrifft die Speicherung. Dabei beschäftigen Sie sich mit Datenbanken, Dateien und Dateigruppen.

Am Tag 5 geht es um das Sicherheitskonzept von SQL Server. Dabei erstellen Sie Benutzernamen und Datenbankbenutzer.

Tag 6 stellt das Konzept der Datenbankrollen vor und zeigt, welche Berechtigungen Datenbankbenutzern und Rollen erteilt werden können.

Tag 7 beginnt mit einem der wichtigsten Aspekte der Datenbankarbeit – den Datensicherungen. Wir Autoren sind oft als paranoid bezeichnet worden, weil wir diesen Datensicherungen (und Wiederherstellungen) eine solch immense Bedeutung zumessen. Andererseits sind wir nie von einer Firma entlassen worden, weil unsere Datenbank nicht wiederherzustellen gewesen wäre. Dieses Kapitel beschreibt den Datensicherungsprozess und gibt einige Tips, damit Sie Ihren Job behalten.

Das sieht am Anfang fast unüberwindbar aus, aber entspannen Sie sich! Angefangen von den Erläuterungen der grundlegenden Prinzipien bis hin zu den Übungen am Ende eines Tages vertiefen Sie Ihr Wissen und schaffen sich eine gute Basis, auf der die anderen beiden Wochen aufbauen.



Relationale Daten- banken und Microsoft SQL Server 2000



Der SQL Server-Entwickler entwirft und implementiert eine Datenbank, während der SQL Server-Administrator die täglich anfallenden Verwaltungsaufgaben wahrnimmt. Mit zunehmender Verbreitung von SQL Server verwischt sich allerdings diese strikte Trennung der beiden Aufgabenbereiche.

Die heutige Lektion bringt zuerst Hintergrundinformationen zu SQL Server und Windows (sowohl Windows 9x als auch Windows NT/2000), geht dann auf Datenbanken und das Client/Server-Modell ein und beschäftigt sich mit dem Aufbau von Datenbanken und ihren Inhalten. Schließlich werfen Sie einen Blick auf den Entwurf von Datenbanken.

1.1 SQL Server auf Erfolgskurs

SQL Server 2000 ist der direkte Nachfolger von SQL Server 7.0 und stellt damit erneut Microsofts Flaggschiff im Datenbankbereich dar. Die hohen Investitionen in die Entwicklung und Vermarktung von SQL Server 7.0 schlagen sich im Produkt SQL Server 2000 nieder. SQL Server 7.0 gehört zu den führenden Datenbanken auf der Windows NT/Windows 2000-Plattform mit einem Marktanteil von 68% (laut Microsoft-Website); SQL Server 2000 soll diesen Trend fortsetzen und den Anteil von Microsoft am Datenbankmarkt erweitern. SQL Server 2000 führt zahlreiche neue Funktionen ein und verbessert verschiedene Features, für die SQL Server 7.0 die Grundlagenarbeit geleistet hat.

SQL Server 2000 läuft unter Windows NT 4.0 (mit Service Pack 5 oder später), Windows 2000, Windows 98 und Windows ME. Außerdem ist eine Version für die Windows CE-Plattform verfügbar. Durch das günstige Preis-/Leistungsverhältnis von SQL Server ist die Leistung eines relationalen Datenbank-Managementsystems (RDBMS) zu einem Bruchteil der noch vor Jahren üblichen Kosten verfügbar. Im TPC-C Benchmark erreichte SQL Server 2000 Enterprise Edition die höchste Rate von Transaktionen pro Minute – die mit hin schnellste Datenbank der Welt ist zudem zu einem günstigen Preis erhältlich. Microsoft SQL Server wird auch in den kommenden Jahren Erweiterungen und Verbesserungen erfahren, so dass die Entscheidung für dieses Produkt auf einer soliden Basis steht.

Von Microsofts SQL Server sind seit der Einführung Millionen von Kopien verkauft worden – 1999 allein 10 Millionen. Die aktuelle Version des Produkts – SQL Server 2000 (intern als Version 8.0 bezeichnet) – ist Gegenstand dieses Buches. Bevor wir uns näher mit dem Einsatz von SQL Server 2000 beschäftigen, lohnt sich ein Blick auf die geschichtliche Entwicklung.



Aktuelle Mitteilungen und Unterstützung für SQL Server finden Sie unter der Webadresse <http://www.microsoft.com/sql/>.

1.2 Die Geschichte von SQL Server



In den siebziger Jahren hat IBM eine Computersprache speziell für Datenbank-abfragen entwickelt: SEQUEL – Structured English Query Language (strukturierte englische Abfragesprache). Im Laufe der Zeit sind verschiedene Erweiterungen hinzugekommen, so dass sich mit dieser Sprache nicht nur Abfragen ausführen, sondern auch Datenbanken erstellen und die Sicherheitsaspekte des Datenbanksystems verwalten lassen. Der Name dieser Sprache wurde später aus rechtlichen Gründen geändert und lautet jetzt SQL (Structured Query Language).

In den heutigen Datenbankmodulen sind verschiedene Versionen oder Dialekte von SQL anzutreffen. In SQL Server handelt es sich um das so genannte *Transact-SQL*. Auch wenn Sie in diesem Buch mit Transact-SQL arbeiten und die Grundlagen der Sprache lernen, konzentriert sich das Buch vor allem darauf, wie man SQL Server installiert, wartet und die Verbindungen herstellt.

Microsoft hat SQL Server (ein Datenbankprodukt, das die Sprache SQL versteht) ursprünglich mit der Firma Sybase für die Plattform IBM OS/2 entwickelt. Microsoft und IBM sind dann eigene Wege gegangen und Microsoft hat OS/2 zugunsten seines neuen Netzbetriebssystems, Windows NT Advanced Server, aufgegeben. Microsoft hat dann das SQL Server-Modul eigenständig für Windows NT weiterentwickelt. Das resultierende Produkt war Microsoft SQL Server 4.2 mit einem nachfolgenden Update auf die Version 4.21. Nach der Trennung von Microsoft und Sybase entwickelte Sybase sein Datenbankmodul für Windows NT weiter (derzeit unter der Bezeichnung Sybase Adaptive Server Enterprise erhältlich), während Microsoft die Produkte SQL Server 6.0 und später SQL Server 6.5 herausbrachte, die ebenfalls unter Windows NT laufen. Die Version SQL Server 7.0 ist sowohl für Windows NT als auch für Windows 95/98 geeignet.

SQL Server 7.0 ist in grundsätzlichen Teilen gegenüber den bisherigen Versionen überarbeitet worden und hat nun kaum noch etwas mit dem ursprünglichen Sybase-Code gemein. Das Datenbankmodul weist eine völlig neue Architektur auf; es sind unter anderem ein intelligenter Abfrageoptimierer und ein erweitertes Datenbankspeichermodule hinzugekommen. Gleichzeitig wurde die Skalierbarkeit, Zuverlässigkeit und Verfügbarkeit des Produkts erweitert; auch die Arbeit des Datenbankadministrators vereinfacht sich.



Wenn Sie SQL Server 2000 auf einem Windows 9x-System ausführen, steht nicht die vollständige Funktionalität von SQL Server zur Verfügung. Beispielsweise müssen Sie auf die Unterstützung mehrerer Prozessoren, Windows NT/2000-Sicherheit, NTFS-Datenträger und vieles mehr verzichten. Deshalb empfiehlt es sich, SQL Server 2000 unter Windows NT oder Windows 2000 und nicht unter Windows 9x auszuführen. Windows NT/2000 bietet darüber hinaus weitere Vorteile. Die NT-Plattform ist für mehrere Benutzer ausgelegt; bei Win-



dows 9x ist das nicht der Fall und die Leistung von SQL Server sinkt rapide ab, wenn man weitere Benutzer hinzufügt. Dieses Buch geht davon aus, dass Sie SQL Server unter Windows 2000 ausführen.

Unter Windows NT Workstation, Windows NT Server und allen Versionen von Microsoft Windows 2000 ist SQL Server 2000 als Dienst implementiert. Bei der Installation unter Windows 95/98 oder Windows ME läuft SQL Server als Anwendung unter dem derzeit angemeldeten Benutzer. Die Dienstprogramme wie SQL Server Enterprise Manager arbeiten als normale Client/Server-Anwendungen, so dass Sie sie von nahezu jeder Stelle im Netzwerk ausführen können.



Ein *Dienst* ist eine Anwendung, die Windows NT oder Windows 2000 entweder automatisch beim Hochfahren des Systems oder manuell auf Anforderung starten kann. Dienste in Windows NT/2000 verfügen über eine generische Anwendungsprogrammierschnittstelle (API), so dass sie sich per Programm steuern lassen. Mithilfe von Diensten ist es möglich, Anwendungen wie Microsoft SQL Server auszuführen, ohne dass sich ein Benutzer am Servercomputer anmelden muss.

1.3 Was ist eine Datenbank?



SQL Server verwendet als Datenbanktyp so genannte *relationale Datenbanken*. Das sind Datenbanken, bei denen die Daten in Form von Tabellen – d.h. in Zeilen und Spalten – organisiert sind. In Tabellen gruppiert man thematisch zusammengehörende Daten. Das Datenbankmodul stellt Beziehungen – oder *Relationen* – zwischen den einzelnen Tabellen her, um die Ergebnisse auf Abfragen bereitzustellen.

Allgemein kann man eine Datenbank als Sammlung verwandter Daten betrachten. Frühere Datenbankprodukte haben eine Datenbank gewöhnlich in einer einzigen Datei gespeichert – etwa in einer Datei *Angestellte.dbf* mit einer einzigen Datentabelle. Die Spaltenstruktur dieser Datei ist auf Angestelltdaten wie Gehalt, Einstellungsdatum, Name, Sozialversicherungsnummer und so weiter ausgelegt. Für jede Person in der Firma ist eine Zeile mit entsprechenden Werten in den jeweiligen Spalten vorgesehen. Separate Dateien enthalten die Indizes für den schnelleren Datenzugriff und alle sicherheitsrelevanten Elemente.

In SQL Server 2000 ist eine Datenbank nicht unbedingt an eine einzige Datei gebunden; eine Datenbank ist hier mehr ein logisches Konzept, das auf einer Sammlung verwandter Objekte basiert. Beispielsweise enthält eine Datenbank in SQL Server nicht nur die Roh-

daten, sondern auch die Struktur der Datenbank, alle Indizes, die Sicherheitselemente der Datenbank und gegebenenfalls andere Objekte wie Sichten oder gespeicherte Prozeduren, die sich auf diese spezielle Datenbank beziehen.

Objekte relationaler Datenbanken



Eine relationale Datenbank setzt sich aus unterschiedlichen Typen von Objekten zusammen. Die einzelnen Lektionen dieses Buches gehen im Detail auf diese Objekte ein. Die folgende Übersicht gibt eine kurze Beschreibung der gebräuchlichsten Objekte:

- *Tabellen* sind die Objekte, die die Datentypen und die tatsächlichen Rohdaten enthalten. Auf Tabellen geht Tag 9 näher ein.
- *Spalten* sind die Teile der Tabelle, die die Daten enthalten. Jeder Spalte ist ein Datentyp und ein eindeutiger Name zuzuweisen.
- *Datentypen* bezeichnen die Art der Daten in einer Spalte, zum Beispiel Zeichen, Zahlen oder Datumswerte. Jeder Spalte innerhalb einer Tabelle ist ein einziger Datentyp zugewiesen.
- *Gespeicherte Prozeduren* sind mit Makros vergleichbar: Mehrere Transact-SQL-Anweisungen lassen sich unter einem Namen zusammenfassen und speichern. Indem man die gespeicherte Prozedur aufruft, führt man die Transact-SQL-Anweisungen der Prozedur aus. Wenn man zum Beispiel einen wöchentlichen Bericht mit Transact-SQL-Anweisungen erstellt, kann man die einzelnen Anweisungen in einer gespeicherten Prozedur zusammenfassen und den Bericht dann einfach durch Aufruf der gespeicherten Prozedur erstellen. Mit gespeicherten Prozeduren kann man auch Sicherheitsmechanismen realisieren.
- *Benutzerdefinierte Funktionen* sind gespeicherten Prozeduren ähnlich und bestehen ebenfalls aus Transact-SQL-Code. Allerdings lassen sich Funktionen in Datenbankabfragen aufrufen, um entweder eine anzuzeigende Datenspalte zu modifizieren oder um Tabellen zurückzugeben, auch wenn diese per Programm oder dynamisch erstellt werden. Zum Beispiel können Sie Ihre eigenen Datumsfunktionen schreiben, um Spalten des Datentyps `datetime` zu modifizieren.
- *Trigger* sind spezielle gespeicherte Prozeduren, die der Benutzer nicht selbst aktiviert. SQL Server ruft einen Trigger auf, wenn die für den Trigger spezifizierte Datenänderung (Hinzufügen, Ändern, Löschen) an der Datenbank – d.h. einer bestimmten Tabelle oder Spalte – vorgenommen wird. Mit Triggern lassen sich Geschäftsregeln oder andere Regeln der Datenintegrität durchsetzen. Zum Beispiel kann man mit einem Trigger sicherstellen, dass jedem Buch in einer Buchhandlung ein gültiger Herausgeber zugeordnet ist.



- *Sichten* sind im Grunde genommen in der Datenbank gespeicherte Abfragen, die eine oder mehrere Tabellen referenzieren können. Sichten kann man erstellen und speichern, um sie später wiederzuverwenden. Gewöhnlich verwendet man Sichten, um bestimmte Spalten aus einer umfangreichen Tabelle auszuschließen oder um mehrere Tabellen miteinander zu verknüpfen. Man kann Sichten auch als Sicherheitsmechanismus einsetzen.
- *Indizes* unterstützen das Organisieren von Daten, so dass Abfragen schneller ablaufen. Tag 13 geht detailliert auf Indizes ein.
- *Primärschlüssel* sind an sich zwar keine Objekte, spielen aber in relationalen Datenbanken eine wichtige Rolle; sie erzwingen die Eindeutigkeit innerhalb der Zeilen und erlauben es, jeden Eintrag eindeutig zu identifizieren.
- *Fremdschlüssel* bestehen aus einer oder mehreren Spalten, die auf Primärschlüssel oder Einschränkungen anderer Tabellen verweisen. SQL Server verwendet Primär- und Fremdschlüssel, um Daten aus verschiedenen Tabellen bei einer Abfrage miteinander zu verknüpfen.
- *Einschränkungen* sind serverbasierte und im System implementierte Mechanismen, mit denen sich die Datenintegrität durchsetzen lässt.
- *Regeln* werden Spalten zugewiesen, so dass sich nur Daten eingeben lassen, die den festgelegten Standards entsprechen. Beispielsweise kann man mit Regeln sicherstellen, dass die Telefonnummer einer Person nur Ziffern enthält. SQL Server 2000 ersetzt Regeln funktionell durch CHECK-Einschränkungen.
- *Standardwerte* legt man für Felder fest, um diese Werte zu übernehmen, wenn eine INSERT-Operation für das betreffende Feld keinen Wert bereitstellt. Zum Beispiel können Sie die Telefonvorwahl für die Stadt, aus der die meisten Kunden kommen, als Standardwert festlegen, so dass Sie diese Vorwahl nicht jedes Mal neu eingeben müssen.

1.4 Relationale Datenbanken entwerfen

Der folgende Abschnitt zum Entwurf von relationalen Datenbanken ist beispielsweise für folgende Situationen wichtig:

- Sie können den Auftrag erhalten, eine relationale Datenbank zu erstellen.
- Sie müssen eine bereits vorhandene relationale Datenbank verwalten und wollen wissen, warum bestimmte Entwurfsentscheidungen getroffen wurden.

Als SQL Server-Administrator müssen Sie höchstwahrscheinlich eine Datenbank betreuen, die Sie nicht selbst entworfen und erstellt haben; das heißt aber nicht, dass Sie ohne Kenntnisse über den Entwurf einer relationalen Datenbank auskommen. Für Ihren Job kann es nur hilfreich sein, wenn Sie die Spielregeln beim Entwurf relationaler Datenbanken kennen und auch wissen, was es mit der Normalisierung auf sich hat.

Der Entwurf einer guten relationalen Datenbank könnte an sich ein ganzes Buch füllen. An dieser Stelle beschränken wir uns auf einige grundlegenden Schritte:

- Analysieren der Situation, um Informationen über die vorgeschlagene Datenbank zusammenzutragen.
- Festlegen der Spalten, Datentypen und Länge von Daten.
- Normalisieren der Daten in Tabellen.
- Erstellen der Datenbank und der Tabellen.

Wenn Sie zusammengehörende Daten in miteinander in Beziehung stehende Tabellen organisieren, wenden Sie die Normalisierungsregeln an, über die Sie in Kürze mehr erfahren.

Zu Beginn des Entwurfs sollten Sie die Geschäftssituation analysieren und ermitteln, was der Kunde erreichen möchte. Als Nächstes machen Sie sich Gedanken darüber, wie sich welche Variablen in Tabellen unterbringen lassen. Dann entwerfen Sie Berichte und Abfragen, die für die Benutzer nützlich sein können, und erarbeiten andere Teile des Datenbankentwurfs, einschließlich Zugriff auf Webseiten.

Die Übersicht »Was Sie tun sollten/was nicht« soll Ihnen beim Entwurf einer Datenbank helfen. Die Übung am Ende dieses Kapitels bringt ein fiktives Interview mit einem Kunden und führt zum Entwurf einer relationalen Datenbank.

Was Sie tun sollten	... was nicht
Fragen Sie die Benutzer, was sie brauchen.	Ignorieren Sie die Benutzer – die so genannten Kunden – nicht.
Erstellen Sie eine Liste von Objekten.	Erstellen Sie keine Objekte, die Sie nie brauchen werden.
Halten Sie die Objektnamen kurz und dennoch aussagekräftig.	Wählen Sie keine komplizierten Namen, Namen mit Leerzeichen oder Namen mit ungewöhnlichen Zeichen, weil diese mühsamer zu schreiben sind.



Was Sie tun sollten	... was nicht
Organisieren Sie die Eigenschaften von Objekten in den richtigen Gruppen. Erstellen Sie identisch benannte Spalten in zusammengehörenden Tabellen, um die Tabellen bei Abfragen verknüpfen zu können. Diese Spalten nutzen Sie für Primär- und Fremdschlüssel. Testen Sie Ihren Entwurf mit einigen Probedaten. Erstellen Sie für abgefragte Tabellen mindestens einen Index. Denken Sie beim Entwurf ihrer Tabellen an die Sicherheit. Dokumentieren Sie die Namen von Tabellen und Spalten sowie die verwendeten Primär- und Fremdschlüssel. Benennen Sie Datenbankobjekte nach einem einheitlichen Schema. Dies kann die Arbeit mit den Objekten wesentlich vereinfachen. Zum Beispiel können Sie die Objekte durch Präfixe kennzeichnen – <code>tblAngestellte</code> für ein Tabellenobjekt namens <code>Angestellte</code> und <code>idxNachname</code> für einen Index, der auf dem Nachnamen basiert.	Verwenden Sie keine Spalten, die mehrere Werte pro Zeile enthalten. Erstellen Sie keine Tabellen mit zu vielen Spalten. Wenn der Entwurf mit 5 Probezeilen funktioniert, sollten Sie nicht einfach daraus schließen, dass die Datenbank auch mit 500.000 Zeilen funktionstüchtig ist. Erstellen Sie pro Tabelle nicht mehr als etwa 5 Indizes. Lassen Sie nicht die Sicherheit Ihrer Daten außer acht. Verlieren Sie Ihre Dokumentation nicht.



Interview mit dem Kunden

Ein guter Datenbankentwurf beginnt mit einer gründlichen Analyse der Situation und dem gewünschten Ergebnis für den Kunden. Deswegen heißen die Leute, die neue Systeme entwerfen, *Analytiker* – sie analysieren das Problem detailliert und suchen Wege zur Lösung des Problems.

Manchmal lässt sich mit einem herkömmlichen Interview am besten herausfinden, was der Kunde wünscht, vor allem, wenn die derzeitige Situation und das Ziel noch nicht vollständig bekannt sind.

Sondieren Sie die Bedürfnisse Ihres Kunden beispielsweise mit folgenden Fragen:

- Wie werden die erforderlichen Arbeiten momentan erledigt?
- Welche Teile des aktuellen Systems würden Sie am liebsten ersetzen?
- Möchten Sie zusätzliche Berichte erstellen können?
- Welche Punkte werden Sie vorrangig verfolgen?
- Sind die Daten vertraulich oder öffentlich?
- Wer muss auf die Daten zugreifen und welche Art von Zugriff sollte jeder Benutzer oder jede Gruppe haben?
- Möchten Sie die Daten ins Internet stellen?
- Möchten Sie, dass die Öffentlichkeit über das Internet Informationen nachschlagen kann?
- Verfügen Sie über genügend Hardware, um sowohl die Datenbankserver- als auch die Clientsoftware auszuführen?
- Was würden Sie in das neue System einbauen, wenn Sie sich weder um die Finanzierung noch um die Technologie kümmern müssten?

Anhand derartiger Fragen können Sie recht schnell feststellen, ob eine Datenbank angebracht ist, und wenn ja, wie sie aussehen muss. Auch wenn Sie wahrscheinlich nicht alles bereitstellen können (wobei Grenzen hinsichtlich des zugewiesenen Budgets, des Zeitrahmens und der Hardware-Möglichkeiten gegeben sind), haben Sie zumindest den Ausgangspunkt für einen längerfristigen Plan zum Wachstum und zur Erweiterung der Datenbank.



Objekte organisieren

Nach dem Interview (Sie haben sich alles genau aufgeschrieben, oder nicht?) ist es am besten, ein Brainstorming über mögliche Objekte einschließlich ihrer Namen, Typen und Längen durchzuführen. Nachdem die Entscheidung über die Objekte gefallen ist, können sie in verwandte Tabellen gruppiert werden.

SQL Server unterstützt verschiedene Datentypen, einschließlich Zeichen, Zahlen, Datumswerte und Währungen. Mehr dazu erfahren Sie am Tag 9.

Nachdem Sie die Tabellenstruktur festgelegt haben, spezifizieren Sie die Eigenschaften (Spalten) innerhalb dieser Tabellen. Verwenden Sie einfache und dennoch aussagekräftige Spaltennamen. Die Länge der Spalten sollte für alle vorkommenden Fälle ausreichen, von extremen Ausnahmen abgesehen. Bei Namen können Sie sich zum Beispiel auf die Anzahl der Zeichen beschränken, die Sie auf einem Adresstikett unterbringen können – und nicht, wie viele Zeichen sich speichern lassen.

Normalisieren der Daten



Nachdem Sie Art und Umfang der erforderlichen Spalten kennen, müssen Sie die Daten auf Tabellen aufteilen, die miteinander in Beziehung stehen. Diesen Vorgang bezeichnet man als *Normalisieren* der Daten.

Durch Normalisieren der Daten versucht man, redundante Daten zu vermeiden. Nehmen wir beispielsweise an, dass derselbe Kunde zwei Autos kauft. In einer Einzeltabellendatenbank wären seine Daten doppelt einzugeben. Zieht der Kunde um, muss man seine Adresse an beiden Stellen ändern; andernfalls sind die Daten nicht mehr konsistent. Indem man die Angaben zu einem Kunden einmalig in einer Kundentabelle erfasst und den betreffenden Datensatz mit jedem Autokauf verknüpft, vermeidet man nicht nur überflüssige (und manchmal widersprüchliche) Daten, sondern muss Änderungen an einem Datensatz lediglich an einer Stelle vornehmen. Abbildung 1.1 zeigt ein Beispiel, wie diese Tabellen aussehen könnten. Es sind getrennte Tabellen für Kunden und Autos vorhanden. In der Tabelle Autos enthält das Feld KundNr lediglich eine Kundennummer und verweist damit auf eine Zeile in der Tabelle Kunden. Wie Sie sehen können, besitzt Ann zwei Autos, während Bob nur ein Auto gekauft hat. Cathy hat überhaupt noch kein Auto; aufgrund des Datenbankmodells lässt sich diese Tatsache aber leicht herausfinden. Ebenso einfach kann man mehrere Autos für Ann protokollieren, während man ihre Adresse nur einmal erfassen muss.

Tabelle Kunden				
KundNr	Name	Adresse	Stadt	etc.
1	Ann	123 AnyStreet	Redmond	
2	Bob	947 Main	Seattle	
3	Cathy	1010 1st Avenue	Bellevue	

Tabelle Autos				
AutoID	Hersteller	Modell	Jahr	KundNr
111AAA	Ford	Explorer	1999	1
222ABC	Honda	Civic	2000	1
112ADE	Toyota	Corolla	1998	2

Abbildung 1.1:
Variablen in einer
relationalen Daten-
bank organisieren

Normalformen



Daten normalisiert man nach bestimmten Regeln, die man in der Datenbanktheorie als *Normalformen* bezeichnet. Die drei wichtigsten Normalformen haben folgende Bedeutung:

- **Erste Normalform:** Diese Regel verlangt, dass eine Spalte nicht mehrere Werte enthalten darf. Um der ersten Normalform zu entsprechen, ist der Name einer Person in Nachname und Vorname(n) aufzuteilen.
- **Zweite Normalform:** Diese Regel verlangt, dass jede Spalte, die keine Schlüsselspalte ist, vom gesamten Schlüssel und nicht nur einem Teil des Primärschlüssels abhängen muss. Wenn Sie beispielsweise eine Kundennummer und eine Teilnummer für einen Schlüssel verwenden, dürfen sich alle Spalten in dieser Tabelle nur auf einen bestimmten Kunden und eine bestimmte Teilnummer gemeinsam beziehen. Somit würde eine Teilbeschreibung nicht in diese Tabelle gehören. Eine Tabelle muss außerdem der ersten Normalform entsprechen, damit sie sich nach der zweiten Normalform organisieren lässt.
- **Dritte Normalform:** Diese Regel ist der vorhergehenden sehr ähnlich und verlangt, dass alle Spalten, die keine Schlüsselspalten sind, nicht von irgendeiner anderen Spalte, die keine Schlüsselspalte ist, abhängen dürfen. Wenn Sie beispielsweise eine Tabelle mit Adressen haben, darf die Postleitzahl nicht von einem anderen Nicht-Schlüsselfeld wie zum Beispiel Bundesland, sondern nur vom gesamten Primärschlüssel abhängen. Natürlich muss die Tabelle auch der zweiten Normalform entsprechen. Manchmal ist es einfacher und bequemer, auf die Normalisierung nach der dritten Normalform zu verzichten.



Neben den hier beschriebenen Normalformen gibt es noch die vierte und fünfte Normalform. Den meisten Datenbankentwicklern genügt aber eine Normalisierung der Daten bis maximal zur dritten Normalform.

Datenbank und Tabellen erstellen

Weil Tabellen die Bausteine der Datenbanken sind, ist offensichtlich, dass gut entworfene Tabellen (und somit die Spalten innerhalb der Tabellen) für den Erfolg der Datenbanken entscheidend sind. Wie bei den meisten Dingen ist das Planen und Entwerfen der mühsamere Teil, während die tatsächliche Erstellung der Datenbank und Tabellen recht einfach ist. Eine Tabelle besteht aus Spalten, die die Eigenschaften einer Tabelle speichern. Tag 4 behandelt Datenbanken und ihre Erstellung detaillierter, Tag 9 geht ausführlich auf Tabellen ein.

1.5 SQL Server und das Client/Server-Modell

Microsofts SQL Server ist ein Datenbankmodul nach dem Client/Server-Modell; deshalb ist es wichtig, dass Sie dieses Modell verstehen.



Eine *Client/Server*-Anwendung gliedert sich grob in zwei Teile: Ein Teil läuft auf einem Server, der andere Teil auf Arbeitsstationen. Die Serverseite der Anwendung liefert Sicherheit, Fehlertoleranz, Leistung, parallele Programmausführung und zuverlässige Datensicherungen. Die Clientseite realisiert die Benutzeroberfläche und kann leere Berichte, Abfragen und Formulare enthalten. Das Konzept des Client/Server-Modells kombiniert die Vorteile beider Teile.

SQL Server arbeitet in diesem Modell auf der Serverseite. Es lassen sich verschiedene Clients wählen, die eine Verbindung zu SQL Server herstellen können. Dazu gehören Dienstprogramme wie SQL Server Query Analyzer, die im Lieferumfang von SQL Server enthalten sind.

SQL Server bietet sowohl für Clients als auch für Server Vorteile, wie Tabelle 1.1 zeigt.

Vorteile auf der Clientseite	Vorteile auf der Serverseite
Leicht zu verwenden	Zuverlässig
Unterstützt mehrere Hardwareplattformen	Parallelprozesse
Unterstützt mehrere Anwendungsprogramme	hochentwickelte Sperren
dem Benutzer vertraut	fehlertolerant
	Hochleistungs-Hardware
	zentralisierte Steuerung

Tabelle 1.1: Vorteile des Client/Server-Modells für SQL Server

Bei der Ausführung einer Abfrage durchsucht der Server in einer Client/Server-Umgebung die Datenbank und liefert nur die Zeilen an den Client zurück, die den Kriterien der Abfrage entsprechen. Einerseits verringert sich dadurch die erforderliche Netzwerkbandbreite, andererseits wird die Abfrage schneller als auf der Arbeitsstation ausgeführt, sofern es sich um einen ausreichend leistungsfähigen Servercomputer handelt.

1.6 Zusammenfassung

Das in der heutigen Lektion vorgestellte Material führt Sie in dieses Buch sowie in die grundlegenden Konzepte relationaler Datenbanken (einschließlich SQL Server) ein. Microsofts SQL Server nimmt mehr und mehr Marktanteile ein und ist ein relationales Datenbankmanagementsystem auf Client/Server-Basis. Als Abfragesprache verwendet SQL Server 2000 den SQL-Dialekt Transact-SQL.

Eine relationale Datenbank besteht aus Tabellen, die Spalten und Zeilen mit Daten enthalten. Die Aufteilung der Daten einer Datenbank auf Tabellen, die miteinander in Beziehung stehen, bezeichnet man als *Normalisierung*.

Der Entwurf einer guten Datenbank beginnt mit der Analyse der Kundenanforderungen. Nach diesem Schritt kann man die Daten auf einzelne Tabellen aufteilen.

1.7 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F *Muss ich Transact-SQL beherrschen?*

A Als Entwickler sollten Sie sich in dieser Sprache auskennen. Wenn Sie hauptsächlich bestehende Datenbanken verwalten müssen, steht Ihnen mit dem SQL Enterprise Manager eine graphische Oberfläche zur Verfügung, über die Sie fast alle Aufgaben erledigen können. Allerdings ist es für Ihre Arbeit mit SQL Server immer von Nutzen, wenn Sie mit Transact-SQL vertraut sind, weil Sie bei



bestimmten Befehlen Transact-SQL-Code eingeben müssen. Letztendlich lassen sich alle Operationen, die Sie im SQL Server Enterprise Manager durchführen, mit Transact-SQL ausdrücken.

F Welche Ähnlichkeiten weist SQL Server mit Sybase, Oracle oder Access auf?

- A** Frühere Versionen von SQL Server (4.x und 6.x) sind eng mit Sybase verwandt. Da Microsoft und Sybase später getrennte Wege gegangen sind, zeigen auch die jeweiligen Produkte unterschiedliche Charakteristika. SQL Server 2000 verrät heute kaum noch die gemeinsamen Wurzeln mit Sybase. Mit Oracle hat SQL Server fast nichts gemein, obwohl sich Oracle-Administratoren problemlos in SQL Server einarbeiten können, weil die zugrunde liegenden Konzepte der relationalen Datenbanken gleich sind. Access ist eine Einzelcomputerdatenbank, kann aber auch bei kleinen Implementierungen (weniger als 20 Benutzer, da die Leistung sonst spürbar leidet) als Server dienen. Access ist ein großartiger Frontend für SQL Server. Allerdings ist das Jet-Datenbankmodul, das frühere Versionen von Access unterstützt, bei weitem nicht so leistungsfähig wie das Datenbankmodul von SQL Server.



Zu Access 2000 gehört die Microsoft Data Engine (MSDE) – die Kerntechnologie von SQL Server 7.0. In SQL Server 2000 ist dieses Modul auf die Microsoft SQL Server 2000 Desktop Engine (MSDE) aktualisiert worden.

Kontrollfragen

1. Was sind die Bausteine einer relationalen Datenbank?
2. Nennen Sie einige Objekte, die in einer Datenbank gespeichert werden.
3. Wer ist für die Datensicherung von SQL Server-Datenbanken verantwortlich?

Übungen

1. Entwerfen Sie selbstständig eine Datenbank. Führen Sie dazu als erstes ein fiktives Interview durch. Basierend auf den gewonnenen Erkenntnissen erstellen Sie dann die Variablen und organisieren sie in Tabellen. Denken Sie daran, dass hier nicht nur eine Antwort richtig ist – es gibt unterschiedliche Wege, das Gleiche zu erreichen.
2. Stellen Sie sich vor, dass Ihr Onkel Joel seit Urzeiten einen Gebrauchtwagenhandel betreibt. Sie haben ihm geholfen, seine Computer und das Netzwerk einzurichten, und nun ruft er Sie in sein Büro.

Joel: Schön, dass Du vorbeikommen konntest. Mein Laden ist so groß geworden, dass ich kaum noch den Überblick behalten kann. Sue hätte fast ein Auto verkauft, das ich gar nicht habe, und Larry hat praktisch eines verschenkt, weil er den falschen Preis aufgeschrieben hat. Ich muss mich neu organisieren.

Sie: Hast du an eine Art Datenbank gedacht?

Joel: Du bist der Computerexperte. Entwirf mir etwas, damit ich meine Autos im Griff behalten kann. Ich habe auch Probleme damit, meine Verkäufer und den von ihnen gebrachten Umsatz im Auge zu behalten. In meinem Einmannbetrieb war das früher nicht so schwierig; heute nimmt es aber zuviel Zeit in Anspruch.

Sie: Brauchst du Berichte über die monatlichen Aktivitäten deiner Verkäufer und andere derartige Berichte?

Joel: Das würde mir sehr weiterhelfen.

Sie: Willst du in die Datenbank Bilder der Autos aufnehmen?

Joel: Es wäre prima, wenn du das machen könntest? Ich habe auch über diese Internetgeschichte gelesen und denke, es wäre toll, die Autos dort zu präsentieren.

Sie: An was hast du also gedacht?

Joel: Ich habe nur vage Vorstellungen. Du bist der Computerexperte.

Sie: Willst du, dass Leute sich im Internet deine Autos und Preise ansehen können?

Joel: Kannst du das machen? Können wir auch Farbbilder zeigen?

Sie: Ja, wir können Bilder, Preise und Extras auf der Webseite unterbringen. Welche Informationen willst du überhaupt in der Datenbank speichern?

Joel: Jahr, Hersteller, Modell, Farbe und gefahrene Kilometer sowie Extras wie Klimaanlage, Allradantrieb, CD-Player, Listenpreis und Einzelhandelspreis sollen für jeden sichtbar sein. Außerdem sollen die Interessenten ein Bild des Autos sehen und die verschiedenen Autos vergleichen können. Zusätzlich sollen Informationen gespeichert werden, die nur für mein Verkaufsteam sichtbar sind – beispielsweise wie hoch die tatsächlichen Kosten sind, ob wir den Wagen möglichst bald wieder abstoßen wollen oder wann das Auto auf den Hof gekommen ist.

Sie: Das sollte für den Anfang reichen. Hast du ein Budget geplant?

Joel: Also ich kann nicht das gesamte Budget dafür ausgeben, aber ich muss etwas machen oder mein Verkaufspersonal verkauft mit Verlust, wenn ich nicht aufpasse.

Sie: Ich mache mir ein paar Gedanken und komme dann wieder auf dich zu.



Zusammenfassung des Interviews

Aus diesem Interview lässt sich nicht nur ableiten, dass Joel eine Datenbank braucht, mit der sich Kosten und Verkaufsinformationen verfolgen lassen, sondern auch, dass die Datenbank eine Verbindung zu einer Webseite haben soll, so dass jeder auf öffentliche Daten über die verschiedenen Autos zugreifen kann, die zum Verkauf stehen.



SQL Server 2000 installieren



In der gestrigen Lektion haben Sie etwas über SQL Server 2000 und relationale Datenbanken im Allgemeinen gelernt. Microsoft SQL Server ist zwar ein ausgereiftes Produkt, aber die Version 2000 enthält eine große Zahl neuer Funktionen. Weiterhin haben Sie gesehen, warum der typische Entwickler oder Datenbankadministrator relationale Datenbanken braucht – die Kunden verlangen es. Sie müssen Eigenschaften und Objekte überwachen, die sich relativ mühelos mit Tabellen abbilden lassen, die aus in Spalten gegliederten Datenzeilen bestehen.

Die heutige Lektion befasst sich mit der Installation von SQL Server 2000. Obwohl die Ausführung des Setup-Programms nicht weiter schwierig ist, müssen Sie einige kritische Entscheidungen treffen, die auf Ihre gesamte Entwicklungsplattform Einfluss nehmen. Es kann sehr zeitaufwendig sein, hier begangene Fehler später zu beheben. Weiterhin müssen Sie die Hardware- und Software-Voraussetzungen kennen, um die beste Umgebung für den SQL Server auszuwählen.

2.1 Die SQL Server-Produktfamilie

Die erste entscheidende Frage lautet: »Welchen SQL Server brauche ich?« Microsoft bietet sechs Editionen von SQL Server 2000 an (mit der speziellen Windows CE Edition sind es sieben Versionen). Nachdem Sie sich mit den Voraussetzungen für die jeweiligen Versionen beschäftigt haben, dürfte Ihnen die Entscheidung nicht schwer fallen. Dabei sollten Sie auch beachten, dass alle Editionen (mit Ausnahme der Windows CE-Version) auf der gleichen Codebasis beruhen, so dass auch die Regeln, Bedingungen und administrativen Aufgaben gleich sind.

Standard Edition

Wenn man über SQL Server 2000 spricht, ist normalerweise die Standard Edition gemeint. Diese Version des Produkts enthält die volle Funktionalität und ist für Computer mit den Betriebssystemen Windows NT Server 4.0 (ab Service Pack 5) oder höher sowie Windows 2000 Server vorgesehen; sie läuft auch unter den Enterprise Editionen beider Betriebssysteme. Die Standard Edition von SQL Server 2000 lässt sich auf Systemen mit symmetrischem Multiprocessing und mit bis zu 4 CPUs und 2 GB RAM einsetzen.

Enterprise Edition

Die Enterprise Edition von SQL Server 2000 ist das Komplettangebot für Highend-Installationen. Diese Version läuft auf Windows NT Server 4.0 (SP5 oder höher) sowie unter Windows 2000 Server, Advanced Server oder Data Center Server. Auf der Hardwareseite unterstützt die Enterprise Edition bis zu 32 CPUs und 64 GB RAM. Außerdem ist eine verbesserte Unterstützung für Failovercluster realisiert. Die Verfügbarkeit dieser Merkmale ist auch vom Betriebssystem abhängig, unter dem SQL Server installiert ist. Dieses Buch bezieht sich auf die Enterprise Edition von SQL Server 2000 unter Windows 2000 Server. Allerdings lässt sich das hier behandelte Material auch auf die anderen Produktversionen anwenden.



Für das Jahr 2001 ist eine 64-Bit-Version der Enterprise Edition geplant, die sich auf den Itanium-Chip von Intel und das Windows 64-Bit-Advanced Server-Betriebssystem stützt.

Personal Edition

Die Personal Edition läuft unter Windows 98 und Windows ME – im Folgenden als Windows 9x bezeichnet – sowie Windows NT 4.0 Workstation oder Windows 2000 Professional. Diese Version ist als Entwicklungs- und Remote-SQL Server-Installation vorgesehen, um einen zentralen Server zu unterstützen. Sie kann mehrere Benutzer gleichzeitig unterstützen (Microsoft hat keine Begrenzung codiert), ist aber in der Regel nur für weniger als 10 Benutzer geeignet. Die Datenbankgröße unterliegt keiner Beschränkung, allerdings unterstützen Windows NT Workstation/Windows 2000 Professional nur zwei Prozessoren und Windows 9x nur einen Prozessor.

Die Personal Edition unterstützt die meisten Features von SQL Server, erlaubt aber keine Veröffentlichung per Transaktionsreplikation (die Replikation ist Thema von Tag 16). Bei Ausführung auf der Windows 9x-Plattform ergeben sich zusätzliche Einschränkungen, auf die diese Lektion später eingeht. Allerdings sind die meisten von ihnen für einen Datenbankentwickler nicht sichtbar, so dass Sie mit der Desktop Edition entwickeln und dann die Datenbank auf der Standard oder Enterprise Edition nutzen können, wobei die 100%ige Codekompatibilität gewährleistet ist.

Developer Edition

Die Developer Edition von SQL Server 2000 umfasst alle Funktionen der Enterprise Edition, beinhaltet jedoch einen besonderen Endbenutzer-Lizenzvertrag (EULA) für Entwicklungen und Tests, der die produktive Nutzung untersagt.



Enterprise Evaluation Edition

Die Evaluation Edition von SQL Server 2000 entspricht ebenfalls der Enterprise Edition, ist jedoch zeitlich begrenzt und weist weitere Beschränkungen hinsichtlich der Nutzung und Bereitstellung auf.

Microsoft SQL Server 2000 Desktop Engine (MSDE)

Die MSDE-Version von SQL Server 2000 ist eine eingeschränkte Version des Vollprodukts. Sie baut auf der gleichen Codebasis auf, enthält jedoch keine Benutzeroberfläche, Verwaltungstools, Analysefunktionen, Unterstützung der Mergereplikation, Clientzugriffslizenzen, Entwicklerbibliotheken oder Onlinedokumentationen.

Diese Edition ist als Backend-Datenspeicher für Anwendungen vorgesehen. Normalerweise erhalten Sie die MSDE-Version, wenn Sie das Microsoft Visual Studio oder die Microsoft Office 2000 Developer Edition kaufen. Dann können Sie Anwendungen entwickeln und MSDE als eingebetteten Teil der Anwendung weiterverbreiten. Wenn Sie über eine Voll- (oder Test-) Version der Tools von SQL Server 2000 verfügen, können Sie damit eine MSDE-Version von SQL Server verwalten, so dass sich dieses Buch ebenso auf MSDE wie auf vollständige Versionen von SQL Server 2000 bezieht.

2.2 Voraussetzungen für die Installation von SQL Server

Bevor Sie mit der Installation von SQL Server 2000 beginnen, müssen Sie wissen, welche Voraussetzungen auf der Hard- und Softwareseite zu erfüllen sind und welche Lizenzbedingungen bestehen. Wenn Sie SQL Server 2000 auf Windows NT/2000-Systemen einsetzen wollen, müssen Sie einige zusätzliche Überlegungen zu den Hardware- und Software-Optionen, Windows NT/2000-Optionen und Lizenzbedingungen anstellen.

Hardware- und Software-Voraussetzungen

Vielleicht sollten Sie als Erstes überprüfen, ob SQL Server 2000 überhaupt auf Ihrem Rechner laufen kann. Wie bei den meisten neuen Software-Versionen von Microsoft benötigt SQL Server 2000 mehr »Pferdestärken« als die Vorgängerversion. Generell kann man sagen, je leistungsfähiger der Computer ist, um so zufriedener werden Sie mit der Leistung von SQL Server 2000 sein.

Unterstützte Hardware

Als Untergrenze für den Einsatz von SQL Server 2000 gibt Microsoft den Intel-Prozessor Pentium 166 MHz an. Derzeit sind die Intel-Prozessoren Pentium, Pentium Pro, Pentium II und Pentium III erhältlich. Natürlich werden auch Systeme mit Pentium-kompatiblen Befehlssatz unterstützt. Für den Arbeitsspeicher sind mindestens 32 MB erforderlich, die Enterprise Edition verlangt mindestens 64 MB.



Obwohl die angegebene Minimalkonfiguration einen Pentium 166 vorsieht, sollte SQL Server 2000 auf jedem Pentium-basierten System (wenn auch langsamer) laufen. 486er-Prozessoren werden nicht unterstützt.

Der erforderliche Festplattenplatz hängt davon ab, welche Komponenten der Software Sie für die Installation auswählen. Eine minimale Installation belegt mindestens 65 MB auf der Festplatte, eine vollständige Installation nimmt 180 MB Festplattenplatz in Anspruch. Zusätzliche SQL Server-Komponenten erfordern weiteren Platz. Zum Beispiel benötigt English Query 12 MB und die Microsoft SQL Server OLAP (Online Analytical Processing) Services können bis zu 95 MB in Anspruch nehmen.

Es wird empfohlen, die Installation von einem lokalen CD-ROM-Laufwerk auszuführen, falls Sie Zugriff auf ein Netzwerk haben, können Sie SQL Server 2000 auch von einer auf einem Netzlaufwerk vorliegenden Kopie der CD-ROM installieren. Diese Installationsmethode ist praktisch, wenn Sie Komponenten automatisch ohne Benutzereingriff installieren wollen.

Auswahl des Betriebssystems

Nachdem die Hardware-Frage geklärt ist, müssen Sie sich für eines der unterstützten Betriebssysteme entscheiden. SQL Server 2000 läuft unter Windows NT (Version 4.0 oder höher mit Service Pack 5 oder höher) in jeder Version (Workstation, Server, Small Business Server oder Server Enterprise Edition). SQL Server lässt sich auch unter jeder Version von Windows 2000 sowie auf Windows 98- und Windows ME-Computern ausführen. Für die Windows 9x-Versionen gelten jedoch einige Einschränkungen.

Einschränkungen unter Windows 9x

Die Client-Komponenten von SQL Server laufen unverändert auf Windows 9x-Plattformen. Die Personal Edition von SQL Server verhält sich auf Windows 9x Systemen aufgrund von betriebssystembedingten Einschränkungen jedoch etwas anders.

- Die Netzbibliothek **Named Pipes**, **Banyan VINES** und **AppleTalk** werden nicht unterstützt.
- **Windows NT-Authentifizierung** (die so genannte integrierte Sicherheit) steht nicht zur Verfügung.



- Serverseitige Multiprotokoll-Verschlüsselung wird nicht unterstützt.
- Asynchron▷ und Scatter-Gather-E/A-Operationen stehen nicht zur Verfügung.
- Da Windows 9x keine Dienste unterstützt, laufen SQL Server-Komponenten nicht als Dienste, sondern als Anwendungen wie jedes andere Programm.
- Der Systemmonitor und die Ereignisanzeige stehen nicht zur Verfügung.
- Die Speicheroptimierung ist fest auf minimalen Speicherverbrauch eingestellt.

Vielleicht sagen Ihnen einige dieser Begriffe im Moment noch nichts, die einzelnen Lektionen dieses Buches erläutern aber alle Einschränkungen und ihre Implementierungsdetails. Die meisten von ihnen werden Sie jedoch gar nicht bemerken.

Windows NT/Windows 2000-Optionen

Nachdem Sie sich mit einigen Bedingungen für die Windows 9x-Plattform bekannt gemacht haben, sollten Sie sich mit den Merkmalen beschäftigen, die für Windows NT und Windows 2000 spezifisch sind. Als Plattform kommt vor allem Windows NT bzw. Windows 2000 infrage, weil hier alle Produktmerkmale von SQL Server zur Verfügung stehen. Dabei empfiehlt es sich, mit Windows 2000 statt mit Windows NT 4.0 zu arbeiten, weil Windows 2000 zuverlässiger ist und ein besseres Sicherheitssystem aufweist.

Die wichtigsten Windows NT/2000-Optionen sind Sicherheit und das NTFS-Dateisystem. Es gibt jedoch noch mehr Gründe, sich für Windows NT/2000 zu entscheiden. Dieses Buch nimmt an, dass alle Funktionen auf den meisten Plattformen verfügbar sind; auf solche, die nur unter Windows NT/2000 zur Verfügung stehen, wird jedoch besonders eingegangen. Da es nur verwirren würde, alle betriebssystemabhängigen Merkmale für jede Version einzeln zu behandeln, haben wir als Basis für dieses Buch das Betriebssystem Windows 2000 gewählt. Sofern also nicht anders erwähnt, beziehen sich alle Teile des Buches, die mit betriebssystemabhängigen Komponenten zu tun haben, auf Windows 2000.

Sicherheitsoptionen



Eine der wichtigsten unter Windows NT/2000 zur Verfügung stehenden Optionen ist die Sicherheit. Windows NT und Windows 2000 sind sichere Betriebssysteme, mit denen Sie festlegen können, wer welche Operationen mit welchen Dateien durchführen darf, und mit denen Sie auch die Zugriffsrechte über Windows NT/2000-Sicherheitskonten steuern können. Dabei handelt es sich um die so genannte *Windows-Authentifizierung* oder *integrierte Sicherheit*. Damit können Sie Benutzer- und Gruppenkonten von Windows direkt in SQL Server verwenden. Auf diese Eigenschaft geht Tag 5 näher ein.

Eine weitere sicherheitsrelevante Überlegung besteht darin, ob SQL Server 2000 auf einem Domänencontroller installiert werden soll. Aus der Sicht des SQL Servers ist es am günstigsten, SQL Server auf einem eigenen Rechner zu installieren, Sie werden dann eine höhere Leistung registrieren können. Um die Möglichkeiten der integrierten Sicherheit von SQL Server 2000 unter Windows NT/2000 ausnutzen zu können, müssen Sie den SQL Server auf einem Computer mit Zugriff auf die Konten Ihrer Domäne installieren. Dies bedeutet (für Windows NT 4.0-Netzwerke), dass der SQL Server auf einem Rechner installiert werden sollte, der Mitglied einer Domäne ist, die Ihrer Masterdomäne vertraut. Bei Windows 2000-Netzwerken installieren Sie SQL Server entweder innerhalb der abwärtskompatiblen Domänenstruktur oder als Computer, der Mitglied des Active Directory ist.



Falls Sie den obigen Absatz nicht verstehen sollten, sehen Sie in Ihrer Windows NT- bzw. Windows 2000-Dokumentation nach oder fragen Sie Ihren Netzwerkadministrator. Das sollte Ihnen helfen, den richtigen Computer in Ihrem Netzwerk für die Installation des SQL Servers auszuwählen.

Dateisystemoptionen

Mit SQL Server 2000 können Sie sowohl FAT-, FAT32- als auch NTFS-Dateisysteme einsetzen. Aus Gründen der Sicherheit und Zuverlässigkeit empfiehlt sich vor allem das Dateisystem NTFS. Bei einer Installation im NTFS-Dateisystem sichert SQL Server die Installationsdateien einschließlich Ihrer Systemdatenbankdateien. NTFS ist zudem bei der Erstellung neuer Datenbanken sehr viel schneller. Unter Windows 2000 können Sie auch die Unterstützung für das verschlüsselnde Dateisystem (EFS, Encrypted File System) nutzen, um die Datenbankdateien zu verschlüsseln – niemand kann dann die Datenbankdateien nutzen, ohne den Benutzernamen und das Kennwort des SQL Server-Dienstkontos zu kennen.

Lizenzierung

Das Lizenzierungsmodell für SQL Server 2000 sieht die Lizenzierungsmodi Pro-Arbeitsplatz und Pro-Prozessor vor. Bei der Pro-Arbeitsplatz-Lizenzierung erwerben Sie je eine Lizenz für jeden Computer, der auf einen SQL Server zugreift. Nachdem Sie eine Pro-Arbeitsplatz-Lizenz erworben haben, können Sie auf eine beliebige Zahl von SQL Servern in Ihrem Netzwerk zugreifen.

Bei der Pro-Prozessor-Lizenzierung erwirbt ein Kunde eine Prozessorlizenz für jeden Prozessor, der die Serversoftware ausführt. Verfügt Ihr Computer beispielsweise über acht Prozessoren, die Sie alle für die Ausführung von SQL Server verfügbar machen wollen, brauchen Sie acht Pro-Prozessor-Lizenzen.



Das Recht zur Installation der Personal Edition von SQL Server 2000 wird nur Benutzern zugestanden, die eine Pro-Arbeitsplatz-Lizenz erworben haben. Wählen Sie also nicht die Option Pro-Prozessor, wenn Sie die Personal Edition von SQL Server installieren möchten.

Bei beiden Lizenzierungsvarianten und mit der Enterprise Edition von SQL Server 2000 können Sie bis zu 16 Instanzen von SQL Server auf ein und demselben Computer installieren. Für die Standard Edition müssen Sie jede zu installierende Instanz separat erwerben, wobei auch hier ein Maximum von 16 Instanzen gilt.



Die Autoren dieses Buches sind keine Rechtsanwälte. Hinsichtlich der korrekten Lizenzierung von SQL Server sollten Sie Ihre Rechtsabteilung konsultieren. Die Übersicht in diesem Buch hat nur informativen Charakter. Die Lizenzierungsmodalitäten können sich ändern oder aufgrund von Verhandlungen besonderen Bedingungen unterliegen.

2.3 SQL Server 2000 installieren

Nachdem nun die Lizenzfrage geklärt ist und Sie ihre Plattform (Hardware und Betriebssystem) ausgewählt haben, können Sie mit der Installation beginnen. Diesem Buch liegt ein System mit Windows 2000 Server auf einem Pentium III mit 600 MHz und 128 MB RAM zugrunde. Außerdem ist SQL Server 6.5 installiert, um alle vorhandenen Auswahlmöglichkeiten zeigen zu können. Alle Screenshots in diesem Buch wurden auf diesem System erstellt. Auf plattformspezifische Besonderheiten wird an den jeweiligen Stellen ausdrücklich hingewiesen.

Start des Installationsvorganges

Um mit der Installation zu beginnen, legen Sie die CD-ROM in Ihr CD-ROM-Laufwerk. Wenn die Autostart-Funktion aktiviert ist, erscheint das Autostart-Dialogfeld (siehe Abbildung 2.1). Falls das nicht automatisch beim Einlegen der CD-ROM geschieht, navigieren Sie mit dem Windows-Explorer zum CD-ROM-Laufwerk und starten das Programm `autorun.exe` im Stammverzeichnis.

Das Autostart-Menü

Bereits im Autostart-Menü gibt es mehrere Auswahlmöglichkeiten. Als ersten Schritt sollten Sie den Punkt *Anmerkungen zu dieser Version lesen* wählen. Daraufhin startet der Editor und zeigt die Infodatei `Readme.txt` mit ergänzenden Informationen zur Dokumentation

von Microsoft SQL Server 2000 an. Diese Informationen können sehr umfangreich sein; Sie sollten sich zunächst nur mit den für die Installation relevanten Punkten näher befassen.



Abbildung 2.1:
Das Autostartmenü
von Microsoft SQL
Server

Falls Sie mit der Onlinedokumentation arbeiten wollen, muss Internet Explorer 5.0 oder höher installiert sein. So können Sie das Installationshandbuch lesen, bevor Sie SQL Server installieren, selbst wenn Ihnen keine gedruckte Dokumentation vorliegt. Allerdings umfasst dieses Kapitel alles, was Sie zur Installation des SQL Servers 2000 wissen müssen. Es gibt hier auch einen Link zu Microsofts SQL Server-Webseite (<http://www.microsoft.com/germany/backoffice/sql/2000/default.htm>), wo Sie spezielle Produktinformationen und Utilities für SQL Server 2000 finden. Die zweite Zeile auf der linken Seite des Autostart-Menüs bezieht sich auf die Voraussetzungen zum Betrieb von SQL Server 2000.

Betriebssystemabhängige Voraussetzungen

Je nach verwendetem Betriebssystem gibt es unterschiedliche Voraussetzungen für die Installation von SQL Server. Für Windows 95 müssen Sie die Winsock 2- und die Bibliothek für Standardsteuerelemente aktualisieren. Windows 95 unterstützt nur die SQL Server-Verwaltungstools und nicht alle Editionen der SQL Server-Produktfamilie. Obwohl Internet Explorer 5.0 (oder höher) nicht über das Autostartmenü von SQL Server verfügbar ist, muss er vorhanden sein, bevor Sie SQL Server oder die Clienttools für SQL Server installieren können. Das hindert Sie nicht daran, einen anderen Webbrowser (z.B. Net-



scape Navigator) zu verwenden, aber Sie müssen eine Minimalinstallation des Internet Explorers durchführen, bevor Sie den Installationsvorgang fortsetzen können. Bei Windows 2000 ist bereits die richtige Version in das System integriert. Bei Windows NT 4.0-Computern müssen Sie außerdem Service Pack 5 installieren. Beide Komponenten (Internet Explorer und das Windows NT Service Pack) müssen installiert sein, bevor Sie mit der Installation von SQL Server 2000 beginnen können.

SQL Server 2000 (vollständig) installieren

Nachdem Sie die Softwarevoraussetzungen installiert haben (falls das überhaupt notwendig ist), können Sie mit der Installation von SQL Server 2000 beginnen. Wählen Sie aus dem Autostartmenü den Eintrag *SQL Server 2000-Komponenten*. Daraufhin erscheint das Menü gemäß Abbildung 2.2.



Abbildung 2.2:
Installation der
Komponenten für
SQL Server 2000

Hier können Sie wählen, ob Sie den Datenbankserver (d.h. den eigentlichen SQL Server), die Analysis Services oder English Query installieren wollen. Für die Themen dieses Buches genügt die erste Option. Die optionalen Komponenten Analysis Services und English Query gehen über die Ziele dieses Buches hinaus und werden hier nicht behandelt.



Die zum Lieferumfang von SQL Server gehörende Personal Edition kann auf jedem unterstützten System laufen – das betrifft die Systeme Windows 98, Windows ME, Windows NT Workstation, Windows 2000 Professional, Win-

dows NT Server, Windows 2000 Server und sogar die Enterprise Edition von Windows NT Server. Es ist auch möglich, die Personal Edition unter Windows 2000 Advanced Server oder Windows 2000 Data Center Server auszuführen.

Dieses Kapitel führt Sie durch die Installation der Vollversion (der Enterprise Edition von SQL Server 2000). Die Installation der Personal Edition ist nahezu identisch. Wenn Sie die Failover-Clusterunterstützung installieren wollen, erfordert die Installation der Enterprise Edition verschiedene spezielle Überlegungen, die über den Rahmen dieses Buches hinausgehen.

Um mit dem Setup ausgehend vom Menü nach Abbildung 2.2 fortzufahren, führen Sie die folgenden Schritte aus:

1. Klicken Sie auf *Datenbankserver installieren*. Das erste Dialogfeld begrüßt Sie beim Microsoft SQL Server-Installations-Assistenten. Klicken Sie auf WEITER. Als Nächstes können Sie wählen, ob Sie SQL Server auf dem lokalen Computer oder auf einem Computer im Netzwerk installieren wollen (siehe Abbildung 2.3). Übernehmen Sie die Standardeinstellung, um das Programm auf dem lokalen Computer zu installieren, und klicken Sie auf WEITER.



Verwenden Sie die Option *Virtueller Server*, wenn Sie SQL Server 2000 in einem Failovercluster installieren.

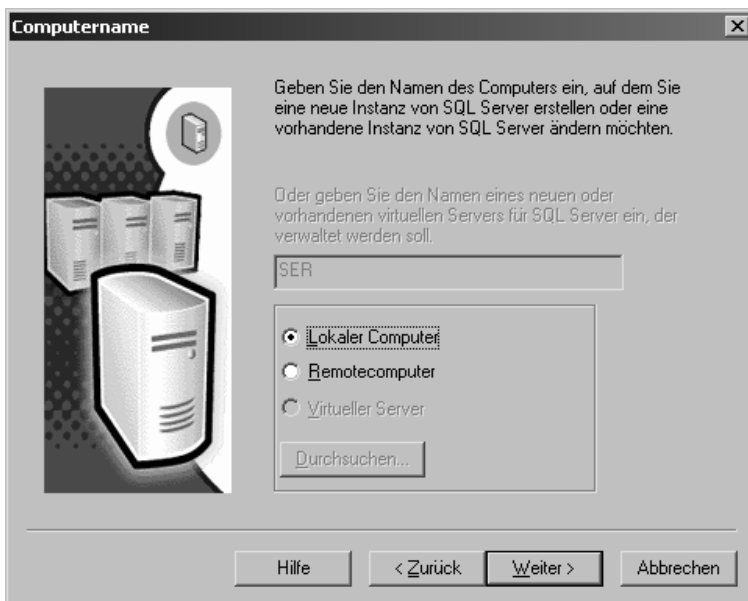


Abbildung 2.3:
Installation auf dem
lokalen oder einem
Remotecomputer?

2. Das Installationsprogramm durchsucht den Computer nach einer bereits vorhandenen Version von SQL Server. Nach dieser Überprüfung gelangen Sie zum Dialogfeld gemäß Abbildung 2.4 (unter der Annahme, dass noch keine SQL Server-Version auf dem Computer vorhanden ist).
3. Übernehmen Sie die Standardeinstellung, um eine neue Instanz von SQL Server zu erstellen, und klicken Sie auf WEITER. Im Dialogfeld BENUTZERINFORMATIONEN geben Sie Ihren Namen und optional den Namen Ihrer Firma ein und klicken auf WEITER. Lesen Sie sich dann die Bestimmungen im Dialogfeld SOFTWARE-LIZENZVERTRAG durch. Wenn Sie mit allen Punkten einverstanden sind, klicken Sie auf JA, um die Vereinbarung zu akzeptieren und mit der Installation von SQL Server fortzufahren. Durch Klicken auf NEIN brechen Sie die Installation ab. Wir gehen hier davon aus, dass Sie dem Lizenzvertrag zustimmen. Klicken Sie also auf JA.



Abbildung 2.4:
Auswahl einer
Installationsoption



Benutzerinformationen

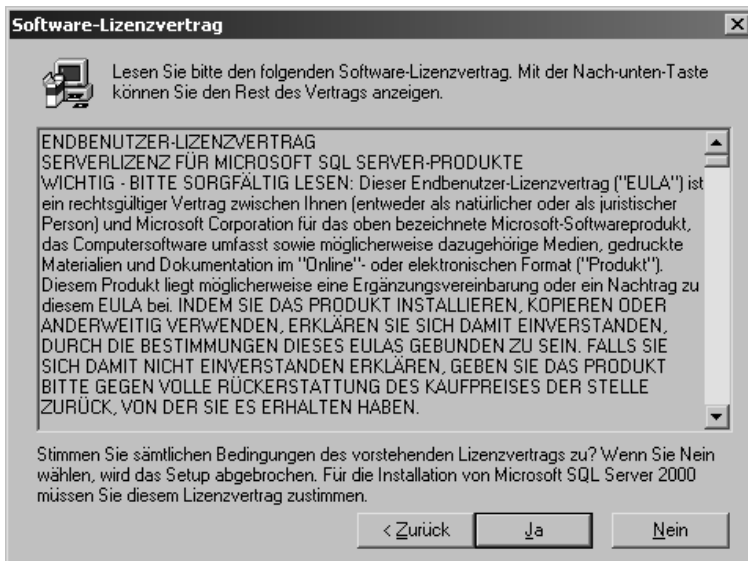
Geben Sie unten Ihren Namen ein. Die Eingabe des Firmennamens ist nicht erforderlich.

Name:

Firma:

< Zurück Weiter > Abbrechen

Abbildung 2.5:
Benutzerinformationen
bereitstellen



Software-Lizenzvertrag

Lesen Sie bitte den folgenden Software-Lizenzvertrag. Mit der Nach-unten-Taste können Sie den Rest des Vertrags anzeigen.

ENDBENUTZER-LIZENZVERTRAG
SERVELIZENZ FÜR MICROSOFT SQL SERVER-PRODUKTE
WICHTIG - BITTE SORGFÄLTIG LESEN: Dieser Endbenutzer-Lizenzvertrag ("EULA") ist ein rechtsgültiger Vertrag zwischen Ihnen (entweder als natürlicher oder als juristischer Person) und Microsoft Corporation für das oben bezeichnete Microsoft-Softwareprodukt, das Computersoftware umfasst sowie möglicherweise dazugehörige Medien, gedruckte Materialien und Dokumentation im "Online"- oder elektronischen Format ("Produkt"). Diesem Produkt liegt möglicherweise eine Ergänzungsvereinbarung oder ein Nachtrag zu diesem EULA bei. INDEM SIE DAS PRODUKT INSTALLIEREN, KOPIEREN ODER ANDERWEITIG VERWENDEN, ERKLÄREN SIE SICH DAMIT EINVERSTANDEN, DURCH DIE BESTIMMUNGEN DIESES EULAS GEBUNDEN ZU SEIN. FALLS SIE SICH DAMIT NICHT EINVERSTANDEN ERKLÄREN, GEBEN SIE DAS PRODUKT BITTE GEGEN VOLLE RÜCKERSTATTUNG DES KAUFPREISES DER STELLE ZURÜCK, VON DER SIE ES ERHALTEN HABEN.

Stimmen Sie sämtlichen Bedingungen des vorstehenden Lizenzvertrags zu? Wenn Sie Nein wählen, wird das Setup abgebrochen. Für die Installation von Microsoft SQL Server 2000 müssen Sie diesem Lizenzvertrag zustimmen.

< Zurück Ja Nein

Abbildung 2.6:
Die Lizenzvereinbarung
zu SQL Server
2000

4. Klicken Sie auf WEITER. Es erscheint das Dialogfeld CD KEY, in das Sie den 25-stelligen CD-Code eingeben müssen. Diesen finden Sie auf dem gelben Aufkleber der CD-Hülle. Ohne gültige SQL Server-Produkt-ID können Sie den Installationsvorgang nicht fortsetzen.

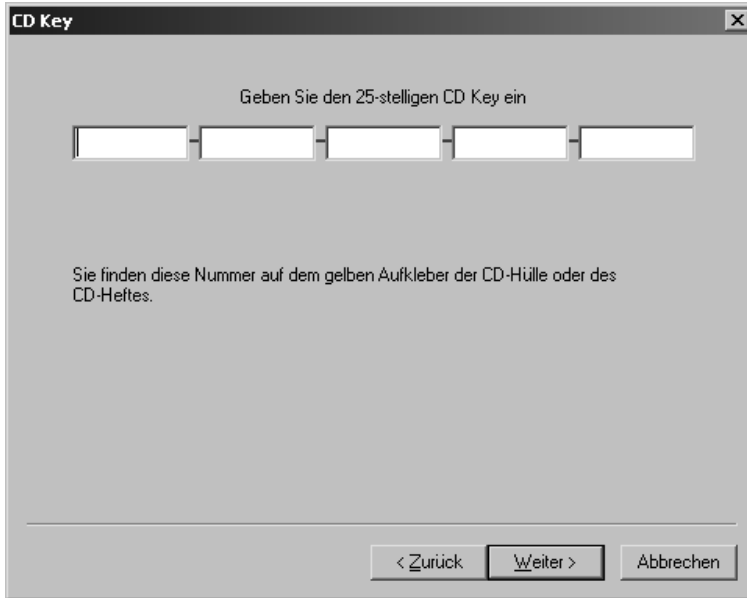


Abbildung 2.7:
Hier geben Sie den
CD Key ein.

5. Nachdem Sie den richtigen Code eingegeben haben, klicken Sie auf WEITER. Damit gelangen Sie zum Dialogfeld INSTALLATIONSDEFINITION (siehe Abbildung 2.8).



Abbildung 2.8:
Hier wählen Sie die
Installationsart aus.

Um SQL Server komplett (die Serversoftware und die Clienttools) zu installieren, übernehmen Sie die Voreinstellungen *Server- und Clienttools*. Wenn Sie keinen Server installieren müssen und eine Verbindung mit einem vorhandenen Server herstellen, wählen Sie die Option *Nur Clienttools*. Möchten Sie lediglich die Microsoft Data Access Components (MDAC) und Netzwerkbibliotheken installieren, wählen Sie die Option *Nur Konnektivität*.

6. Wählen Sie die Standardoption und klicken Sie auf WEITER. Daraufhin gelangen Sie zum Dialogfeld INSTANZNAME (siehe Abbildung 2.9).
7. Übernehmen Sie zunächst die Standardoption, um eine Standardinstanz von SQL Server zu installieren. Was eine benannte Instanz ist, untersuchen Sie später in dieser Lektion. Dann installieren Sie auch eine benannte Instanz von SQL Server.
8. Klicken Sie auf WEITER, um zum Dialogfeld SETUP-TYP zu gelangen (siehe Abbildung 2.10).



Abbildung 2.9:
Den Instanznamen
wählen

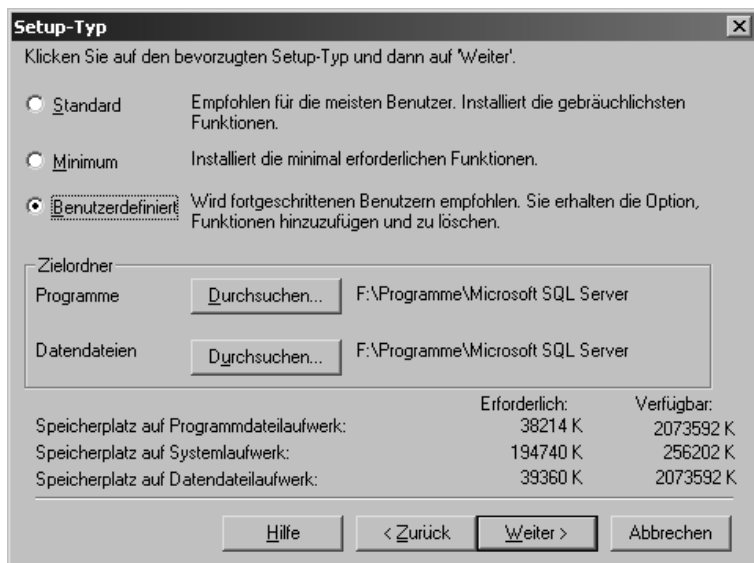


Abbildung 2.10:
Den Setup-Typ
wählen

Wenn Sie die Standardinstallation wählen, übernimmt das Setupprogramm alle Vorgabeeinstellungen und Sie gelangen direkt zum Dialogfeld entsprechend Abbildung 2.15. Alle Komponenten mit Ausnahme der Entwicklungswerkzeuge und Codebeispiele werden installiert; allerdings erhalten Sie den Transact-SQL-Debugger auch bei der Standardinstallation. Bei einer Minimalinstallation durchlaufen Sie die gleichen Schritte in Bezug auf die Dialogfelder, es werden aber nur die Komponenten für den Clientzugriff (ODBC usw.) und die SQL Server-Kerndienste installiert; hier fehlen alle Verwaltungswerkzeuge und die Onlinedokumentation.

In Abbildung 2.10 ist die benutzerdefinierte Installation markiert. Damit können Sie jede Setup-Option durchlaufen und selbst festlegen. In dieser Lektion erfahren Sie auch, welche Komponenten bei den einzelnen Dialogfeldern zur Standard- oder minimalen Installation gehören.

Nachdem Sie den Setup-Typ gewählt haben, legen Sie fest, wo die SQL Server-Programmdateien (die zur Ausführung von SQL Server 2000 erforderlichen Dateien) und die Datendateien (die Systemdatenbanken und der Standardstandort der Benutzerdatenbanken) installiert werden sollen. Der jeweilige Platzbedarf erscheint im unteren Teil des Dialogfelds. Da bei dem als Beispiel verwendeten Computer auf dem Systemlaufwerk D: nicht genügend Platz zur Verfügung steht (und Laufwerk C: für Windows 98 reserviert ist), soll SQL Server auf einem anderen Laufwerk – hier F: – installiert werden. Die meisten Benutzer können jedoch das Vorgabelaufwerk C: verwenden.



Es empfiehlt sich, den vorgegebenen Pfad `\Programme\Microsoft SQL Server` beizubehalten, selbst wenn man ein anderes Festplattenlaufwerk wählt. Diese Standardeinstellung ist deutlich als SQL Server-Verzeichnis zu erkennen. Dadurch verringert man die Gefahr, versehentlich Dateien in diesen Verzeichnissen zu löschen. Außerdem ist diese Einstellung zum Windows 2000-Logo für SQL Server 2000 kompatibel.

Die benutzerdefinierte Installation

Klicken Sie auf WEITER, um die zu installierenden Komponenten von SQL Server 2000 festzulegen. Von dieser Auswahl hängt der Umfang des benötigten Festplattenplatzes ab. Außerdem bestimmen Sie damit die Funktionalität, die Ihnen nach der Installation von SQL Server zur Verfügung steht.

Komponenten auswählen

Ihr erster Schritt besteht darin, die zu installierenden Komponenten auszuwählen (siehe Abbildung 2.11). Hier sind einige grundlegende Entscheidungen zu treffen. Es besteht auch die Möglichkeit, nur die für Clients benötigten Komponenten zu installieren. Jede der vorhandenen Auswahlmöglichkeiten wird im Folgenden beschrieben.

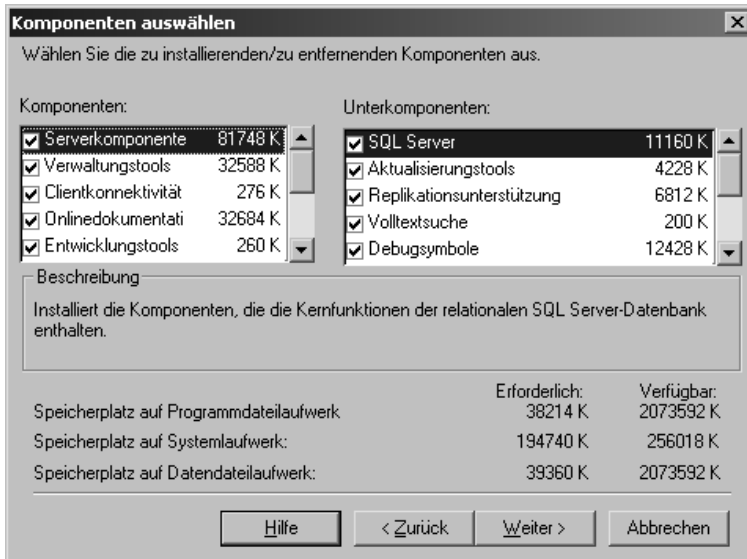


Abbildung 2.11:
Zu installierende
Serverkomponenten
auswählen



Damit Sie alle Beispiele in diesem Buch durcharbeiten können, sollten Sie alle verfügbaren Optionen wählen.



Serverkomponenten

Bei den Serverkomponenten stehen folgende Unterkomponenten zur Auswahl:

- *SQL Server*: Dies sind die Kerndatenbankkomponenten und der Managementunterstützungsdienst (SQL Server-Agent). Diese Option ist per Voreinstellung ausgewählt.
- *Aktualisierungstools*: Dies ist die Unterstützung für die Datenbankkonvertierung; mit dieser Komponente können Sie den Versionskonvertierungsassistenten ausführen, um eine SQL Server 6.5- Installation zu aktualisieren. SQL Server 6.5 muss nicht auf demselben Computer wie SQL Server 2000 installiert sein, um die Daten auf den Server zu migrieren, allerdings muss der Computer unter Windows NT oder Windows 2000 laufen. Diese Option ist per Voreinstellung ausgewählt.
- *Replikationsunterstützung*: Falls Sie von der Datenreplikation Gebrauch machen wollen, sollten Sie diese Option (wie in der Voreinstellung) ausgewählt lassen.
- *Volltextsuche*: Eine interessante Fähigkeit von SQL Server 2000 ist die Möglichkeit, den Microsoft Index Server für die Erstellung von Indizes für textbasierte Daten zu nutzen. Diese Möglichkeit wird in Lektion 13 näher untersucht. Im Gegensatz zu SQL Server 7.0 wird diese Funktion per Voreinstellung installiert; sie ist nur auf Computern mit den Betriebssystemen Windows NT und Windows 2000 verfügbar. Außerdem ist bei SQL Server 2000 im Gegensatz zu SQL Server 7.0 eine vollständige Unterstützung der Volltextsuche in einer Failover-Clusterkonfiguration gegeben.
- *Debugsymbole* sind in der Voreinstellung aktiviert und stellen Debugdateien bereit, falls Sie die Problembehebung von SQL Server mit Microsoft Product Support durchführen müssen. Diese Option sollten Sie aktiviert lassen.
- *Leistungsindikatoren* sollten (wie in der Standardeinstellung) aktiviert bleiben, wenn Sie die Leistung von SQL Server mit dem Dienstprogramm Systemmonitor von Windows 2000 überwachen wollen.

Verwaltungstools

Bei den Verwaltungstools stehen Ihnen folgende Auswahlmöglichkeiten zu Verfügung (siehe Abbildung 2.12):

- *Enterprise Manager* ist eine grafische Benutzeroberfläche zur Unterstützung der Entwicklung und Administration von SQL Server 2000.
- Der *Profiler* ist ein perfektes Werkzeug, um die Aktivitäten des SQL Servers zu überwachen. Mit diesem Tool können Abfragen und ihre Ausführung überwacht sowie Leistungsmessungen vorgenommen werden.
- Im *Query Analyzer* lassen sich Transact-SQL Anweisungen eingeben und ausführen. Sie können sich auch ansehen, wie SQL Server Ihre Abfragen optimiert und ausführt. Erfahrene Transact-SQL-Entwickler greifen gern auf dieses Tool zurück.

- *DTC-Clientunterstützung* erlaubt den Einsatz des DTC (Distributed Transaction Coordinator), um modifizierende Datenabfragen über mehrere Systeme hinweg auszuführen.
- *Konfliktanzeige* installiert den Konfliktlöser für die Mergereplikation.

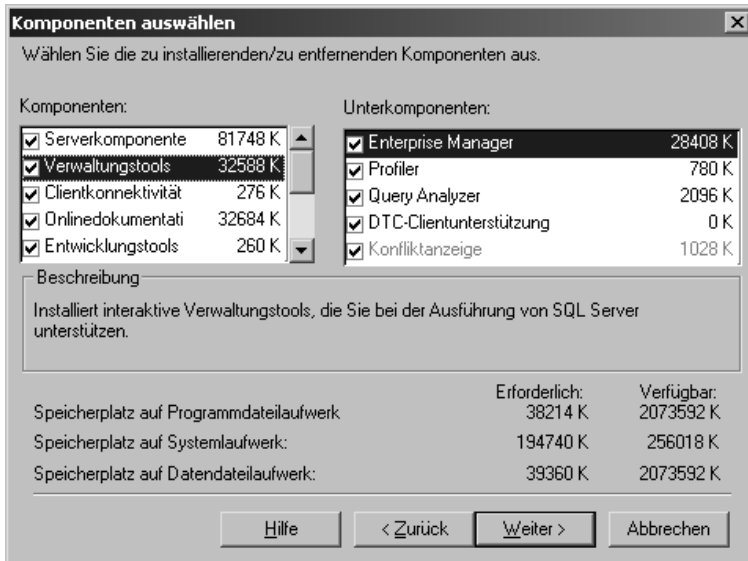


Abbildung 2.12:
Auswahl der zu
installierenden Ver-
waltungstools

Clientkonnektivität

Diese Komponenten ermöglichen die Kommunikation mit SQL Server. Enthalten sind die Bibliotheken für ODBC (Open Database Connectivity), OLEDB (Object Linking and Embedding Database) und DBLIB (DB-Library). Jede dieser Bibliotheken erlaubt Ihnen Programme zu schreiben oder zu benutzen, die eine Verbindung zum SQL Server aufbauen. Ohne diese Komponenten – die unter der Bezeichnung MDAC (Microsoft Data Access Components) zusammengefasst sind – können Sie relativ wenig ausrichten; deshalb ist diese Option per Voreinstellung aktiviert.

Onlinedokumentation

Hier haben Sie die Wahl, ob Sie die Onlinedokumentation auf der lokalen Festplatte installieren, Verknüpfungen zur Dokumentation auf der SQL Server-CD einrichten (um Platz zu sparen) oder die Dokumentation überhaupt nicht installieren wollen. Es empfiehlt sich allerdings, die Dokumentation mit jeder Instanz von SQL Server zu installieren. Nichts ist frustrierender, als mit dem Server zu arbeiten und nicht auf die Handbücher zugreifen zu können, wenn man sie am dringendsten braucht.

Entwicklungstools

Für die Installation der Entwicklungstools haben Sie folgende Auswahlmöglichkeiten (siehe Abbildung 2.13):

- *Header und Bibliotheken:* Das sind Include- und Bibliotheksdateien für C/C++, die Sie für die Entwicklung von SQL Server-Programmen brauchen.
- *MDAC SDKs:* Das sind die Software Development Kits für XML und die Microsoft Data Access Components. Mithilfe dieser SDKs können Sie Programme mit XML und MDAC entwickeln.
- *API zum Sichern/Wiederherstellen:* Zu dieser Auswahl gehören ein Beispielprogramm, die nötigen C/C++-Dateien und die Dokumentation für die Entwicklung von Programmen zur Sicherung und Wiederherstellung von Daten.
- *Debuggerschnittstelle:* Installiert die erforderlichen Komponenten, damit sich gespeicherte Prozeduren mit Microsoft Visual Studio debuggen lassen.

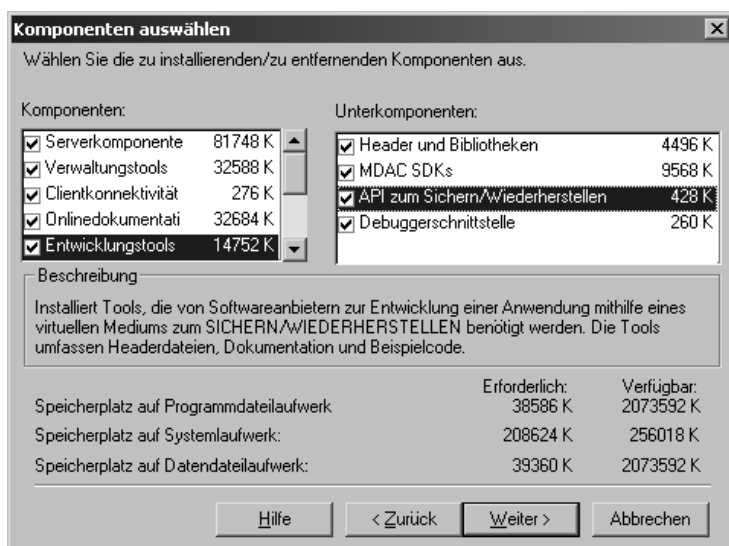


Abbildung 2.13:
Auswahl der zu installierenden Entwicklungstools

Codebeispiele

In der Standardeinstellung werden keine Codebeispiele installiert. Es stehen folgende Unterkomponenten zur Auswahl (siehe Abbildung 2.14):

- *ADO:* Enthält Codebeispiele zu ActiveX Data Objects (ADO).
- *DBLIB:* Enthält Codebeispiele für die DB-Library-API. DB-Library ist die native Datenbank-API früherer SQL Server-Versionen und steht in SQL Server 2000 nur wegen der Abwärtskompatibilität zur Verfügung.

- *Desktop*: Umfasst Codebeispiele für die Microsoft SQL Server Desktop Engine (MS-DE) zum Einrichten von Operationen bei der unbeaufsichtigten Installation.
- *DTS*: Enthält Codebeispiele für Data Transformation Services (DTS). DTS ist eine Möglichkeit, Daten von einer Quelle zur anderen zu verschieben.
- *ESQLC*: Enthält Codebeispiele mit Embedded SQL für die Programmiersprache C.
- *MSDTC*: Enthält Codebeispiele für Microsoft Distributed Transaction Coordinator (MS DTC).
- *ODBC*: Enthält Codebeispiele für die Programmierschnittstelle Open Database Connectivity (ODBC) in SQL Server.
- *ODS*: Enthält Codebeispiele für die Programmierschnittstelle Open Data Services (ODS) für SQL Server.
- *OLE-Automatisierung*: Enthält Codebeispiele für die Unterstützung von OLE-Automatisierung für SQL Server.
- *Replikation*: Enthält Codebeispiele für SQL Server-Replikation.
- *SQLDMO*: Enthält Codebeispiele für die SQL-Distributed-Management-Objects-Programmierschnittstelle.
- *SQLNS*: Enthält Codebeispiele für die SQL-Namespaces-Programmierschnittstelle.
- *Utils*: Enthält Beispieldienstprogramme.
- *XML*: Enthält XML-Beispiele.

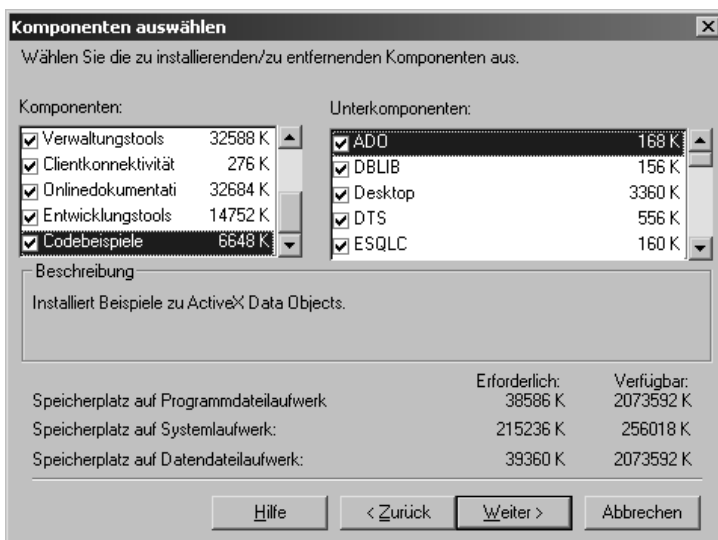


Abbildung 2.14:
Die zu installierenden
Codebeispiele
auswählen

Dienstkonten

Klicken Sie auf WEITER, nachdem Sie alle zu installierenden Komponenten festgelegt haben. Damit gelangen Sie zum Dialogfeld DIENSTKONTEN (siehe Abbildung 2.15). Dieses Dialogfeld erscheint nur auf Windows NT/2000-Computern. Windows 9x unterstützt das Konzept der Dienste nicht, so dass SQL Server immer im Kontext des gerade angemeldeten Benutzers läuft. Weil SQL Server unter Windows NT/2000 als Hintergrunddienst läuft, muss niemand angemeldet sein, um Dienste von SQL Server zu aktivieren. Allerdings müssen sich auch Dienste beim Betriebssystem anmelden um zu funktionieren.

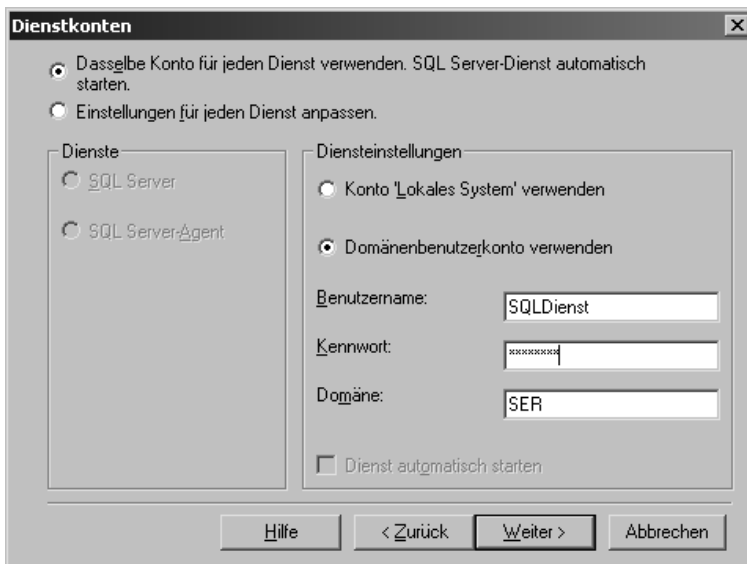


Abbildung 2.15:
Die Dienstkonten
auswählen

Unter Windows NT/2000 stellt das lokale Systemkonto normalerweise das Betriebssystem selbst dar. Die Auswahl des lokalen Systemkontos funktioniert immer. Allerdings hat dieses spezielle Konto keine Zugriffsrechte für das Netzwerk. Wenn Sie also mehrere SQL Server im Netzwerk integrieren oder SQL Server mit anderen BackOffice-Diensten, wie zum Beispiel Exchange Server, betreiben wollen, sollten Sie die SQL Server-Dienste unter einem Benutzerkonto ausführen.



Vielleicht haben Sie gehört, dass das lokale Systemkonto unter Windows 2000 Netzwerkzugriff hat. Das stimmt zwar, gilt aber nur für reine Windows 2000-Netzwerke, die mit anderen Windows 2000-basierten Computern kommunizieren.

Das Installationsprogramm wählt standardmäßig das Konto des Benutzers, der die Installation durchführt. Das ist nur selten die richtige Wahl. Sie sollten ein eigenes Benutzerkonto

für den SQL Server-Dienst einrichten und dann dieses Konto zum Mitglied der lokalen Administratorgruppe für diesen Computer machen. Dies ist zwar nicht zwingend erforderlich, erleichtert Ihnen aber die Arbeit.



Falls Sie Sicherheitsprobleme befürchten, wenn Sie das Dienstkonto zum Mitglied der Gruppe Administratoren machen, konsultieren Sie die Online-dokumentation nach weiteren Informationen, wie sich SQL Server ohne administrative Rechte ausführen lässt.

Weiterhin können Sie hier wählen, ob Sie dasselbe Konto für beide standardmäßigen SQL Server-Dienste verwenden wollen (eine Erklärung dafür folgt später in dieser Lektion). Sie können verschiedene Konten für jeden Dienst verwenden, es ist aber einfacher, die Voreinstellung zu übernehmen. Falls es sich um Ihren eigenen SQL Server handelt, verwenden Sie Ihr Administratorkonto oder erstellen (wie in Abbildung 2.15 gezeigt) ein eigenes Benutzerkonto mit den Dienstprogrammen *Benutzerverwaltung* oder *Benutzerverwaltung für Domänen* in Windows NT 4.0 bzw. *Computerverwaltung* in Windows 2000 (oder mit *Active Directory-Benutzer und -Computer*, wenn es sich um eine Windows 2000-Domäne handelt). Falls Sie sich in einem organisierten Firmennetzwerk befinden, lassen Sie sich ein entsprechendes Benutzerkonto erstellen, das Sie benutzen können.

Stellen Sie sicher, dass Ihr Konto neben der Administratoreigenschaft folgende Eigenschaften aufweist:

- Benutzer kann Kennwort nicht ändern
- Kennwort läuft nie ab
- Alle Anmeldezeiten erlaubt

Abbildung 2.16 zeigt, wie man diese Konfiguration für Windows 2000 über die Computerverwaltung einstellt.

Authentifizierungsmodus

Klicken Sie auf WEITER, nachdem Sie Benutzername und Kennwort für Ihre Dienstkonten eingegeben haben. Daraufhin gelangen Sie zum Dialogfeld AUTHENTIFIZIERUNGSMODUS (siehe Abbildung 2.17). Dieses Dialogfeld erscheint auf allen Betriebssystemplattformen, wobei aber der Windows-Authentifizierungsmodus nicht auf Windows 9x-Systemen verfügbar ist. Bei allen Windows NT/2000-Computern ist der Windows-Authentifizierungsmodus per Vorgabe eingeschaltet.

Der Authentifizierungsmodus bestimmt, ob SQL Server vom Betriebssystem abhängig ist, um Benutzerkonten zu erstellen und zu verwalten (die Standardkonfiguration ist Ja), oder ob eine Kombination von Betriebssystemkonten und Konten, die Sie in SQL Server 2000 anlegen, möglich ist. Diese Optionen untersuchen Sie ausführlich am 5. Tag. Übernehmen Sie hier die Standardeinstellungen.

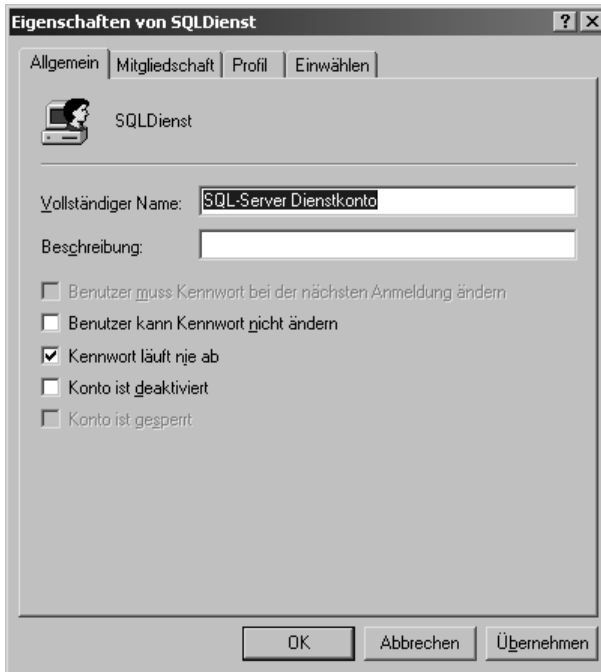


Abbildung 2.16:
Dienstkonten erstellen oder konfigurieren

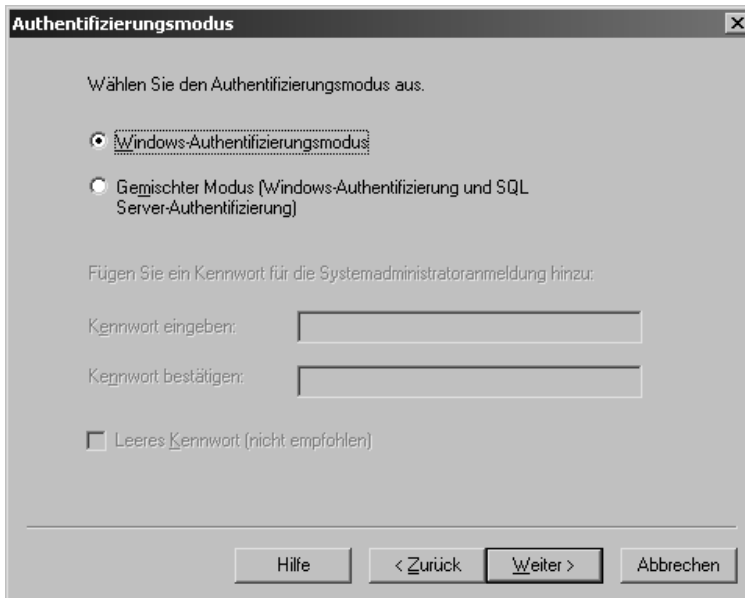


Abbildung 2.17:
Den Authentifizierungsmodus wählen



Wenn Sie in den gemischten Modus wechseln, müssen Sie ein Kennwort für die Anmeldung des Systemadministrators (*sa*) bei SQL Server festlegen. Wenn Sie auf einem leeren Kennwort beharren, müssen Sie explizit ein Kontrollkästchen einschalten. Dieses Dialogfeld wurde hinzugefügt, weil zu viele SQL Server-Systeme mit einem leeren *sa*-Kennwort über das Internet zugänglich gemacht wurden. Wenn Sie Glück haben, verwendet Ihr System die Windows-Authentifizierung. Andernfalls haben Sie zumindest ein Kennwort für Ihre *sa*-Anmeldung.

Sortierungseinstellungen



Wenn Sie nach der Installation von SQL Server 2000 den Zeichensatz, die Sortierreihenfolge oder die Unicode-Sortierreihenfolge ändern möchten, müssen Sie die Datenbank *master* neu erstellen. Im Gegensatz zu vorherigen Versionen von SQL Server müssen Sie jedoch nicht die Daten neu laden und die Indizes neu erstellen, weil SQL Server 2000 datenspezifische (und sogar spaltenspezifische) Sortierungseinstellungen unterstützt.

Klicken Sie auf WEITER, um zum Dialogfeld SORTIERUNGSEINSTELLUNGEN zu gelangen, in welchem Sie eine weitere kritische Entscheidung über Ihre SQL Server-Installation treffen (siehe Abbildung 2.18). Hier müssen Sie wählen, mit welchem Zeichensatz die Nicht-Unicode-Daten zu speichern sind, wie die Daten bei Rückgabe aus SQL Server sortiert werden und welche Art der Unicode-Sortierreihenfolge Sie verwenden wollen (d.h. wie Unicode-Daten sortiert und an den Benutzer zurückgegeben werden).



Was ist Unicode? Bisher hat man jedes auf dem Bildschirm angezeigte Zeichen in einem Byte gespeichert. Damit sind aber nur 256 verschiedene Zeichen möglich. In der Regel sind die ersten 128 Zeichen überall gleich, während die letzten 128 Zeichen je nach dem ausgewählten Zeichensatz (auch als Code-seite bezeichnet) variieren.

Aus globaler Sicht stehen damit nicht annähernd genug Zeichen zur Verfügung, um alle Sprachen dieser Welt darzustellen. Die Unicode-Darstellung verwendet nun für jedes Zeichen zwei Bytes. Damit verdoppelt sich zwar der Speicherbedarf, man kann aber Daten in jeder Sprache auf dem Server ablegen, ohne sie jemals übersetzen zu müssen – sie werden immer in der richtigen Weise gespeichert. Unicode löst damit ein schwieriges Problem, wenn es um internationale Anwendungen unter Einbeziehung verschiedener Sprachen geht – Deutsche können ihre Daten in Deutsch, Chinesen in Chinesisch etc. speichern, alle in derselben Spalte Ihrer Datenbank.

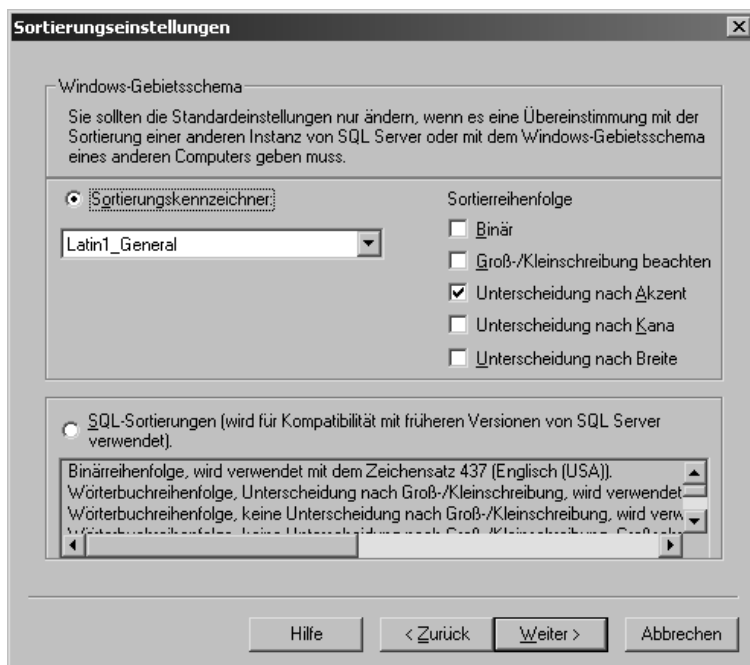


Abbildung 2.18:
Das Dialogfeld zum
Einstellen der Sor-
tierreihenfolge

Als Sortierungskennzeichner ist *Latin1_General* voreingestellt. Diese Sortierung eignet sich für US-Englisch und die meisten westeuropäischen Sprachen. Falls keine zwingenden Gründe vorliegen (z. B. weil Sie in China sind und deshalb den chinesischen Zeichensatz verwenden wollen), sollten Sie diesen Zeichensatz verwenden. Außerdem können Sie festlegen, wie die Daten sortiert werden:

- Bei *Latin1_General* bedeutet die Sortierreihenfolge UNTERSCHIEDUNG NACH AKZENT, dass die Daten in der gleichen Weise wie im Wörterbuch sortiert zurückgegeben werden, wobei aber eine Unterscheidung der Buchstaben mit und ohne Akzent erfolgt, d. h. »é« ist weder mit »E« noch mit »e« gleichwertig.
- Viele Anwendungen verwenden eine binäre Sortierreihenfolge (d. h. die Ergebnisse werden in ihrer ASCII-Reihenfolge zurückgeliefert) und übernehmen dann selbst die Verantwortung für die Sortierung. Schalten Sie für diesen Fall das Kontrollkästchen BINÄR ein.
- Eine andere Möglichkeit besteht darin, den Server anzuweisen, zwischen Groß- und Kleinschreibung zu unterscheiden, so dass »s« und »S« unterschiedlich einsortiert werden. Dann können Sie wählen, ob groß- oder kleingeschriebene Zeichen zuerst einzuordnen sind. Allerdings hat diese Option einen tiefgreifenden Nebeneffekt. Wenn Sie in SQL Server eine Tabelle namens »Verkauf« erzeugen, ist sie nicht identisch mit einer Tabelle namens »VERKAUF«. Sie zwingen damit die Anwender (und sich selbst)

dazu, die Schreibweise sklavisch einzuhalten. Auch die Suche nach einem Namen – beispielsweise »Waymire« – scheitert, sofern Sie nicht den ersten Buchstaben groß und die übrigen Buchstaben klein schreiben. Wenn Sie also »waymire« eingeben, finden Sie die gewünschte Information nicht. Um diese Sortierreihenfolge zu aktivieren, schalten Sie das Kontrollkästchen GROSS-/KLEINSCHREIBUNG BEACHTEN ein.

- Die Option UNTERSCHIEDUNG NACH KANA legt fest, ob SQL Server die beiden japanischen Kana-Zeichensätze (Hiragana und Katakana) als gleich betrachtet. Wenn Sie diese Option einschalten, gelten gleichlautende Zeichen aus beiden Zeichensätzen als unterschiedliche Zeichen. In der Standardeinstellung (Kontrollkästchen ausgeschaltet) werden sie als gleichwertige Zeichen behandelt.
- In manchen Sprachen kann ein und dasselbe Zeichen sowohl in einer Darstellung mit einem Byte als auch in einer Darstellung mit zwei Bytes auftreten. Wenn Sie das Kontrollkästchen UNTERSCHIEDUNG NACH BREITE aktivieren, behandelt SQL Server derartige Zeichen bei Vergleichen als unterschiedlich. In der Standardeinstellung (Kontrollkästchen ausgeschaltet) gelten sie als gleichwertig. Wenn Sie nicht mit Doppelbyte-Sprachen arbeiten, brauchen Sie sich um diese Einstellung keine Gedanken zu machen.

Alternative Sortierungseinstellungen können Sie auch wählen, um Abwärtskompatibilität mit früheren Installationen von SQL Server zu wahren, die einen anderen Zeichensatz und eine abweichende Sortierreihenfolge verwendet haben. In diesem Fall wählen Sie die Option *SQL-Sortierungen* – die Einstellungen ergeben sich dann aus vorherigen bzw. vorhandenen Installationen von SQL Server 7.0 oder früher.

Falls Sie bei der vorherigen Installation die Standardeinstellungen übernommen haben, sollten Sie jetzt die Einstellungen *Wörterbuchreihenfolge*, *Groß-/Kleinschreibung beachten* wählen, was abwärtskompatibel mit dem Zeichensatz 1252 ist. Diese Option ist die Standardeinstellung in SQL Server 6.5 und SQL Server 7.0.



SQL Server stellt Unicode auch für die Windows-9x-Plattformen zur Verfügung, auch wenn Sie unter diesen Betriebssystemen normalerweise keine Unicode-Unterstützung vorfinden werden.

Netzwerkbibliotheken

Klicken Sie auf WEITER, nachdem Sie die Sortierungseinstellungen festgelegt haben. Dadurch gelangen Sie zum Dialogfeld NETZWERKBIBLIOTHEKEN (siehe Abbildung 2.19). Hier müssen Sie die Netzwerkbibliotheken auswählen, die SQL Server unterstützen soll. Es geht darum, welche Netzwerkbibliotheken Ihre Clients verwenden können, um mit Ihrem SQL Server zu kommunizieren. Auf Windows NT/2000-Computern sind per Vorgabe Named Pipes und TCP/IP-Sockets ausgewählt. Auf Windows-9x-Systemen ist nur die Option TCP/IP-Sockets aktiviert.

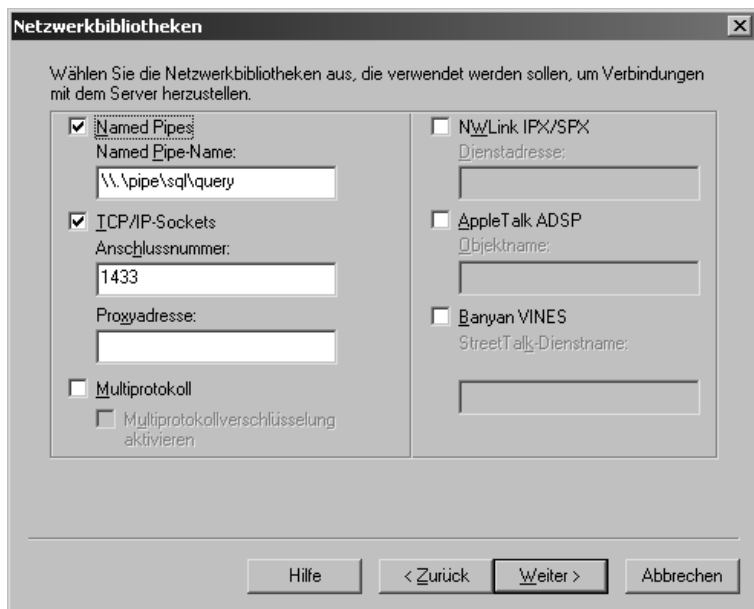


Abbildung 2.19:
Die zu installierenden Netzwerk-
bibliotheken aus-
wählen

Bevor Sie fortfahren, sollten Sie sich einen Überblick über die Netzwerkbibliotheken verschaffen.

Named Pipes

Named Pipes bedeuten einen dateisystembasierten Zugang zu Netzwerkdiensten. Wenn Sie sich mit einem freigegebenen Verzeichnis verbinden, geben Sie einen UNC- (Universal Naming Convention) Pfad zum Dateiserver an: \\FILESERVER\Freigabename. Um eine Verbindung zu einer Named Pipe herzustellen, geben Sie einen Freigabennamen der Form \\COMPUTER\\pipe\\sql\\query an. Die von SQL Server zu verwendende Named Pipe können Sie zwar ändern, sollten das aber nur als erfahrener Administrator tun und wenn Sie sich über die Auswirkungen genau im Klaren sind.

Auf Windows NT-Systemen sind Named Pipes erforderlich, so dass Sie die Auswahl nicht aufheben sollten. Die letzten Versionen von SQL Server haben normalerweise mit dieser Bibliothek gearbeitet – falls Sie also SQL Server-6.x-Clients in Ihrem Netz betreiben, werden diese wahrscheinlich Named Pipes für den Verbindungsaufbau zum Server verwenden. In SQL Server 2000 ersetzen die TCP/IP-Sockets den bisherigen Kommunikationsmechanismus über Named Pipes.

Named Pipes stehen unter Windows 9x (auf der Serverseite) nicht zur Verfügung. Sie können sie für den Verbindungsaufbau zu einem Server verwenden, aber der serverspezifische Teil der Named Pipes ist nicht vorhanden, deshalb ist diese Option bei der Installation einer Desktop Edition von SQL Server in Windows 9x-Umgebungen nicht verfügbar.

TCP/IP-Sockets

Diese Bibliothek verwendet die Netzwerkfähigkeiten der TCP/IP-Sockets für die Serververbindung. Die vorgegebene Anschlussnummer für SQL Server ist 1433. Sie verwenden übrigens jeden Tag Sockets (z.B. um sich über Anschlussnummer 80 mit einem WWW-Server zu verbinden). Alle Betriebssysteme unterstützen dieses Protokoll und es ist die vorgegebene Netzwerkbibliothek bei allen neuen Installationen von SQL Server 2000-Software.

Wenn Sie eine benannte Instanz (später in der heutigen Lektion erläutert) installieren, wird die Socketnummer auf 0 gesetzt (um den Anschluss dynamisch zu bestimmen). Eine benannte Instanz von SQL Server fragt bei jedem Start Windows nach einem verfügbaren Socket ab. Kann SQL Server den beim letzten Start benutzten Socket verwenden, geschieht das auch. Möchten Sie aus bestimmten Gründen die Nummer eines Sockets auf einen festen Wert setzen, können Sie hier die Standardeinstellung überschreiben und die Nummer eines bekannten freien Sockets Ihres Servers eingeben.



Warum verwendet Microsoft nicht einfach 1434 oder irgend eine andere Nummer? Der Grund hierfür liegt darin, dass jede Nummer durch die Internet Assigned Number Authority (IANA) zugewiesen wird und für Microsoft SQL Server nur die beiden Anschlüsse 1433 und 1434 vorgesehen sind. Auch wenn SQL Server über die Nummer 1434 verfügt, verwendet SQL Server den Anschluss 1434 UDP (im Gegensatz zum Anschluss 1434 TCP), um SQL Server-Installationen in einem Netzwerk zu lokalisieren und abzufragen. Da also keine anderen Nummern für SQL Server zugewiesen sind, ist Windows für die Zuteilung zuständig (oder Sie, falls Sie einen Anschluss manuell auswählen).

Multiprotokoll

Multiprotokoll unterstützt jede verfügbare Kommunikationsmethode zwischen Computern mithilfe von Remoteprozeduraufrufen (RPC). Diese Netzwerkbibliothek bietet vor allem die Möglichkeit, den Datenverkehr im Netzwerk zu verschlüsseln. Für alle sicherheitsrelevanten Umgebungen (Banken, Regierung usw.) bietet sich dieses Protokoll an, um die Daten auf dem Weg durch das Netzwerk zu schützen.

Multiprotokoll wird aus den folgenden zwei Gründen nicht mehr standardmäßig installiert:

- In SQL Server 2000 unterstützen alle Netzwerkbibliotheken die Verschlüsselung (Einzelheiten dazu erfahren Sie in der morgigen Lektion).
- Multiprotokoll arbeitet nicht mit benannten Instanzen von SQL Server. Die Multiprotokollverschlüsselung ist nicht für SQL Server verfügbar, die unter Windows 9x laufen.



NWLink IPX/SPX

NWLink IPX/SPX dient dazu, ältere Novell-Umgebungen zu unterstützen. Wählen Sie diese Option nur, falls sie noch IPX/SPX für die Verbindung mit Novell-Clientsoftware verwenden.

AppleTalk ADSP

AppleTalk (wie der Name schon sagt) unterstützt die Netzwerkkommunikation über Macintosh- und Apple-basierte Netzwerke. Sie müssen auch die Dienste für Macintosh-Software für Windows NT/2000 installieren, um diese Netzwerkbibliothek nutzen zu können. Die Bibliothek wird nicht mehr gepflegt und wahrscheinlich zugunsten der TCP/IP-Sockets verdrängt. Weiterhin funktioniert sie nicht mit benannten Instanzen von SQL Server 2000.

Banyan VINES

Banyan VINES wird in Banyan VINES Netzwerkumgebungen eingesetzt. Wenn Sie ein Banyan-StreetTalk-Netzwerk betreiben, wählen Sie diese Option. Auch hier sind zusätzliche Softwarekomponenten für Windows NT/2000 erforderlich, um diese Funktionalität zu aktivieren. Wie für AppleTalk gilt, dass diese Netzwerkbibliothek nicht mehr erweitert und zugunsten von TCP/IP-Sockets aufgegeben wird. Die Bibliothek funktioniert auch nicht mit benannten Instanzen von SQL Server 2000.

Die Installation beenden

Nachdem Sie die benötigten Netzwerkbibliotheken ausgewählt haben, klicken Sie auf WEITER. Damit gelangen Sie zum Dialogfeld gemäß Abbildung 2.20; es zeigt an, dass das Installationsprogramm nun mit dem Kopiervorgang beginnen kann.

Klicken Sie auf WEITER. Daraufhin erscheint das Dialogfeld LIZENZIERUNGSMODUS AUSWÄHLEN (siehe Abbildung 2.21). Wählen Sie hier den Lizenzierungsmodus und die Anzahl der Medien bzw. Prozessoren. Am Beginn dieser Lektion haben Sie Empfehlungen zu den jeweiligen Modi erhalten. Klicken Sie dann auf WEITER, um den Kopiervorgang zu starten.

Das Setupprogramm von SQL Server installiert nun die ausgewählten Komponenten, und zwar zunächst die Komponenten für die Clientkonnektivität (MDAC, Microsoft Data Access Components), dann das Modul für die Volltextsuche, anschließend (bei Bedarf) die Microsoft Management Console (MMC) – eine Benutzeroberfläche, die den SQL Server Enterprise Manager enthält. Daran schließt sich die Installation des (später in dieser Lektion erläuterten) MSDTC (Microsoft Distributed Transaction Coordinator) an, gefolgt vom Modul für die HTML-Hilfe. Dann werden die SQL Server-Programmdateien auf die Festplatte kopiert. SQL Server kopiert nur die Dateien, die für die von Ihnen festgelegten Installationsoptionen erforderlich sind.



Abbildung 2.20:
Setup ist bereit,
die Dateien zu
kopieren.



Abbildung 2.21:
Wählen Sie Ihren Lizenzierungsmodus.

Zum Abschluss der Installation kann das Setupprogramm dazu auffordern, den Computer neu zu starten. Eine derartige Aufforderung erscheint, wenn Sie die MDAC-Komponenten das erste Mal auf Ihrem Computer einrichten. Starten Sie in diesem Fall den Computer neu.



2.4 Weitere optionale SQL Server-Komponenten

Nachdem Sie SQL Server 2000 installiert haben, können Sie drei weitere Dienste installieren: das Microsoft Volltextsuchmodul, die SQL Server 2000 Analysis Services und Microsoft English Query. Die Volltextsuche lässt sich bereits während der normalen Installation von SQL Server installieren; die beiden anderen Dienste können Sie im Anschluss an die Installation des Servers installieren, aber auch vollkommen unabhängig von SQL Server installieren und einsetzen.

Microsoft SQL Server Analysis Services

Microsoft SQL Server Analysis Services (in SQL Server 7.0 als OLAP Services bezeichnet) erlauben den schnellen Zugriff auf Datawarehouse-Daten in SQL Server. Die Server Analysis Services helfen Ihnen, OLAP-Daten (d.h. Daten der analytischen Onlineverarbeitung) aufzubauen, um eine detaillierte Trendanalyse zu erstellen. Die Dienste bieten die Möglichkeit, diese so genannten Datenwürfel (Cubes) zu erstellen und zu überwachen, und sie liefern eine Benutzeroberfläche, über die sich die Datenwürfel erstellen, verwalten und abfragen lassen. Die serverseitigen Komponenten können nur unter Windows NT 4.0 oder Windows 2000 installiert werden. Die Clientkomponenten und die Benutzeroberfläche stehen auch unter Windows 9x zur Verfügung.

Microsoft English Query

Microsoft English Query ermöglicht es einem Datenbankadministrator, ein Datenbankschema zu erstellen, mit dessen Hilfe die Anwender Ihre Datenbankabfragen in Englisch statt in Transact-SQL durchführen können. Besonders hilfreich ist dies sicherlich für Internet-Anwendungen, in denen Sie die Anwender nicht zur Verwendung von Transact-SQL zwingen wollen. Zum Beispiel können Sie anstelle einer komplizierten Transact-SQL Befehlsfolge einfach »Show me the number of books sold for each author this year« eingeben. Microsoft English Query kann auf jeder von SQL Server 2000 unterstützten Plattform installiert werden.

2.5 Nach der Installation: Bestandsaufnahme

Über welche Komponenten verfügen Sie nun nach der Installation von SQL Server 2000? Es gibt einen Satz von Systemdiensten (oder Anwendungen bei Ausführung unter Windows 9x), einen Satz von Werkzeugen und einen Satz von Handbüchern. Außerdem

haben Sie einige neue Dateien auf Ihrem Computer und die Windows-Registrierung wurde verändert. Außerdem hat das Setupprogramm mehrere SQL Server-Standarddatenbanken installiert und die Standardsicherheitskonfiguration eingerichtet.

Die SQL Server-Dienste

Die folgende Übersicht beschreibt alle SQL Server-Dienste, die auf Ihrem System installiert sein können:

- *MSSQLServer*: Dieser Dienst ist der eigentliche Datenbankserver. Wenn Sie SQL Server starten oder anhalten, bedeutet dies normalerweise, den MSSQLServer-Dienst zu starten oder anzuhalten.
- *SQLServerAgent*: Dieser Dienst stellt die Unterstützung für Auftragsplanung, Warnungen, Ereignisbehandlung und Replikation zur Verfügung. Diesen Dienst lernen Sie in Lektion 18 näher kennen.
- *MSDTC (Distributed Transaction Coordinator)* unterstützt verteilte Transaktionen über mehrere Server. Auf verteilte Transaktionen geht Tag 12 näher ein.
- *Microsoft Search* unterstützt die Indizierung von Textfeldern in SQL Server.
- *MSSQLServerOLAPService* unterstützt Microsoft SQL Server Analysis Services (die oben bereits erwähnt wurden).
- *MSSQLServerADHelper* unterstützt die Active Directory-Integration von Microsoft SQL Server. Dieser Dienst wird hier nicht weiter untersucht und benötigt keine weitere administrative Kontrolle (außerdem läuft dieser Dienst nur selten).

Die genannten Dienste lassen sich nach verschiedenen Methoden steuern. Am Einfachsten ist es, mit dem SQL Server-Dienst-Manager oder dem SQL Server Enterprise Manager zu arbeiten. Diese Werkzeuge lernen Sie in der morgigen Lektion kennen. Weiterhin können Sie auf Werkzeuge von Windows NT/2000 zurückgreifen, wie zum Beispiel das Programm Dienste im Ordner Verwaltung von Windows 2000 oder das Dienstprogramm Servermanager.

Viele Administratoren schwören auf die gute alte Befehlszeile. Mit den Befehlen `NET START` und `NET STOP` können Sie jeden Dienst starten bzw. anhalten – natürlich auch die SQL Server-Dienste. Öffnen Sie eine Eingabeaufforderung (über `START / PROGRAMME / ZUBEHÖR / EINGABEAUFFORDERUNG`) und geben Sie `NET START` ein, um eine Liste der ausgeführten Dienste zu erhalten (siehe Abbildung 2.22).

Um den SQL Server Agent-Dienst anzuhalten, geben Sie `net stop sqlserveragent` ein. Den Dienst starten Sie mit `net start sqlserveragent`. Für die anderen Dienste treffen diese Befehle sinngemäß zu.



Installationsordner

Wenn Sie die erste Kopie von SQL Server 2000 installieren, werden zwei Gruppen von Installationsordnern angelegt. Eine Gruppe ist für die installierte Instanz von SQL Server vorgesehen, die andere für Tools, Dienstprogramme und COM-Komponenten, die für alle auf Ihrem Computer installierten Kopien von SQL Server 2000 gemeinsam genutzt werden.



Abbildung 2.22:
Die Liste der auf
einem Computer
laufenden Dienste

Installationsordner für die Standardinstanz

Tabelle 2.1 zeigt die auf Ihrem System erstellten Unterverzeichnisse mit ihrem jeweiligen Inhalt (setzen Sie vor die angegebenen Standorte jeweils `C:\Programme\Microsoft SQL Server\`, wenn Sie den Standardpfad wie weiter oben beschrieben beibehalten haben):

Dateistandort	Installierte Elemente
\MSSQL	Alle anderen Unterstützungsdateien für die Dienste SQL Server und SQL Server Agent sowie die für die Deinstallation benötigten Dateien und die Datei <i>readme.txt</i>
\MSSQL\Backup	Das Vorgabeverzeichnis für SQL Server-Backups. Dieser Ordner ist zunächst leer.
\MSSQL\Binn	Alle SQL Server-Programmdateien und DLLs mit Ausnahme der SQL Server-Werkzeuge
\MSSQL\Data	Die SQL Server-Datendateien (die Systemdatenbanken sowie Ihre eigenen Datenbanken)
\MSSQL\FTData	Die Indizes für den Microsoft-Suchdienst
\MSSQL\Install	Die während der Installation ausgeführten SQL-Skripts sowie die .OUT-Dateien, die Erfolgs- und Fehlermeldungen dieser Skripts enthalten
\MSSQL\Jobs	Der Ordner, in dem Aufträge Daten temporär speichern; enthält anfänglich keine Daten.
\MSSQL\Log	SQL Server-Fehlerprotokolle
\MSSQL\ReplData	Dieser Ordner wird während der Replikation intensiv genutzt, ist aber leer, bis diese verwendet wird.
\MSSQL\Upgrade	Alle Programme und Dateien, die für die Aktualisierung von einer Vorgängerversion von SQL Server auf SQL Server 2000 erforderlich sind
\Programme\OLAP Services	Die Komponenten der Analysis Services
\Programme\Microsoft English Query	Die Komponenten von English Query

Tabelle 2.1: SQL Server-Ordner für die Standardinstanz

Installationsordner für Tools und Dienstprogramme

Tabelle 2.2 zeigt die auf Ihrem System angelegten Ordner und deren Inhalt (setzen Sie bei einer Standardinstallation das Präfix *C:\Programme\Microsoft SQL Server\80* vor die angegebenen Dateistandorte).



Dateistandort	Installierte Tools
\Tools\Binn	Die Programmdateien und Unterstützungs-DLLs für alle SQL Server-Tools
\Tools\Books	Die kompilierten HTML-Dateien für die Onlinedokumentation
\Tools\DevTools	Hier finden Sie die unterstützenden Dateien für die Anwendungsentwicklung (z. B. C-Headerdateien)
\Tools\Html	Alle von der MMC verwendeten HTML-Dateien
\Tools\Scripts	Transact-SQL-Skripts für Tools. Bei Bedarf können Sie ebenfalls auf diese Skripts zurückgreifen.
\Tools\Templates	Vorlagendateien für das Erstellen von Abfragen oder Verfolgungen in SQL Server Profiler und SQL Server Query Analyzer
\COM	Die von allen installierten SQL Server-Instanzen sowie den SQL Server-Tools und Utilities verwendeten COM-Dateien
\COM\Resources\1033	Sprachspezifische DLL-Erweiterungen

Tabelle 2.2: SQL Server-Unterverzeichnisse für Tools

Einträge in der Windows-Registrierung

Ihre Registrierungsdatenbank wurde während der Installation der SQL Server-Dienste an einigen Stellen geändert, um sie bei den Windows NT/2000-Anwendungen *Systemmonitor* und *Ereignisanzeige* zu registrieren und um die erforderlichen Startverzeichnisse für die Dienste bereitzustellen. Hier sollten Sie vor allem den Schlüssel `HKEY_LOCAL_MACHINE\Software\Microsoft\MSSQLServer` kennen. Wenn Sie *regedit.exe* (oder *regedt32.exe* auf Windows NT/2000-Computern, obwohl *regedit.exe* auch funktioniert) aufrufen, können Sie bis zu diesem Schlüssel navigieren. Abbildung 2.23 zeigt ein Beispiel für die betreffenden Registrierungsschlüssel.

Normalerweise müssen Sie hier nichts ändern; es schadet aber auch nichts, wenn man für den Notfall weiß, wo diese Einträge zu finden sind. Weitere Registrierungseinträge mit den persönlichen Voreinstellungen der jeweiligen Benutzer der Clienttools finden Sie unter dem Eintrag `HKEY_CURRENT_USER\Software\Microsoft\MSSQLServer`.

Die einzelnen Dienste sind unter `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\<Dienstname>` gespeichert.

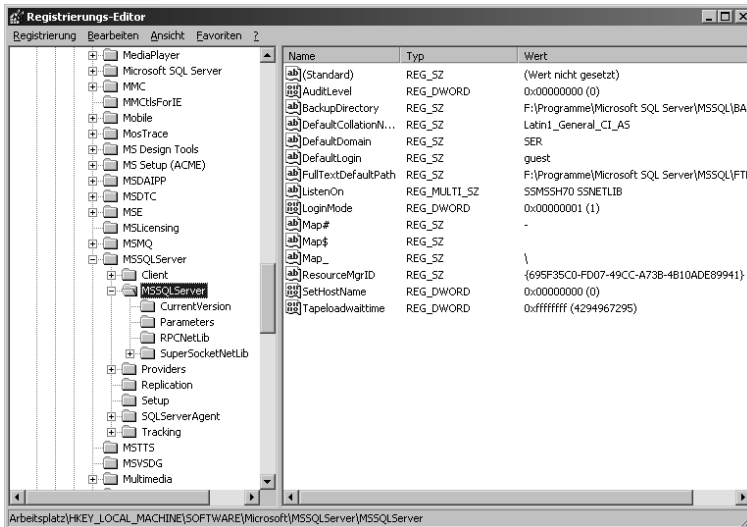


Abbildung 2.23:
Die Registrierungsschlüssel für SQL Server

Die Standarddatenbanken

Bei der Installation von SQL Server 2000 wurden auch mehrere Datenbanken (*master*, *model*, *tempdb*, *msdb*, *pubs*, und *Northwind*) angelegt, auf die die nächsten Abschnitte eingehen. Natürlich können Sie später Ihre eigenen Datenbanken hinzufügen; auf jeden Fall aber sind die genannten Dateien auf Ihrem System vorhanden. Einige davon (*master*, *model*, *tempdb* und *msdb*) sind Systemdatenbanken – diese Datenbanken können Sie nicht löschen, da sie für die Funktion von SQL Server erforderlich sind. Dagegen sind die beiden Datenbanken *pubs* und *Northwind* lediglich Beispieldatenbanken, die Ihnen zum Erlernen von SQL Server zur Verfügung stehen. Auf dem Produktionssystem von SQL Server können Sie diese beiden Datenbanken ohne weiteres löschen.

Die Datenbank master

Wie bereits erwähnt, ist *master* die Schlüsseldatenbank für den Betrieb von SQL Server. Sie enthält für jede auf Ihrem System installierte Datenbank einen Verweis auf deren Hauptdatei, sowie weitere für den gesamten Server wesentliche Informationen. Das sind unter anderem systemweit gültige Fehlermeldungen, Anmeldeinformationen, Systemprozeduren und Informationen über weitere angeschlossene Server. Die Datenbank *master* kann im Fall einer Katastrophe nur mit Hilfe besonderer Techniken wiederhergestellt werden, die Tag 8 behandelt. Nach der Installation nimmt diese Datei etwa 12 MB in Anspruch.



Die Datenbank model

Diese Datenbank kann man sich am besten als Vorlagendatenbank vorstellen. Wenn Sie eine neue Datenbank anlegen, erstellt SQL Server eine Kopie der Datenbank *model* und wendet dann die von Ihnen gewünschten Änderungen der Datenbankgröße und anderer Parameter auf diese Kopie an. Deshalb taucht jedes Objekt, das in dieser Datenbank enthalten ist, in jeder neu erstellten Datenbank auf. Beispielsweise können Sie sofort nach der Installation von SQL Server eine Tabelle oder einen Benutzernamen in die Datenbank *model* einfügen. Von nun an werden die Tabelle und der Benutzername in jede neu angelegte Datenbank übernommen.

Die Datenbank *model* belegt nach der Installation 768 KB. Da diese Datenbank beim Anlegen einer neuen Datenbank kopiert wird, kann keine Datenbank kleiner als die Datenbank *model* sein.

Die Datenbank tempdb

Die Datenbank *tempdb* wird während der Ausführung von Sortier-, Verknüpfungs- und anderen Operationen verwendet, die zeitlich begrenzten Speicherplatz benötigen. Nach der Installation beträgt die Dateigröße etwa 2 MB, wobei diese Datenbank wie alle SQL Server 2000-Datenbanken gemäß Voreinstellung bei Bedarf wachsen kann. Die Datenbank *tempdb* wird bei jedem Start von SQL Server (des SQL Server-Dienstes) neu initialisiert.

Die Datenbank msdb

Die Datenbank *msdb* unterstützt den SQL Server-Agent-Dienst und speichert Informationen über Aufträge, Warnungen, Ereignisse und die Replikation. Außerdem nimmt diese Datenbank ein Protokoll aller Sicherungs- und Wiederherstellungsvorgänge auf. Die Datenbank *msdb* ist anfänglich etwa 12 MB groß.

Die Datenbank pubs

Diese Datenbank ist als Lernhilfe gedacht. Sie enthält eine Beispieldatenbank für Verleger mit Autoren, Büchern und Verkaufszahlen. Die meisten Beispiele in der Onlinedokumentation (und in diesem Buch) beziehen sich auf diese Datenbank. Die meisten Datenbankfunktionen werden anhand ihrer Umsetzung in dieser Datenbank verdeutlicht. Die Datenbank *pubs* weist nach der Installation eine Größe von knapp 2 MB auf.

Die Datenbank Northwind

Diese Datenbank ist eine weitere Lernhilfe als Alternative zur Datenbank *pubs*. Die Datenbank *Northwind* gehört schon seit längerer Zeit als Beispieldatenbank zu Microsoft Access. Da immer mehr Access-Anwender auf SQL Server umsteigen, wurde diese Datenbank übernommen, um den Access-Benutzern die Funktionen von SQL Server mit einer vertrauten Datenbank näher zu bringen. Die Standardgröße der Datenbank *Northwind* beträgt etwa 3,5 MB.

Vorgegebene Benutzernamen in SQL Server

Sicherlich wollen Sie sich nach der Installation von SQL Server als Erstes beim Server anmelden. Da Sie die Standardeinstellungen inklusive Windows-Authentifizierungsmodus übernommen haben, wählen Sie einfach die Option Windows NT-Authentifizierung in jedem Dialogfeld, das bei einer Verbindung zu SQL Server erscheint. Wenn Sie sich für den gemischten Modus entschieden haben, können Sie entweder wie eben beschrieben vorgehen oder die Standardanmeldung für SQL Server verwenden: *sa* (klein geschrieben, wenn Ihr Server bei der Sortierreihenfolge zwischen Groß-/Kleinschreibung unterscheidet). Die Buchstaben *sa* stehen als Abkürzung für *Systemadministrator*.

sa

Der Benutzername *sa* ist Mitglied der festen Serverrolle *sysadmin*. Als Mitglied dieser Rolle hat *sa* alle Zugriffsrechte in SQL Server. Das Konto *sa* existiert immer und lässt sich nicht löschen. Im Windows-Authentifizierungsmodus ist es jedoch nicht verfügbar. Im gemischten Modus können Sie sich für eine Anmeldung unter diesem Konto entscheiden. Wahrscheinlich aber wählen Sie die nächste Option, die Anmeldung über Ihre Mitgliedschaft in Ihrer lokalen Administratorengruppe.

Die lokale Administratorengruppe in Windows NT/2000

Wenn Sie sich an einem Windows NT/2000-Computer befinden und Mitglied der lokalen Administratorengruppe sind, müssen Sie nicht die SQL Server-Authentifizierung verwenden. Während der Installation fügt SQL Server 2000 der Rolle *sysadmin* die lokale Windows NT/2000-Gruppe Administratoren hinzu, genau wie es bei *sa* der Fall ist. Im Ergebnis sind alle lokalen Administratoren in der Standardeinstellung auch SQL Server-Administratoren. Auf Windows 9x-Computern gibt es keine Windows-Authentifizierung, so dass Sie sich hier als *sa* anmelden müssen. Wenn Sie mit Windows-Authentifizierung arbeiten, ist in SQL Server keine Kennwortkontrolle notwendig; SQL Server verwendet einfach die Zertifikate der Windows NT/2000-Anmeldung.



2.6 Benannte und Standardinstanzen

SQL Server 2000 hat die Fähigkeit eingeführt, mehrere unabhängige Kopien des Datenbankservers auf einem einzigen Windows-basierten Computer auszuführen. Microsoft unterstützt bis zu 16 Kopien von SQL Server 2000, die auf ein und demselben Computer installiert sind. Bei der Unterscheidung der Instanzen tauchen die Begriffe *benannt* und *Standard* auf, die in den folgenden Abschnitten erläutert werden.

Standardinstanzen

Für eine Verbindung zur Standardinstanz von SQL Server 2000 brauchen Sie nur den Servernamen anzugeben, wenn Sie die Verbindung über eine Anwendung herstellen. In allen vorherigen Versionen von SQL Server haben sich die Benutzer auf diese Weise verbunden: Sie haben den Namen des Computers, auf dem SQL Server installiert war, als Name des SQL Servers, mit dem sie kommunizieren wollten, angegeben. Heißt zum Beispiel der eigene Computer *SER*, gibt man als Name für den SQL Server *SER* an, um die Verbindung zu SQL Server herzustellen.

Die Standardinstanz wird in den Verzeichnissen installiert, die die Tabellen 2.1 und 2.2 weiter vorn in dieser Lektion genannt haben, und die Registrierungsschlüssel befinden sich (zum größten Teil) am gleichen Standort wie bei SQL Server 6.5 und SQL Server 7.0 – also im Schlüssel `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer`. Die Programmdateien werden im Standardpfad `\MSSQL` installiert. Tabelle 2.3 gibt die Dienstenamen für die eigentlichen SQL Server-Dienste an.

Dienststart	Dienstname
SQL Server	MSSQLServer
SQL Server-Agent	SQLServerAgent

Tabelle 2.3: Dienste von SQL Server für die Standardinstanz

Zu einem bestimmten Zeitpunkt kann jeweils nur eine Kopie der Standardinstanz von SQL Server laufen und zwar unabhängig von der Version. Ist zum Beispiel SQL Server 6.5 installiert, lässt sich nicht gleichzeitig SQL Server 6.5 und die Standardinstanz von SQL Server 2000 starten. Allerdings kann man benannte Instanzen von SQL Server 2000 auch gleichzeitig mit einer Standardinstanz ausführen.

Benannte Instanzen

Eine benannte Instanz von SQL Server 2000 ist eine Instanz, die Sie während der Installation benennen. Wenn Sie eine Verbindung zu einer benannten Instanz herstellen möchten, geben Sie sowohl den Dienstnamen als auch den Instanznamen an, den Sie während der Installation festgelegt haben. Später in dieser Lektion sollen Sie eine benannte Instanz *Handel* installieren. Wenn Sie sich dann zu dieser SQL Server-Instanz auf dem Computer *SER* verbinden wollen, geben Sie *SER\Handel* als Servername an. Damit weisen Sie SQL Server an, mit dem Computer namens *SER* zu kommunizieren und nach einer Instanz namens *Handel* auf diesem Computer zu suchen.

Benannte Instanzen sind mit dem Windows 2000-Logo kompatibel. Sie verwenden `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\Instanzname` als Stammschlüssel in der Registrierung. Die Installation erfolgt in den Programmdateipfad `\MSSQL$Instanzname` statt einfach nur `\MSSQL`. Diese Instanzen sind sowohl von der Standardinstanz als auch jeder anderen benannten Instanz vollständig getrennt, weil keine zwei Instanzen auf demselben Computer denselben Instanznamen haben können. Die Dienste werden ebenfalls mit eindeutigen Namen angelegt, wie es Tabelle 2.4 zeigt.

Dienststart	Dienstname
SQL Server	MSSQL\$Instanzname
SQL Server-Agent	SQLAgent\$Instanzname

Tabelle 2.4: Dienstnamen bei benannten Instanzen

Gemeinsame Komponenten

Einige Komponenten werden zwischen den Installationen von SQL Server 2000 gemeinsam genutzt; genau genommen gehören sie weder zur Standardinstanz noch zu irgend einer der benannten Instanzen, die Sie eventuell installiert haben. Zu den gemeinsamen Komponenten zählen die SQL Server-Tools sowie Systemkomponenten wie MDAC 2.6.



Wenn Sie eine benannte Instanz von SQL Server 2000 installieren wollen und SQL Server 7.0 installiert haben (dabei handelt es sich per definitionem um eine Standardinstanz), werden Ihre 7.0-Tools durch die SQL Server 2000-Tools ersetzt und die SQL Server 7.0-Tools gelöscht. Die SQL Server 2000-Werkzeugsammlung arbeitet korrekt, wenn sie auf eine SQL Server 7.0-Installation verweist.

Eine benannte Instanz installieren

Installieren Sie nun eine benannte Instanz. Wählen Sie die Instanz namens *Handel* (das Kontrollkästchen für die Standardoption sollte schattiert dargestellt sein, wenn Sie sich für die Installation einer neuen Instanz entscheiden, weil auf Ihrem Computer bereits eine Instanz existiert). Führen Sie eine Standardinstallation durch, ändern Sie aber den Sicherheitsdialog. Übernehmen Sie nicht den vorgegebenen Windows-Authentifizierungsmodus, sondern wählen Sie den gemischten Modus, und legen Sie Ihr Kennwort fest (für das gesamte übrige Buch gilt als Kennwort *kennwort*). Nehmen Sie das gleiche Dienstkonto wie früher. Gelegentlich greifen Sie im Verlauf dieses Buches auf die benannte Instanz zurück.

2.7 Aktualisierung von einer vorherigen SQL Server-Version

Bei einer Aktualisierung von SQL Server 7.0 auf SQL Server 2000 wählen Sie zu Beginn der Installation nicht die Option *Eine neue Instanz von SQL Server erstellen*, sondern die Option *Eine vorhandene Instanz von SQL Server aktualisieren, entfernen oder Komponenten hinzufügen* (vgl. Abbildung 2.4); markieren Sie die Standardinstanz; und wählen Sie dann die Aktualisierung Ihrer vorhandenen Installation. Folgen Sie den Aufforderungen; sie sind den bereits gezeigten für eine neue Installation ähnlich. Die Aktualisierung findet an Ort und Stelle statt; wenn sie abgeschlossen ist, haben Sie Ihre 7.0-Installation auf SQL Server 2000 aktualisiert. Die Tools werden durch die SQL Server 2000-Tools ersetzt.

Aktualisierung von SQL Server 6.5

Die Aktualisierung von SQL Server 6.5 läuft etwas anders ab. Zuerst installieren Sie eine Standardinstanz von SQL Server 2000, wie Sie es bereits durchgeführt haben. Nachdem das erledigt ist, finden Sie einen neuen Menüeintrag im START-Menü: MICROSOFT SQL SERVER – VERSIONSUMSTELLUNG. Dieses Menü enthält Optionen, um zu SQL Server 6.5 zurückzuschalten, den SQL Server-Aktualisierungs-Assistenten auszuführen oder SQL Server 6.x zu deinstallieren.

Versionsumstellung

Die auszuführende Standardinstanz können Sie zwischen SQL Server 6.5 und SQL Server 2000 umschalten. Nachdem Sie eine Standardinstanz von SQL Server 2000 installiert haben, wird Ihre SQL Server 6.5-Installation deaktiviert. Zu dieser Installation können Sie

zurückschalten, indem Sie sie aus diesem Menü auswählen. Daraufhin wird Ihre Standardinstanz von SQL Server 2000 deaktiviert und SQL Server 6.5 neu gestartet. Die Tools von SQL Server 6.5 erscheinen im START-Menü (genau genommen sind sie nur als versteckt markiert; wenn Sie zur Version 6.5 wechseln, wird die Markierung aufgehoben).

Die Version können Sie erneut wechseln, indem Sie wieder in dieses Menü gehen. Jetzt heißt der Programmeintrag allerdings Microsoft SQL Server 2000. Wenn Sie ihn wählen, deaktivieren Sie die SQL Server 6.5-Installation und kehren zur Standardinstanz von SQL Server 2000 zurück.

Im Allgemeinen empfiehlt sich dieses Hin- und Herschalten zwischen den Versionen nicht. Diese Option ist eigentlich nur als Unterstützung für den als Nächstes erläuterten Aktualisierungs-Assistenten gedacht.

Der Aktualisierungs-Assistent

Der SQL Server-Aktualisierungs-Assistent aktualisiert eine SQL Server 6.5-Installation auf SQL Server 2000. Nachdem Sie Ihre Standardinstanz installiert haben, sollten Sie diesen Programmbefehl aufrufen. Der Assistent migriert alle Ihre Datenbanken sowie die System-einstellungen nach SQL Server 2000. Detaillierte Informationen zur Aktualisierung finden Sie zum entsprechenden Thema in der Onlinedokumentation.

SQL Server 6.5 deinstallieren

Nachdem Sie Ihr SQL Server 6.5-System auf SQL Server 2000 aktualisiert haben, müssen Sie die alte Version manuell entfernen. Wählen Sie dazu den betreffenden Menüeintrag. Keinesfalls dürfen Sie SQL Server 6.5 ausführen, um diese Version zu deinstallieren; dadurch stören Sie die Standardinstanz von SQL Server 2000. Das spezielle Deinstallationsprogramm entfernt SQL Server 6.5 sorgfältig und bewahrt dabei die Standardinstanz von SQL Server 2000.

2.8 Problembehebung bei der Installation

Die Installation von SQL Server 2000 ist ein relativ unkomplizierter Vorgang. Falls dennoch etwas schief geht, sollten Sie die Ursache möglichst genau ermitteln, damit Sie Abhilfe schaffen können. Fehlschläge bei der Installation von SQL Server 2000 sind allerdings extrem selten, so dass Sie die folgenden Informationen vielleicht nie brauchen werden.



sqlstp.log

Diese Protokolldatei befindet sich im Windows-Verzeichnis (*d:\winnt* beim Beispielsystem). Es handelt sich um die Ausgabedatei des Installationsprogramms. Alle aufgetretenen Probleme werden in dieser Datei aufgezeichnet. Sollten Sie auf ein Problem stoßen, suchen Sie in dieser Datei nach einer Meldung zum betreffenden Fehler. Normalerweise ist die Fehlermeldung gegen Ende der Datei zu finden.

*.OUT-Dateien

Im Verlauf der Installation werden auch Skripts aus dem Verzeichnis *\mssql7\install* ausgeführt. Falls Sie in der Datei *sqlstp.log* Hinweise auf Fehler in einem dieser Scripts erhalten haben, können Sie die Ausführungsergebnisse in den Dateien mit der Erweiterung *.OUT* in diesem Verzeichnis finden. Die Meldungen sind normalerweise ohne weiteres verständlich – z.B. »xxx konnte nicht erstellt werden«.

SQL Server-Fehlerprotokoll

Wenn SQL Server (der SQL Server-Dienst) gestartet wurde und ein Fehler aufgetreten ist, finden Sie wahrscheinlich hilfreiche Informationen im Fehlerprotokoll des SQL Servers. Diese Dateien stehen im Verzeichnis *\MSSQL\Log* und sind ihrem Alter nach nummeriert. Das aktuelle Protokoll trägt den Namen *Errorlog* (ohne Dateierweiterung). Das nächst ältere heißt *Errorlog.1*, dann folgt *Errorlog.2* und so weiter.

Windows NT/2000-Ereignisprotokoll

Unter Windows NT/2000 können Sie Ihre Fehlerprotokolle auch mit der Windows-Ereignisanzeige überprüfen. Unter Windows 2000 wählen Sie dazu *START / PROGRAMME / VERWALTUNG / EREIGNISANZEIGE*. In der Anwendung Ereignisanzeige erscheinen drei verschiedene Protokolle: das Systemprotokoll, das Sicherheitsprotokoll und das Anwendungsprotokoll. Klicken Sie in der Ereignisanzeige auf Anwendungsprotokoll, um dieses Protokoll im Detailbereich anzuzeigen. Suchen Sie nach Fehlermeldungen mit einem roten Stoppzeichen. Untersuchen Sie derartige Meldungen genauer, um weitere Hinweise zur Problembehebung zu erhalten.

2.9 SQL Server entfernen

Falls Sie SQL Server aus irgendeinem Grund wieder aus Ihrem System entfernen müssen, ist das relativ einfach zu erledigen. Wählen Sie in der Systemsteuerung das Programm *Software*, markieren Sie *Programme ändern oder entfernen* und Microsoft SQL Server 2000 erscheint in der Liste der Programme – genau wie jede andere Anwendung auf Ihrem Computer. Sie können auch die Verknüpfung in der SQL Server-2000-Programmgruppe verwenden, um SQL Server 2000 zu entfernen. Damit werden alle Dateien und Registrierungseinträge von SQL Server gelöscht, die installierten gemeinsam verwendeten Komponenten (z.B. die MDAC-Komponenten) werden dabei nicht entfernt. Außerdem bleiben die Tools erhalten, wenn es sich nicht um die letzte zu entfernende Instanz von SQL Server handelt. Wenn Sie die letzte Instanz von SQL Server entfernen, werden auch die Tools gelöscht.

Bei einem Computer, auf dem bereits SQL Server 6.5 installiert ist, müssen Sie bei der Aktualisierung auf SQL Server 2000 einen wichtigen Punkt beachten. Wenn Sie sicher sind, dass Sie SQL Server 6.5 nicht mehr einsetzen wollen, entfernen Sie die Vorgängerversion von SQL Server *nicht* über das zu SQL Server 6.5 gehörende Setupprogramm. Das könnte nämlich Ihre SQL Server-2000-Installation zerstören. Microsoft hat ein spezielles Deinstallationsprogramm für diesen Zweck geschrieben und im Menü START / PROGRAMME / MICROSOFT SQL SERVER – VERSIONSUMSTELLUNG eine Verknüpfung zu diesem Programm unter dem Menüpunkt SQL SERVER 6.X DEINSTALLIEREN vorgesehen.

2.10 Zusammenfassung

SQL Server 2000 lässt sich relativ einfach installieren. Die Schwierigkeit besteht nur darin, im Laufe der Installation die richtigen Entscheidungen zu treffen, da Sie mit diesen Einstellungen für die Dauer Ihrer Arbeit mit dem Produkt leben müssen. Das Installationsprogramm startet automatisch, wenn Sie die CD einlegen, und es hilft Ihnen gegebenenfalls bei der Installation einiger Voraussetzungen. Auf allen Systemen, auf denen Sie SQL Server 2000 installieren wollen, muss Internet Explorer 5.0 vorhanden sein.

Entscheiden Sie sich, welche Version von SQL Server 2000 Sie installieren wollen. Unter Windows NT-Server können Sie die Standard Edition wählen. Es bietet sich auch die Enterprise Edition an, insbesondere wenn Sie alle Merkmale von SQL Server nutzen wollen. Vielleicht brauchen Sie auch die Developer oder Evaluation Versionen des Produkts, die (mit Ausnahme der Lizenzierung) mit der Enterprise Edition identisch sind. Schließlich läuft die Personal Edition unter Windows 98, Windows ME, Windows NT Workstation (4.0 oder höher) und Windows 2000 Professional.



Nachdem Sie die Voraussetzungen erfüllt haben, können Sie mit der Installation von SQL Server beginnen. Wenn Sie die Voreinstellungen übernehmen möchten (nachdem Sie sie jetzt kennen), wählen Sie einfach die Standardinstallation. Der Rest läuft dann überwiegend automatisch ab. Denken Sie daran, ein Dienstkonto zu erstellen und zu verwenden, falls Sie SQL Server unter Windows NT/2000 installieren und Ihren SQL Server mit anderen SQL Servern oder mit anderen Microsoft BackOffice-Komponenten verbinden.

2.11 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Welcher Installationstyp hat den geringsten Speicherplatzbedarf auf der Festplatte?*
- A** Der Installationstyp Minimal.
- F** *Wie kann ich nur die für einen Client benötigten Komponenten installieren?*
- A** Führen Sie eine benutzerdefinierte Installation durch und deaktivieren Sie die Serverkomponenten.
- F** *Welche Netzwerkbibliotheken werden standardmäßig unter Windows NT/2000 installiert?*
- A** Named Pipes und TCP/IP-Sockets.
- F** *Welche Datei sollte ich mir ansehen, bevor ich mit der Installation beginne?*
- A** Die Datei readme.txt – im Autostartmenü finden Sie eine Verknüpfung zu dieser Datei.

Kontrollfragen

1. Welche Sortierreihenfolge wählen Sie, wenn die Daten in der ASCII-Reihenfolge zurückgegeben werden sollen?
2. Worauf hat die Unicode-Sortierreihenfolge Einfluss?
3. Welche Komponente installiert die OLE DB- und ODBC-Treiber?
4. Welches Benutzerkonto verwendet man, um die SQL Server-Dienste zu steuern?
5. Wie lässt sich SQL Server starten, wenn er nicht läuft?
6. Welches Dateisystem ist für den Betrieb von SQL Server unter Windows NT/2000 zu bevorzugen?
7. Wie kann man den Microsoft-Suchdienst installieren?

Übungen

1. Installieren Sie SQL Server 2000, falls Sie es nicht bereits durchgeführt haben. Wählen Sie eine benutzerdefinierte Installation und installieren Sie alle Komponenten, die Ihre Version zulässt.
2. Sehen Sie sich die Fehlerprotokolldatei und die Datei *sqlstp.log* in Ihrem Windows-Verzeichnis an. Überprüfen Sie auch die *.out*-Dateien im Verzeichnis *mssql\install* auf eventuelle Probleme.
3. Installieren Sie die Microsoft SQL Server Analysis Services. Auch wenn dieses Produkt hier nicht näher untersucht wird, könnte es für Sie hilfreich sein.
4. Überprüfen Sie die Registrierungseinträge für den SQL Server, damit Sie im Notfall wissen, wo sie zu finden sind.



SQL Server 2000 Tools und Utilities

Gestern haben Sie gelernt, wie Sie SQL Server 2000 installieren. Wie Sie sicherlich festgestellt haben, ist die Ausführung des Setupprogramms eine fast triviale Angelegenheit. Es gibt nur wenige Dinge, die Sie genauestens von vornherein planen müssen. Das betrifft vor allem die Festlegung der Sortierreihenfolge, die sich im Nachhinein nur schwer ändern lässt. Gegenüber SQL Server 7.0 ist dieser Punkt unter SQL Server 2000 allerdings nicht mehr so kritisch, weil man die Sortiereinstellungen beginnend bei der Ebene der Datenbank bis hinunter zur Spaltenebene jeder Tabelle separat festlegen kann. Andere Einstellungen lassen sich mit den zum Produkt gehörenden Dienstprogrammen ebenfalls leichter verändern.

Die heutige Lektion befasst sich mit den Tools und Dienstprogrammen, die Sie – entsprechend dem Hinweis in Lektion 2 – sicherlich komplett installiert haben. Dabei lernen Sie nicht nur die Dienstprogramme kennen, die im START-Menü erscheinen, sondern auch die Dienstprogramme für die Befehlszeile sowie andere Dienstprogramme, für die einfach keine Verknüpfung im START-Menü vorgesehen ist. Auch wenn diese Programme etwas versteckt sind, sollten Sie ihren Wert nicht unterschätzen.

3.1 Die Programmgruppe Microsoft SQL Server 2000

Um die verfügbaren Werkzeuge zu untersuchen, bietet sich als Ausgangspunkt die Programmgruppe Microsoft SQL Server im Start-Menü an. Diese erreichen Sie über START / PROGRAMME / MICROSOFT SQL SERVER 2000 (siehe Abbildung 3.1). Beachten Sie, dass nur eine Kopie der Werkzeuge installiert ist, unabhängig davon, wie viele Instanzen von SQL Server selbst vorhanden sind. Weiterhin ist festzustellen, dass bei einer Aktualisierung von SQL Server 7.0 auf SQL Server 2000 alle Werkzeuge der Version 7.0 durch die neuen Werkzeuge ersetzt wurden. Es ist garantiert, dass die Werkzeuge von SQL Server 2000 mit SQL Server 7.0 zusammenarbeiten.

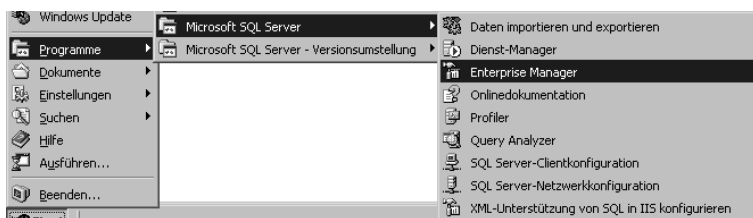


Abbildung 3.1:
Die Programmgruppe
Microsoft SQL
Server

Wenn Sie Datenbankanwendungen entwickeln oder Administrationsaufgaben erfüllen, arbeiten Sie überwiegend mit den Werkzeugen in dieser Programmgruppe. Es ist wichtig, dass Sie mit allen Werkzeugen vertraut sind, da Sie mit ihnen im Verlauf dieses Buches

ständig zu tun haben. Die heutige Lektion behandelt diese Werkzeuge in der Reihenfolge, in der Sie sie wahrscheinlich benutzen wollen, und nicht in der Reihenfolge, wie sie im START-Menü erscheinen.

Die Datei readme.txt

Es empfiehlt sich, mit der Datei *Readme.txt* zu beginnen. Diese Datei enthält wichtige Informationen, die Sie vor der Installation lesen sollten, aber auch Informationen, die erst nach der Drucklegung der Handbücher verfügbar wurden. Gehen Sie die Informationen in dieser Datei durch, um sicher zu sein, dass Sie vor Ihrem Umstieg auf SQL Server 2000 keine weiteren Änderungen vornehmen müssen. Aber wahrscheinlich haben Sie das schon gestern vor der Installation getan, oder nicht?

Onlinedokumentation

Abgesehen von diesem Buch bildet die SQL Server-Onlinedokumentation Ihre Hauptquelle für Informationen. Wenn Sie Fragen zu SQL Server haben, finden Sie die Antwort hier. Die Onlinedokumentation wird anstelle gedruckter Handbücher mitgeliefert; die Bücher können Sie aber bei Microsoft bestellen. Die Dokumentation besteht aus kompilierten HTML-Seiten, die Sie sich mit dem Microsoft Internet Explorer 5 oder höher ansehen können. Abbildung 3.2 zeigt die Onlinedokumentation von SQL Server.



Abbildung 3.2:
Die SQL Server
2000-Onlinedoku-
mentation

Die Onlinedokumentation bietet Ihnen alles, was Sie von Handbüchern erwarten können – Sie können sich Informationen ansehen, nach ihnen suchen und sie ausdrucken. Um ein bestimmtes Thema zu drucken, klicken Sie einfach mit der rechten Maustaste auf den gewünschten Eintrag und wählen **DRUCKEN**. Sie können dann wählen, ob Sie nur die aktuelle Seite oder auch alle untergeordneten Seiten ausdrucken wollen.



Falls Sie nicht ganze Wälder auf dem Gewissen haben wollen, sollten Sie beim Drucken nicht in den Hauptverzeichniseinträgen beginnen.

Die linke Seite der Onlinedokumentation ist standardmäßig in vier Registerkarten gegliedert.

- **INHALT:** Diese Registerkarte zeigt jedes SQL Server-Buch wie ein normales Handbuch. Wenn Sie wissen, in welchem Buch eine bestimmte Information enthalten ist, können Sie die Anzeige des Buches erweitern (indem Sie auf das Pluszeichen links vom Titel klicken), um sich den Inhalt anzusehen. Die meisten Bücher haben mehrere untergeordnete Kapitel, durch die Sie sich bewegen müssen, um die gewünschte Information zu finden.
- **INDEX:** Diese Registerkarte enthält eine alphabetisch sortierte Liste aller Schlüsselwörter aus den Büchern. Geben Sie einfach das Wort ein, nach dem Sie suchen wollen, und die Anzeige springt automatisch auf den Indexeintrag, der Ihrer Eingabe am ehesten entspricht. Durch einen Doppelklick auf das markierte Schlüsselwort können Sie sich das gewünschte Thema anzeigen lassen (falls es zu diesem Schlüsselwort mehrere Einträge gibt, können Sie das Thema aus einer Liste auswählen).
- **SUCHEN:** Auf dieser Registerkarte werden Sie normalerweise landen. Geben Sie einfach ein Konzept, eine Anweisung oder eine Option ein, über die Sie mehr erfahren wollen, und klicken Sie auf **THEMEN AUFLISTEN**. Das Suchresultat erscheint dann in der Auswahlliste. Hier können Sie sich wieder durch einen Doppelklick auf den Eintrag die Informationen in der rechten Fensterhälfte anzeigen lassen.
- **FAVORITEN:** Genau wie im Internet Explorer können Sie eine Liste Ihrer Lieblingsseiten in der Dokumentation erstellen und diese dann einfach hier auswählen, statt die Einträge erneut suchen zu müssen. Für das jeweils angezeigte Thema können Sie auf die Schaltfläche **HINZUFÜGEN** klicken (siehe Abbildung 3.3). Wenn Sie später zu diesem Thema zurückkehren wollen, doppelklicken Sie einfach auf den entsprechenden Eintrag in der Themenliste der Registerkarte **FAVORITEN**.

Die SQL Server-Onlinedokumentation ist in mehrere »Bücher« unterteilt, die Sie auf der Registerkarte **INHALT** finden. Die folgenden Abschnitte stellen die einzelnen Bücher kurz vor.

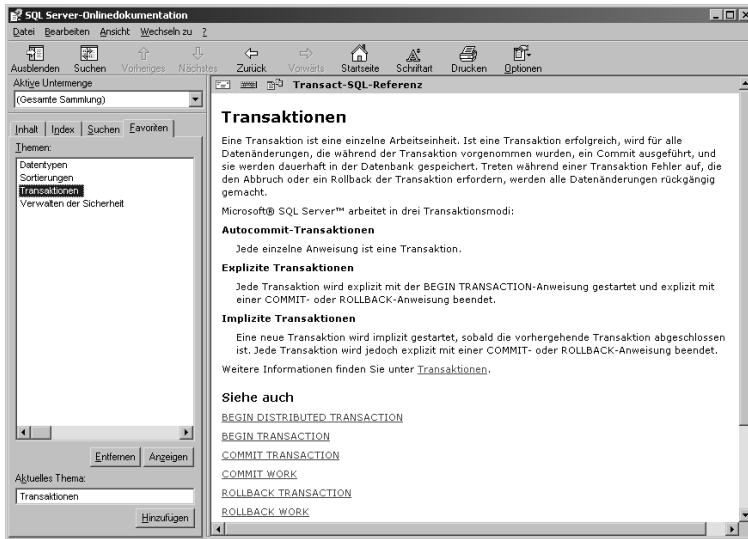


Abbildung 3.3:
Die Registerkarte
Favoriten der
Online-
dokumentation

Erste Schritte

Dieses Buch bietet Verweise auf die anderen Bücher, erläutert die verwendeten typographischen Konventionen (als Hilfe zum Verständnis der Syntaxdarstellung und anderer Textauszeichnungen) und enthält eine Liste der vorhandenen Werkzeuge, Assistenten und anderer Funktionen von SQL Server. Es enthält weiterhin Hinweise, wie Sie Kontakt zur SQL Server-Supportabteilung aufnehmen können. Zusätzlich finden Sie hier Informationen für Personen mit Behinderungen und zur Abwärtskompatibilität. Jeder Benutzer sollte dieses Buch lesen – es ist recht kurz, aber für die spätere Nutzung der Onlinedokumentation sehr hilfreich.

Neuigkeiten

Dieses Buch liefert einen kurzen Überblick über die Funktionen des Produkts, die für Benutzer der Vorgängerversionen von SQL Server 2000 neu sind. Falls Sie bisher mit einer früheren Version von SQL Server gearbeitet haben, sollten Sie sich dieses Buch ansehen, um sich über die Änderungen zu informieren.

Installieren von SQL Server

Dieses Buch enthält eine detaillierte Beschreibung der Installation von SQL Server 2000. Da Sie die Lektion des zweiten Tages »SQL Server 2000 installieren« bereits gelesen haben, brauchen Sie sich wahrscheinlich nicht mehr mit diesem Buch zu befassen.



SQL Server-Architektur

Dieses Buch gibt Ihnen einen ausführlichen Überblick über den Aufbau von SQL Server 2000. Es liefert viele Einzelheiten zur Funktionsweise von SQL Server und zur Zusammenarbeit der einzelnen Komponenten. Die meisten der hier enthaltenen Informationen werden Ihnen noch im Verlauf dieses Buches begegnen. Zu empfehlen ist auf jeden Fall der Abschnitt »Details zur Implementierung«. Hier sind alle Grenzen von SQL Server dokumentiert – beispielsweise wie groß eine Datenbank sein kann, wie viele Tabellen in einer Datenbank enthalten sein können und viele weitere Detailangaben.

Verwalten von SQL Server

Dieses Buch beschreibt die Administration von SQL Server. Da Ihnen »SQL Server 2000 in 21 Tagen« alle notwendigen Grundlagen (und einige erweiterte Optionen) zur Verwaltung erläutert, brauchen Sie wahrscheinlich gar nicht auf dieses Buch der Onlinedokumentation zurückzugreifen.

Erstellen und Verwalten von Datenbanken

Dieses Buch beschreibt, wie Sie Datenbanken und alle darin enthaltenen Objekte erstellen. Auch hier gilt, dass Sie sich mit »SQL Server 2000 in 21 Tagen« alle notwendigen Kenntnisse aneignen können und dieses Buch der Onlinedokumentation eigentlich nicht brauchen.

Erstellen und Verwenden von Datawarehouses

In diesem Buch erfahren Sie, wie Sie Datawarehouses – eine spezielle Form relationaler Datenbanken – erstellen. Die Themen rund um Datawarehouses bilden einen eigenen Komplex, der sich von der allgemeinen Konstruktion von Datenbanken abhebt. Wenn Sie Datawarehouses aufbauen oder unterstützen müssen, sollten Sie sich mit diesem Buch der Onlinedokumentation beschäftigen. Datawarehouses setzen grundlegende Kenntnisse über den Aufbau »normaler« Datenbanken voraus. Das Wissen, dass Sie sich in diesem 21-Tage-Kurs aneignen, bietet Ihnen eine solide Basis für das Erstellen von Datawarehouses.

Zugreifen auf und Ändern von relationalen Daten

Dieses Buch erklärt, wie Sie Daten in SQL Server abfragen und ändern. Die Erläuterungen sind so umfangreich, dass Sie weitgehend ohne Onlinedokumentation auskommen werden.

XML- und Internetunterstützung

Dieses Buch dokumentiert die Unterstützung von SQL Server 2000 für XML. Auch wenn Lektion 21 ausführlich auf dieses Thema eingeht, sollten Sie sich mit diesem Buch der Onlinedokumentation befassen, um weiterführende Einzelheiten zu erfahren.

Transact-SQL-Referenz

In dieser Referenz finden Sie die genaue Syntax, nach der die einzelnen Objekte in SQL Server aufzurufen sind. Wenn Ihnen die Syntax für einen Transact-SQL-Befehl, eine gespeicherte Prozedur oder Funktion entfallen ist, können Sie hier nachschlagen. Natürlich behandelt »SQL Server 2000 in 21 Tagen« alle wichtigen Befehle, dennoch ist dieses Buch der Onlinedokumentation immer verfügbar, um schnell ein bestimmtes Element aufzusuchen.

Optimieren der Datenbankleistung

Dieses Buch bietet Informationen zur Leistungsoptimierung von SQL Server. Auf einige grundlegende Optimierungs- und Überwachungsaufgaben geht Tag 20 näher ein und das ganze Buch hindurch finden Sie Hinweise zur Leistungsoptimierung. Da aber die Leistungsoptimierung ein sehr weites Feld ist, sollten Sie sich diesen Abschnitt der Onlinedokumentation näher ansehen (am besten, nachdem Sie dieses Buch durchgearbeitet haben).

Replikation

Dieses Buch beschreibt die Replikationstechnologie von SQL Server 2000. Wahrscheinlich kommen Sie aber auch ohne dieses Buch der Onlinedokumentation aus, da die Tage 16 und 17 ausführlich auf die Replikation eingehen.

Data Transformation Services

Data Transformation Services (DTS) sind das Thema dieses Buches der Onlinedokumentation. Tag 19 behandelt zwar einen großen Teil der DTS, da es sich aber um ein recht anspruchsvolles Werkzeug handelt, sollten Sie sich bei komplexeren Transformationen mit diesem Buch der Onlinedokumentation beschäftigen.



Analysis Services

Dieses Buch der Onlinedokumentation erläutert die Analysis Services, in SQL Server 7.0 als OLAP Services bezeichnet. Da dieser 21-Tage-Kurs aus Platzgründen nicht weiter auf die Analysis Services eingeht, sollten Sie bei Bedarf auf diesen Teil der Onlinedokumentation zurückgreifen.

English Query

Hier erfahren Sie mehr zu den Abfragemechanismen auf Basis der englischen Sprache. Dieser Teil der Onlinedokumentation ist nur in englisch verfügbar. Der 21-Tage-Kurs geht aus Platzgründen nicht weiter auf English Query ein.

Meta Data Services

Die in SQL Server 7.0 als Microsoft Repository bezeichneten Meta Data Services sind Thema dieses Buches der Onlinedokumentation. Auf die Meta Data Services geht der 21-Tage-Kurs ebenfalls nicht ein.

Building SQL Server Applications

Dieser Teil der Onlinedokumentation ist sehr umfangreich. Er enthält die Programmierreferenzen für verschiedene Client- und Servertechnologien. Wie der Titel des Buches vermuten lässt, liegt er nur in der englischen Version vor. Das Buch »SQL Server 2000 in 21 Tagen« bringt aber fast alles, was Sie über SQL Server-Befehle wissen müssen. Der 21-Tage-Kurs soll jedoch keine Programmierreferenz für Clientanwendungen sein, so dass Sie sich bei Bedarf mit diesem Buch der Onlinedokumentation auseinander setzen müssen. Das gilt insbesondere, wenn Sie Clientanwendungen schreiben, die ODBC, OLE DB oder ADO (ActiveX Data Objects) verwenden. SQL-DMO (SQL Distributed Management Objects) stellt die Programmierschnittstelle für die Verwaltung von SQL Server dar und wird ebenfalls in diesem Teil der Onlinedokumentation erläutert.

Übersicht über SQL Server-Tools

Dieses Buch der Onlinedokumentation behandelt praktisch alles, womit Sie sich heute beschäftigen – es erläutert jedes SQL Server-Tool. Im übrigen Teil dieses Buches setzen Sie diese Tools ein, so dass Sie am Ende des 21-Tage-Kurses problemlos damit umgehen können.

Problembhebung

Dieses Kapitel beschreibt die am häufigsten auftretenden Probleme und ihre Lösungsmöglichkeiten. Nach Abschluss des 21-Tage-Kurses sollten sie die meisten Probleme verstehen und lösen können, ohne auf die Onlinedokumentation zurückgreifen zu müssen. Aber für den Fall der Fälle ist dieses Kapitel auf Ihrem Computer verfügbar.

Glossar

Das Glossar ist ein Verzeichnis mit Erklärungen der in der Onlinedokumentation zu SQL Server verwendeten Begriffe.

Vorgehensweise

Dieses Buch der Onlinedokumentation erläutert, wie bestimmte Aufgaben unter Verwendung von Transact-SQL, der SQL Server-Tools und anderer Programmierschnittstellen durchgeführt werden können. Normalerweise gehen Sie nicht direkt zu diesem Buch, sondern gelangen über Hyperlinks aus anderen Büchern zu den entsprechenden Abschnitten.

Dienst-Manager

Mit dem SQL Server-Dienst-Manager können Sie die Dienste von SQL Server (oder jeden SQL Server im Netzwerk) steuern; Abbildung 3.4 zeigt dieses Tool.



Abbildung 3.4:
Das Dienstprogramm SQL Server Dienst-Manager

Das Feld SERVER enthält den Namen des Servers, den Sie überwachen. Im Feld DIENSTE erscheint der untersuchte Dienst, darunter sehen Sie eine grafische Darstellung für den Zustand dieses Dienstes. Ein laufender Dienst ist durch den grünen Pfeil gekennzeichnet;

in Abbildung 3.4 ist das der Dienst SQL Server. Wie Sie am zweiten Tag bereits gelernt haben, ist der Dienst-Manager die Hauptkomponente, die SQL Server ausführt.



In SQL Server 7.0 sind die eigentlichen Dienstnamen wie *MSSQLServer* und *SQLServerAgent* in diesem Dienstprogramm zu sehen. Mit der Einführung mehrerer Instanzen in SQL Server 2000 hat sich das SQL Server-Team dafür entschieden, in diesem Dialogfeld einfach »SQL Server« und »SQL Server-Agent« anzuzeigen. Wenn Sie also eine Standardinstanz von SQL Server steuern, lautet der Dienstname *MSSQLServer* für SQL Server und *SQLServerAgent* für den Agent-Dienst. Bei benannten Instanzen dagegen steuern Sie beim Starten oder Anhalten von SQL Server tatsächlich den Dienst *MSSQL\$Instanzname*, wie Sie es gestern gelernt haben.

Sie können einen Dienst auch beenden oder anhalten. Einige Dienste unterstützen keinen Anhalten-Modus; Dienste, die sich anhalten lassen, zeigen dabei ein unterschiedliches Verhalten. Beispielsweise funktioniert der SQL Server-Dienst im angehaltenen Zustand weiter, erlaubt aber keine neuen Verbindungen zum Server.



Das Dienstprogramm *Dienst-Manager* enthält auch eine Option zum automatischen Starten jedes Dienstes beim Systemstart. Diese Option ist vor allem praktisch bei Windows 9x-Computern, weil dieses Betriebssystem keine Dienste kennt, die man für den automatischen Start konfigurieren kann. Schalten Sie für jeden Dienst, den Sie automatisch starten wollen, das Kontrollkästchen **DIENT BEI BETRIEBSSYSTEMSTART AUTOMATISCH STARTEN** ein.



Abbildung 3.5:
Benannte Instanzen im Dienstprogramm SQL Server-
Dienst-Manager

Das Listenfeld **DIENTE** zeigt, welche Dienste sich mit diesem Tool steuern lassen. Hier finden Sie die Dienste SQL Server, SQL Server-Agent, MSDTC und (falls installiert) den Microsoft-Suchdienst (Microsoft Search). Über das Listenfeld **SERVER** können Sie einen

anderen Server auswählen. Im Listenfeld sind auch benannte Instanzen aufgeführt; zum Beispiel zeigt Abbildung 3.5 den Server `SER\HANDEL` als separaten Eintrag (die benannte Instanz Handel haben Sie in Lektion 2 installiert).



Die Dienste MSDTC und Microsoft Search erscheinen im Dienst-Manager für jede benannte Instanz sowie für die Standardinstanz. Diese Anzeige dient nur dem Komfort; von jedem Dienst läuft trotzdem nur eine Kopie auf Ihrem Computer – alle Instanzen von SQL Server auf diesem Computer nutzen diese Dienste gemeinsam.

Es kann sein, dass die Serverliste nicht vollständig ist; Sie können aber auch den Namen des zu untersuchenden Servers (oder `Servername\Instanzname`) eintippen. Nachdem Sie einen Servernamen eingegeben haben, klicken Sie auf die Dropdown-Liste DIENSTE. Der Dienst-Manager versucht dann, die Verbindung zum Remoteserver herzustellen und die installierten Dienste aufzuzählen.

Alle Funktionen dieser Anwendung finden Sie auch im SQL Server Enterprise Manager. Der Dienst-Manager befindet sich jedoch in der Taskleiste und ist deshalb leichter zu erreichen.



Beachten Sie, dass zu diesem Zeitpunkt die Sicherheit überprüft wird. Sie müssen ein Windows NT/2000-Administrator oder ein Serveroperator für den Server sein, mit dem Sie eine Verbindung herstellen wollen, um die Dienste verwalten zu können (falls der ausgewählte Rechner ein Windows NT/2000-Computer ist). Die SQL Server-Berechtigungen sind unabhängig von Ihren Windows 2000-Berechtigungen – als SQL Server-Administrator müssen Sie also nicht unbedingt auch die SQL Server-Dienste steuern können.



Abbildung 3.6:
Die zusätzlichen Merkmale des Dienstprogramms
SQL Server-Dienst-Manager

Der SQL Server-Dienst-Manager hat noch eine weitere, etwas versteckte Funktion. Klicken Sie auf das Symbol in der oberen linken Ecke der Anwendung. Daraufhin erscheint das normale Anwendungs-menü mit einigen Erweiterungen (siehe Abbildung 3.6). Beachten Sie die beiden zusätzlich verfügbaren Menüoptionen: VERBINDEN und OPTIONEN.

Verbinden

Falls Sie den Namen eines Remotecomputers in der Serverauswahlliste eingegeben oder ausgewählt haben, können Sie die Verbindung über das Menü aktivieren, statt auf die Dienstaushlliste zu klicken. In beiden Fällen passiert dasselbe – es wird der Versuch unternommen, eine Verbindung zum entfernten Rechner aufzubauen, um den Zustand der SQL Server-Dienste auf diesem Rechner zu ermitteln.

Optionen

Wenn Sie Optionen wählen, erscheint das Dialogfeld gemäß Abbildung 3.7. In diesem Dialogfeld legen Sie die Optionen für den Dienst-Manager fest, beispielsweise den Standarddienst.



Abbildung 3.7:
Das Dialogfeld SQL Server-Dienst-Manager

Den Standarddienst festlegen

Der Standarddienst bestimmt, welcher SQL Server-Dienst standardmäßig überwacht wird (und in der Taskleiste zu sehen ist), wenn Sie die Anwendung *Dienst-Manager* starten (normalerweise beim Anmelden). Wenn Sie mehr an einer Überwachung des SQL Server-Agent-Dienstes anstelle des SQL Server-Dienstes interessiert sind, können Sie hier die Standardeinstellung ändern.

Abrufintervall setzen

Das Abrufintervall legt fest, wie häufig der Dienst-Manager den Zustand der überwachten Dienste überprüft. Das Vorgabeintervall beträgt 5 Sekunden. Das bedeutet, dass der Dienst-Manager den Server alle 5 Sekunden nach dem Zustand aller SQL Server-bezogenen Dienste fragt.

Dienstkontrollaktion überprüfen

Die Option DIENSTKONTROLLAKTION ÜBERPRÜFEN ist per Vorgabe eingeschaltet. Wenn Sie auf ein entsprechendes Symbol klicken, wird die gewünschte Aktion erst nach einer Rückfrage (siehe Abbildung 3.8) ausgeführt. Um diese Überprüfung auszuschalten, und die Dienste einfach per Doppelklick auf das entsprechende Symbol anhalten und starten zu können, deaktivieren Sie das Kontrollkästchen in diesem Dialogfeld.

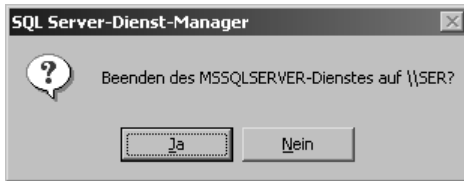


Abbildung 3.8:
Das Bestätigungsdiaologfeld des SQL Server-Dienst-Managers

SQL Server-Clientkonfiguration

Das SQL Server-Clientkonfigurationsprogramm ist nicht kompliziert, hat aber entscheidenden Einfluss auf die Konnektivität des Clientcomputers zu Ihrem SQL Server. Wenn Sie dieses Programm starten, erscheint ein Dialogfeld entsprechend Abbildung 3.9.

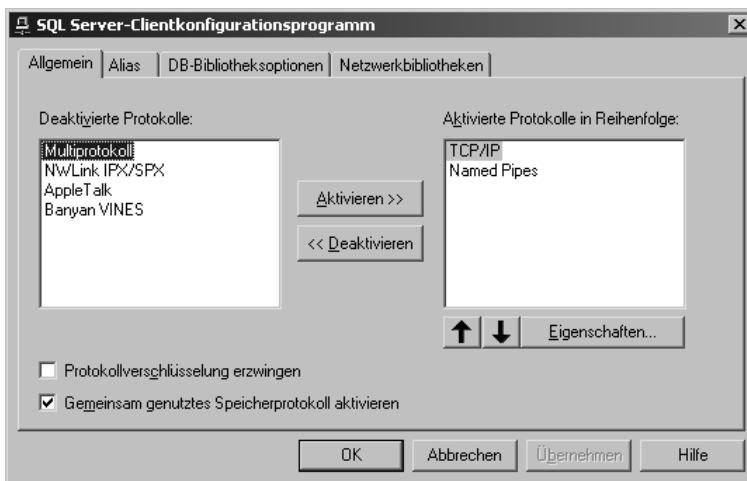


Abbildung 3.9:
Das Dienstprogramm für die Clientkonfiguration von SQL Server

Dieses Dialogfeld zeigt, welche Netzwerkbibliothek standardmäßig für einen Verbindungsaufbau zu einem SQL Server von der SQL Server-Clientsoftware verwendet wird. Stellen Sie sich das als eine Art Sprache vor. Wenn Ihr Server nur Englisch und Französisch ver-

steht und Sie ihn in Deutsch ansprechen, versteht Sie der Server nicht und kann Ihnen nicht antworten. SQL Server 2000 ändert die Standardnetzwerkbibliothek unabhängig vom Betriebssystem immer auf TCP/IP-Sockets.

In SQL Server 7.0 hängt die Wahl der Clientnetzwerkbibliothek vom Betriebssystem ab: Auf Windows NT-Computern gilt als Standard die Named Pipes-Netzwerkbibliothek, während auf Windows 9x-Computern standardmäßig die TCP/IP-Sockets-Netzwerkbibliothek eingesetzt wird.

Die gestrige Lektion hat diese Netzworbbibliotheken im Rahmen der Installation behandelt. Wählen Sie einfach diejenige aus, die in Ihrer Systemumgebung am sinnvollsten ist. Die »universelle« Netzworbbibliothek ist TCP/IP; wenn Sie also diese Option übernehmen, sollten Sie sich mit jedem SQL Server unabhängig vom Betriebssystem, auf dem er läuft, verbinden können. Auf Windows NT/2000 ist Named Pipes erforderlich, so dass diese Auswahl ebenfalls funktioniert. Allerdings geht Microsoft von Named Pipes ab, so dass die Wahl von TCP/IP-Sockerts zukunftsicher ist.



Ein Hinweis zur Bequemlichkeit: Wenn Sie eine Verbindung unter dem Servernamen »(local)« oder ».« (ein einzelner Punkt) herstellen, umgehen Sie die vorgegebene Netzworbbibliothek und verbinden sich zur lokal installierten Standardinstanz. Bei einer benannten Instanz geben Sie *.Instanzname* an. Wenn Sie diese Option wählen, verwenden Sie (in der Voreinstellung) die Shared Memory Netzworbbibliothek. Bei den installierten Clienttools von SQL Server sollte die Umgehung der Netzworbbibliotheken und die Verwendung des Shared Memory automatisch vonstatten gehen, selbst wenn Sie die Verbindung nach der Syntax *Servername* oder *Servername\Instanzname* herstellen. Haben Sie jedoch bereits vorher mit SQL Server gearbeitet, müssen Sie wahrscheinlich den einzelnen Punkt verwenden, um die lokale Verbindung herzustellen.

Falls Sie eine Netzworbbibliothek auswählen, die der Server nicht unterstützt, endet der Verbindungsversuch mit der Fehlermeldung: »Die Verbindung mit Server kann nicht hergestellt werden *Servername*«. Das kann bedeuten, dass auf dem Remotecomputer kein SQL Server läuft oder dass der dortige Server die von Ihnen für die Verbindung ausgewählte Netzworbbibliothek nicht unterstützt.

Zwei weitere Optionen im unteren Teil des Dialogfelds sind PROTOKOLLVERSCHLÜSSELUNG ERZWINGEN und GEMEINSAM GENUTZTES SPEICHERPROTOKOLL AKTIVIEREN. Das gemeinsam genutzte Speicherprotokoll dient dazu, eine Verbindung zu einer lokalen Installation von SQL Server herzustellen. Diese Option können Sie zwar deaktivieren, allerdings ist die Leistung besser, wenn die lokale Verbindung über das gemeinsam genutzte Speicherprotokoll hergestellt wird.

Die Option PROTOKOLLVERSCHLÜSSELUNG ERZWINGEN bewirkt, dass jede Verbindung von Ihrem Clientcomputer eine Verschlüsselung mit dem SQL Server, zu dem Sie die Verbindung herstellen wollen, aushandelt. Da SQL Server-Versionen vor SQL Server 2000 nicht über diese Verschlüsselung verfügen, können Sie weder Verbindungen zu früheren Versionen herstellen, noch Anwendungen auf Ihrem Computer ausführen. Selbst bei SQL Server 2000-Computern muss der Server über ein Zertifikat für den installierten Server verfügen. Auf Zertifikate geht diese Lektion später näher ein.

Die Registerkarte Alias

Sie können die Standardeinstellungen auch ändern und die Verbindung zu einem benannten Server mit einem bestimmten Netzwerkprotokoll herstellen. Angenommen, Sie haben einen Server namens *Verkauf* und die auf diesem Server installierte Kopie von SQL Server unterstützt nur die Netzwerkbibliothek NWLink IPX/SPX, während die übrigen Server mit TCP/IP-Sockets arbeiten. Dafür können Sie in diesem Dialogfeld einen entsprechenden Eintrag aufnehmen. Klicken Sie auf HINZUFÜGEN; daraufhin erscheint ein Dialogfeld wie in Abbildung 3.10. Wählen Sie die Option NWLink IPX/SPX (oder welche Netzwerkbibliothek Sie benutzen wollen), geben Sie den Aliasnamen des Servers (in diesem Falle *Verkauf*) ein und nehmen Sie alle weiteren erforderlichen Einstellungen vor, die für jede Netzwerkbibliothek unterschiedlich sind. Klicken Sie auf OK, um den Server mit dem ausgewählten Protokoll hinzuzufügen. Wenn Sie dann *Verkauf* als Servername in einer SQL Server-Anwendung oder einem Dienstprogramm von diesem Computer eingeben, stellen Sie die Verbindung über die Netzwerkbibliothek NWLink IPX/SPX her.

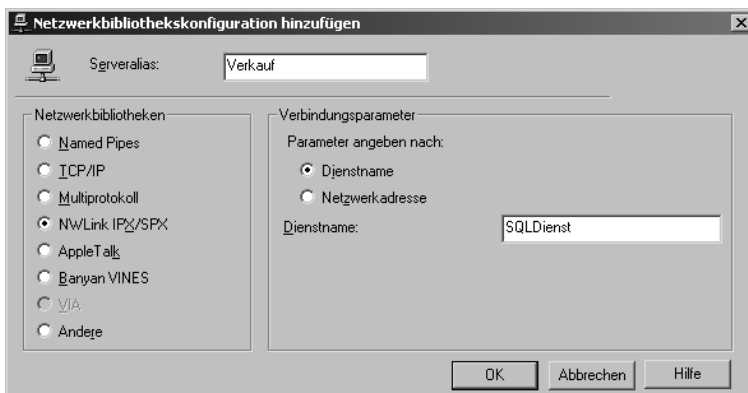


Abbildung 3.10:
Das Dialogfeld
Netzwerkbiblio-
thekskonfiguration
hinzufügen



Gestern haben Sie erfahren, dass MDAC 2.6 installiert sein muss (die Version, die zu SQL Server 2000 gehört), um eine Verbindung zu einer benannten Instanz von SQL Server 2000 herzustellen. Allerdings können Sie das mit einem Alias umgehen. Um beispielsweise eine Verbindung zur benannten

Instanz *HANDEL* auf dem Computer *SER* herzustellen, lautet der Name der benannten Pipe `\\ser\pipe\MSSQL$Handel\sqlquery`. Geben Sie also einen Alias für irgendeinen Servernamen (wie etwa *BenInstanz*) ein, wobei Sie Named Pipes als Netzwerkbibliothek ausgewählt haben, und legen Sie dann dafür diesen Pipe-Namen fest. Wenn Sie ab jetzt *BenInstanz* als Servername angeben, gelangen Sie zur benannten Pipe der benannten Instanz und stellen die Verbindung zu Ihren benannten Instanzen her. Diese Lösung ist recht praktisch, wenn aus bestimmten Gründen keine Aktualisierung der Clientsoftware auf Ihren Computern möglich ist und Sie dennoch Verbindungen zu benannten Instanzen herstellen wollen.

Die Registerkarte DB-Bibliotheksoptionen

Abbildung 3.11 zeigt die Registerkarte DB-BIBLIOTHEKSOPTIONEN, die in SQL Server 2000 nur aus Gründen der Abwärtskompatibilität existiert. Für die Verbindung zu SQL Server 2000 sind ODBC und OLE DB die bevorzugten Mechanismen. Alle im Lieferumfang von SQL Server 2000 enthaltenen Clientdienstprogramme (mit Ausnahme von `isql.exe`) verwenden ODBC. Das Dienstprogramm `isql.exe` arbeitet weiterhin mit DB-Library und auch ältere, vom Benutzer geschriebene Anwendungen können noch auf diese Netzwerkbibliothek zurückgreifen.

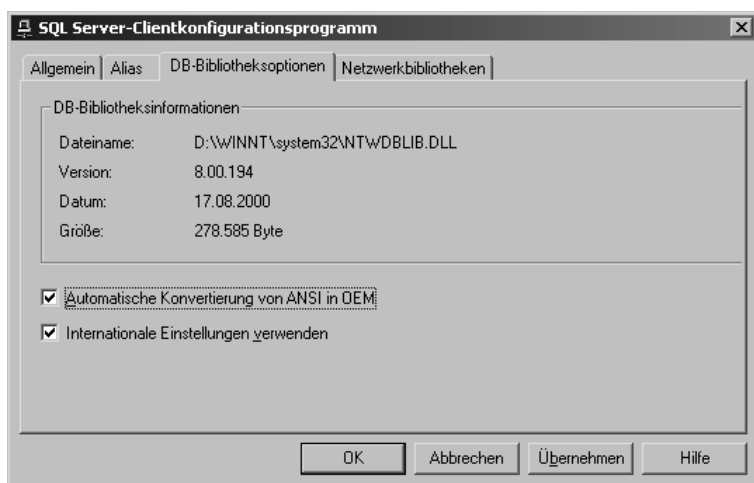


Abbildung 3.11:
Die Registerkarte
DB-Bibliotheks-
optionen

Diese Registerkarte zeigt an, welche Version der DB-Bibliotheks-DLL auf Ihrem System aktiv ist. Hier erfahren Sie auch den physikalischen Standort, das Datum und die Größe sowie den Namen der DLL. Außerdem sind zwei Optionen verfügbar: AUTOMATISCHE KONVERTIERUNG VON ANSI IN OEM UND INTERNATIONALE EINSTELLUNG VERWENDEN.

- Die erste Option verwendet bestimmte Einstellungen wie Datums- und Zeitformate aus Windows 9x, Windows NT und Windows 2000. Falls Sie zum Beispiel Windows 2000 verwenden und von den US-Einstellungen abweichen, haben Ihre lokalen Einstellungen Vorrang vor den SQL Server-Einstellungen.
- Die Konvertierung von ANSI in OEM übersetzt die von SQL Server kommenden Daten automatisch in die vom Client verwendete lokale Codeseite.

Normalerweise sollten Sie diese Einstellungen nicht deaktivieren.

Die Registerkarte Netzwerkbibliotheken

Die Registerkarte NETZWERKBIBLIOTHEKEN listet alle auf Ihrem Computer installierten Netzwerkbibliotheken mit ihrem Dateinamen, der Version, Dateidatum und Größe auf. Diese Informationen sind besonders hilfreich, wenn Sie Unterstützung benötigen. Die Versionsinformationen sagen Ihrem Anbieter für technischen Support, welche Service Packs und welche Versionen der Netzwerkbibliotheken Sie verwenden. Alle Netzwerkbibliotheken werden per Vorgabe auf Ihrem Rechner installiert.

SQL Server-Netzwerkconfiguration

Die SQL Server-Netzwerkconfiguration ist der SQL Server-Clientconfiguration sehr ähnlich. Beide stehen, wie Sie wahrscheinlich schon erraten haben, in enger Beziehung zueinander. Während aber die SQL Server-Clientconfiguration bestimmt, wie Ihre Anwendungen die Verbindung zu SQL Server herstellen, legt die SQL Server-Netzwerkconfiguration fest, welche Netzwerkbibliotheken SQL Server 2000 verwendet (siehe Abbildung 3.12). Um bei dem bereits bekannten Vergleich mit natürlichen Sprachen zu bleiben, repräsentiert dieses Programm die Liste der Sprachen, die Ihr Server spricht. Falls Sie versuchen, eine Verbindung unter Verwendung einer anderen Netzwerkbibliothek herzustellen, wird Ihr Server Sie nicht verstehen.

Genau wie in der SQL Server-Clientconfiguration können Sie einfach auf AKTIVIEREN klicken, um der Liste weitere Netzwerkbibliotheken hinzuzufügen. Zur Neukonfiguration einer Netzwerkbibliothek klicken Sie auf EIGENSCHAFTEN; eine Netzwerkbibliothek deaktivieren Sie, indem Sie sie markieren und auf DEAKTIVIEREN klicken. Die vorgenommenen Änderungen treten erst beim nächsten Neustart des SQL Server-Dienstes in Kraft.

Sehen Sie sich kurz Abbildung 3.13 an. Wie bereits in der gestrigen Lektion erwähnt, unterstützen die benannten Instanzen von SQL Server 2000 nur eine Untermenge der Netzwerkbibliotheken (TCP/IP-Sockets, Named Pipes, NWLink IPX/SPX und VIA). Die Unterstützung für die anderen Netzwerkbibliotheken hat einen ähnlichen Status wie bei der DB-Library: Für die Zukunft sind keine Erweiterungen vorgesehen, was auch für die Konnektivität zu benannten Instanzen gilt.

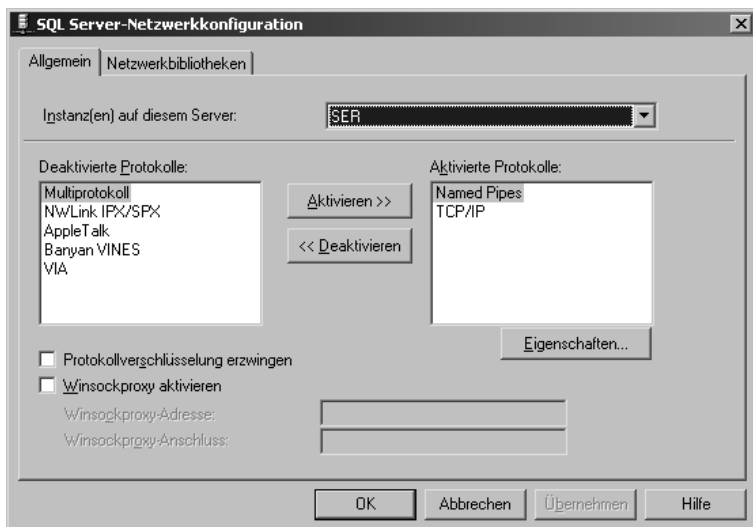


Abbildung 3.12:
Das Dienstprogramm SQL Server-
Netzwerkkonfiguration

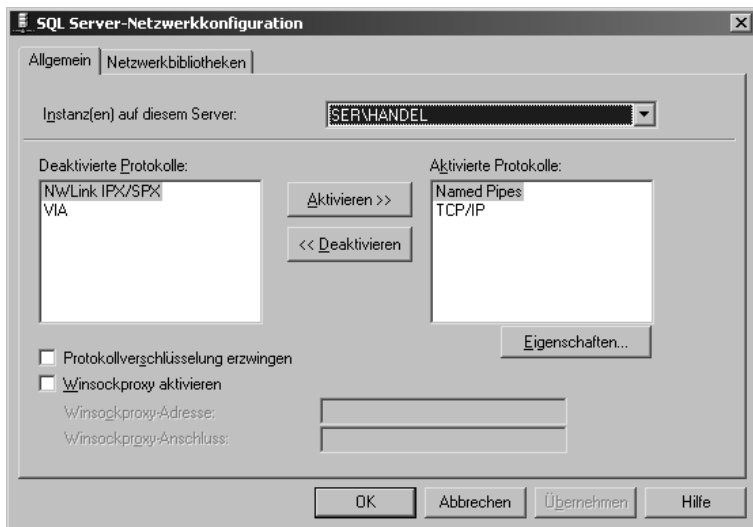


Abbildung 3.13:
Das Dienstprogramm SQL Server-
Netzwerkkonfiguration für eine
benannte Instanz

Die Registerkarte NETZWERKBIBLIOTHEKEN entspricht vollständig der gleichnamigen Registerkarte des Dienstprogramms SQL Server-Clientkonfiguration.

Protokollverschlüsselung erzwingen

Wenn Sie dieses Kontrollkästchen aktivieren, *erzwingt* der Server die Verschlüsselung für alle Verbindungsversuche. Lässt sich zwischen Client und Server keine Verschlüsselung aushandeln, ist keine Verbindung möglich. Die Verschlüsselung in SQL Server 2000 unterscheidet sich wesentlich von der in SQL Server 7.0, die von der Multiprotokoll-Netzwerkbibliothek abhängig ist.

SQL Server 2000 verwendet SSL/TLS (Secure Sockets Layer/Transport Level Security), um den gesamten SQL Server-Verkehr zu verschlüsseln (falls Sie es wünschen). Diese Fähigkeit ist vor allem sinnvoll, wenn Ihr SQL Server vertrauliche Daten speichert und Sie verhindern wollen, dass unbefugte Personen die Daten während der Übertragung im Netzwerk ausspionieren können.

Das klingt zwar interessant, aber wie funktioniert es? Zuerst brauchen Sie einen öffentlichen Schlüssel für Ihren Server. Er muss für den Namen des vollständig qualifizierten DNS-Namens des Computers ausgelegt sein. Dann müssen Sie das Zertifikat auf dem SQL Server-Computer über Internet Explorer installieren. Wenn SQL Server ein Zertifikat findet, das dem Servernamen entspricht, verwendet er einfach dieses Zertifikat.

Verwenden Sie Zertifikate in Ihrer Firma oder Organisation, sollten Sie diese Seite des Buches Ihrem Zertifikatverwalter zeigen; er kann Ihnen dann bei der Auswahl und Installation des richtigen Zertifikats helfen. Firmen verfügen oftmals über ihre eigene Infrastruktur für Zertifikate, so dass die folgenden Anweisungen möglicherweise nicht auf Sie zutreffen.

Wenn Sie das Verfahren selbst durchführen, besorgen Sie sich einfach ein Zertifikat von einer Zertifizierungsstelle wie VeriSign (<http://www.verisign.com>), der Ihre Internet Explorer-Clients vertrauen (VeriSign bietet sogar ein kostenloses Probezertifikat an). Um den DNS-Namen Ihres Computers zu ermitteln, klicken Sie mit der rechten Maustaste auf ARBEITSPLATZ und wählen EIGENSCHAFTEN aus dem Kontextmenü. Hier finden Sie den vollständigen Namen Ihres Computers auf der Registerkarte NETZWERKIDENTIFIKATION. Gegebenenfalls müssen Sie in dieser Registerkarte auf EIGENSCHAFTEN klicken. Beispielsweise kann der Computernamen `SER.COMPUTER.MYNETWORK.COM` lauten. Fordern Sie dann ein »Server-Identitätszertifikat« oder ein »Webserverzertifikat« an. Wenn Sie es erhalten haben, installieren Sie es mit Internet Explorer (über EXTRAS / INTERNETOPTIONEN / INHALT / ZERTIFIKATE in IE 5.0) und wählen IMPORTIEREN, um das per E-Mail von VeriSign erhaltene Zertifikat zu laden. Wenn alles korrekt funktioniert, sollte die Verschlüsselung automatisch ihre Arbeit aufnehmen, sobald Sie Ihren SQL Server-Dienst das nächste Mal starten.

Falls die Verschlüsselung aus welchen Gründen auch immer nicht funktioniert, erhalten Sie auf dem Client eine entsprechende Meldung, dass ein Aushandeln der Verschlüsselung mit dem Server nicht möglich gewesen ist. In der Regel bedeutet diese Meldung, dass der Client dem Zertifikat nicht vertraut. Die Namen der Zertifizierungsstellen (Firmen), denen Internet Explorer vertraut, können Sie sich an der gleichen Stelle wie eben beschrie-

ben ansehen (über EXTRAS / INTERNETOPTIONEN / INHALT / ZERTIFIKATE in IE 5.0), außer dass Sie jetzt auf HERAUSGEBER klicken. Die Firma VeriSign wird hier empfohlen, weil sie schon lange auf dem Markt ist und sich in allen Listen von vertrauten Herausgebern findet.

Winsockproxy aktivieren

Mit der Option *Winsockproxy aktivieren* kann man einen Proxyserver veranlassen, SQL Server-Anforderungen vom Internet (oder von jedem anderen Netzwerk) an SQL Server weiterzuleiten. Schalten Sie das Kontrollkästchen Winsockproxy aktivieren ein und füllen Sie dann die Felder für die TCP/IP-Adresse des Proxyservers sowie den Anschluss, der als Proxy für SQL Server fungieren soll, aus. Diese Option erlaubt, dass Anforderungen an den Proxyserver zum SQL Server umgeleitet werden und dass SQL Server auf diese Anforderungen antworten kann. Die meisten SQL Server-Computer, die für Internet-E-Commerce eingesetzt werden, stellen den Backend für Websites dar und müssen nicht direkt mit dem Internet verbunden sein. Wenn Sie jedoch so vorgehen wollen, steht diese Option zur Verfügung.

Query Analyzer

Der SQL Server Query Analyzer ist Ihr wichtigstes Werkzeug um Transact-SQL-Abfragen oder gespeicherte Prozeduren auszuführen. Dieses Dienstprogramm setzen Sie im weiteren Verlauf dieses 21-Tage-Kurses ständig ein, so dass Sie sich mit den verfügbaren Optionen vertraut machen sollten. Auch wenn einige Funktionen nicht auf Anhieb verständlich sind, werden die einschlägigen Abschnitte alles genau erläutern.

Diese Anwendung können Sie einfach starten, indem Sie START / AUSFÜHREN wählen und dann den Namen der Anwendung eingeben. Aus historischen Gründen heißt diese Anwendung ISQLW.exe. Geben Sie also in das Dialogfeld AUSFÜHREN den Programmnamen isqlw ein, um Query Analyzer zu starten.

Wie Sie noch in Lektion 5 lernen, ist jede Operation in SQL Server sicher. Deshalb müssen Sie sich zunächst bei jedem SQL Server, mit dem Sie sich verbinden wollen, anmelden und identifizieren (siehe Abbildung 3.14). Wie schon bei der Behandlung der Clientkonfiguration erwähnt, können Sie »(local)« oder ».« eingeben, um sich mit Ihrer lokalen Instanz von SQL Server zu verbinden. Den Servernamen können Sie auch frei lassen. Der Query Analyzer erkennt Ihre Absichten und stellt eine Verbindung zur Standardinstanz von SQL Server her. Falls Sie an dem Computer arbeiten, auf dem der SQL Server läuft, mit dem Sie sich verbinden wollen, wird die standardmäßige lokale Netzwerkverbindung verwendet (TCP/IP-Sockets). Anderenfalls wird die von Ihnen konfigurierte

Clientnetzwerkbibliothek verwendet. Um die Verbindung zu einer benannten Instanz herzustellen, geben Sie einfach die Kombination *Servername\Instanzname* in das Feld SQL SERVER ein.



Abbildung 3.14:
Das Dialogfeld *Verbindung mit SQL Server herstellen* beim Start von *SQL Server-Query Analyzer*

Man kann auch veranlassen, dass SQL Server automatisch startet, wenn man versucht, eine Verbindung herzustellen und der SQL Server-Dienst noch nicht läuft. Diese Option bietet sich vor allem für lokale oder Windows 9x-Computer an; bei Produktionsservern sollten Sie mit dieser Option vorsichtig umgehen. Warum ist der Server heruntergefahren? Falls es aus Wartungsgründen geschehen ist, was passiert dann bei einem vorzeitigen Start von SQL Server?

Außerdem müssen Sie die Anmeldeinformationen eingeben. Auf Windows 9x-Computern können Sie nur die SQL Server-Authentifizierung verwenden. Geben Sie als Benutzernamen *sa* ein und lassen Sie das Kennwort leer. Für Server, die zwischen Groß- und Kleinschreibung unterscheiden, gilt dies auch für die Anmeldung – geben Sie den Anmeldenamen und das Kennwort also in der richtigen Schreibweise an. Falls Sie an einem Windows 2000-Computer arbeiten oder eine Verbindung zu einem SQL Server herstellen wollen, der unter Windows NT/2000 läuft, können Sie die Option *Windows-Authentifizierung* wählen, um sich mit Ihren Windows NT/2000-Benutzerrechten anzumelden. Diese Art der Anmeldung ist natürlich einfacher. Standardmäßig können sich Mitglieder der lokalen Windows NT/2000-Gruppe Administratoren an SQL Server anmelden. In beiden Fällen erfolgt die Anmeldung als Systemadministrator oder »Superuser« bei SQL Server. Das lässt sich nicht ändern – seien Sie also entsprechend vorsichtig.

Nachdem Sie auf OK geklickt oder die Eingabetaste betätigt haben, werden Sie bei der entsprechenden Instanz von SQL Server angemeldet (vorausgesetzt, dass Sie den richtigen Benutzernamen und das richtige Kennwort angegeben haben oder Ihre Windows NT/2000-Berechtigungen den Zugang erlauben). Damit gelangen Sie zu einem Bildschirm, wie ihn Abbildung 3.15 zeigt.

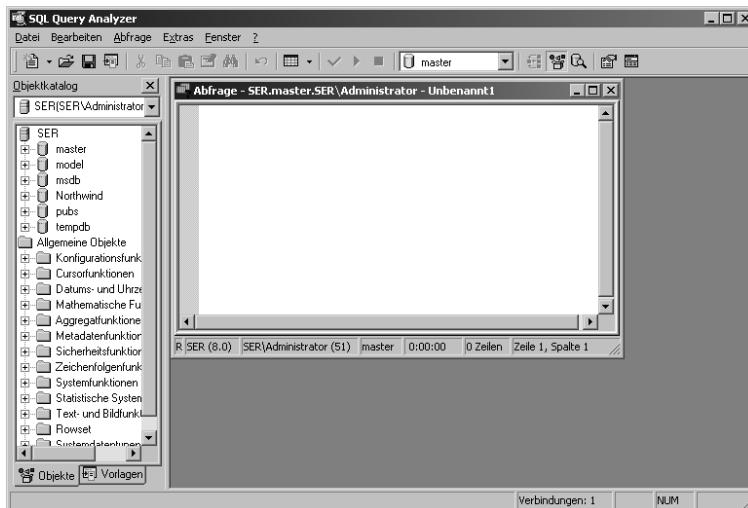


Abbildung 3.15:
Der SQL Server-
Query Analyzer

Falls die Anmeldung fehlschlägt, erscheint ein Meldungsfeld wie in Abbildung 3.16. Das passiert gewöhnlich, wenn Sie das Kennwort falsch geschrieben haben. Geben Sie es einfach noch einmal ein. Überprüfen Sie auch, dass Sie Ihren Benutzernamen richtig geschrieben haben. Falls Sie das Kennwort für `sa` geändert haben, müssen Sie auch dieses Kennwort im Verbindungsdialogfeld nach Abbildung 3.14 verwenden. Diesen Fehler können Sie leicht provozieren, wenn Sie sich – wie in Lektion 2 beschrieben – mit der Instanz `SER\Handel` (bzw. der von Ihnen installierten Instanz) verbinden möchten und dabei ein falsches Kennwort für `sa` angeben.



Abbildung 3.16:
Meldungsfeld bei
gescheiterter Anmel-
dung

Es kann auch eine Fehlermeldung wie in Abbildung 3.17 erscheinen, wenn Sie versucht haben, eine Verbindung über SQL Server-Authentifizierung herzustellen. In diesem Fall unterstützt der Server (Ihre Standardinstanz) ausschließlich die Windows-Authentifizierung (auch als integrierte Sicherheit bezeichnet) und diese Fehlermeldung weist Sie darauf hin. Gehen Sie zurück zum Anmeldedialogfeld und wählen Sie die Option Windows-Authentifizierung; dann sollte die Verbindung klappen.

Abbildung 3.18 zeigt eine andere mögliche Fehlermeldung. Falls Sie diesen Fehler erhalten, müssen Sie mehrere Dinge überprüfen. Als Erstes stellen Sie sicher, dass der SQL Ser-

ver-Dienst auf dem Computer läuft, bei dem Sie sich anmelden wollen. Ist das der Fall, kann das Problem in der Clientkonfiguration bestehen (siehe dazu den Abschnitt zur Clientkonfiguration weiter vorn in dieser Lektion). Es kann auch sein, dass Sie den Namen des Servers falsch geschrieben haben. Um eine Liste der im Netzwerk vorhandenen Server anzuzeigen, statt den Namen des Servers einzugeben, können Sie auf die drei Punkte neben dem Eingabefeld für den Servernamen klicken; Sie erhalten dann eine Liste der in Ihrem Netzwerk aktiven SQL Server-Computer. Diese Liste zeigt manchmal nur die halbe Wahrheit – wenn Ihr Server nicht in der Liste auftaucht, können Sie die Verbindung aufbauen, indem Sie den Namen des Servers direkt eingeben.

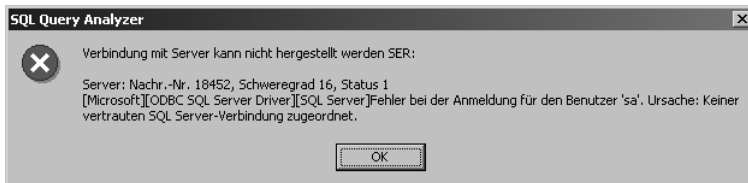


Abbildung 3.17:
Anmeldung gescheitert – der Benutzer ist keiner vertrauten SQL Server-Verbindung zugeordnet.



Abbildung 3.18:
Probleme mit der Netzwerk-konnektivität

Abbildung 3.19 zeigt ebenfalls einen leeren Query Analyzer, der aber folgende Besonderheit aufweist: Es bestehen hier zwei offene Verbindungen. Der Query Analyzer kann gleichzeitig bis zu 32 offene Verbindungen verwalten. Der Titelzeile für jede Verbindung können Sie entnehmen

- mit welchem Computer Sie verbunden sind,
- welche Datenbank Sie verwenden,
- unter welchem Benutzernamen Sie angemeldet sind,
- welche Namen die geöffneten Abfragen tragen (mehr dazu in Kürze) und
- welche Nummer das jeweilige Fenster hat.

Diese Informationen sind vor allem dann unentbehrlich, wenn Sie 40 oder 50 Verbindungen gleichzeitig geöffnet haben. Im Beispiel von Abbildung 3.19 bestehen lediglich zwei Verbindungen – eine zur Standardinstanz und eine zur benannten Instanz `.\Handel`. Beide Abfragen beziehen sich auf die Datenbank `master` und zwar zu separaten Instanzen. Es gibt keinen Grund, dass beide Verbindungen nicht zu ein und derselben Instanz erfolgen kön-

nen, auch wenn beispielsweise die Standardinstanz nur die Windows-Authentifizierung akzeptiert und die benannte Instanz mit SQL Server-Authentifizierung arbeitet.

Die Abfrage-Symbolleiste

Nachdem Sie sich erfolgreich angemeldet haben, können Sie Abfragen mit Transact-SQL-Anweisungen ausführen (dazu mehr in Lektion 10). Im Query Analyzer lassen sich mehrere Fenster öffnen. Klicken Sie dazu auf das Symbol NEUE ABFRAGE (das Symbol am linken Rand der Symbolleiste). Beachten Sie, dass das etwas anderes ist, als der Aufbau mehrerer Verbindungen, denn diese können zu verschiedenen Computern bestehen. Jedes neue Abfragefenster stellt eine eigene Verbindung zu SQL Server her; wenn Sie zu viele Abfragefenster öffnen, verschwenden Sie Ressourcen auf Ihrem SQL Server.

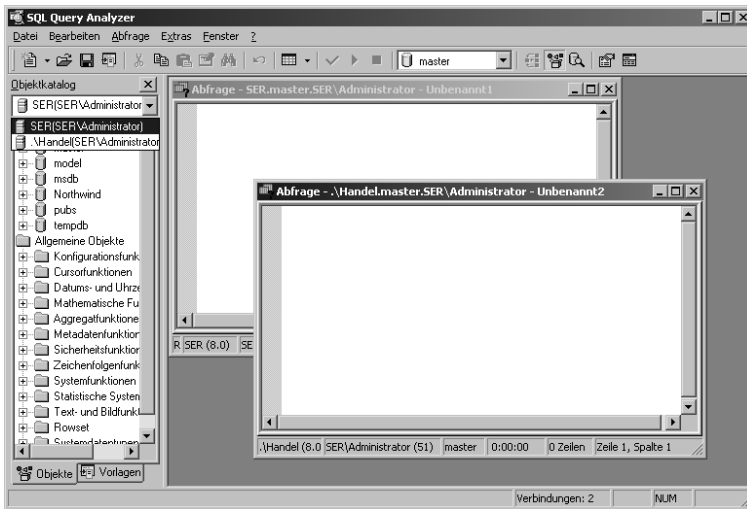


Abbildung 3.19:
Mehrere Verbindungen in einer einzelnen Instanz des Query Analyzers



Abbildung 3.20:
Die Symbolleiste des Query Analyzers

Wenn Sie das zweite Symbol (das wie das Windows-Standardsymbol für DATEI ÖFFNEN aussieht) anklicken, öffnen Sie damit ein Standarddialogfeld, über das Sie Transact-SQL-Skripts (mit der Standarderweiterung *.sql*) laden können. Das nächste Symbol (ABFRAGE/ ERGEBNIS SPEICHERN) speichert entweder den Text aus Ihrem Abfragefenster in einer Datei oder – falls Sie eine Abfrage ausgeführt und das Ergebnisfenster ausgewählt (oder angeklickt) haben – die Ergebnisse der Abfrage. Über das nächste Symbol (VORLAGE EINFÜGEN) laden Sie eine »Vorlagen-SQL-Abfrage«, so dass Sie schnell Variationen bereits fertiger Programmteile entwickeln können. Dann folgen die Schaltflächen für die Win-

dows-Standardoperationen Ausschneiden, Kopieren und Einfügen. Die nächste Schaltfläche löscht den Inhalt des aktuellen Abfragefensters. Daran schließt sich die Schaltfläche für die Standardoperation Suchen an. Diese Schaltflächen werden im geeigneten Kontext aktiv und sind andernfalls deaktiviert dargestellt. Mit der nächsten Schaltfläche lässt sich eine Operation rückgängig machen (beispielsweise das Löschen eines versehentlich gelöschten Textes aufheben).

Die als Nächstes folgende Schaltfläche **AUSFÜHRUNGSMODUS** ist sehr leistungsfähig. Wenn Sie auf diese Schaltfläche klicken, öffnet sich eine Dropdown-Liste mit Optionen, die bestimmen, wie und wo die Ergebnisse einer Abfrage erscheinen. Die Standardoption *Ergebnisse in Raster* ist oftmals praktisch, weil viele Namen in SQL Server 2000 eine Länge von 128 Unicode-Zeichen haben können und die normale Textanzeige alle Namen in voller Länge – gegebenenfalls mit Leerzeichen aufgefüllt – ausgibt. Die Option *Ergebnisse in Raster* lässt in der Regel nur Platz für die breiteste anzuzeigende Datenspalte. Wenn Sie sich den gesamten Text ansehen wollen, schalten Sie einfach in den Modus *Ergebnisse in Text*. Die Ergebnisse von Abfragen können Sie sogar in eine Datei umleiten. Abbildung 3.21 zeigt die einzelnen Optionen.

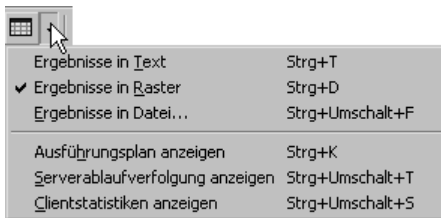


Abbildung 3.21:
Auswahl des Abfragemodus

Der nächste Befehl im Menü **AUSFÜHRUNGSMODUS** (Abbildung 3.21) heißt **AUSFÜHRUNGSPLAN ANZEIGEN**. Diese Option zeigt die Methoden und Indizes, nach denen SQL Server die mit den Abfragen angeforderten Daten sucht. Anders als bei den Optionen zur Text- oder Rasterdarstellung, von denen jeweils eine eingeschaltet sein muss, können Sie diese Option unabhängig von der gewählten Darstellungsart ein- oder ausschalten.

Der Menübefehl **SERVERABLAUFVERFOLGUNG ANZEIGEN** ist nur für Systemadministratoren von SQL Server verfügbar und zeigt alle zugrunde liegenden Befehle und mehrere Statistiken zu diesen Befehlen aus SQL Server. Es handelt sich praktisch um eine Miniversion des SQL Server-Profilers, auf den diese Lektion später eingeht.

Der letzte Menübefehl **CLIENTSTATISTIKEN ANZEIGEN** bringt alle Angaben, die Query Analyzer von SQL Server bei der Ausführung des letzten Befehls erhalten hat, sowie einige Durchschnittsdaten bezogen auf zusätzlich ausgeführte Befehle.

Schalten Sie diese drei Anzeigeeoptionen ein, damit Sie deren Wirkung verfolgen können. Achten Sie darauf, dass die Standardoption *Ergebnisse in Raster* aktiv ist, wenn alle drei Anzeigeeoptionen eingeschaltet sind.

Am Einfachsten lässt sich die Wirkung dieser Optionen an einem Beispiel zeigen. Geben Sie den folgenden Text in das Abfragefenster ein:



```
use pubs
select * from authors
```



Diese Abfrage ist unkompliziert, wenn Sie verstehen, was die einzelnen Anweisungen bedeuten. Die erste Anweisung, `use pubs`, weist den Server an, die Datenbank `pubs` zu verwenden. Diese Datenbank ist eine Beispieldatenbank, die auf allen SQL Servern installiert wird. Die zweite Anweisung, `select * from authors`, fordert den Server auf, alle Zeilen und Spalten der Tabelle oder der Ansicht `authors` zurückzugeben (und da Sie mit der vorherigen Anweisung die Datenbank `pubs` aktiviert haben, wissen Sie, dass sich die Tabelle bzw. Ansicht in dieser Datenbank befindet).

Auf Deutsch lauten diese Anweisungen also:

Wechsle zur Datenbank `pubs`
Zeige mir alle Daten über Autoren

Nachdem Sie diese beiden Anweisungen eingetippt haben, können Sie überprüfen lassen, ob sie korrekt eingegeben wurden. In der Symbolleiste sind jetzt verschiedene Optionen verfügbar. Klicken Sie auf das blaue Häkchen, um die Eingaben zu überprüfen. SQL Server analysiert die Abfrage – d.h., prüft, ob die Abfrage syntaktisch richtig ist. Wenn alles o.k. ist, erscheint die Meldung:



Befehl(e) erfolgreich abgeschlossen.

Um die Abfrage nun tatsächlich auszuführen, klicken Sie auf das nächste Symbol – die Schaltfläche **ABFRAGE AUSFÜHREN**. Sie können auch **ABFRAGE / AUSFÜHREN** wählen oder **[F5]** bzw. **[Alt]+[X]** auf der Tastatur drücken. Nachdem Sie die Abfrage ausgeführt haben, erscheint das Ergebnisfenster mit allen Informationen zu *authors* der Datenbank *pubs* (siehe Abbildung 3.22).

Während SQL Server die Abfrage ausführt, ist die Schaltfläche **ABFRAGEAUSFÜHRUNG ABBRECHEN** aktiviert. Damit können Sie eine laufende Abfrage abbrechen. Normalerweise werden die Abfragen jedoch so schnell durchgeführt, dass Sie kaum eine Chance haben, sie abzubrechen.

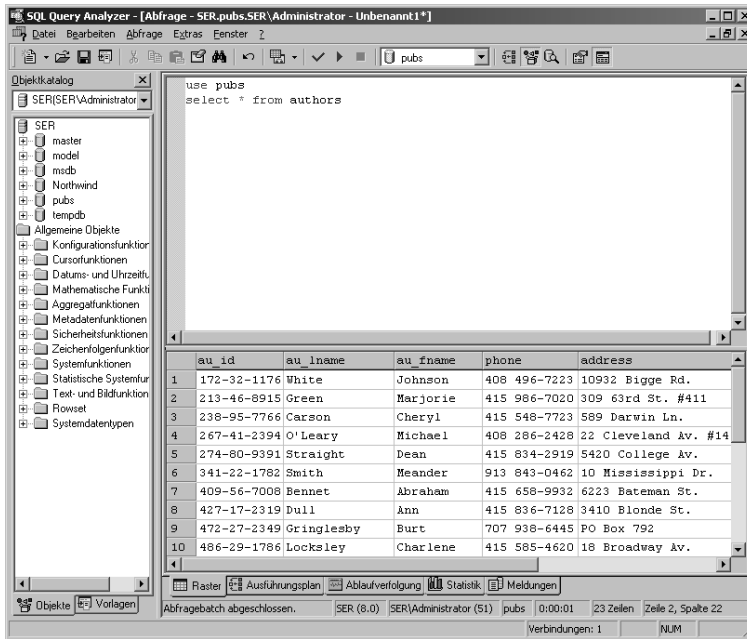


Abbildung 3.22:
Die Abfrage-
ergebnisse

Kommen wir nun zurück zum Unterschied zwischen Text- und Rasterdarstellung. Klicken Sie auf die Schaltfläche **AUSFÜHRUNGSMODUS** und wechseln Sie zur Option *Ergebnisse in Text*. Diese Darstellungsart können Sie auch mit **[Strg] + [T]** wählen. Wiederholen Sie die Abfrage und sehen Sie sich das Ergebnisfenster an. Alle Register sind ebenfalls wieder unten aufgeführt, nur die Abfrageergebnisse sind etwas anders formatiert.

Erweitern Sie dieses Beispiel, indem Sie im Fenster mit den Transact-SQL-Anweisungen eine weitere Anweisung wie folgt einfügen:



```
use pubs
select * from authors
select * from titles
```

Führen Sie diese Abfragen aus (mit einer der oben angegebenen Methoden); Query Analyzer stapelt die Ergebnisse in ein und demselben Ergebnisfenster einfach übereinander. Für jede ausgeführte Abfrage erstellt Query Analyzer einfach ein neues Raster oder trennt den Text innerhalb desselben Fensters.

Auf der Symbolleiste finden Sie als nächste Option ein Dropdown-Listefeld; es enthält eine Liste der installierten Datenbanken des Servers, mit dem Sie verbunden sind. Wenn Sie hier die Auswahl ändern, arbeitet das ausgeführte SQL-Skript mit dieser Datenbank, sofern Sie nicht im Skript selbst eine Datenbank explizit angeben. Wird im Skript die aktu-

elle Datenbank geändert, spiegelt sich die neue Auswahl in diesem Listenfeld nach Ausführung des Skripts wider.

Mithilfe der nächsten Schaltfläche **GESCHÄTZTEN AUSFÜHRUNGSPLAN ANZEIGEN** können Sie anzeigen, wie SQL Server die Ausführung der Abfrage intern einschätzt (im Gegensatz zur vorher festgelegten Option *Ausführungsplan anzeigen*, die anzeigt, wie die Abfrage tatsächlich läuft). Warum zwei Optionen? Wie verhält es sich, wenn eine Abfrage mehrere Stunden dauert, Sie aber lediglich sehen wollen, wie SQL Server die Ausführung der Abfrage geplant hat? Auf dieses Merkmal geht Lektion 13 im Detail ein.

Mit der nächsten Schaltfläche können Sie den Objektkatalog ein-/ausschalten. Die bisherigen Abbildungen haben den Objektkatalog im linken Teil des Fensters gezeigt. Damit können Sie in einfacher Weise alle Datenbankobjekte für jede Verbindung durchsuchen. Diese Funktion ist leistungsfähiger als es auf den ersten Blick scheint. Objekte lassen sich per Drag & Drop in das Abfragefenster ziehen. Wenn Sie mit der rechten Maustaste auf das Objekt klicken und es in das Abfragefenster ziehen, erscheint ein Popup-Fenster (siehe Abbildung 3.23). Hier stehen verschiedene Optionen für SQL Server-Anweisungen zur Verfügung – beispielsweise können Sie eine Anweisung zum Erstellen einer Tabelle oder eine Auswahlanweisung schreiben lassen. Auf diese Möglichkeit werden Sie in den nächsten Tagen sicherlich öfter zurückgreifen.

Im Query Analyzer gibt es noch weitere neue Merkmale. Die nächste Schaltfläche auf der Symbolleiste heißt **OBJEKTSUCHE**. Wenn Sie darauf klicken, erscheint ein Dialogfeld wie in Abbildung 3.24 gezeigt. Wissen Sie nicht mehr, wo Sie die Tabelle mit der Spalte für das Gehalt untergebracht haben? Mit der Objektsuche des Query Analyzers können Sie überall nach dieser Tabelle suchen. Insbesondere bei großen oder komplexen Systemen bzw. bei Systemen, die sich nicht auf Anhieb überblicken lassen, ist diese Funktion sehr nützlich.

Die nächste Schaltfläche (**AKTUELLE VERBINDUNGSEIGENSCHAFTEN**) behandelt der nachfolgende Abschnitt näher. Mit der letzten Schaltfläche der Symbolleiste können Sie schließlich festlegen, dass das Ergebnissenster immer zu sehen ist, selbst wenn Sie noch keine Abfrage ausgeführt haben.

Optionen für SQL Server-Query Analyzer konfigurieren

Über die Schaltfläche mit der Hand, die ein Blatt Papier hält (die vorletzte Schaltfläche der Symbolleiste), öffnen Sie das Dialogfeld **AKTUELLE VERBINDUNGSEIGENSCHAFTEN** (siehe Abbildung 3.25). Zu diesem Dialogfeld gelangen Sie auch über **ABFRAGE / AKTUELLE VERBINDUNGSEIGENSCHAFTEN**.

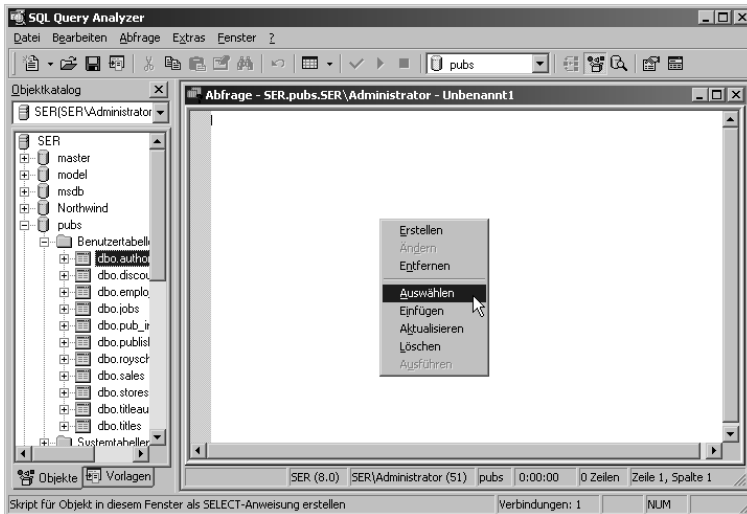


Abbildung 3.23:
Drag & Drop vom
Objektkatalog

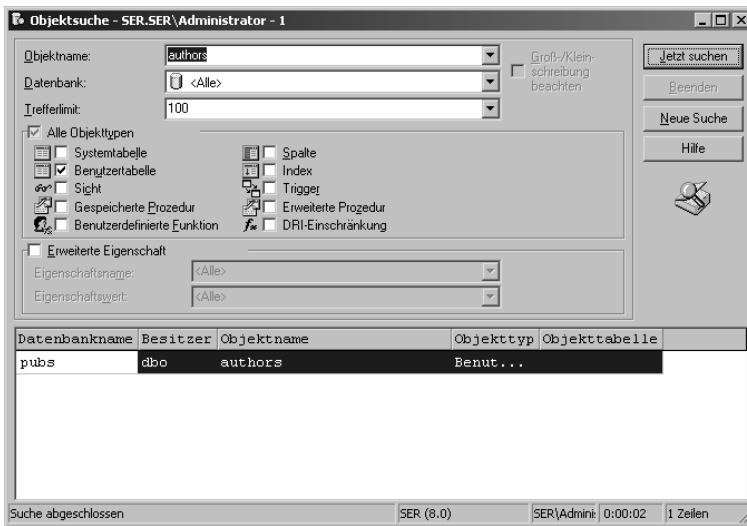


Abbildung 3.24:
Das Dialogfeld
Objektsuche

Die Optionen des Dialogfelds AKTUELLE VERBINDUNGSEIGENSCHAFTEN sollten Sie erst ändern, wenn Sie sich mit den erweiterten Merkmalen von SQL Server auskennen. Zumindest wissen Sie aber erst einmal, wo sich diese Optionen bei Bedarf ändern lassen.

Die Optionen von Query Analyzer können Sie noch an einer anderen Stelle konfigurieren und ändern. Das in Abbildung 3.26 dargestellte Dialogfeld erreichen Sie über den Befehl OPTIONEN aus dem Menü EXTRAS. Hier finden Sie zahlreiche Einstellmöglichkeiten. Die Onlinedokumentation erläutert alle Optionen im Detail, aber Sie werden die meisten die-

ser Einstellungen erst ändern, wenn Sie Erfahrungen als SQL Server-Administrator oder -Programmierer gesammelt haben.

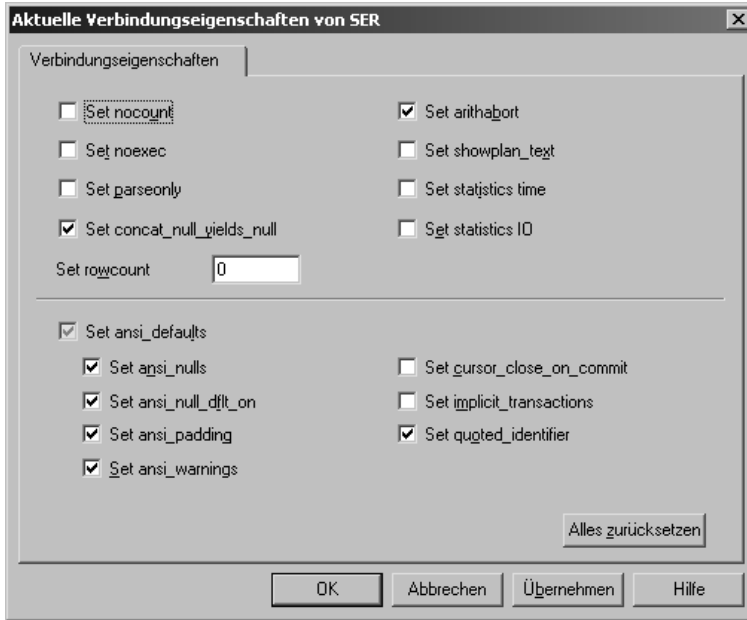


Abbildung 3.25:
Das Dialogfeld
Aktuelle Verbin-
dungseigenschaften



Verschiedentlich wurde erwähnt, dass ein Skript ausgeführt wird. Ein Transact-SQL-Skript ist eine Gruppe von SQL-Anweisungen, die gemeinsam gespeichert und ausgeführt werden. Über die Menübefehle DATEI / SPEICHERN und DATEI / ÖFFNEN können Sie eine Abfrage (oder mehrere Abfragen) speichern bzw. zur Ausführung öffnen. In der Standardeinstellung haben Skripts die Erweiterung .sql. Diese Erweiterung können Sie auf der Registerkarte Allgemein ändern, was aber nur in Ausnahmefällen sinnvoll ist. Außerdem können Sie auf dieser Registerkarte die Standardstandorte für Skripts bestimmen sowie die Optionen für die Standorte der Ergebnisse aus Abfragen und die für die betreffenden Dateien zu verwendenden Standarderweiterungen festlegen.

Auf den Registerkarten EDITOR und ERGEBNISSE können Sie die Entwicklungsumgebung an Ihre Arbeitsgewohnheiten anpassen. Über die Registerkarte SCHRIFTARTEN ändern Sie die Standardschriften, die für die Anzeige von Abfragen und Ergebnissen in der Benutzeroberfläche verwendet werden. Die Registerkarte VERBINDUNGEN zeigt die gleichen Verbindungseigenschaften, die Sie gerade eben für die aktuelle Verbindung kennen gelernt haben (wobei die Standardwerte für alle Verbindungen eingestellt sind).

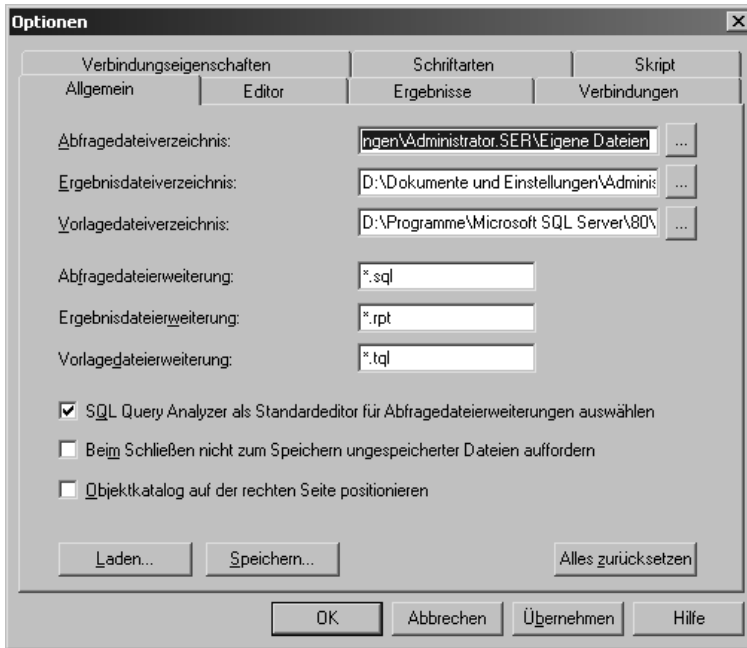


Abbildung 3.26:
Das Dialogfeld
Optionen des Query
Analyzers

Auf der Registerkarte VERBINDUNGEN können Sie einstellen, wie lange Query Analyzer bei Anmeldeversuchen zu SQL Server warten soll. Es lässt sich auch konfigurieren, wie lange auf länger andauernde Abfragen gewartet werden soll (0 bedeutet unbegrenzt warten; jeder andere Wert gibt die Anzahl der Sekunden an). Außerdem legen Sie auf dieser Registerkarte fest, ob die Zeichendaten zu übersetzen und die Ländereinstellungen zu verwenden sind.

Ändern Sie diese Optionen mit Bedacht; sie haben Einfluss auf die zurückgegebenen Ergebnisse und man vergisst leicht, dass man in diesem Dialogfeld Änderungen vorgenommen hat.

Hilfoptionen

Als Option untersuchen wir hier die Hilfoptionen des Query Analyzers. Falls Sie sich nicht mehr an die Syntax einer Anweisung erinnern können (z. B. wie man die Anweisung richtig schreibt oder welche Parameter erforderlich sind), markieren Sie ein Schlüsselwort und drücken $\text{Alt} + \text{F1}$. Daraufhin erscheint ein Hilfefenster für diese Anweisung. Hilfe ist für nahezu jede Transact-SQL-Anweisung verfügbar. Im Gegensatz zu vorherigen Versionen von SQL Server startet die Hilfe von SQL Server 2000 immer die Onlinedokumentation und bringt Sie zum passendsten Thema für die angeforderte Hilfe. Diese Funktion

gewährleistet eine präzise Hilfestellung, da lediglich eine definite Informationsquelle zur Syntax und zu Abfragoptionen existiert.

Eine weitere nützliche Option ist die Tastenkombination **[Alt] + [F1]**. Wenn Sie den Namen eines SQL Server-Objekts, z.B. einer Tabelle, markieren, erscheint die Standardhilfe im Ergebnisfenster. Welche Hilfe Sie konkret erhalten, hängt von dem Objekt ab, für das Sie die Hilfe anfordern – normalerweise zeigt die Hilfe die nützlichen Eigenschaften des Objekts an.



Wenn Sie einen Befehl im Abfragefenster markieren und dann die Abfrage ausführen, wird nur der markierte Text ausgeführt. Auf diese Weise brauchen Sie nicht alle Anweisungen in einem Fenster auszuführen.

Enterprise Manager

Der Query Analyzer ist zweifellos ein leistungsfähiges Werkzeug. In SQL Server 2000 hat er viele Erweiterungen erfahren, um den Ansprüchen des Programmierers und Datenbankadministrators besser gerecht zu werden. Darüber hinaus steht Ihnen der SQL Server Enterprise Manager als grafische Benutzeroberfläche für die meisten Verwaltungs- und Entwicklungsaufgaben unter SQL Server 2000 zur Verfügung. Die meisten Aufgaben, die Sie mit dem SQL Server Enterprise Manager erledigen können, lassen sich auch mit Transact-SQL-Anweisungen im Query Analyzer ausführen. Allerdings lässt es sich oftmals mit dem Enterprise Manager bequemer arbeiten (insbesondere wenn Sie mit SQL Server 2000 noch nicht so vertraut sind).

SQL Server Enterprise Manager ist ein sogenanntes Snap-In für die Microsoft Management Console (MMC). Die MMC ist eine allgemeine Oberfläche, die von Microsoft und anderen Anbietern als gemeinsame Verwaltungsoberfläche für die jeweiligen Produkte genutzt werden kann. Alle BackOffice-Produkte verwenden MMC als Hauptschnittstelle für die Verwaltung.

Einen Server registrieren

Wenn Sie den SQL Server Enterprise Manager starten, müssen Sie gegebenenfalls zunächst einen Server registrieren. Arbeiten Sie an einem Computer, auf dem SQL Server installiert wurde, ist Ihr lokaler SQL Server bereits während der Installation registriert worden. Falls Sie auf dem Windows 2000-Rechner, auf dem Sie SQL Server 2000 installiert haben, bereits eine Vorgängerversion installiert hatten, gibt es möglicherweise auch eine SQL Server-6.x-Gruppe. Um einen Server zu registrieren, erweitern Sie in der Konsolenstruktur den Knoten *Microsoft SQL Servers*; es erscheint die Standardgruppe *SQL Server-Gruppe*. Markieren Sie den Knoten *SQL Server-Gruppe* und wählen Sie dann aus dem

Menü VORGANG den Befehl NEUE SQL SERVER-REGISTRIERUNG. Daraufhin startet der SQL Server-Registrierungs-Assistent (siehe Abbildung 3.27).



Abbildung 3.27:
Der SQL Server-
Registrierungs-Assistent

Es empfiehlt sich, das Kontrollkästchen AB SOFORT MÖCHTE ICH DIESE AUFGABE OHNE HILFE EINES ASSISTENTEN DURCHFÜHREN einzuschalten; die Registrierung eines SQL Servers gehört nämlich zu den leichteren Aufgaben, die Sie schneller ohne den Assistenten durchführen können. Klicken Sie auf WEITER. Daraufhin gelangen Sie zum Standarddialogfeld für die Registrierung eines SQL Servers (siehe Abbildung 3.28).

Geben Sie den Namen Ihres Computers (wie in Abbildung 3.28 zu sehen ist) ein und wählen Sie die gewünschte Authentifizierungsart. Sie können entweder die Windows-Authentifizierung (falls der SQL Server auf einem Windows NT/2000-Computer läuft) oder die SQL Server-Authentifizierung (die in der Voreinstellung nur auf Windows 9x-Computern funktioniert) wählen. Markieren Sie die gewünschte Authentifizierungsart und füllen Sie bei SQL Server-Authentifizierung die zur Anmeldung nötigen Felder aus. Da Sie bereits die Standardinstanz im integrierten Windows-Modus installiert haben, müssen Sie die Option Windows-Authentifizierung wählen.

Bei SQL Server-Authentifizierung können Sie auch festlegen, dass bei jedem Verbindungsaufbau die Abfrage von Benutzername und Kennwort erfolgt. Wählen Sie diese Option insbesondere auf Windows 9x-Computern, um die Sicherheit zu erhöhen. Wenn die Sicherheit der Daten eine wichtige Rolle spielt, sollten Sie vorzugsweise Windows 2000 einsetzen; dieses Betriebssystem ist bedeutend sicherer als die Windows 9x-Systeme.

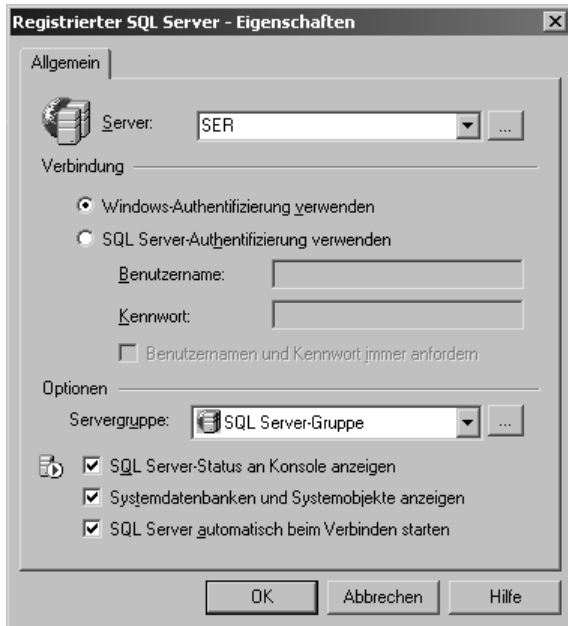


Abbildung 3.28:
Das Eigenschaftsdialogfeld des registrierten SQL Servers

Außerdem können Sie festlegen, ob Sie den Server in die Standardservergruppe aufnehmen oder eine neue Gruppe anlegen möchten. Diese Gruppierung ist lediglich ein organisatorisches Mittel Ihrer Arbeitsoberfläche. Die SQL Server im Netzwerk haben keine Ahnung von dieser Gruppierung. Im unteren Teil des Dialogfelds sind noch drei Optionen zu sehen, beispielsweise ob Sie die Systemdatenbanken und Systemobjekte anzeigen oder SQL Server automatisch beim Verbinden starten möchten. Das Buch geht davon aus, dass alle drei Kontrollkästchen wie in der Voreinstellung aktiviert sind.

Klicken Sie auf OK – das war alles! Sie haben soeben die erste Registrierung für den Enterprise Manager erstellt.



Natürlich haben Sie gerade die Fehlermeldung »Ein Server mit diesem Namen ist bereits vorhanden« erhalten, oder nicht? Bei der Installation von SQL Server registriert nämlich das Setupprogramm automatisch Ihre lokalen Installationen in Ihrer Kopie von Enterprise Manager. Bei Ihren lokalen Kopien brauchen Sie die Server also nicht zu registrieren, sie sind bereits bekannt. Allerdings müssen Sie die Registrierung bei allen Remotekopien durchführen.

Die Arbeitsweise des Enterprise Managers

Schließen Sie das Dialogfeld zur Registrierung, indem Sie auf die Schaltfläche ABBRECHEN klicken. Im linken Fensterbereich des Enterprise Managers – der so genannten Konsolenstruktur – ist zu sehen, dass Ihre Server bereits registriert sind, wenn Sie die SQL Server-Gruppe und dann jeden Server erweitern (siehe Abbildung 3.29). Das Serversymbol ist mit einem grünen Dreieck gekennzeichnet, wenn zu diesem Server eine Verbindung besteht. In Abbildung 3.29 ist zu sehen, dass sowohl die Verbindung zur Standardinstanz als auch zur benannten Instanz von SQL Server auf dem lokalen Computer (SER) hergestellt ist.

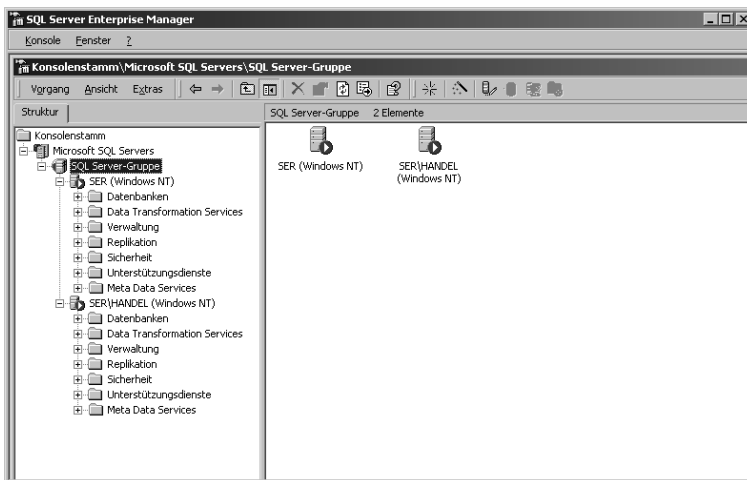


Abbildung 3.29:
Enterprise Manager
mit den registrierten
Servern

Innerhalb eines jeden Servers enthält der Ordner *Datenbanken* eine Liste der auf dem jeweiligen Server installierten Datenbanken. Der Ordner *Data Transformation Services* zeigt alle von Ihnen eingerichteten DTS-Pakete sowie alle Meta Data Services-Pakete. Der Ordner *Verwaltung* zeigt verschiedene administrative Instrumente für SQL Server. Im Ordner *Replikation* finden Sie alle Änderungen, die Sie an der Replikationskonfiguration Ihres SQL Servers vorgenommen haben. Der Ordner *Sicherheit* erlaubt es, die Sicherheitseinstellungen in SQL Server zu ändern. Schließlich steuern Sie über den Ordner *Unterstützungsdienste* andere SQL Server-bezogene Dienste auf Ihrem Computer, zum Beispiel den Distributed Transaction Coordinator und die Volltextsuche. Abbildung 3.30 zeigt einen Ausschnitt der Konsolenstruktur, wobei alle Ordner des Servers SER erweitert sind. Wenn Sie einen Server markieren, erscheinen Informationen über diesen Server als HTML-Seite im rechten Teil – dem Detailbereich – des Enterprise Managers.

Erweitern Sie nun den Datenbankordner *pubs* und markieren Sie den Knoten für diese Datenbank in der Konsolenstruktur. Im Detailbereich erscheint daraufhin eine neue Webseite (siehe Abbildung 3.31). Der Enterprise Manager zeigt Containerobjekte norma-

erweise in der Konsolenstruktur und den Inhalt der Containerobjekte oder andere Informationen im Detailbereich an.

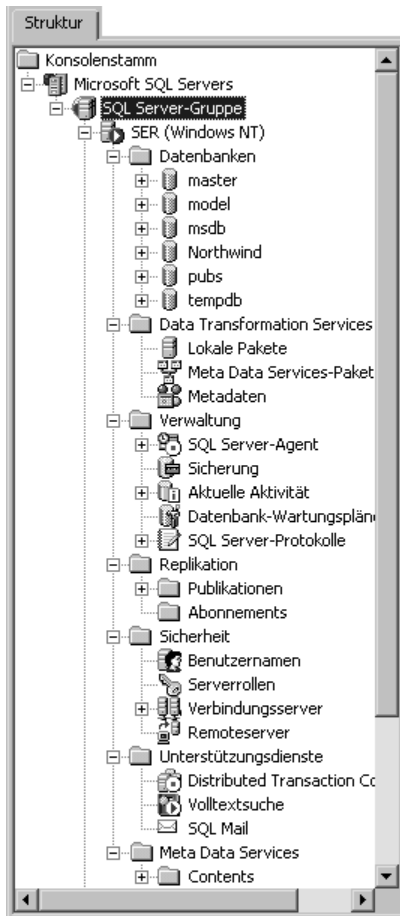


Abbildung 3.30:
Der linke Bereich (die Konsolenstruktur) des Enterprise Managers mit erweiterten Ordnern



Bei einigen Dialogfeldern gelangen Sie zu weiteren Dialogfeldern oder Fenstern. Klicken Sie zum Beispiel mit der rechten Maustaste auf den Datenbankordner *pubs* und wählen Sie EIGENSCHAFTEN im erscheinenden Kontextmenü. Über dem Enterprise Manager öffnet sich damit ein neues Dialogfeld (siehe Abbildung 3.32). In derartigen eigenständigen Dialogfeldern sind oftmals so genannte *Eigenschaftsblätter* (Dialogfelder mit Beschreibungen zu den Eigenschaften eines Objekts) realisiert. Klicken Sie auf ABBRECHEN, um das Eigenschaftsblatt der Datenbank *pubs* zu verlassen, ohne eventuelle Änderungen zu übernehmen.

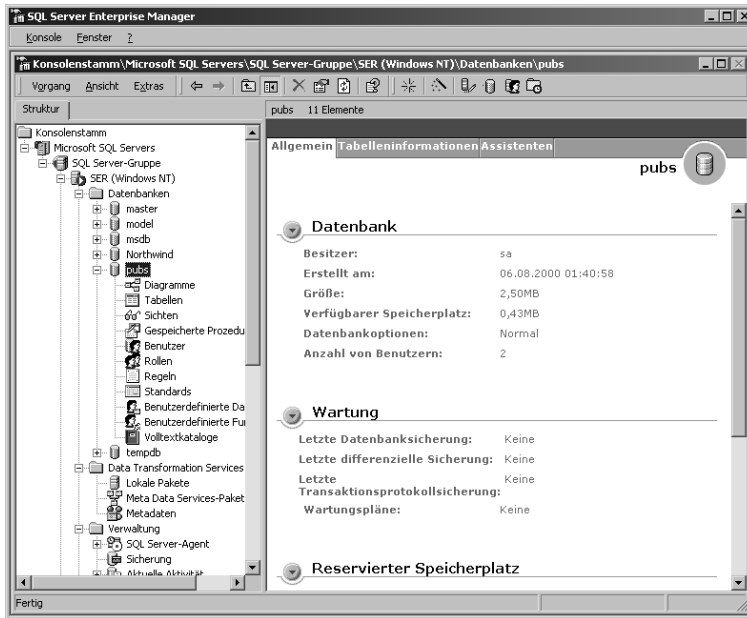


Abbildung 3.31:
Eigenschaften der
Datenbank *pubs* im
rechten Bereich
(Detailbereich) des
Enterprise Mana-
gers

Ebenfalls erwähnenswert sind die Menüoptionen. Die verfügbaren Optionen ändern sich je nachdem, was Sie markiert haben. Zum Beispiel erscheinen die in Abbildung 3.33 dargestellten Optionen, wenn die Datenbank *pubs* markiert ist und Sie das Menü EXTRAS wählen. Über die Bedeutung dieser Menübefehle brauchen Sie sich im Moment noch keine Gedanken zu machen, im Verlauf dieses 21-Tage-Kurses lernen Sie alles genau kennen.

Die Erklärungen zum Enterprise Manager könnten weitere 20 bis 30 Seiten füllen. Sie sollten aber bereits jetzt genug wissen, um dieses Werkzeug für die anstehenden Aufgaben einsetzen zu können. Mit dem Enterprise Manager haben Sie ab jetzt in diesem Kurs jeden Tag zu tun, so dass Sie sich am Tag 21 als Enterprise Manager-Experte bezeichnen können!

Daten importieren und exportieren

Diese Menüauswahl startet den DTS-Import/Export-Assistenten, den Sie am Tag 19 näher kennen lernen werden.

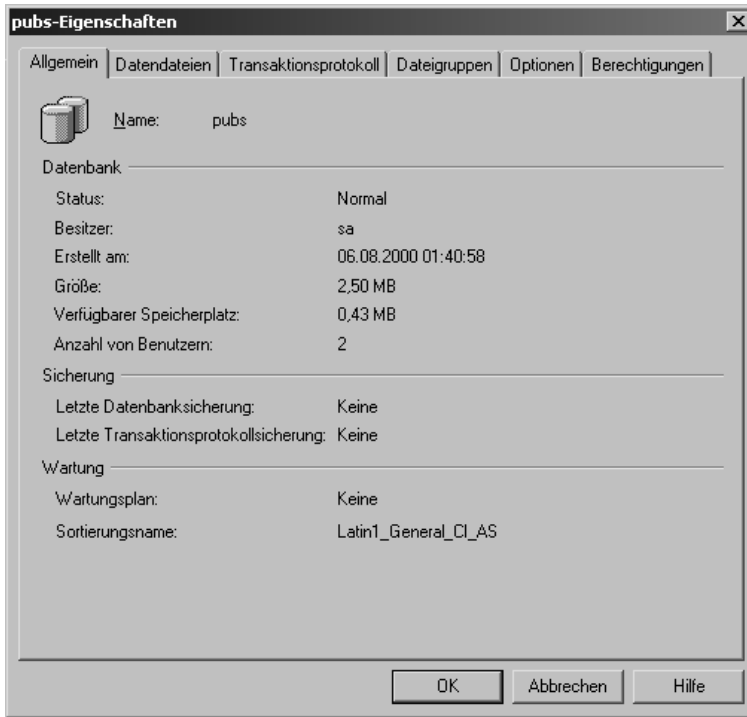


Abbildung 3.32:
Die Eigenschaften
der Datenbank pubs

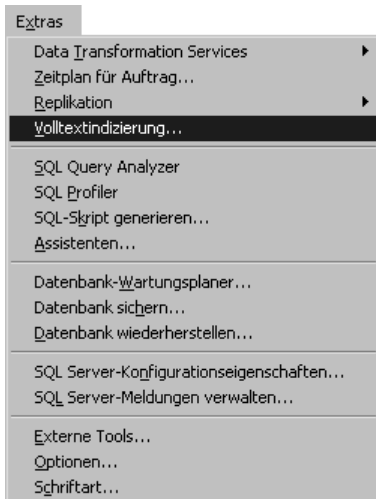


Abbildung 3.33:
Menüoptionen in SQL Server Enterprise Manager

XML-Unterstützung von SQL in IIS konfigurieren

Über diesen Menüpunkt können Sie Ihren Webserver so einrichten, dass direkte Abfragen nach SQL Server mittels XML (Extensible Markup Language) möglich sind. Auf diese Option geht Lektion 21 näher ein.

Profiler

Das Dienstprogramm SQL Profiler erlaubt es Ihnen, alle Aktivitäten Ihres SQL Servers zu überwachen. Sie können den Profiler auch verwenden, um die Leistung Ihrer Anwendungen zu steigern, indem Sie z.B. die Ausführungspläne untersuchen, nach denen SQL Server Abfragen bearbeitet. Der Profiler bietet Ihnen ausgefeilte Filtermechanismen, so dass sich die Verfolgung nur auf bestimmte Benutzer, Anwendungen oder Befehlstypen einschränken lässt. Es gibt etwa 100 verschiedene Ereignisse, die Sie überwachen können. Auf den Profiler geht Tag 20 näher ein.

3.2 Die Programmgruppe Microsoft SQL Server – Versionsumstellung

Die zweite Programmgruppe existiert nur in Installationen unter Windows NT und Windows 2000 und enthält drei mögliche Optionen: SQL Server-Aktualisierungs-Assistent, SQL Server 6.x deinstallieren und gegebenenfalls einen Eintrag für den Wechsel zu SQL Server 6.5.

SQL Server-Aktualisierungs-Assistent

Den SQL Server-Aktualisierungs-Assistenten haben Sie bereits gestern im Rahmen der SQL Server-2000- Installation kennen gelernt. Der SQL Server-Aktualisierungs-Assistent hilft Ihnen auf Windows NT-Computern bei der Umstellung von SQL Server 6.5-Datenbanken und Servereigenschaften auf SQL Server 2000. Sie können eine einzelne Datenbank, mehrere Datenbanken oder Server- und Aufgabeneigenschaften von einer Vorgängerversion des SQL Servers aktualisieren.

SQL Server 6.5 und SQL Server 7.0 sind die einzigen Vorgängerversionen, für die eine Aktualisierung möglich ist. Frühere Versionen werden nicht unterstützt. Den SQL Server-Aktualisierungs-Assistenten finden Sie nur auf Windows NT/2000-Systemen, da die Version 6.5 von SQL Server nur unter Windows NT bzw. Windows 2000 läuft.

Der Aktualisierungsvorgang belässt Ihre SQL Server 6.5-Datenbanken an ihrem ursprünglichen Speicherort. Wenn Sie die Aktualisierung durchführen, werden die Daten aus SQL Server 6.5 nach SQL Server 2000 exportiert und neue Datenbanken und Dateien unter SQL Server 2000 angelegt. Die Aktualisierung können Sie entweder von Ihrer vorherigen Version auf demselben Rechner durchführen oder Sie können SQL Server 2000 auf einem zweiten Rechner installieren und die Aktualisierung dann zu dem neuen SQL Server 2000-Computer vornehmen.

Microsoft SQL Server 6.5

Dieser Eintrag existiert nur auf Systemen, auf denen bereits SQL Server 6.5 installiert war. Wenn Sie diesen Eintrag wählen, schalten Sie wieder auf die SQL Server 6.5-Installation zurück. Das Startmenü ändert sich entsprechend und zeigt die SQL Server 6.5-Dienstprogramme an (obwohl Ihre SQL Server 2000-Tools ebenfalls verfügbar sind). Die vorherige Version von SQL Server ist damit zur Ausführung bereit. Der Eintrag zum Wechsel der Version im Startmenü ist in *SQL Server 2000* umbenannt worden. Wenn Sie diese Option wählen, kehren Sie zu SQL Server 2000 zurück und die SQL Server 6.5-Tools werden erneut ausgeblendet.



Diese Option gilt nur für die Umschaltung zwischen SQL Server 6.5 und einer Standardinstanz von SQL Server 2000. Wie bereits gestern angemerkt, können Sie SQL Server 6.5, SQL Server 7.0 oder SQL Server 2000 als eine Standardinstanz betreiben und dazu bis zu 16 benannte Instanzen von SQL Server gleichzeitig ausführen.

SQL Server 6.x deinstallieren

Auch dieser Eintrag existiert nur auf Systemen, auf denen eine vorherige Version von SQL Server installiert ist. Wenn Sie diesen Eintrag auswählen, deinstallieren Sie die alte SQL Server-Version. Dies ist ein angepasstes Deinstallationsprogramm, das Ihre vorherige Version des SQL Servers sicher entfernt.



Entfernen Sie die alte Version von SQL Server 6.5 nicht durch Ausführen des Setupprogramms. Nur mit dem speziellen Deinstallationsprogramm, das mit SQL Server 2000 installiert wird, lässt sich die vorherige Version sicher entfernen, ohne die SQL Server 2000-Installation zu gefährden.

3.3 Weitere SQL Server Tools und Utilities

Verschiedene Dienstprogramme haben im Start-Menü keinen Eintrag. Dennoch können sie manchmal unverzichtbar sein. Man kann diese Programme in die Kategorien Verbindungswerkzeuge, Serverdiagnose und Wartungswerkzeuge aufteilen.

Verbindungswerkzeuge

Verbindungswerkzeuge sind die Kommandozeilenprogramme, die eine Abfrageschnittstelle oder die Möglichkeit zum Testen des Netzwerks bieten. Zunächst untersuchen wir die Tools `OSQL` und `ISQL`, mit denen Sie die Verbindung zu SQL Server herstellen und Transact-SQL-Anweisungen ausführen können. Sie stellen das kommandozeilenorientierte Pendant zum SQL Query Analyzer dar. Die zweite Gruppe von Werkzeugen, `makepipe/readpipe` und `ODBCPing`, dient dazu, die Verbindung zu SQL Server zu testen.

OSQL

Das Programm `OSQL.exe` bietet eine Abfrageschnittstelle zu SQL Server. Die Verbindung zu SQL Server stellt das Dienstprogramm mittels ODBC her. Mit diesem Programm können Sie beispielsweise Stapelabfragen zur Unterstützung von Produktionsaufgaben durchführen. In Windows-Befehlsdateien können Sie `OSQL` aufrufen und Transact-SQL-Anweisungen ausführen, um in Ihren Datenbanken neue Daten hinzuzufügen, Daten zu ändern oder Daten zu löschen. Sie können auch Skripts erstellen (so wie Sie es im Abschnitt zum SQL Server Query Analyzer gesehen haben) und sie ausführen lassen, ohne mit einer grafischen Benutzeroberfläche zu arbeiten. `OSQL` lässt sich mit zwei unterschiedlichen Sätzen von Optionen aufrufen:



```
osql [-L] [-?]
```

Die Parameter haben folgende Bedeutung:

- `-L` zeigt eine Liste aller im Netzwerk gefundenen SQL Server an.
- `-?` ist die Standardanforderung für diese Syntaxliste.



```
osql {{-U Benutzername [-P Kennwort]] [-E]}
      [-S Servername[Instanzname]] [-e] [-p] [-n] [-i] [-b] [-O] [-R]
      [-d Datenbanknamen verwenden]
      [-q "Befehlszeilenabfrage"]
      [-Q "Befehlszeilenabfrage"]
```

```
[-c Befehlsende] [-h Header] [-w Spaltenbreite]
[-s Spaltentrennzeichen] [-t Abfragetimeout] [-m Fehlerebene]
[-r {0 | 1}] [-H Hostname] [-i Eingabedatei]
[-o Ausgabedatei] [-a Paketgröße] [-l Anmeldungstimeout]
[-D DSN]
```

Die Parameter haben folgende Bedeutung:

- **-U** ist ihr Anmelde-name bei SQL Server.
- **-P** ist das Kennwort für die Anmeldung bei SQL Server. Wenn Sie kein Kennwort eingeben, fordert Sie OSQL beim Programmstart zur Eingabe auf.
- **-E** fordert eine Verbindung im Windows-Authentifizierungsmodus an, so dass Sie die Parameter **-U** und **-P** weglassen können.
- **-S** gibt an, zu welchem Server OSQL eine Verbindung herstellen soll. Wenn kein Server spezifiziert ist, stellt OSQL die Verbindung zur lokalen Standardinstanz her.
- **-e** wiederholt den Text jeder ausgeführten Anweisung in der Ausgabe dieser Anweisung.
- **-p** gibt die Leistungsstatistik für die Abfragen aus.
- **-n** entfernt die Nummerierung und das Symbol für die Eingabeaufforderung (>), die OSQL normalerweise in jeder Zeile ausgibt, wenn Sie mehrere Anweisungen eingeben.
- **-I** aktiviert die QUOTED_IDENTIFIER-Verbindungsoption. Damit legen Sie fest, ob Elemente in Anführungszeichen als Zeichenfolgenkonstanten (»hello«) oder als Spalten-/Tabellennamen gelten.
- **-b** gibt an, dass OSQL beendet wird und einen DOS ERRORLEVEL-Wert zurückgibt, wenn ein Fehler auftritt. OSQL liefert den Wert 1 zurück, wenn eine Fehlermeldung mit einem Schweregrad von 10 oder höher auftritt.
- **-O** weist OSQL an, das Verhalten von ISQL zu emulieren, um Kompatibilität zur Vorgängerversion herzustellen.
- **-u** weist OSQL an, dass die Ergebnisse der Abfrage im Unicodeformat zurückzugeben sind.
- **-R** gibt an, dass die Clienteneinstellungen bei der Umwandlung von Währungs-, Datums- oder Zeitdaten verwendet werden sollen.
- **-d** gibt an, zu welcher Datenbank nach dem Verbindungsaufbau gewechselt werden soll.
- **-q** teilt OSQL mit, dass die in Anführungszeichen eingeschlossene Abfrage beim Start ausgeführt werden soll. OSQL läuft nach der Ausführung der Abfrage weiter. Falls die

Abfrage Anführungszeichen enthält, schließen Sie die Abfrage in doppelte Anführungszeichen ein und verwenden innerhalb der Abfrage einfache Anführungszeichen.

- **-Q** teilt OSQL mit, dass die in Anführungszeichen eingeschlossene Abfrage beim Start ausgeführt und das Programm anschließend beendet werden soll.
- **-c** legt das *Batchtrennzeichen* fest. In SQL Server-Skripts veranlasst das Schlüsselwort GO, dass die Abfragen als Gruppe (als so genannter *Stapel* oder *Batch*) an SQL Server geschickt werden. Dieses Verhalten können Sie überschreiben und ein eigenes Trennzeichen für OSQL festlegen, um die Abfragen an SQL Server abzuschicken. Normalerweise sollten Sie diese Option nicht überschreiben.
- **-h** gibt an, wie viele Leerzeilen zwischen den Spaltenüberschriften und den Abfrageergebnissen eingefügt werden sollen. Wenn Sie **-h-1** angeben, werden keine Spaltenüberschriften für Ihre Abfrage erzeugt.
- **-w** erlaubt es, für die Breite der Ausgabe einen anderen Wert als die Standardeinstellung 80 Zeichen festzulegen.
- **-s** ändert das Trennzeichen zwischen zwei Spalten (Vorgabewert ist ein Leerzeichen).
- **-t** gibt an, wie lange OSQL warten soll, bis die Verbindung zum Server als gescheitert gilt.
- **-m** ändert das Fehlermeldungsverhalten. Die Syntax ist **-m n**, wobei *n* den Schweregrad des Fehlers angibt. Den Schweregrad von Fehlermeldungen erläutert Lektion 20.
- **-r** gibt an, dass Fehlermeldungen nach STDERR geschickt werden. Wenn Sie diesen Wert auf 0 setzen, werden nur Fehlermeldungen des Schweregrades 17 oder höher nach STDERR geschickt, eine 1 bedeutet, dass alle Meldungen nach STDERR gehen.
- **-H** gibt den Namen der Arbeitsstation an, wenn Sie ihn an SQL Server senden wollen.
- **-i** enthält den Dateinamen und Pfad eines Transact-SQL-Skripts, das Sie ausführen wollen.
- **-o** gibt die Datei an, in der die Ergebnisse Ihres Skripts gespeichert werden sollen. Falls die Eingabedatei im Unicodeformat vorliegt, wird auch die Ausgabedatei in diesem Format erstellt.
- **-a** gibt die im Netzwerk zu verwendende Paketgröße an.
- **-l** (ein kleines L, nicht die Ziffer 1) teilt OSQL den Wert für das Anmeldungstimeout mit (wie lange OSQL wartet, bevor angenommen wird, dass der Server nicht läuft).
- **-D** teilt OSQL den Namen einer SQL Server-Datenquelle mit, die im Programm DATENQUELLEN (ODBC) im Ordner VERWALTUNG (oder denjenigen, die per Programm erstellt wurden) definiert ist. Diese Option funktioniert nur für SQL Server-Datenquellen.

Wenn Sie die weiter vorn in dieser Lektion mit dem SQL Server Query Analyzer ausgeführten Befehle von der Befehlszeile ausführen, erscheint Folgendes:



Bei diesem (und den meisten) Befehlszeilenwerkzeugen können Sie Parameter entweder mit einem Minuszeichen (-) oder einem Schrägstrich (/) spezifizieren. Deshalb können Sie zum Beispiel beim Parameter für den Servernamen entweder `osql /s` oder `osql -s` schreiben.



```
C:\>osql /Usa
Kennwort:
1> use pubs
2> select * from authors
3> go
```



Daraufhin erscheinen die Abfrageergebnisse, die folgendermaßen enden:



```
893-72-1158 McBadden                                Heather
              707 448-4982 301 Putnam
              Vacaville                CA    95688        0
899-46-2035 Ringer                                  Anne
              801 826-0752 67 Seventh Av.
              Salt Lake City            UT    84152        1
998-72-3567 Ringer                                  Albert
              801 826-0752 67 Seventh Av.
              Salt Lake City            UT    84152        1

(23 rows affected)
1>
```



Das Schlüsselwort `GO` ist das Signal für `OSQL`, die bis dahin eingegebene(n) Anweisung(en) auszuführen. Die `1>` am Ende ist das Zeichen dafür, dass `OSQL` dazu bereit ist, weitere Transact-SQL-Anweisungen entgegenzunehmen. Außer den bisher beschriebenen Optionen sei noch auf die Befehle `Reset` und `Exit` hingewiesen:

- `Reset` stoppt jeden Befehl und bringt Sie wieder zur Eingabeaufforderung `1>` zurück.
- `Exit` beendet `OSQL`.

Eine Stapelverarbeitungsdatei könnte folgendermaßen aussehen:

```
osql -E -I"D:\Programme\Microsoft SQL Server\mssql\runquery.sql"
      -oc:\results.txt
```

In die Eingabedatei *runquery.sql* schreiben Sie Ihre Abfragen, die Ergebnisse erscheinen in der Ausgabedatei *results.txt*. Diese Anweisung stellt eine Verbindung per Windows-Authentifizierung zu SQL Server her. Sie können diese Stapelverarbeitungsdatei jederzeit erneut ausführen und sogar zeitgesteuert ausführen lassen.

ISQL

ISQL.exe ist das befehlszeilenorientierte Abfrageprogramm früherer Versionen von SQL Server. Es verwendet die Netzbibliothek DB-Library, um die Verbindung zu SQL Server herzustellen. Deswegen unterstützt es die neuen Features von SQL Server 2000 nicht; das gilt auch für die Darstellung in Unicode. Aus diesem Grund sollten Sie *isql* nicht mehr in bereits vorhandenen Stapeldateien verwenden und vor allem nicht für Neuentwicklungen einsetzen.



```
isql -U login_id [-e] [-E] [-p] [-n] [-d db_name] [-q "query"]
[-Q "query"] [-c cmd_end] [-h headers] [-w column_width]
[-s col_separator] [-t time_out] [-m error_level] [-L] [-?]
[-r {0 | 1}] [-H wksta_name] [-P password] [-S server_name]
[-i input_file] [-o output_file] [-a packet_size]
[-b] [-O] [-l time_out] [-x max_text_size]
```

Die meisten dieser Parameter ähneln denen von *osql.exe*. Generell sollten Sie dieses Dienstprogramm unter SQL Server 2000 nur noch einsetzen, um bestehende Jobs aus früheren Versionen zu unterstützen.

Makepipe/Readpipe

Mit diesen Dienstprogrammen können Sie die Integrität des Named Pipe-Dateisystems überprüfen. Da SQL Server 2000 Named Pipes normalerweise nicht mehr verwendet, werden diese Dienstprogramme nicht mehr in Standardeinstellung installiert. Falls Probleme mit Named Pipes auftreten, können Sie die Programme aus dem Verzeichnis *x86\binn* der Installations-CD in das Verzeichnis *\binn* für Ihren SQL Server kopieren. Weitere Hinweise zu diesen Programmen finden Sie in der Onlinedokumentation.

ODBCPing

Mit dem Dienstprogramm *ODBCPing* lässt sich testen, ob eine funktionierende ODBC-Verbindung zum SQL Server hergestellt werden kann.



```
odbcping [-?] | [{-Sserver | -Ddatasource} [-Ulogin] [-Ppassword]]
```



Die Parameter haben folgende Bedeutung:

- -S ist der Server, zu dem Sie eine Verbindung herstellen wollen.
- -D ist der Name einer ODBC-Datenquelle.
- -U ist der Benutzername für die Anmeldung bei SQL Server.
- -P ist das Kennwort für den von Ihnen gewählten Benutzernamen.

Standardmäßig installiert SQL Server 2000 dieses Programm nicht. Allerdings befindet es sich im Verzeichnis *x86\bin* der SQL Server 2000-CD. Kopieren Sie das Dienstprogramm in das Verzeichnis *\bin* des SQL Server-Pfades (zum Beispiel *D:\Programme\Microsoft SQL Server\mssql\bin* für die Standardinstanz) und probieren Sie dann Folgendes aus, um diese Verbindung zu testen:

Öffnen Sie ein Befehlszeilenfenster und geben Sie den folgenden Befehl ein, um sich mit Ihrer Kopie von SQL Server zu verbinden. Ersetzen Sie dabei *SER* durch den Namen Ihres Servers.



```
odbcping -SSER -Usa -P
```

Der Server sollte etwa folgende Antwort bringen:



```
CONNECTED TO SQL SERVER
```

```
ODBC SQL Server Driver Version: 03.80.0194
```

```
SQL Server Version: Microsoft SQL Server 2000 - 8.00.194 (Intel X86)  
Aug 6 2000 00:57:48  
Copyright (c) 1988-2000 Microsoft Corporation  
Enterprise Edition on Windows NT 5.0 (Build 2195: )
```

Diese Meldung besagt, dass ODBC funktioniert. Sie können auch eine Verbindung zu einem ODBC DSN (Data Source Name) herstellen, d.h. einer voreingestellten Konfiguration zu einem Server, die Sie über die Anwendung DATENQUELLEN (ODBC) im Ordner VERWALTUNG einrichten.



Das Tool ODBCPing unterstützt keine Verbindungen mit integrierter Sicherheit, so dass Sie es nicht für eine Standardinstallation von SQL Server, für die nur der integrierte Sicherheitsmodus festgelegt ist (beispielsweise Ihre Standardinstanz) verwenden können.

Server-Diagnose- und -Wartungswerkzeuge

Die Server-Diagnose- und -Wartungswerkzeuge gehören zu einer Gruppe von Dienstprogrammen, auf die Sie nach der Installation von SQL Server 2000 hin und wieder zurückgreifen müssen.

SQLServr.exe

SQLServr.exe ist das eigentliche Programm, das SQL Server ausführt (bei einer Standardinstanz den Dienst MSSQLServer). Allerdings können Sie SQL Server bei Bedarf auch von der Eingabeaufforderung starten. Normalerweise geht man so vor, wenn SQL Server im so genannten Einzelbenutzermodus laufen muss. Mehr dazu erfahren Sie am Tag 8, wenn es um die Wiederherstellung von Datenbanken geht.



```
sqlservr [-sinstance_name] [-c] [-f] [-dmaster_path]
          [-lmaster_log_path] [-m] [-n] [-eerror_log_path]
          [-pprecision_level] [-Ttrace#] [-v] [-x]
          [-g memory_to_reserve] [-O] [-y error_number]
```

Die Parameter haben folgende Bedeutung:

- *-s* spezifiziert den Namen der zu startenden Instanz. Geben Sie diesen Parameter nicht an, wenn Sie eine Standardinstanz von SQL Server starten. Andernfalls müssen Sie diese Option spezifizieren, selbst wenn Sie *SQLServr.exe* aus dem Verzeichnis *\binn* einer benannten Instanz aufrufen.
- *-c* gibt an, dass SQL Server als Programm und nicht als Windows 2000-Dienst gestartet wird. Dadurch verkürzt sich die erforderliche Zeit für das Starten von SQL Server an einer Eingabeaufforderung.
- *-f* gibt an, dass SQL Server im minimal konfigurierten Modus startet. Diese Option müssen Sie verwenden, wenn Sie manuell Konfigurationsänderungen vorgenommen haben, die verhindern, dass SQL Server normal startet. Dies ist eine Notfalloption, die es Ihnen ermöglicht, Ihre Fehler zu korrigieren.
- *-d* gibt den vollgekennzeichneten Pfad der *master*-Datenbankdatei an. Falls Sie hier nichts angeben, werden die während der Installation erstellten Vorgabewerte aus der Windows-Registrierung verwendet. Die Vorgabe lautet *LW:\Programme\Microsoft SQL Server\mssql\data\master.mdf* (wobei LW: für das von Ihnen bei der Installation gewählte Laufwerk steht).
- *-l* gibt den vollgekennzeichneten Pfad der Transaktionsprotokolldatei der *master*-Datenbank an. Falls Sie hier nichts angeben, werden die während der Installation erstellten Vorgabewerte aus der Windows-Registrierung verwendet. Die Vorgabe lautet

LW:\Programme\Microsoft SQL Server\mssql\data\master.ldf (wobei *LW*: für das von Ihnen bei der Installation gewählte Laufwerk steht).

- *-m* gibt an, dass SQL Server im Einzelbenutzermodus startet, so dass nur ein Benutzer eine Verbindung zu SQL Server herstellen kann. Diese Methode verwendet man bei Wiederherstellungsaufgaben nach dem Verlust kritischer Datendateien (z.B. bei der Wiederherstellung der *master*-Datenbank aus einer Sicherung).
- *-n* verhindert die Aufzeichnung von Fehlern im Windows NT/2000-Anwendungsprotokoll (nicht empfohlen).
- *-e* gibt den vollgekennzeichneten Pfad der Fehlerprotokolldatei an. Die Vorgabe lautet *LW:\Programme\Microsoft SQL Server\mssql\log\errorlog* (wobei *LW*: für das von Ihnen bei der Installation gewählte Laufwerk steht).
- *-p* spezifiziert die maximale Genauigkeit für die Datentypen *decimal* und *numeric*. Standardmäßig erlaubt SQL Server für diese Datentypen bis zu 38 signifikante Stellen. Die Anzahl der Stellen können Sie mit diesem Parameter von 1 bis 38 festlegen. Verwenden Sie diese Option, wenn Sie keine derart großen Zahlen in SQL Server benötigen oder auf Abwärtskompatibilität zu vorherigen SQL Server-Versionen achten müssen.
- *-T* spezifiziert ein Ablaufverfolgungsflag für SQL Server. Ein *Ablaufverfolgungsflag* ist ein numerischer Schalter, mit dem sich in SQL Server ein spezielles (vom Standardfall abweichendes) Verhalten einstellen lässt. Normalerweise verwenden Sie diese Option nur, um Informationen für Supporttechniker von SQL Server bereitzustellen. Um mehrere Flags anzugeben, verwenden Sie *-T* mehrfach.
- *-v* zeigt die Versionsnummer von *sqlservr.exe* an.
- *-x* schaltet die Leistungsstatistiken von SQL Server ab (nicht empfohlen).
- *-g* spezifiziert die Größe des Arbeitsspeichers, der für andere Anwendungen (wie zum Beispiel erweiterte gespeicherte Prozeduren) reserviert werden soll, die in SQL Server laufen. Diese Konfigurationsoption sollten Sie nur ändern, wenn Sie explizit dazu aufgefordert werden.
- *-O* schaltet DCOM und verteilte Abfragen ab. Setzen Sie diese Option, wenn Sie wissen, dass Sie keine verteilten Abfragen ausführen.
- *-y* übernimmt eine Fehlernummer als Parameter. Stellt SQL Server 2000 eine Fehlermeldung fest, die in dieser Option angegeben ist, schreibt es die Ablaufverfolgung des Symptomstacks in das Fehlerprotokoll. Auch diese Option sollten Sie nur verwenden, wenn Sie ein SQL Server-Supporttechniker dazu auffordert.

Halten Sie den SQL Server beispielsweise mit dem SQL Server-Dienst-Manager oder dem SQL Server Enterprise Manager an und öffnen Sie dann eine Eingabeaufforderung.

Wechseln Sie in das Verzeichnis `\binn` der Standardinstanz und geben Sie folgenden Text ein:



`C:\Programme\Microsoft SQL Server\MSSQL\Binn>sqlservr.exe -c`

SQL Server startet von dieser Eingabeaufforderung und meldet sich wie in Abbildung 3.34 dargestellt, wenn der Server dazu bereit ist, Ihre Anmeldung mit einem der Abfrage-Tools entgegenzunehmen.

```

Eingabeaufforderung - sqlservr.exe -c
F:\Programme\Microsoft SQL Server\MSSQL\Binn>sqlservr.exe -c
2000-12-25 10:47:44.91 server Microsoft SQL Server 2000 - 8.00.194 (Intel X86)
Aug 6 2000 00:57:48
Copyright (c) 1988-2000 Microsoft Corporation
Enterprise Edition on Windows NT 5.0 (Build 2195: )
2000-12-25 10:47:44.92 server Copyright (C) 1988-2000 Microsoft Corporation.
2000-12-25 10:47:44.92 server Alle Rechte vorbehalten.
2000-12-25 10:47:44.92 server Serverprozess-ID ist 680.
2000-12-25 10:47:44.93 server Protokolliert SQL Server-Meldungen in Datei 'F:\Programme\Microsoft SQL Server\MSSQL\log\ERRORLOG'.
2000-12-25 10:47:44.98 server SQL Server startet mit Prioritätsklasse 'normal' (1 CPU vorgefunden).
2000-12-25 10:47:46.24 server SQL Server wurde für die Verarbeitung in thread-Modus konfiguriert.
2000-12-25 10:47:46.25 server Die dynamic-Sperrenreservierung wird verwendet. [2500] Sperrenblöcke, [5000] Sperrenbesitzerblöcke.
2000-12-25 10:47:46.28 server Initialisiert Distributed Transaction Coordinator.
2000-12-25 10:47:46.93 server Failed to obtain TransactionDispenserInterface: Result Code = 0x8004d01b
2000-12-25 10:47:46.99 spid3 Startet Datenbank 'master'.
2000-12-25 10:47:47.59 server Verwendet 'SSNETLIB.DLL', Version '8.0.194'.
2000-12-25 10:47:47.59 spid5 Startet Datenbank 'model'.
2000-12-25 10:47:47.63 spid3 Servername ist 'SER'.
2000-12-25 10:47:47.66 spid8 Startet Datenbank 'msdb'.
2000-12-25 10:47:47.66 spid9 Startet Datenbank 'pubs'.
2000-12-25 10:47:47.66 spid10 Startet Datenbank 'Northwind'.
2000-12-25 10:47:48.52 spid5 Löscht tempdb-Datenbank.
2000-12-25 10:47:51.81 spid5 Startet Datenbank 'tempdb'.
2000-12-25 10:47:52.66 spid3 Wiederherstellung abgeschlossen.
2000-12-25 10:47:55.46 server SQL Server überwacht TCP, Shared Memory, Named Pipes.
2000-12-25 10:47:55.46 server SQL Server überwacht 192.168.0.3:1433, 127.0.0.1:1433.
2000-12-25 10:47:55.47 server SQL Server ist bereit für Clientverbindungen
    
```

Abbildung 3.34: SQL Server bei Ausführung von einer Eingabeaufforderung

Um den Server anzuhalten, aktivieren Sie das Fenster der Eingabeaufforderung und drücken `[Strg] + [C]`. Daraufhin erscheint die Meldung:

Do you wish to shutdown SQL Server (Y/N)?

Tippen Sie `[Y]`, um SQL Server anzuhalten. Dann können Sie SQL Server als Dienst neu starten.

Das Dienstprogramm Rebuild Master (rebuildm.exe)

Das Rebuild Master-Dienstprogramm stellt die *master*-Datenbank und die Datenbanken *msdb*, *model*, *tempdb*, *pubs* und *Northwind* wieder her. Dieses Dienstprogramm benötigen Sie in folgenden Fällen:



- Eine Festplatte ist ausgefallen und Sie besitzen keine Sicherheitskopien.
- Sie wollen die Sortierreihenfolge oder den Zeichensatz ändern.
- Die *master*-Datenbank ist beschädigt worden.

Wenn Sie das Rebuild Master-Dienstprogramm ausführen, werden obige Datenbanken neu von Ihrer SQL Server-2000-CD erstellt und daraufhin Zeichensatz und Sortierreihenfolge ihren Angaben gemäß angepasst. Alle anderen Datenbanken (zusätzlich zu den oben erwähnten) sind nach diesem Vorgang für den SQL Server nicht mehr definiert und Sie müssen sie entweder neu erstellen oder sie erneut beim Server anmelden.

Die Wiederherstellung der *master*-Datenbank geschieht normalerweise in einem Katastrophenszenario; in diesem Zusammenhang lernen Sie am Tag 8 das Dienstprogramm Rebuild Master näher kennen.

Die Option Registrierung wiederherstellen

Die Option *Registrierung wiederherstellen* erlaubt es, die Registrierung für eine beschädigte Installation von SQL Server wiederherzustellen. Dabei werden einfach nur die Registrierungsschlüssel wiederhergestellt, die während des vorherigen Setups erstellt wurden.

Um diesen Prozess durchzuführen, starten Sie das Setupprogramm, wählen die Wartung der vorhandenen Installation, spezifizieren den Instanznamen (wenn Sie mit einer benannten Instanz arbeiten) und wählen im Dialogfeld INSTALLATIONS-AUSWAHL die Option *Erweiterte Optionen*. Im Dialogfeld ERWEITERTE OPTIONEN wählen Sie die Option *Registrierung wiederherstellen* und folgen dann den Anweisungen des Setupprogramms. Wählen Sie immer genau die gleichen Optionen wie beim ursprünglichen Setup, um die Registrierung in einen brauchbaren Zustand zurückzubringen.

SQLMaint.exe

Dieses Programm dient zur Durchführung routinemäßiger Datenbankwartungsaufgaben, einschließlich Datensicherung, Konsistenzprüfung und Indexwartung. Der SQL Server Enterprise Manager enthält den Datenbank-Wartungsplanungs-Assistenten. Dieser hilft Ihnen, Wartungspläne und Sicherungsaufgaben für Ihre Datenbanken zu konfigurieren, ohne dass Sie sich mit den genauen Abläufen hinter den Kulissen auskennen müssen. Der Assistent erstellt einen SQL Server-Auftrag, der die festgelegten Wartungstasks automatisch in den angegebenen Intervallen ausführt. Um die vielen Optionen des Programms müssen Sie sich momentan noch keine Gedanken machen. Wenn Sie sich in Lektion 18 mit dem SQL Server-Agenten beschäftigen, um regelmäßig wiederkehrende Aufgaben zu konfigurieren, sollten Sie auch die hier angegebene Syntax wiederholen und sich ansehen, wie Sie

mit diesem Dienstprogramm Wartungsaufgaben automatisch ausführen können. Die zahlreichen Parameter von *sqlmaint* lassen allerdings bereits vermuten, dass die manuelle Konfiguration etwas knifflig sein kann.



```
sqlmaint
[-?] |
[ [-S <Servername>[\<Instanzname>]]
  [-U <Benutzername> [-P <Kennwort>]]
  {
    [ -D <Datenbankname> | -PlanName <Name> | -PlanID <GUID>]
    [-Rpt <Textdatei>]
    [-To <Operatorname>]
    [-HtmlRpt <HTML-Datei> [-DelHtmlRpt <Zeitabschnitt>]]
    [-RmUnusedSpace <Minimum der DB in MB><% freier Speicherplatz>]
    [-CkDB | -CkDBNoIdx] [-CkAl | -CkAlNoIdx] [-CkCat]
    [-UpdSts] [-UpdOptiStats <% der Datenbank für Stichprobe>]
    [-RebldIdx <% freier Speicherplatz>] [-WriteHistory]
    [
      {-BkUpDB [<Sicherungspfad>] | -BkUpLog [<Sicherungspfad>] }
      {-BkUpMedia
        {FESTPLATTE [ [-DelBkUps <Zeitabschnitt>]
          [-CrBkSubDir ] [ -UseDefDir ]
        }
      }
      |BAND
      }
    [-BkUpOnlyIfClean]
    [-VrfyBackup]
  ]
]

<Zeitabschnitt> ::=
Zahl[minutes | hours | days | weeks | months]
```

Die Parameter haben folgende Bedeutung:

- -S bezeichnet den Namen des Zielservers (*Servername* bei einer Standardinstanz, *Server\Instanzname* bei einer benannten Instanz).
- -U ist der Benutzername, den Sie bei der Durchführung des Programms verwenden wollen.
- -P ist das dazugehörige Kennwort. Wenn Sie die Optionen -U und -P weglassen, wird eine vertraute Verbindung zu SQL Server hergestellt.
- -D ist der Name der Datenbank, die Sie warten wollen.

- *-Rpt* gibt an, wohin die Ausgaben (z.B. Fehlermeldungen) des Dienstprogramms zu schreiben sind. Geben Sie hier den vollständigen Pfad und Dateinamen an.
- *-To* gibt die Mail-Adresse an, an die eine Berichtskopie gesandt werden soll. Sie werden die Mail-Integration (SQL Mail) am Tag 19 näher kennen lernen.
- *-HTMLRpt* gibt an, wo eine Ausgabedatei in Form einer Webseite mit den Meldungen des Programms erstellt werden soll. Geben Sie hier einen vollständigen Pfad und Dateinamen an.
- *-RmUnusedSpace* teilt *sqlmaint.exe* mit, dass freier Speicherplatz aus der Datenbank entfernt werden soll, wenn dieser die in *Minimum der DB in MB* angegebene Größe überschreitet und die Datenbank so definiert ist, dass sie automatisch größer wird (was der Vorgabe für Datenbanken entspricht). Die Datenbank wird verkleinert, wobei der Prozentsatz an freiem Speicher, der im Parameter *% freier Speicherplatz* angegeben wurde, zurückbleibt. Somit sinkt, falls Sie für *% freier Speicherplatz* 10 angegeben haben, die Größe einer 100 Mbyte großen Datenbank, in der nur 50 Mbyte belegt waren, auf 55 Mbyte (5 Mbyte entsprechen 10% der belegten 50 Mbyte, deshalb bleiben 5 Mbyte freier Speicherplatz in der Datenbank erhalten).
- *-CkDB* | *-CkDBNoIdx* führen die DBCC-Anweisung CHECKDB aus. Diese Anweisung lernen Sie am Tag 7 kennen.
- *-CkAI* | *-CkAINoIdx* führen die DBCC-Anweisung CHECKALLOC aus. Mehr darüber erfahren Sie am Tag 7.
- *-CkTxtAI* ist die DBCC-Anweisung TEXTALL. Auch hier gilt, warten Sie Tag 7 ab.
- *-CkCat* ist die DBCC-Anweisung CHECKCATALOG. Sie wissen schon ... Tag 7.
- *-UpdSts* führt die Anweisung Update Statistics aus. Mit Statistiken befasst sich Tag 13.
- *-UpdOptiSts* führt die Anweisung Update Statistics mit einem zusätzlichen Parameter aus. Mit Statistiken befasst sich Tag 13.
- *-RebldIdx* führt die DBCC-Anweisung DBREINDEX aus, um den Füllfaktor der Indizes neu zu erstellen – auch hier gilt, siehe Tag 13.
- *-WriteHistory* zeichnet die Durchführung des Wartungsplans in einer Systemtabelle der *msdb*-Datenbank auf (in der Tabelle *sysdbmaintplan_history*).
- *-BkUpDB* | *-BkUpLog* gibt an, wo Ihre Sicherheitskopien gespeichert werden. Sie können entweder die komplette Datenbank oder nur das Transaktionsprotokoll sichern. Sicherheitskopien behandelt Tag 7.
- *-BkUpMedia* gibt an, ob Sie eine Sicherung auf Band oder auf Festplatte vornehmen.

- *-DelBkUps* gibt an, wie lange Ihre Sicherheitskopien erhalten bleiben, wenn Sie sie auf der Festplatte speichern.
- *-BkUpOnlyIfClean* gibt an, dass eine Sicherung nur vorgenommen werden soll, wenn die Datenbank unbeschädigt ist. Das ist die mächtigste Funktion dieses Dienstprogramms; mehr dazu am Tag 7.
- *-VrfyBackup* überprüft nach Abschluss des Sicherungsvorgangs die Lesbarkeit der Sicherung. Auch diesen Parameter behandelt Tag 7.

Zum Beispiel können Sie mit dem Befehl

```
sqlmaint -S ser -D master -CkDB -Rpt d:\maint.txt
```

die Ausführung eines DBCC-Befehls *CHECKDB* in der Datenbank *master* für die Standardinstanz von SQL Server veranlassen und einen Bericht in die Datei *maint.txt* im Stammverzeichnis von Laufwerk D schreiben.



Wir wollen hier nichts vor Ihnen verheimlichen, es ist jedoch so, dass die meisten dieser Optionen Ihnen erheblich mehr sagen werden, nachdem Sie dieses Buch gelesen haben, als sie es zum jetzigen Zeitpunkt tun. Deshalb steht hier immer wieder »Mehr darüber erfahren Sie an Tag x«.

SQLDiag.exe

Das Dienstprogramm SQLDiag dient dazu, einen Anruf beim SQL Server-Produktsupport vorzubereiten. SQLDiag sammelt Fehlerprotokolle, Konfigurationsinformationen des Servers, die SQL Server-Version (einschließlich aller angewandten Service Packs), Betriebssysteminformationen, Angaben über das Computersystem und andere nützliche Informationen für die Fehlerbehebung. Das Dienstprogramm schreibt alle diese Angaben in eine einzige Datei – *SQLDIAG.TXT* – und legt sie standardmäßig im Verzeichnis *ms-sql\log* ab.



```
sqldiag [-S Server][-U Benutzername][-P Kennwort][-E][-O Ausgabedatei]
```

In dieser Syntax bedeuten:

- *-S* ist der Name des Zielservers (oder *Server\Instanzname* bei einer benannten Instanz).
- *-U* ist der Benutzername, den Sie für die Ausführung des Dienstprogramms verwenden.
- *-P* ist das zum Benutzernamen gehörende Kennwort. Wenn Sie die Optionen *-U* und *-P* weglassen, wird eine vertraute Verbindung zu SQL Server hergestellt.



- -E spezifiziert, dass Sie eine Verbindung mittels integrierter Sicherheit herstellen möchten.
- -O spezifiziert den Pfadnamen und Dateinamen für die Berichtsdatei.

Drucken Sie diese Datei aus (oder speichern Sie sie an einem sicheren Ort), denn sie enthält viele wertvolle Informationen über Ihr System. Die Ausführung dieses Programms und die Bereitstellung dieser Informationen wird die Bearbeitung Ihrer Serviceanfragen durch den Microsoft Software Service sehr erleichtern.

BCP – ein eigener Komplex

BCP bedeutet Bulk Copy Program. Dieses Dienstprogramm lädt Daten aus einer Datei nach SQL Server oder exportiert Daten aus SQL Server in eine Datei. Einzelheiten zu diesem Dienstprogramm (einschließlich der Syntax) erfahren Sie am Tag 19.

3.4 Zusammenfassung

Microsoft liefert mit SQL Server 2000 die bisher umfangreichste Sammlung von Dienstprogrammen aus. Die gesamte Organisation lässt sich leicht von einer einzigen grafischen Benutzeroberfläche verwalten. Mit den verschiedenen Dienstprogrammen können Sie Abfragen ausführen, Informationen zur Leistung einholen und die SQL Server-Dienste überwachen. Sie können die Netzwerkverbindungen konfigurieren und auf Ihrem System sind bereits Dienstprogramme zur Fehlerbehebung installiert. Einige Dienstprogramme können Sie aus dem START-Menü aufrufen, während sich andere nur von der Eingabeaufforderung im Verzeichnis `mssql\binn` starten lassen. Bei Ihrer praktischen Arbeit kommen Sie früher oder später mit allen diesen Dienstprogrammen in Berührung – auf jeden Fall aber in den verbleibenden 18 Tagen dieses Buches.

3.5 Fragen & Antworten

- F** *Mit welchen SQL Server-Dienstprogrammen kann ich die SQL Server-bezogenen Dienste auf meinem Server steuern?*
- A** Mit dem SQL Server-Dienst-Manager und dem SQL Server Enterprise Manager.
- F** *Welche Dienstprogramme erlauben mir die Ausführung von Abfragen an SQL Server 2000?*
- A** Der SQL Server Query Analyzer, `osql.exe` und `isql.exe`.

- F** *Welches der Dienstprogramme stellt den überwiegenden Teil der Funktionen bereit, die ich als Administrator (bzw. als Entwickler) für SQL Server benötige?*
- A** Der SQL Server Enterprise Manager.
- F** *Welches Tool bietet mir Hilfestellung zur Syntax von Befehlen?*
- A** Die SQL Server-Onlinedokumentation oder die Hilfefunktion im SQL Server Query Analyzer.

3.6 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Kontrollfragen

1. Wie verhindern Sie, dass sich andere Benutzer mit Ihren Sicherheitszertifikaten von Enterprise Manager aus an SQL Server anmelden?
2. Wo legen Sie die standardmäßig verwendete Netzwerkbibliothek für Ihren Clientcomputer fest?
3. Wo können Sie für Ihren Server die Netzwerkbibliothek NWLink IPX/SPX hinzufügen?
4. Mit welchen Dienstprogrammen können Sie Leistungsdaten über SQL Server 2000 sammeln?
5. Welches Dienstprogramm sollten Sie ausführen, bevor Sie sich mit Microsofts Software Service in Verbindung setzen?



Übungen

1. Arbeiten Sie mit der Onlinedokumentation, um weitere Informationen über die heute behandelten Dienstprogramme zu finden.
2. Ändern Sie die Standardeinstellung für die Netzwerkbibliothek Ihres Clients auf eine von Ihrem Server nicht unterstützte Einstellung und versuchen Sie, mit dem SQL Server Query Analyzer eine Verbindung zum Server herzustellen.
3. Machen Sie sich mit dem Enterprise Manager vertraut. Beobachten Sie, was beim Anklicken bestimmter Objekte passiert und welche Menüoptionen jeweils verfügbar sind. Klicken Sie Objekte mit der rechten Maustaste an und untersuchen Sie die Auswahlmöglichkeiten im daraufhin erscheinenden Kontextmenü. Je besser Sie sich im Enterprise Manager auskennen, um so leichter fällt Ihnen die weitere Arbeit mit SQL Server.



**Datenbanken und
Datendateien erstellen
und verwalten**



Tag 3 hat die verschiedenen Tools und Dienstprogramme vorgestellt, die zum Lieferumfang von SQL Server 2000 gehören. Sie haben den SQL Server Enterprise Manager kennen gelernt, ein Dienstprogramm, das als Snap-In in der Microsoft Management Console (MMC) läuft. Einen Großteil der in der heutigen Lektion beschriebenen Aufgaben erledigen Sie mit dem SQL Server Enterprise Manager und dem SQL Server Query Analyzer.

Heute lernen Sie, wie Sie Datenbanken erstellen, ändern und löschen. Eine Datenbank wird in mindestens zwei getrennten Dateien gespeichert: Eine Datei nimmt die Daten, Systemtabellen und andere Datenbankobjekte auf, während die andere Datei das Transaktionsprotokoll enthält. In SQL Server 2000 haben Sie die Möglichkeit, Ihre Datenbanken dynamisch wachsen zu lassen, indem Sie der Datenbank oder dem Transaktionsprotokoll Wachstumsparameter zuweisen.



Ein komplizierteres Thema sind Dateigruppen, deren Grundlagen Sie ebenfalls in dieser Lektion kennen lernen. Mithilfe von *Dateigruppen* kann man Datenbankobjekte, wie Tabellen oder Indizes, explizit einer bestimmten Datei (oder Gruppe von Dateien) zuzuweisen. Dies kann Vorteile für die Verwaltung und Wartung der Datenbank haben und möglicherweise die Leistung umfangreicherer Installationen von SQL Server steigern.

Sie werden außerdem noch einen Blick auf die verschiedenen Optionen zur Konfiguration von Datenbanken werfen und erfahren, welchen Einfluss sie auf Ihre Datenbanken haben.

4.1 Eine Datenbank erstellen

In SQL Server können Sie eine neue Datenbank nach folgenden drei Methoden erstellen:

- Mit dem Datenbankerstellung-Assistenten
- Mit dem SQL Server Enterprise Manager
- Mit der Anweisung `CREATE DATABASE`

Wenn Sie eine neue Datenbank erstellen, erzeugen Sie eigentlich nur eine Kopie der Datenbank `model`. Denken Sie daran, dass alles, was in der `model`-Datenbank enthalten ist, auch in der neuen Datenbank auftaucht. Nachdem Sie die Datenbank durch Kopieren der `model`-Datenbank erzeugt haben, erweitert sie SQL Server auf die angeforderte Größe und füllt den zusätzlichen Platz mit leeren Speicherseiten.

Datenbanken benötigen Dateien, um die in ihnen enthaltenen Daten physikalisch auf der Festplatte zu speichern. Wenn Sie eine neue Datenbank erstellen, müssen Sie mindestens eine Datei spezifizieren, um die Daten und Systemtabellen zu speichern, und eine Datei,

die das Transaktionsprotokoll aufnimmt. Wie Abbildung 4.1 zeigt, können die Datenbank und das Transaktionsprotokoll auch mehrere Dateien umfassen.

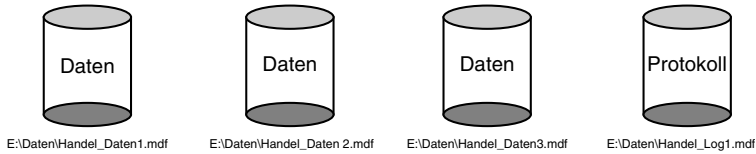


Abbildung 4.1:
Die eigentliche
Datenbank und das
zugehörige Transak-
tionsprotokoll
können sich über
mehrere Datenbank-
dateien erstrecken.



Die Datenbankdateien, die Sie erstellen, lassen sich nicht durch irgendeine andere Datenbank oder ein anderes Transaktionsprotokoll gemeinsam nutzen.

Dieser Abschnitt erläutert die einzelnen Teile der Anweisung `CREATE DATABASE`. Nachdem Sie mit den Elementen und Abläufen vertraut sind, erfahren Sie, wie man eine Datenbank mithilfe des SQL Server Enterprise Managers erstellt.



Die Syntax der Anweisung `CREATE DATABASE` lautet:

```
CREATE DATABASE database_name
[ON {[PRIMARY]
(NAME = logical_name,
FILENAME = 'physical_name'
[,SIZE = size]
[,MAXSIZE = max_size | UNLIMITED]
[,FILEGROWTH = growth_increment]]
},...n]]
[LOG ON
((NAME = logical_name,
FILENAME = 'physical_name'
[,SIZE=size | UNLIMITED]
[,MAXSIZE = max_size | UNLIMITED]
[,FILEGROWTH = growth_increment]))
[,...n]]
[,COLLATE collation_name]
[FOR LOAD | FOR ATTACH]
```

In SQL Server 2000 ist als einziger Parameter der logische Name der Datenbank erforderlich, um eine Datenbank zu erstellen. Allerdings ist das nicht die empfohlene Methode. Besser ist es, mindestens die folgenden Parameter zu spezifizieren: logischer Datenbank-



name, Dateiname und Größe für die Datendatei sowie Dateiname und Größe für die Transaktionsprotokolldatei. Die folgende Liste enthält die Bedeutung der einzelnen Parameter:

- *database_name* bezieht sich auf die Datenbank als Ganzes.
- *ON PRIMARY* gibt an, zu welcher Dateigruppe diese Datenbank gehört. Die standardmäßige Dateigruppe ist *Primary*. Auf Dateigruppen geht diese Lektion später ein.
- *NAME* spezifiziert den logischen Namen, über den Sie sich in *SQL Server* auf die physikalische Datenbankdatei auf der Festplatte beziehen.
- *FILENAME* gibt den Pfad- und Dateinamen für den Standort der Daten auf der Festplatte an. Es muss sich um eine lokale Festplatte handeln.
- *SIZE* gibt die Größe der Datenbankdatei – entweder in MB oder in KB – an. Der Standardwert ist die Größe der Datenbank *model*. Um eine bestimmte Größe festzulegen, geben Sie den Zahlenwert und das Suffix MB bzw. KB im Parameter an. Eine Datei der Größe 10 MB ist im Parameter als 10MB zu spezifizieren.



Die Größenangaben lassen sich nur als ganze Zahlen spezifizieren. Um eine Datenbank der Größe 2,5 MB zu erstellen, müssen Sie den Wert in Kilobyte angeben: 2560KB.

- *MAXSIZE* gibt an, bis zu welcher maximalen Größe die Datenbank dynamisch wachsen kann. Falls Sie hier nichts angeben und die automatische Vergrößerung zugelassen ist, kann die Datenbank bis zur Größe der Festplatte anwachsen. Diesen Parameter geben Sie ebenfalls entweder in MB oder in KB an.
- *FILEGROWTH* legt die Schrittweite für die dynamische Vergrößerung der Datenbank fest. Diesen Wert können Sie als MB bzw. KB oder als Prozentwert bezogen auf die Dateigröße zum Zeitpunkt der Größenänderung angeben. Als Standardwert gilt 1MB. Der Wert für *FILEGROWTH* darf die *MAXSIZE*-Einstellung nicht überschreiten.
- *LOG ON* gibt an, wo sich die Transaktionsprotokolle befinden und welche Größe sie besitzen.
- *COLLATE* ist ein neuer Parameter in *SQL Server 2000* und gibt die Standardsortierung für die Datenbank an. Dabei muss es sich entweder um einen Windows-Sortierungsnamen oder einen *SQL*-Sortierungsnamen handeln. Wenn Sie diesen Parameter nicht angeben, wird der Datenbank die Standardsortierung der Instanz von *SQL Server* zugewiesen. Die Sortierung lässt sich auch auf der Ebene von Tabellen oder einzelnen Spalten spezifizieren.

- *FOR LOAD* markiert die Datenbank für die ausschließliche Verwendung durch den Datenbankbesitzer (dbo). Dieser Parameter ist nur aus Gründen der Abwärtskompatibilität zu SQL Server 6.5 vorhanden und sollte in SQL Server 2000 nicht verwendet werden.
- *FOR ATTACH* fügt einen Satz von Dateien an, die eine Datenbank darstellen. Diese Dateien müssen vorher erstellt und dann von SQL Server 2000 getrennt worden sein. Später in dieser Lektion erfahren Sie mehr zu dieser Option.

Listing 4.1 zeigt den erforderlichen Code, um eine Datenbank mit einer anfänglichen Größe von 25 MB zu erzeugen – 20 MB für die eigentlichen Daten und 5 MB für das Transaktionsprotokoll. Die Dateien können eine maximale Größe von 115 MB annehmen – 100 MB für die Daten und 15 MB für das Transaktionsprotokoll. Es gilt die Standardsortierung von SQL Server.



Bei SQL Server 2000 empfiehlt es sich, die derzeit benötigte Größe für die Daten und Protokolle anzugeben, statt den Umfang zu spezifizieren, der in Zukunft erforderlich sein wird. Mit den Parametern *FILEGROWTH* und *MAXSIZE* kann man festlegen, wie die Datenbank bei Bedarf zu vergrößern ist. Auf diese Weise lässt sich Festplattenplatz sparen.

Listing 4.1: Eine Datenbank mit einer anfänglichen Größe von 25 MB erstellen

```
USE master
GO
CREATE DATABASE Frogger ON PRIMARY
( NAME = FroggerData,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\FroggerData.mdf',
  SIZE = 20 MB,
  MAXSIZE = 100 MB,
  FILEGROWTH = 10 MB )
LOG ON
( NAME = FroggerLog,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\FroggerLog.ldf',
  SIZE = 5 MB,
  MAXSIZE = 15 MB,
  FILEGROWTH = 1 MB )
GO
```



Der CREATE DATABASE-Prozess reserviert 20.00 MB auf dem Datenträger 'FroggerData'.
Der CREATE DATABASE-Prozess reserviert 5.00 MB auf dem Datenträger 'FroggerLog'.



Der Code im Beispiel von Listing 4.2 erstellt eine Datenbank, die mehrere Dateien sowohl für die Daten als auch für das Transaktionsprotokoll umfasst. Die Namen der Transaktionsprotokolldateien der Datendateien verwenden die von Microsoft empfohlenen Dateierweiterungen. Demnach soll die erste Datendatei die Erweiterung *.mdf*, alle weiteren Datendateien die Erweiterung *.ndf* und die Protokolldateien die Erweiterung *.ldf* erhalten.

Listing 4.2: Erzeugen einer Datenbank, die mehrere Dateien umfasst

```
USE master
GO
CREATE DATABASE Leap ON PRIMARY
( NAME = LeapData1,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\LeapData1.mdf',
  SIZE = 5,
  MAXSIZE = 20,
  FILEGROWTH = 1 ),
( NAME = LeapData2,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\LeapData2.ndf',
  SIZE = 5,
  MAXSIZE = 20,
  FILEGROWTH = 5 )
LOG ON
( NAME = LeapLog1,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\LeapLog1.ldf',
  SIZE = 2,
  MAXSIZE = 20,
  FILEGROWTH = 1 ),
( NAME = LeapLog2,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\LeapLog2.ldf',
  SIZE = 2,
  MAXSIZE = 10,
  FILEGROWTH = 2 )
GO
```



Der CREATE DATABASE-Prozess reserviert 5.00 MB auf dem Datenträger 'LeapData1'.
Der CREATE DATABASE-Prozess reserviert 5.00 MB auf dem Datenträger 'LeapData2'.
Der CREATE DATABASE-Prozess reserviert 2.00 MB auf dem Datenträger 'LeapLog1'.
Der CREATE DATABASE-Prozess reserviert 2.00 MB auf dem Datenträger 'LeapLog2'.

Wenn Sie mehrere Datendateien angeben, verteilt SQL Server die Daten automatisch gleichmäßig über alle spezifizierten Dateien. Dies kann helfen, Konflikte beim Datenbankzugriff und Zugriffskonzentrationen auf bestimmte Stellen der Datenbankdatei zu verringern. Dagegen verteilt SQL Server die Einträge in das Transaktionsprotokoll nicht gleichmäßig über die angegebenen Dateien, sondern füllt die Dateien jeweils nacheinander auf. Ist eine Protokolldatei voll, kommen die weiteren Informationen in die nächste Datei.



Falls Sie nicht RAID 5 oder höher einsetzen, wird dringend empfohlen, die Transaktionsprotokolldateien auf physikalisch getrennten Festplatten zu speichern. Beim Versagen einer Festplatte haben Sie so bessere Chancen für die Wiederherstellung Ihrer Daten. Ein weiterer Vorteil besteht darin, dass Schreibzugriffe auf das Transaktionsprotokoll nicht mit denen auf Ihre Datenbank kollidieren.

Listing 4.3 zeigt, wie man eine Datenbank erstellt, die eine mit dem Befehl `COLLATE` festgelegte Sortierreihenfolge verwendet. In diesem Fall ist angegeben, dass SQL Server 2000 eine Datenbank mit der Codeseite Latin 1 oder 1251, der Wörterbuchsorierreihenfolge (General), ohne Beachtung der Groß-/Kleinschreibung (CI) und ohne Beachtung von Akzenten (AI) erstellt.

Listing 4.3: Erzeugen einer Datenbank mit einer nicht standardmäßigen Sortierreihenfolge

```
USE master
GO
CREATE DATABASE Swim ON PRIMARY
( NAME = SwimData,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\SwimData1.mdf',
  SIZE = 20MB,
  MAXSIZE = 100MB,
  FILEGROWTH = 10MB )
LOG ON
( NAME = SwimLog,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\SwimLog1.ldf',
  SIZE = 5MB,
  MAXSIZE = 15MB,
  FILEGROWTH = 1MB )
COLLATE Latin1_General_CI_AI
GO
```



Der CREATE DATABASE-Prozess reserviert 20.00 MB auf dem Datenträger 'SwimData'.

Der CREATE DATABASE-Prozess reserviert 5.00 MB auf dem Datenträger 'SwimLog'.

Eine neue Datenbank können Sie auch mit dem SQL Server Enterprise Manager erzeugen. Führen Sie dazu die folgenden Schritte aus:

1. Starten Sie den SQL Server Enterprise Manager über START / PROGRAMME / MICROSOFT SQL SERVER / ENTERPRISE MANAGER.
2. Stellen Sie die Verbindung zu einer Instanz von SQL Server her.
3. Erweitern Sie den Ordner *Datenbanken* gemäß Abbildung 4.2.
4. Klicken Sie mit der rechten Maustaste entweder auf das Datenbankverzeichnis oder auf eine leere Stelle im Detailbereich und wählen Sie aus dem Kontextmenü NEUE DATENBANK.
5. Damit gelangen Sie zum Dialogfeld DATENBANKEIGENSCHAFTEN. Geben Sie auf der Registerkarte ALLGEMEIN einen Datenbanknamen an (im Beispiel *Croak*). Wechseln Sie zur Registerkarte DATENDATEIEN; die neue Datenbankdatei namens *Croak_Data* hat eine Anfangsgröße von 1 MB und befindet sich im Ordner ...\\data. Im Abschnitt DATEIEIGENSCHAFTEN im unteren Teil des Dialogfelds ist das Kontrollkästchen DATEI AUTOMATISCH VERGRÖßERN eingeschaltet und die Werte für die Dateivergrößerung sind gesetzt. Als maximale Dateigröße ist die Option *Unbeschränkt vergrößerbar* gewählt.
6. Um die Eigenschaften der Datenbankdatei zu ändern, klicken Sie einfach auf das entsprechende Feld und nehmen die Änderungen vor. (Das Beispiel arbeitet mit den Standardwerten.) Um zusätzliche Datenbankdateien hinzuzufügen, gehen Sie jeweils in das nächste Feld unter DATEINAME der Liste DATENBANKDATEIEN und legen die zugehörigen Dateieigenschaften fest.
7. Klicken Sie auf die Registerkarte TRANSAKTIONSPROTOKOLL. Das Transaktionsprotokoll hat den Standardnamen *Croak_Log* und die Anfangsgröße 1 MB.
8. Klicken Sie auf OK, nachdem Sie die Eigenschaften wie gewünscht bearbeitet haben. Der Enterprise Manager sollte nun Abbildung 4.4 entsprechen. Falls die bisher angelegten Datenbanken Frogger, Leap oder Croak nicht zu sehen sind, klicken Sie mit der rechten Maustaste auf den Ordner *Datenbanken* und wählen AKTUALISIEREN aus dem Kontextmenü.

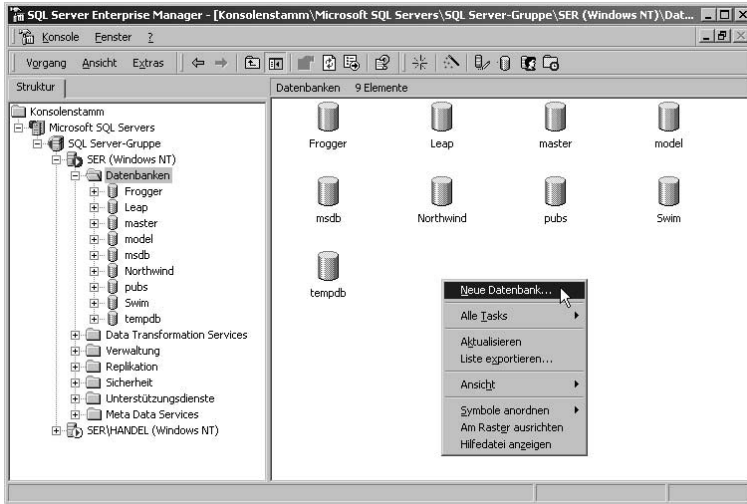


Abbildung 4.2:
Der Detailbereich
des SQL Server-
Enterprise Mana-
gers zeigt die Daten-
banken an. Über
das Kontextmenü
können Sie die Sym-
bole anordnen oder
Aufgaben ausfüh-
ren.

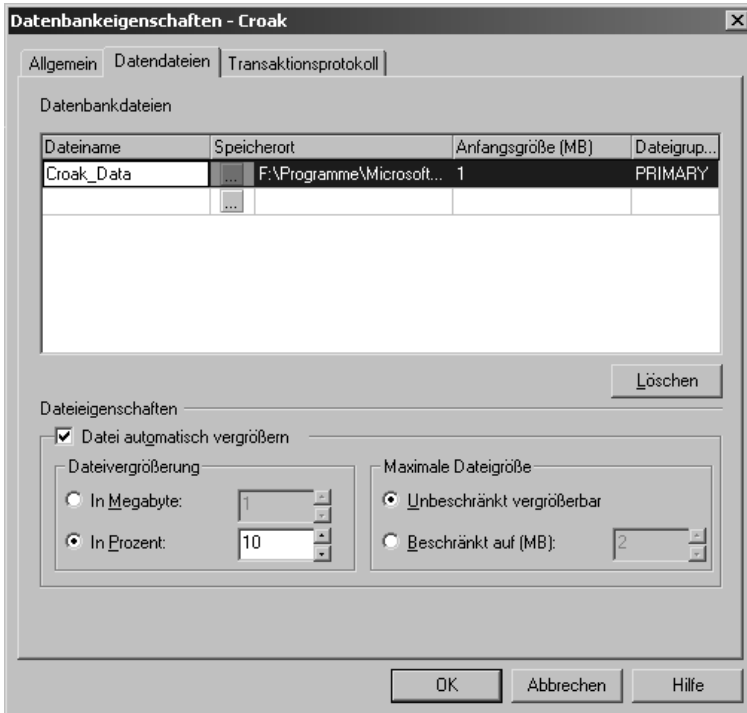


Abbildung 4.3:
Diese Registerkarte
zeigt die Eigenschaf-
ten einer Daten-
bank, beispielsweise
Name, Größe und
Dateigruppe.

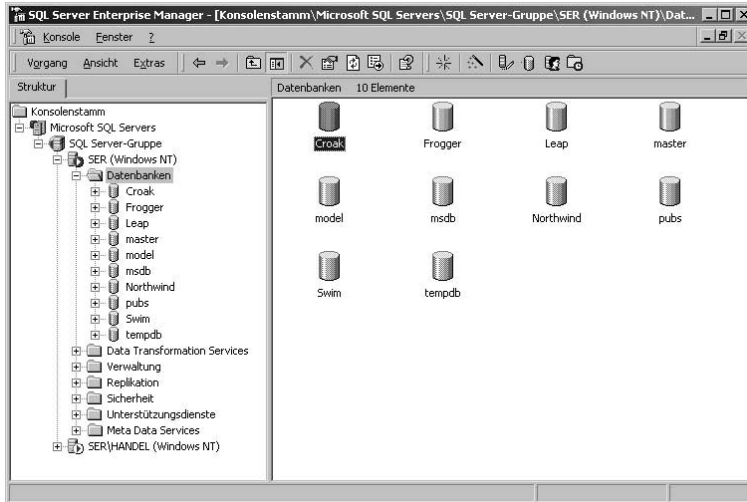


Abbildung 4.4:
Der Enterprise
Manager mit den
neu hinzugefügten
Datenbanken

4.2 Informationen über Datenbanken

Es dürfte Sie nicht überraschen, dass sich Informationen über Datenbanken nach verschiedenen Methoden einholen lassen. Beispielsweise können Sie mit dem SQL Server Enterprise Manager arbeiten oder im SQL Server Query Analyzer gespeicherte Systemprozeduren aufrufen.

Mit der gespeicherten Systemprozedur `sp_helpdb` können Sie einzelne Datenbanken untersuchen oder Informationen über alle unter SQL Server verfügbaren Datenbanken erhalten. Wenn Sie `sp_helpdb` aufrufen, ohne eine Datenbank zu spezifizieren, erhalten Sie Zusammenfassungen zu allen Datenbanken in SQL Server.



Das folgende Beispiel zeigt nur einen Teil der zurückgegebenen Daten. Außer den hier wiedergegebenen Informationen erhalten Sie viele weitere Angaben wie zum Beispiel Wiederherstellbarkeit, Aktualisierungsfähigkeit und Sortierreihenfolge.



```
EXEC sp_helpdb
```



name	db_size	owner	dbid	created
-----	-----	-----	-----	-----
Croak	2.00 MB	SER\Admi	10	Dez 28 2000
Frogger	25.00 MB	SER\Admi	7	Dez 28 2000
Leap	14.00 MB	SER\Admi	9	Dez 28 2000
master	17.00 MB	sa	1	Aug 6 2000
model	1.25 MB	sa	3	Aug 6 2000
msdb	14.00 MB	sa	4	Aug 6 2000
Northwind	4.25 MB	sa	6	Aug 6 2000
pubs	2.50 MB	sa	5	Aug 6 2000
Swim	25.00 MB	SER\Admi	8	Dez 28 2000
tempdb	8.75 MB	sa	2	Dez 27 2000

Die gespeicherte Prozedur `sp_helpdb` liefert also zusammenfassende Informationen zu allen unter SQL Server vorhandenen Datenbanken. Zum Beispiel hat die Datenbank *Frogger* eine Größe von 25 MB; als Besitzer (owner) ist das Benutzerkonto *SER\Administrator* angegeben. Um mehr Informationen über eine bestimmte Datenbank zu erhalten, formulieren Sie die Anweisung mit der gespeicherten Systemprozedur `sp_helpdb` folgendermaßen:



```
EXEC sp_helpdb Croak
```



name	db_size	owner	dbid	created
-----	-----	-----	-----	-----
Croak	2.00 MB	SER\Admini	10	Dez 28 2000

name	fileid	filename
-----	-----	-----
Croak_Data	1	F:\Programme\...\MSSQL\data\Croak_Data.MDF
Croak_Log	2	F:\Programme\...\MSSQL\data\Croak_Log.LDF

Außer den Informationen, die die Anweisung des letzten Beispiels geliefert hat, bekommen Sie jetzt noch Informationen über die vorhandenen Dateien und ihre Zuordnung. Die Ausgabe der obigen Anweisung zeigt, dass die Datendatei *Croak_Data* 2 MB und die Transaktionsprotokolldatei *Croak_Log* 1 MB groß ist.

Die Informationen zu Datenbanken können Sie auch mit dem SQL Server Enterprise Manager einholen. Starten Sie den SQL Server Enterprise Manager und erweitern Sie den Ordner *Datenbanken*. Wenn Sie im Detailbereich auf eine Datenbank doppelklicken, erscheint das bereits bekannte Dialogfeld DATENBANKEIGENSCHAFTEN. Informationen über eine Datenbank erhalten Sie auch, indem Sie in der Konsolenstruktur einfach auf eine Datenbank klicken. Zum Beispiel zeigt Abbildung 4.5 allgemeine Informationen zur Datenbank *Leap*. Weitere Detailebenen lassen sich einblenden, indem Sie die Optionen

unter den Symbolen DATENBANK, WARTUNG oder RESERVIERTER SPEICHERPLATZ wählen. Abbildung 4.6 zeigt die Optionen, die unter dem Symbol DATENBANK verfügbar sind.

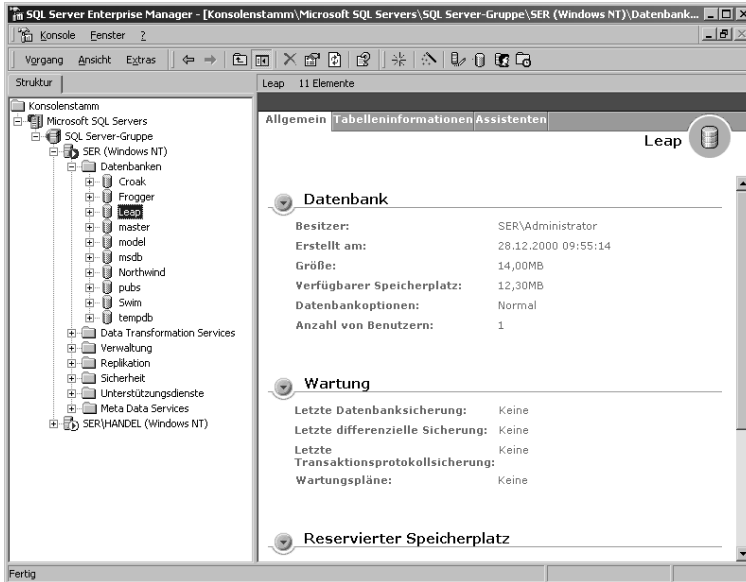


Abbildung 4.5:
Allgemeine Informationen zur Datenbank Leap

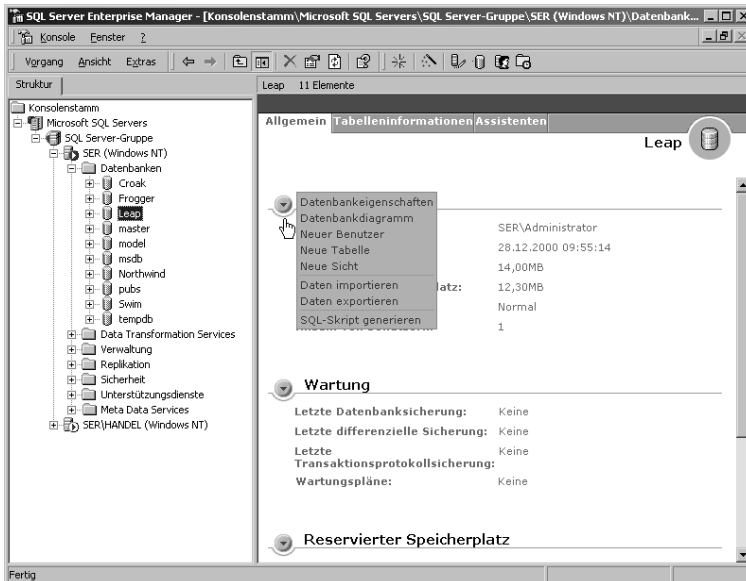


Abbildung 4.6:
Zusätzliche datenbankbezogene Optionen

4.3 Datenbankoptionen festlegen

Dieser Abschnitt beschäftigt sich näher mit den Optionen, die sich auf Datenbanken anwenden lassen. Wie üblich können Sie die Datenbankoptionen entweder mit gespeicherten Systemprozeduren im SQL Server Query Analyzer oder über den SQL Server Enterprise Manager einstellen.

Um die Datenbankoptionen mit dem SQL Server Enterprise Manager anzusehen oder zu ändern, navigieren Sie durch die Konsolenstruktur, bis Sie die gewünschte Datenbank (*pubs* in diesem Beispiel) erreicht haben. Nun können Sie auf die Datenbank doppelklicken oder mit der rechten Maustaste darauf klicken und EIGENSCHAFTEN aus dem Kontextmenü wählen. Gehen Sie im Dialogfeld PUBS-EIGENSCHAFTEN auf die Registerkarte OPTIONEN (siehe Abbildung 4.7).

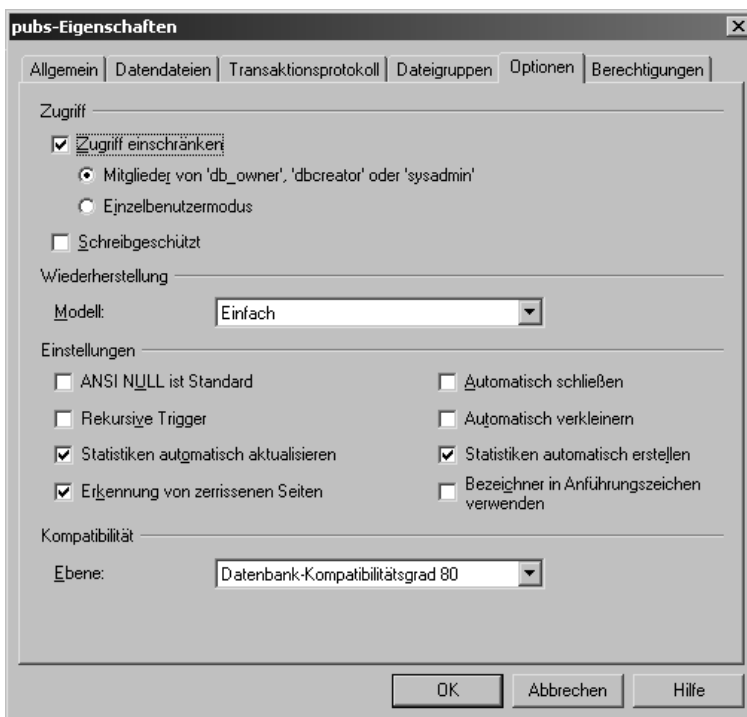


Abbildung 4.7:
Die Registerkarte
Optionen der Eigen-
schaftenseite für die
Datenbank *pubs*

Die Datenbankoptionen sind auf dieser Registerkarte in die Kategorien ZUGRIFF, EINSTELLUNGEN und KOMPATIBILITÄT gegliedert. Die einzelnen Einstellungen haben folgende Bedeutung:

- Mitglieder von 'db_owner', 'dbcreator' oder 'sysadmin' gibt an, dass nur Mitglieder von db_owner, dbcreator oder sysadmin auf die Datenbank zugreifen können. Diese Option setzt der Datenbankadministrator, wenn er Wartungsarbeiten ungestört von normalen Benutzern durchführen möchte.
- *Einzelbenutzermodus* legt fest, dass jeweils nur ein Benutzer auf die Datenbank zugreifen kann. Bei dieser Option lässt sich in der Tat nur eine einzige Benutzerverbindung zur Datenbank herstellen. Man verwendet diese Option vor allem während der Wiederherstellung einer Datenbank.
- *Schreibgeschützt* gibt an, dass die Benutzer der Datenbank nur Daten abrufen, nicht aber ändern können.
- Modell bestimmt den Typ des Wiederherstellungsmodells für die Datenbank, d.h. welches Risiko des Datenverlustes Sie eingehen möchten. Zur Wahl stehen die Optionen *Einfach*, *Massenprotokolliert* und *Vollständig*. Die maximale Wiederherstellbarkeit erreichen Sie mit der Option *Vollständig*, am leichtesten lässt sich die Wiederherstellung mit der Option *Einfach* realisieren.
- *ANSI NULL ist Standard* legt fest, ob neue Datenbankspalten standardmäßig als NULL oder NOT NULL definiert werden. Auf Tabellen und Spalten geht Tag 9 näher ein.
- *Rekursive Trigger* lässt das rekursive Auslösen von Triggern zu. Rekursion tritt auf, wenn derselbe Trigger infolge der von ihm ausgeführten Aktion erneut ausgelöst wird. SQL Server 2000 unterstützt die Rekursion bis zu einer Tiefe von 32 Ebenen.
- *Statistiken automatisch aktualisieren* arbeitet mit der Option *Statistiken automatisch erstellen* zusammen. Die Informationen in den Datenspalten ändern sich im Laufe der Zeit, während die Statistiken unverändert bleiben. Um diese Diskrepanz zu beseitigen, müssen Sie die Statistiken regelmäßig aktualisieren. Wenn Sie dieses Kontrollkästchen einschalten, geschieht die Aktualisierung automatisch.
- *Erkennung von zerrissenen Seiten* legt fest, dass unvollständig geschriebene Seiten (eine Form der Beschädigung von Daten) erkannt werden. Es empfiehlt sich, diese Option einzuschalten.
- *Automatisch schließen* legt fest, dass die Datenbank automatisch heruntergefahren wird, wenn der letzte Benutzer die Arbeit damit beendet hat. Bei Datenbanken, die nicht häufig benutzt werden, spart diese Option Ressourcen auf dem Server.
- *Automatisch verkleinern* gibt an, dass die Datenbankdateien regelmäßig automatisch verkleinert werden. Transaktionsprotokolle werden automatisch verkleinert, nachdem eine Sicherung des Transaktionsprotokolls durchgeführt wurde. Die Datenbankdateien werden automatisch verkleinert, wenn eine periodische Überprüfung feststellt, dass mehr als 25% des zugewiesenen Speicherplatzes nicht verwendet wird. Der Vorgang verkleinert die Datenbank so, dass nur noch 25% des belegten Speicherplatzes unbenutzt ist. Beachten Sie, dass dieser Vorgang Datenbanken nicht unter ihre Originalgröße verkleinert.

- *Statistiken automatisch erstellen* generiert automatisch Statistiken über die Verteilung der Werte in einer Datenspalte. Der SQL Server Abfrageoptimierer erstellt anhand dieser Informationen einen Ausführungsplan, der die Kosten bei der Einbeziehung verschiedener Spalten berücksichtigt. Es empfiehlt sich, diese Option einzuschalten.
- *Bezeichner in Anführungszeichen verwenden* erlaubt die Verwendung von doppelten Anführungszeichen als Teil eines SQL Server-Bezeichners. Ein *Bezeichner* ist der Name eines Datenbankobjekts; das kann beispielsweise eine Variable oder eine Tabelle sein. Bezeichner in Anführungszeichen sind hilfreich, wenn der Name eines Datenbankobjektes einem reservierten Wort von SQL Server entspricht oder wenn der Name Leerzeichen enthält – zum Beispiel eine Tabelle »Table« oder eine Tabelle »Summa Summarum«.
- *Kompatibilitätsebene* gibt an, welchen Grad der Abwärtskompatibilität die Datenbank unterstützen soll. Der Kompatibilitätsgrad 80 gilt für SQL Server 2000 (abgeleitet von SQL Server 8.0), der Grad 70 für SQL Server 7.0, der Grad 65 für SQL Server 6.5 und schließlich der Kompatibilitätsgrad 60 für SQL Server 6.0.

Die genannten Optionen können Sie auch mit der gespeicherten Systemprozedur `sp_dboption` im SQL Server Query Analyzer festlegen. Um zum Beispiel den Zugriff auf die Datenbank *pubs* auf Mitglieder von `db_owner`, `dbcreator` oder `sysadmin` einzuschränken, führen Sie den Code gemäß Abbildung 4.8 im Query Analyzer aus. Die Anweisung lautet:

```
exec sp_dboption pubs, 'DBO Use Only', TRUE
```

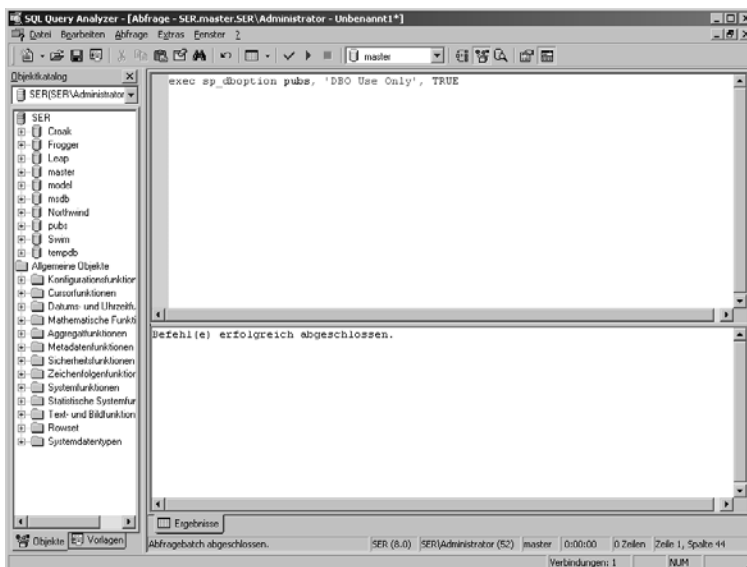


Abbildung 4.8:
Mit diesem Code
schränken Sie den
Zugriff auf die
Datenbank
pubs ein.

Nachdem Sie diese Anweisung ausgeführt haben, können Sie mithilfe der gespeicherten Systemprozedur `sp_helpdb pubs` prüfen, ob die Option tatsächlich angewandt wurde (siehe Abbildung 4.9). Neben anderen standardmäßig gesetzten Optionen erscheint im Ergebnisfenster auch die Option *dbo use only*.

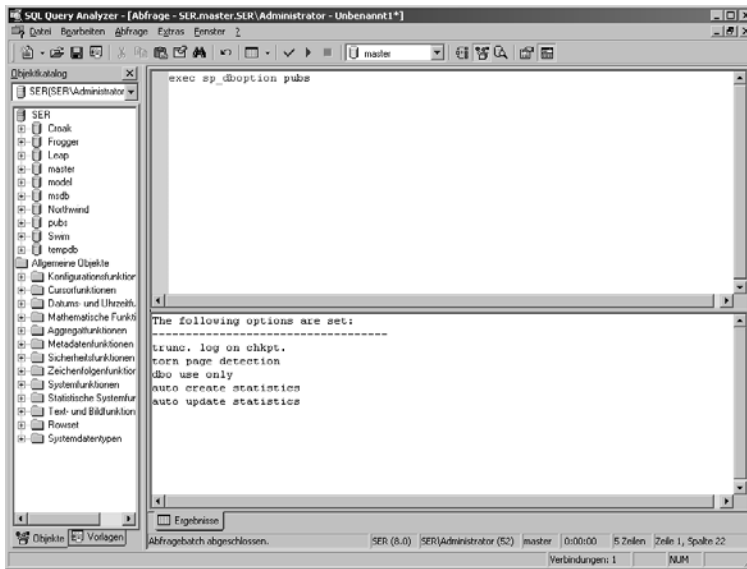


Abbildung 4.9:
Das Ergebnisfenster
zeigt auch die geän-
derte Option.

Um den vorherigen Zustand wiederherzustellen, führen Sie einfach die folgende Anweisung

```
sp_dboption pubs, 'DBO Use Only', False
```

aus.

Als Parameter der gespeicherten Systemprozedur `sp_dboption` können Sie sieben zusätzliche Optionen angeben, die im Enterprise Manager nicht zur Verfügung stehen:

- *Concat NULL Yields NULL* arbeitet so ähnlich wie eine Multiplikation mit Null. Alles was Sie mit Null multiplizieren, ergibt wieder Null. Diese Option besagt, dass alles was Sie mit NULL verbinden, wieder NULL ergibt – beispielsweise 'Hello World' + NULL = NULL.
- *Cursor close on commit* gibt an, dass alle offenen Cursor geschlossen werden, wenn eine Transaktion beendet wird. Ein Cursor ist das Ergebnis einer Abfrage.
- *Default to local cursor* gibt an, dass ein Cursor, der ohne die Verwendung des Schlüsselwortes GLOBAL erstellt wird, nur in dem lokalen Stapel, Trigger, der gespeicherten Prozedur und so weiter verfügbar ist, in deren Kontext er erstellt wurde.

- *Merge publish* gibt an, dass die Datenbank für eine Mergereplikation publiziert werden kann. Replikationsszenarios behandelt Tag 17.
- *Offline* schließt die Datenbank und fährt sie herunter. Nachdem sich eine Datenbank im Status offline befindet, kann man sie auf ein Wechselmedium verschieben und verteilen.
- *Published* gibt an, dass die Datenbank für die Replikation publiziert werden kann.
- *Subscribed* gibt an, dass die Datenbank als Empfänger für publizierte Daten dient.

4.4 Die Größe von Datenbanken ändern

Wenn Sie die Dateidefinitionen und Größenangaben für eine Datenbank ändern möchten, können Sie das mit der Anweisung `ALTER DATABASE` oder mit dem SQL Server Enterprise Manager erledigen. Um die Größe der Datenbank zu verringern, müssen Sie entweder die Anweisung `DBCC SHRINKDATABASE` oder `DBCC SHRINKFILE` verwenden. Dateigruppen können Sie der Datenbank mit der Anweisung `ALTER DATABASE` hinzufügen. Dieser Abschnitt untersucht zunächst die Anweisung `ALTER DATABASE`, wobei Sie einige Änderungen an den bislang erstellten Datenbanken vornehmen. Dann erfahren Sie, wie man Datenbanken mit der Anweisung `DBCC SHRINKDATABASE` verkleinern kann. Zum Abschluss dieses Abschnitts erhalten Sie einen kurzen Überblick über Dateigruppen.



Damit Sie die Anweisung `ALTER DATABASE` ausführen können, brauchen Sie `CREATE DATABASE`-Berechtigungen.



Die Syntax für die Anweisung `ALTER DATABASE` lautet:

```
ALTER DATABASE database
{
    ADD FILE <File specification> [...n] [TO FILEGROUP filegroup_name]
| ADD LOG FILE <File specification> [...n]
| REMOVE FILE logical_name
| ADD FILEGROUP filegroup_name
| REMOVE FILEGROUP filegroup_name
| MODIFY FILE <File specification>
| MODIFY NAME = new_dbname
| MODIFY FILEGROUP filegroup_name, filegroup_property
| SET <optionsspec> [...n][WITH <termination>]
| COLLATE <collation_name>
}
```

```
<File specification> ::=
(NAME = logical_name,
FILENAME = 'physical_name'
[,SIZE = size]
[,MAXSIZE = max_size [UNLIMITED]
[,FILEGROWTH = growth_increment])
```

Die Parameter haben folgende Bedeutung:

- *database* ist der Name der zu ändernden Datenbank.
- *ADD FILE* gibt an, dass eine Datei hinzugefügt wird.
- Alle Optionen von *File specification* sind die gleichen Spezifikationen, die bereits bei der Anweisung *CREATE DATABASE* angegeben wurden.
- *TO FILEGROUP* gibt an, zu welcher Dateigruppe diese Datei hinzugefügt werden soll. Ist keine Dateigruppe angegeben, wird die Datei der Standarddateigruppe (PRIMARY) hinzugefügt.
- *ADD LOGFILE* fügt der Datenbank eine neue Protokolldatei hinzu.
- *REMOVE FILE* entfernt eine Datei aus der Datenbank. Die Datei muss leer sein, damit sie entfernt werden kann. Mit der Anweisung *DBCC SHRINKFILE* können Sie die Datei leeren. Mehr dazu erfahren Sie später in dieser Lektion.
- *ADD FILEGROUP* fügt eine neue Dateigruppe hinzu. Sie müssen außerdem den neuen Namen für die Dateigruppe angeben.
- *REMOVE FILEGROUP* entfernt eine Dateigruppe aus der Datenbank und löscht alle Dateien in der Dateigruppe. Die Dateien in der Dateigruppe müssen leer sein. Mit der Anweisung *DBCC SHRINKFILE* können Sie die Dateien leeren. Mehr dazu erfahren Sie später in dieser Lektion.
- *MODIFY FILE* erlaubt die Änderung der Eigenschaften einer Datei. Hierzu gehören die Optionen *physical_name*, *filegrowth* und *maxsize*. Wenn Sie die Größenangaben ändern, muss der neue Wert größer als der momentan eingestellte sein. Der Parameter *FILENAME* kann nur für Dateien geändert werden, die sich in *tempdb* befinden; die Änderung tritt erst nach einem Neustart von SQL Server in Kraft.



Obwohl man die Größe geringer als die aktuelle Größe der Datei festlegen kann, lässt sich die Datenbank von dieser Stelle aus nicht verkleinern. Dazu muss man den Befehl *DBCC SHRINKDATABASE* oder *DBCC SHRINKFILE* aufrufen.

- *MODIFY NAME* benennt die Datenbank um.
- *MODIFY FILEGROUP* erlaubt Änderungen an den Eigenschaften von Dateigruppen einschließlich der Eigenschaften *READONLY*, *READWRITE* und *DEFAULT*.
- *SET* erlaubt es, die Eigenschaften *READONLY*, *READWRITE* und *DEFAULT* festzulegen. Der Parameter *termination* spezifiziert, wann ein Rollback auftritt – entweder *ROLLBACK AFTER n SECONDS* oder *ROLLBACK IMMEDIATE*.
- *COLLATE* spezifiziert die für die Datenbank zu verwendende Sortierreihenfolge. Hier ist der *SQL*-Sortierungsname anzugeben. Wenn die Sortierreihenfolge nicht spezifiziert ist, gilt die Standardsortierung von *SQL Server*.

Eine Datenbank erweitern

Datenbanken lassen sich erweitern, indem man ihnen zusätzliche Dateien für die Vergrößerung hinzufügt. Das gilt sowohl für den Datenanteil als auch den Protokollanteil. Sofern Sie die automatische Größenänderung nicht ausdrücklich abgeschaltet haben, wachsen die Datenbankdateien automatisch so lange an, bis kein Platz mehr auf der Festplatte vorhanden ist. Wie bereits erwähnt, werden die Daten gleichmäßig über mehrere Dateien hinweg verteilt gespeichert. Es stellt sich nun die Frage, warum man mehrere Dateien für die Daten anlegen soll, wenn die Datenbank ohnehin automatisch bei Bedarf wächst. Es scheint, als ob sich dadurch nur die Wartung der Datenbank erschwert. Prinzipiell stimmt das zwar, aber eine auf mehrere Dateien verteilte Datenbank hat auch folgende Vorteile:

- Man kann die Dateien auf verschiedene Festplatten verteilen.
- Die Leistung lässt sich verbessern, da Lese- und Schreibzugriffe auf die Datenbank mit höherer Wahrscheinlichkeit über verschiedene Festplattencontroller abgewickelt werden.
- Datenbankdateien lassen sich unabhängig voneinander sichern.
- Wenn Sie Dateigruppen verwenden, können Sie bestimmte Teile der Daten in vorgegebenen Dateien speichern – beispielsweise die Lohntabelle in einer eigenen Dateigruppe und dort in einer bestimmten Datei.



Sie sollten eine Maximalgröße für die Datenbank und das Transaktionsprotokoll festlegen. Wenn Sie keine Maximalgröße festlegen und das automatische Wachstum nicht explizit unterdrückt haben, kann die Datenbank letztendlich die gesamte Festplatte belegen. Wenn eine Festplattenpartition in Windows voll wird, generiert Windows Fehler und verweigert die weitere Arbeit, bis Sie den Platz auf der betreffenden Partition freigeben.



Listing 4.4 zeigt, wie man vom Query Analyzer aus eine neue Datendatei in die Datenbank Croak hinzufügt.

Listing 4.4: Hinzufügen einer neuen Datendatei zur Datenbank Croak



```
ALTER DATABASE croak
ADD FILE
( NAME = CroakData2,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\CroakData2.ndf',
  SIZE = 2,
  MAXSIZE = 10,
  FILEGROWTH = 2)
```



Die Datenbank wird um 2.00 MB auf dem Datenträger 'CroakData2' erweitert.

Mit der gespeicherten Systemprozedur `sp_helpdb` können Sie nun überprüfen, ob die Datenbank erfolgreich vergrößert wurde:

```
sp_helpdb croak
```

Die Datenbank sollte jetzt eine Größe von 4 MB haben und die Dateien *Croak_Data*, *Croak_Log* und *CroakData2* enthalten.

Jetzt können Sie noch das Transaktionsprotokoll erweitern:



```
ALTER DATABASE croak
ADD LOG FILE
( NAME = CroakLog2,
  FILENAME =
    'F:\Programme\Microsoft SQL Server\MSSQL\DATA\CroakLog2.ndf',
  SIZE = 2,
  MAXSIZE = 10,
  FILEGROWTH = 2)
```



Die Datenbank wird um 2.00 MB auf dem Datenträger 'CroakLog2' erweitert.

Überprüfen Sie die Ergebnisse, indem Sie `sp_helpdb croak` ausführen. Die Datenbank sollte jetzt 6 MB groß sein.

Die gleichen Ergebnisse können Sie auch mit dem SQL Server Enterprise Manager erreichen. Führen Sie die folgenden Schritte aus, um die weiter vorn in dieser Lektion erstellte Datenbank Frogger zu modifizieren:

1. Starten Sie SQL Server Enterprise Manager.
2. Erweitern Sie das Datenbankverzeichnis und öffnen sie das Dialogfeld DATENBANK-EIGENSCHAFTEN für die Datenbank *Frogger* (klicken Sie dazu mit der rechten Maustaste auf die Datenbank *Frogger* und wählen Sie EIGENSCHAFTEN aus dem Kontextmenü).
3. Klicken Sie auf der Registerkarte DATENDATEIEN des Dialogfelds FROGGER-EIGENSCHAFTEN in das leere Feld unter *FroggerData* und fügen Sie *FroggerData2* gemäß Abbildung 4.10 hinzu.
4. Geben Sie in der Spalte SPEICHERORT den neuen Dateinamen *F:\Programme\Microsoft SQL Server\MSSQL\DATA\FroggerData2.NDF* ein oder übernehmen Sie den Vorgabewert *FroggerData2_Data.NDF*. (Der Laufwerksbuchstabe kann je nach Ihrer konkreten Installation anders lauten.)
5. In der Spalte RESERVIERTER SPEICHER tragen Sie eine **2** ein.
6. Belassen Sie die Dateigruppe auf PRIMARY.
7. Schalten Sie im Abschnitt DATEIEIGENSCHAFTEN das Kontrollkästchen DATEI AUTOMATISCH VERGRÖßERN ein.
8. Wählen Sie im Abschnitt DATEIVERGRÖßERUNG die Option *In Megabyte* und geben Sie in das zugehörige Bearbeitungsfeld den Wert **2** ein.
9. Wählen Sie im Abschnitt MAXIMALE DATEIGRÖSSE die Option *Beschränkt auf (MB)* und geben Sie hier **4** ein.
10. Nachdem Sie die neue Datendatei hinzugefügt haben, erweitern Sie noch das Transaktionsprotokoll. Statt eine neue Transaktionsprotokolldatei hinzuzufügen, ändern Sie hier nur den reservierten Speicher von 5 MB auf 10 MB.
11. Klicken Sie auf OK, sobald Sie fertig sind.
12. Um zu überprüfen, ob die Datenbank geändert wurde, klicken Sie in der Konsolenstruktur des Enterprise Managers auf die Datenbank *Frogger*. Im Detailbereich spiegelt der Abschnitt RESERVIERTER SPEICHERPLATZ die geänderten Werte wider, wie es Abbildung 4.11 zeigt (gegebenenfalls müssen Sie den Befehl TASKPAD aus dem Menü ANSICHT wählen).

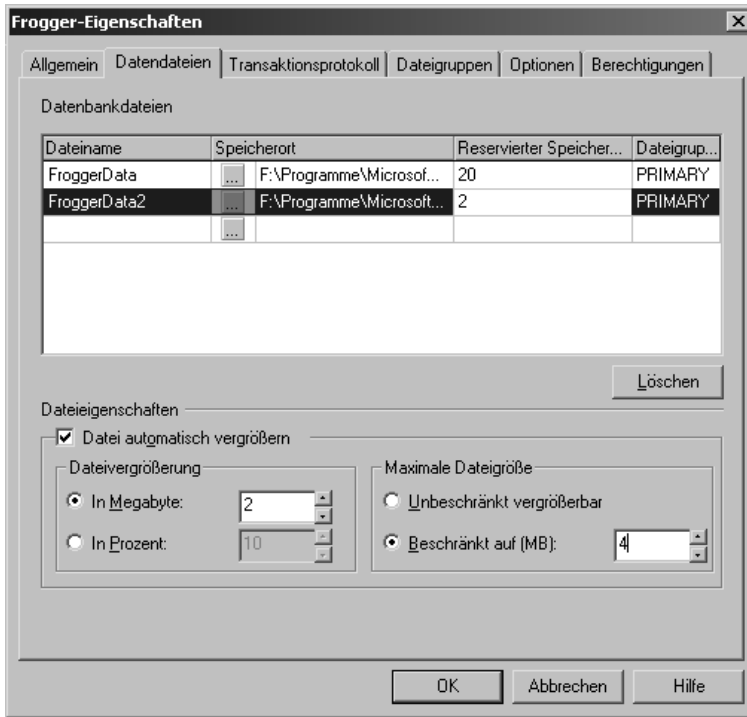


Abbildung 4.10:
Die Datendatei
FroggerData2 mit
einem reservierten
Speicher von 2 MB
zur Datenbank
Frogger hinzufügen

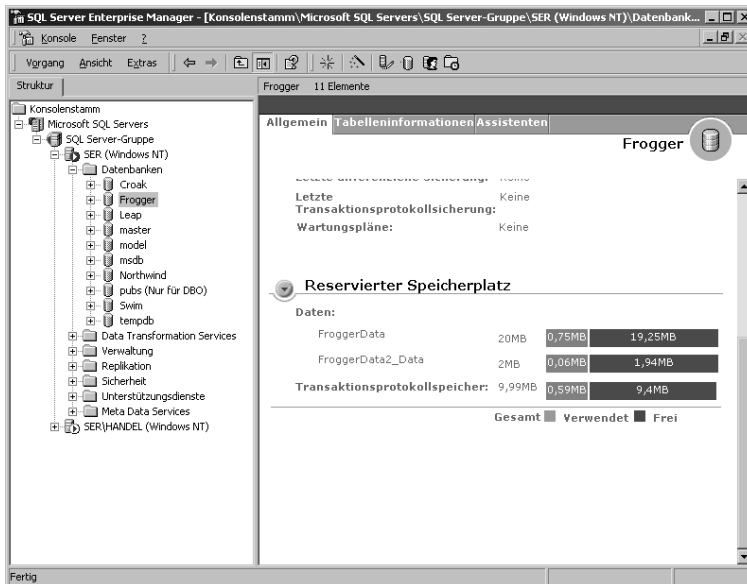


Abbildung 4.11:
Überprüfen, dass die
Datenbank über den
zugewiesenen Platz
verfügt.

Datenbanken verkleinern

Mit der Anweisung `DBCC SHRINKDATABASE` lässt sich eine gesamte Datenbank verkleinern. Die Syntax sieht folgendermaßen aus:



```
DBCC SHRINKDATABASE (  
    (database_name  
    [, target_percent]  
    [, {NOTRUNCATE | TRUNCATEONLY}])  
)
```

Die Parameter haben folgende Bedeutung:

- *database_name* ist der Name der zu verkleinernden Datenbank.
- *target_percent* ist der gewünschte Prozentsatz an freiem Speicher, der nach der Verkleinerung noch in der Datenbank übrig bleiben soll.
- `NOTRUNCATE` bewirkt, dass der freigegebene Speicherplatz in den Datenbankdateien beibehalten wird. Die Option *target_percent* wird ignoriert. Normalerweise geht freigegebener Speicherplatz wieder an das Betriebssystem zurück.
- `TRUNCATEONLY` bewirkt die Rückgabe des gesamten freien Speicherplatzes aus den Datendateien an das Betriebssystem. Wenn Sie `TRUNCATEONLY` angeben, wird der Parameter *target_percent* ignoriert und der gesamte freie Speicherplatz dem Betriebssystem zurückgegeben.

Was bewirkt dieser Befehl und wie funktioniert er? Im Wesentlichen versucht die Anweisung `SHRINKDATABASE`, alle Datendateien in der Datenbank zu verkleinern und dabei den spezifizierten prozentualen Anteil freizuhalten. Benutzte Seiten am Ende der Datendateien werden dabei vor die angegebene prozentuale Schwelle verschoben. Beispielsweise können Sie bei einer 10 MB großen Datenbank mit nur 5 MB gespeicherten Daten den Prozentwert auf 20 setzen. Das heißt, Sie möchten 20 Prozent der Datenbank als freien Speicher beibehalten. Alle in der Datenbank befindlichen Zeilen werden neu positioniert, wobei der freie Platz am Beginn der Datenbank verschwindet. Dieser Vorgang ist mit der Defragmentierung von Festplatten vergleichbar. Da nur 5 MB eigentliche Daten vorhanden sind und Sie 20 Prozent der Datenbank als freien Speicherplatz beibehalten möchten, ist als freier Speicherplatz 20 Prozent von 5 MB gleich 1 MB freizuhalten. Demzufolge ist die Datenbank auf 6 MB zu verkleinern, die anderen 4 MB werden an das Betriebssystem zurückgegeben.



Die Datenbank können Sie nicht kleiner machen, als es für die Speicherung der Daten erforderlich ist, und die Datenbank kann auch nicht kleiner als die Datenbank *model* werden. Außerdem lässt sich die Datenbank nicht unter die ursprünglich im Befehl `CREATE DATABASE` spezifizierte Größe bringen. Falls Sie die Originalgröße unterschreiten wollen, müssen Sie mit dem Befehl `DBCC SHRINKFILE` arbeiten.

Wenn Sie die Option `NOTRUNCATE` angegeben haben, werden die Daten verschoben, um am Ende der Datei freien Platz zu schaffen. Dieser Platz verbleibt aber in der Datenbank und wird nicht an das Betriebssystem zurückgegeben.

Die Option `TRUNCATEONLY` setzt nur den unbenutzten Speicherplatz frei, ohne die Daten zu verschieben. Falls am Ende der Datei freie Seiten liegen, werden diese an das Betriebssystem zurückgegeben.

Mit der Anweisung `DBCC SHRINKFILE` können Sie auch einzelne Dateien der Datenbank verkleinern, im Gegensatz zur Anweisung `DBCC SHRINKDATABASE`, die die gesamte Datenbank verkleinert. Die Syntax der Anweisung `DBCC SHRINKFILE` lautet:



```
DBCC SHRINKFILE {
    (file_name | file_id )
    [, target_size]
    [, {EMPTYFILE | NOTRUNCATE | TRUNCATEONLY}])
}
```

Im Befehl `DBCC SHRINKFILE` müssen Sie entweder den logischen Namen der Datei in der Datenbank oder die Datei-ID angeben. Die Datei-ID können Sie mit der gespeicherten Systemprozedur `sp_helpdb` ermitteln, die sie weiter oben in dieser Lektion bereits kennen gelernt haben.

Der Parameter *target_size* wird genauso verwendet, wie oben bereits für die Anweisung `SHRINKDATABASE` erläutert. Die Optionen `TRUNCATEONLY` und `NOTRUNCATE` sind ebenfalls dieselben wie oben erwähnt. Der Parameter `EMPTYFILE` ist insofern interessant, als dass er bewirkt, dass alle benutzten Seiten aus der angegebenen Datenbankdatei in andere Dateien derselben Dateigruppe verschoben werden. Nachdem alle Seiten verschoben wurden, wird die Datei als leer gekennzeichnet und die Speicherung von Daten in diese Datei wird verhindert. Das kann nützlich sein, wenn Sie die Parameter `REMOVE FILE` oder `REMOVE FILEGROUP` in der Anweisung `ALTER DATABASE` verwenden wollen, weil hier die angegebenen Dateien leer sein müssen.

Der Code in Listing 4.5 leert die Datei *FroggerData2* und verwendet die Option `REMOVE FILE`, um die Datei aus der Datenbank *Frogger* zu entfernen.

Listing 4.5: Leeren einer Datei und die Verwendung von REMOVE FILE



```
USE frogger
GO
DBCC SHRINKFILE (FroggerData2, EMPTYFILE)
GO
ALTER DATABASE Frogger REMOVE FILE FroggerData2
```



DbId	FileId	CurrentSize	MinimumSize	UsedPages	EstimatedPages
7	3	256	256	0	0

(1 row(s) affected)

DBCC-Ausführung abgeschlossen. Falls DBCC Fehlermeldungen ausgegeben hat, wenden Sie sich an den Systemadministrator.
Datei 'FroggerData2' wurde entfernt.

Im ersten Teil leert der Code von Listing 4.5 die Datei *FroggerData2* mit der Option *EMPTYFILE*. Danach ändert die zweite Hälfte des Abfragestapels die Datenbank und entfernt die Datei. Mit der Anweisung `exec sp_helpdb Frogger` können Sie überprüfen, ob die Datei tatsächlich entfernt wurde:



```
exec sp_helpdb Frogger
```



name	db_size	owner	dbid	created
Frogger	30.00 MB	SER\Admin	8	Dez 28 2000

name	fileid	filename	Filegroup	...
FroggerData	1	F:\Programme\Microsoft SQL ...	PRIMARY	.../
FroggerLog	2	F:\Programme\Microsoft SQL ...	NULL	.../

...	size	maxsize	growth	Usage
...	20480 KB	102400 KB	10240 KB	data only
...	10240 KB	15360 KB	1024 KB	log only

Diese Ausgabe zeigt, dass *FroggerData2* nicht mehr in der Datenbank *Frogger* enthalten ist. Mit dem Explorer können Sie sich davon überzeugen, dass auch die Datei *FroggerData2.ndf* im Verzeichnis ...\\DATA vom System gelöscht wurde.

4.5 Datenbanken umbenennen

Es gibt verschiedene Gründe, warum man eine Datenbank umbenennen möchte. Das kann zum Beispiel im Rahmen einer Umstrukturierung erfolgen, wenn Buchhaltung und Finanzabteilung zusammengelegt werden oder weil Sie eine Entwicklungsdatenbank in die Produktionsumgebung überführen wollen. In jedem Fall ist die Aufgabe an sich unkompliziert.

Um eine Datenbank umzubenennen, rufen Sie die gespeicherte Systemprozedur `sp_renamedb` auf. Dabei sind folgende Einschränkungen zu beachten bzw. Voraussetzungen zu erfüllen:

- Sie müssen ein Mitglied der festen Serverrolle *sysadmin* sein.
- Transact-SQL-Skripts können von Datenbanknamen abhängen. Überprüfen Sie die Datenbank nach derartigen Namen, damit die Skripts weiterhin korrekt laufen.
- Die Datenbank muss sich im Einzelbenutzermodus befinden.
- Die Namen von Datenbankdateien und Dateigruppen werden durch eine Umbenennung nicht beeinflusst.
- Sie müssen sich in der *master*-Datenbank befinden, um die gespeicherte Systemprozedur `sp_renamedb` ausführen zu können.

Im folgenden Beispiel benennen Sie die Datenbank *Frogger* in *TsingWa* um (Tsing Wa bedeutet im Chinesischen kleiner Frosch):



```
USE master
GO
EXEC sp_dboption Frogger, 'Single User', True
EXEC sp_renamedb 'Frogger', 'TsingWa'
EXEC sp_dboption TsingWa, 'Single User', False
GO
```



Der Datenbank-Name 'TsingWa' wurde festgelegt.

Mit `sp_helpdb` überprüfen Sie die Namensänderung:



```
EXEC sp_helpdb
```



name	db_size	owner	dbid	created
Croak	6.00 MB	SER\Administrator	10	Dez 28 2000
Leap	14.00 MB	SER\Administrator	9	Dez 28 2000
master	17.00 MB	sa	1	Aug 6 2000
model	1.25 MB	sa	3	Aug 6 2000
msdb	14.00 MB	sa	4	Aug 6 2000
Northwind	4.25 MB	sa	6	Aug 6 2000
pubs	2.00 MB	sa	5	Aug 6 2000
Swim	25.00 MB	SER\Administrator	8	Dez 28 2000
tempdb	8.50 MB	sa	2	Dez 29 2000
TsingWa	30.00 MB	SER\Administrator	7	Dez 29 2000

4.6 Arbeit mit Dateigruppen

Mithilfe von Dateigruppen ist es möglich, Datenbankobjekte wie Tabellen oder Indizes explizit einer bestimmten Gruppe von Datenbankdateien zuzuordnen. Das kann sowohl für die Verwaltung als auch für die Leistung von Vorteil sein. Wenn Sie zum Beispiel nicht mit RAID 5 arbeiten, können Sie Dateien in einer Dateigruppe auf physikalisch separaten Festplattenlaufwerken unterbringen und dann bestimmte Tabellen, Indizes und andere Datenbankobjekte den Dateien zuordnen, die zu dieser Dateigruppe gehören. Das kann zu einer Leistungssteigerung führen, weil Lese- und Schreibzugriffe auf die Datenbank gleichzeitig auf unterschiedlichen Festplatten erfolgen können. Für die Verwaltung ergibt sich der Vorteil, dass man einzelne Dateien einer Dateigruppe oder die gesamte Dateigruppe sichern und wiederherstellen kann.

Unter SQL Server gibt es zwei Arten von Dateigruppen: die Dateigruppe PRIMARY (die Standardgruppe) und benutzerdefinierte Dateigruppen. Die Dateigruppe PRIMARY muss die primäre Datendatei und alle anderen Dateien, die keiner anderen Dateigruppe zugeordnet sind, enthalten. Benutzerdefinierte Dateigruppen erstellt man, wenn man die Datenbank ändert und Dateien einer bestimmten Dateigruppe hinzufügt.

Für die Arbeit mit Dateigruppen sind ein paar Regeln zu beachten:

- Eine Datei kann nicht Mitglied mehrerer Dateigruppen sein.
- Die primäre Datendatei muss sich in der Dateigruppe PRIMARY befinden.
- Einer Dateigruppe können Sie Tabellen, Indizes, Text, ntext und Bilddaten zuordnen.



Die Bezeichner `ntext`, `nchar` und `nvarchar` beziehen sich auf Unicode-Datentypen. Unicode verwendet mehrere Bytes (normalerweise 2), um jedes angezeigte Zeichen darzustellen, während ASCII (Datentypen `char`, `varchar` und `text`) nur 1 Byte verwendet, um ein Zeichen zu speichern. Mehr zu Datentypen erfahren Sie am Tag 9.

- Alle Systemdateien müssen sich in der Dateigruppe `PRIMARY` in der primären Datei befinden.
- Wenn für die Dateigruppe `PRIMARY` kein Platz mehr zur Verfügung steht, werden neue Seiten nicht automatisch auf die benutzerdefinierten Dateigruppen verteilt.

In den folgenden Beispielen erstellen Sie eine neue Dateigruppe und fügen ihr eine Datendatei hinzu. Dann ändern Sie die Datenbank – Sie modifizieren die Eigenschaften der Dateigruppe, um sie zu einem Teil der Standarddateigruppe zu machen. Die Standarddateigruppe `PRIMARY` wird angelegt, wenn Sie die Datenbank neu erstellen. Es kann jeweils nur eine Dateigruppe die Standarddateigruppe sein. Wenn Sie eine neue Dateigruppe als Standardgruppe markieren, kommen neue Dateien in diese Dateigruppe, sofern Sie nicht ausdrücklich `PRIMARY` angeben. Am Ende der Beispiele löschen Sie die Dateigruppe wieder.

Eine neue Dateigruppe erstellen Sie mit den folgenden Anweisungen:



```
ALTER DATABASE Leap
ADD FILEGROUP LeapGroup1
GO
```



Befehl(e) erfolgreich abgeschlossen.

Mit den folgenden Anweisungen können Sie sich vergewissern, dass die Dateigruppe Teil der Datenbank *Leap* ist:



```
use Leap
GO
EXEC sp_helpfilegroup
```



groupname	groupid	filecount
PRIMARY	1	2
LeapGroup1	2	0

Jetzt ändern Sie die Datenbank und fügen der Dateigruppe eine neue Datei hinzu:



```
ALTER DATABASE Leap
ADD FILE
(Name = LeapDataG1, FILENAME =
'F:\Programme\Microsoft SQL Server\MSSQL\DATA\LeapDataG1.ndf',
SIZE = 2)
TO FILEGROUP LeapGroup1
GO
```



Die Datenbank wird um 2.00 MB auf dem Datenträger 'LeapDataG1' erweitert.

Führen Sie die gespeicherte Prozedur `sp_helpfilegroup` erneut aus:



```
USE Leap
GO
EXEC sp_helpfilegroup
```



groupname	groupid	filecount
-----	-----	-----
PRIMARY	1	2
LeapGroup1	2	1

Die benutzerdefinierte Dateigruppe `LeapGroup1` können Sie zu einem Teil der Standarddateigruppe machen, indem Sie ihre Dateigruppeneigenschaften verändern. Die Dateigruppe können Sie auch als `READONLY` oder `READWRITE` markieren. Weiterführende Informationen zu diesen Optionen finden Sie in der Onlinedokumentation.

Um die neue Dateigruppe als Standarddateigruppe zu kennzeichnen, führen Sie folgende Anweisungen aus:



```
USE Leap
GO
ALTER DATABASE Leap
MODIFY FILEGROUP LeapGroup1 DEFAULT
```



Die Dateigruppe-Eigenschaft 'DEFAULT' wurde festgelegt.

Um Ihre neue Vorgabedateigruppe zu testen, können Sie eine neue Datei ohne Angabe der Dateigruppe hinzufügen:



```
USE pubs
GO
ALTER DATABASE Leap
ADD FILE
(NAME = LeapDataTest,
 FILENAME =
 'F:\Programme\Microsoft SQL Server\MSSQL\DATA\LeapDataTest.ndf',
 SIZE = 2)
GO
```



Die Datenbank wird um 2.00 MB auf dem Datenträger 'LeapDataTest' erweitert.

Die Zuordnung der Dateien zu den Dateigruppen können Sie sich mithilfe der gespeicherten Systemprozedur `sp_helpfile` anzeigen lassen.

Um eine Dateigruppe zu entfernen, müssen Sie zunächst die Dateigruppe `PRIMARY` zur Standarddateigruppe machen. Dann leeren Sie Dateien in der alten Dateigruppe. Schließlich können Sie die Dateien und dann die Dateigruppe wie im folgenden Beispiel sicher entfernen:



```
USE Leap
GO

ALTER DATABASE Leap
MODIFY FILEGROUP [PRIMARY] DEFAULT
GO

ALTER DATABASE Leap
REMOVE FILE LeapDataG1
Go

ALTER DATABASE Leap
REMOVE FILE LeapDataTest
GO

ALTER DATABASE Leap
REMOVE FILEGROUP LeapGroup1
GO
```

Als Ergebnis sollten Sie Meldungen erhalten, dass die Dateigruppe `LeapDataG1` und die darin enthaltenen Dateien gelöscht wurden. Die erfolgreiche Ausführung können Sie wieder mit `sp_helpfile` überprüfen.

Die beschriebenen Aufgaben lassen sich auch mit dem Enterprise Manager durchführen, genau wie beim Erstellen von Dateien. Führen Sie die folgenden Schritte aus, um eine neue Datenbankdatei zu erstellen und sie einer neuen Dateigruppe hinzuzufügen:

1. Erweitern Sie im Enterprise Manager die Konsolenstruktur bis zur Datenbank Croak, öffnen Sie das Eigenschaftsblatt der Datenbank und gehen Sie zur Registerkarte DATENDATEIEN.
2. Um eine neue Dateigruppe anzulegen und ihr eine Datei hinzuzufügen, klicken Sie auf das leere Feld unter CroakData2 und geben CroakDataG1 ein.
3. Legen Sie *F:\Programme\Microsoft SQL Server\Mssql\Data\CroakDataG1.ndf* als Speicherort fest.
4. In der Spalte RESERVIERTER SPEICHERPLATZ (MB) geben Sie 1 ein.
5. In der Spalte DATEIGRUPPE fügen Sie die Gruppe CroakGroup1 hinzu (siehe Abbildung 4.12).

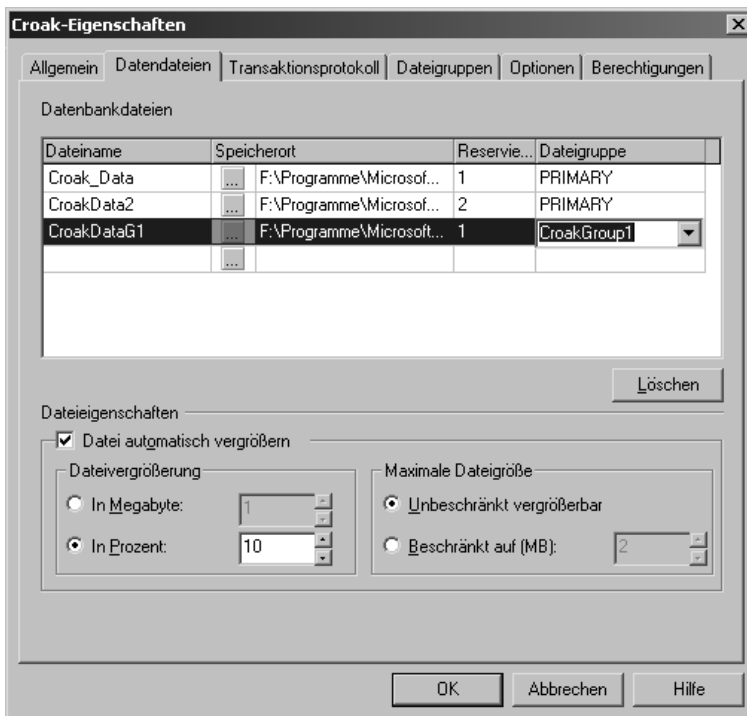


Abbildung 4.12:
Wählen Sie die
Dateigruppe in den
Eigenschaften für
die Datenbank aus.



6. Klicken Sie auf OK, wenn Sie fertig sind.
7. Zur Kontrolle öffnen Sie erneut das Eigenschaftsblatt für die Datenbank *Croak* und gehen auf die Registerkarte DATEIGRUPPEN. Hier sollten sowohl *CroakGroup1* als auch *Primary* aufgelistet sein.

Dieser Abschnitt bietet lediglich einen Überblick über Dateigruppen und wie man sie anlegt. Dateigruppen können sehr komplex sein und werden in diesem Buch nicht weiter behandelt. Für weiterführende Informationen sei auf die einschlägige Literatur und die Onlinedokumentation verwiesen.

4.7 Eine Datenbank löschen

Von Zeit zu Zeit müssen Sie eine Datenbank aus dem System entfernen, beispielsweise weil die Datenbank nicht mehr benötigt wird. Auch das Entfernen einer Datenbank von SQL Server ist unkompliziert. Wie bei den bisher beschriebenen Operationen lässt sich auch diese Aufgabe sowohl mit dem Enterprise Manager als auch per Transact-SQL durchführen.

Bevor Sie eine Datenbank tatsächlich entfernen, sollten Sie folgende Punkte beachten:

- Das Entfernen einer Datenbank entfernt die Informationen über diese Datenbank aus den Systemtabellen und löscht die Daten- und Protokolldateien aus dem System.
- Eine gelöschte Datenbank lässt sich nur durch das Einlesen einer Sicherheitskopie wiederherstellen.
- Um eine Datenbank löschen zu können, dürfen keine Benutzer mit der Datenbank verbunden sein.
- Sie müssen Mitglied der Datenbankrolle *db_owner* (oder der Serverrolle *sysadmin*) sein, um eine Datenbank entfernen zu können.
- Wenn Sie die Anweisung `DROP DATABASE` ausführen, muss die Datenbank *master* aktiv sein.



Die Syntax der Anweisung `DROP DATABASE` lautet:

```
DROP DATABASE database_name, database_name2...
```

Mit der Anweisung `DROP DATABASE` lassen sich mehrere Datenbanken auf einmal löschen. Im Beispiel löschen Sie sowohl die Datenbank *TsingWa* (vormals *Frogger*) als auch die Datenbank *Croak*. Führen Sie die folgenden Anweisungen im SQL Server Query Analyzer aus:



```
USE MASTER
GO
DROP DATABASE TsingWa, Croak
GO
```



Die Datenbankdatei 'F:\Programme\Microsoft SQL Server\MSSQL\DATA\FroggerLog.ldf' wird gelöscht.

Die Datenbankdatei 'F:\Programme\Microsoft SQL Server\MSSQL\DATA\FroggerData.mdf' wird gelöscht.

Die Datenbankdatei 'F:\Programme\Microsoft SQL Server\MSSQL\data\CroakDataG1_Data.NDF' wird gelöscht.

Die Datenbankdatei 'F:\Programme\Microsoft SQL Server\MSSQL\DATA\CroakLog2.ndf' wird gelöscht.

Die Datenbankdatei 'F:\Programme\Microsoft SQL Server\MSSQL\DATA\CroakData2.ndf' wird gelöscht.

Die Datenbankdatei 'F:\Programme\Microsoft SQL Server\MSSQL\data\Croak_Log.LDF' wird gelöscht.

Die Datenbankdatei 'F:\Programme\Microsoft SQL Server\MSSQL\data\Croak_Data.MDF' wird gelöscht.

Wie die Ausgabe zeigt, wurden alle Dateien, die mit den beiden Datenbanken verbunden sind, gelöscht und die Datenbanken aus dem System entfernt. Zur Überprüfung können Sie `sp_helpdb` ausführen.

Eine Datenbank können Sie natürlich auch mit dem SQL Server Enterprise Manager löschen. Führen Sie dazu die folgenden Schritte aus:

1. Starten Sie den SQL Server Enterprise Manager und erweitern Sie die Konsolenstruktur bis zum Datenbankordner.
2. Klicken Sie mit der rechten Maustaste auf die Datenbank *Leap* und wählen Sie LÖSCHEN aus dem Kontextmenü.
3. Klicken Sie im Bestätigungsdialogfeld DATENBANK LÖSCHEN – LEAP auf JA.

Mehr ist nicht zu tun. Die Datenbank ist aus dem System verschwunden und die Datenbankdateien sind ebenfalls gelöscht worden.

4.8 Zusammenfassung

Heute haben Sie gelernt, wie Sie Datenbanken erstellen, ändern und löschen und wie Sie die verschiedenen Datenbankoptionen einstellen können. Wenn Sie eine Datenbank erzeugen, wird sie in mindestens zwei Dateien auf der Festplatte abgelegt. Eine Datei enthält die Daten, die Systemtabellen und andere Datenbankobjekte, während die andere Datei das Transaktionsprotokoll aufnimmt. Unter SQL Server 2000 können die Datenbanken dynamisch anwachsen, wenn Sie die entsprechenden Optionen für die Größenanpassung der Datenbank oder des Transaktionsprotokolls angeben. Außerdem hat diese Lektion kurz erläutert, was Dateigruppen sind und welche Vorteile sie für die Leistung und Verwaltung bringen. Am Ende der Lektion haben Sie noch gelernt, wie man eine Datenbank löscht.

4.9 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F *Werde ich in der Praxis viele Datenbanken erzeugen müssen?*

A Das hängt davon ab, in welcher Arbeitsumgebung Sie sich befinden. In meiner Beratertätigkeit erstelle ich nicht sehr viele Datenbanken. Diejenigen, die ich erzeuge, sind zum größten Teil Entwicklungsdatenbanken, in denen Testdaten gespeichert und mit denen Anwendungen entwickelt werden. Wenn die Entwicklung abgeschlossen ist, werden sie in der Produktionsumgebung neu erstellt und in Betrieb genommen.

F *Sollte ich den SQL Server Enterprise Manager oder den SQL Server Query Analyzer für die Erzeugung und Veränderung von Datenbanken verwenden?*

A Falls die Datenbank für eine Produktionsumgebung bestimmt ist oder häufig neu erstellt werden muss, bietet sich der SQL Server Query Analyzer an. Das hat den Vorteil, dass man leicht Skriptdateien für die spätere Wiederverwendung speichern kann. Wenn die Datenbank nur der Entwicklung dient und sich schnell und einfach wiederherstellen lässt, empfiehlt sich der SQL Server Enterprise Manager.

F *Wie groß sollten meine Datenbanken und Transaktionsprotokolle sein?*

A Die Anfangsgröße der Datenbank sollte ausreichen, um alle geplanten Daten aufzunehmen. Auch die Indizes sind in die Größenberechnung einzubeziehen. Mehr zu Datenbankgrößen erfahren Sie am Tag 9, wenn es um Datentypen und deren Speicherung geht. Die Größe des Transaktionsprotokolls hängt davon ab, wie viele Änderungen an der Datenbank vorgenommen werden sollen. Wenn die Datenbank nur einen lesenden Zugriff zulässt, kann das Transaktionsprotokoll sehr klein ausfallen – vielleicht 5% der Größe der Datenbankdateien. Wenn in der Datenbank eine große Zahl von Änderungs-, Einfüge- und Löschoperationen durchzuführen sind, brauchen Sie wahrscheinlich ein größeres Transaktionsprotokoll. Eine Größe von etwa 25 bis 30% der Datenbankgröße ist angemessen. Bei normaler Verwendung der Datenbank sollte das Transaktionsprotokoll etwa 10 bis 15% der Datenbankgröße ausmachen.

F *Sollte ich lieber viele kleine oder besser wenige große Dateien in meiner Datenbank haben?*

A Das hängt von Ihnen ab. Viele Dateien bedeuten einen höheren Verwaltungsaufwand. Sie erhalten dadurch allerdings auch gewisse Vorteile. Statt eine Sicherheitskopie einer riesigen Datei durchzuführen, können Sie zum Beispiel viele kleinere Dateien sichern.

F *Welche Datenbankoptionen werden am häufigsten geändert?*

A Das hängt davon ab, was Sie mit Ihrer Datenbank tun. Ich habe zum Beispiel einige Datenbanken, die jede Nacht mit Daten aus Textdateien von einer AS400 gefüllt werden. Am Einfachsten lassen sich diese Textdateien in die Datenbanktabellen kopieren, wenn man den Parameter Auswahl/Massenkopieren (select into/bulkcopy) auf TRUE setzt und dann ein Skript zum Massenkopieren verwendet. Es kann auch vorkommen, dass Sie die Option 'Nur für DBO' verwenden müssen. Im Laufe dieses Buches werden Sie noch auf Beispiele stoßen, die diese Optionen benötigen.

F *Wie wichtig sind Dateigruppen?*

A Dateigruppen sind hauptsächlich ein Hilfsmittel zur Leistungsoptimierung. Die meisten Datenbanken funktionieren ohne benutzerdefinierte Dateigruppen genauso gut.

F *Warum sollte ich eine Datenbank löschen wollen und wie oft passiert so etwas?*

A Es gibt viele Gründe, um Dateien aus dem System zu entfernen. Beispielsweise legt man in der Entwicklungsphase Datenbanken an, die man nach dem Testen nicht mehr benötigt. Man kann auch eine defekte Datenbank entfernen, um sie komplett wiederherzustellen.



F Sollte ich die automatische Datenbankvergrößerung verwenden oder deaktivieren?

- A Die Möglichkeit, Datenbanken automatisch vergrößern zu lassen, ist eine der größten Stärken von SQL Server 2000. Damit wird eine dynamische Größenanpassung der Datenbank mit geringem Verwaltungsaufwand ermöglicht. Falls Sie diese Option verwenden, empfiehlt es sich, auch die maximale Dateigröße für die Datenbank anzugeben.

Kontrollfragen

1. Welche SQL-Anweisungen brauchen Sie, um folgende Datenbank zu erzeugen?

Datenbank: *Accounting*

Dateigröße: 20 MB mit einer maximalen Größe von 40 MB und Dateivergrößerung in 2 MB-Schritten

Protokollgröße: 5 MB mit einer maximalen Größe von 10 MB und Dateivergrößerung in 1 MB-Schritten

2. Was bewirkt das folgende Programm?

```
ALTER DATABASE Test
ADD FILE
(NAME = TestData2,
FILENAME =
'F:\Programme\Microsoft SQL Server\mssql\data\TestData2.ndf',
SIZE = 10)
```

3. Welche Anweisungen müssen Sie ausführen, um die folgenden vier Datenbanken zu löschen: Von, Ron, Don und Christina?

Übungen

Diese Übung besteht aus einer einzigen Aufgabe, die in mehrere Schritte unterteilt ist. Der jeweils nächste Schritt der Aufgabe erfordert es, dass Sie die vorangehenden Schritte erfolgreich absolviert haben. Bitte lesen Sie sich vor der Durchführung jeder Teilaufgabe die gesamte Anweisung für den jeweiligen Schritt durch.

1. Erstellen Sie eine neue Datenbank namens *Frog* mit den folgenden Eigenschaften:
- ▶ Zwei Datendateien mit den Namen *FrogData1* und *FrogData2*. Die anfängliche Speicherreservierung beträgt 3 MB. Die Datendateien sollen mit einer Schrittweite von 2 MB auf maximal 20 MB anwachsen können.

- ▶ Zwei Transaktionsprotokolldateien namens *FrogLog1* und *FrogLog2*. Die anfängliche Größe dieser Dateien beträgt 1 MB. Die maximale Größe der Transaktionsprotokolldateien beträgt jeweils 5 MB.
 - ▶ Standardwerte für die automatische Dateivergrößerung.
2. Fügen Sie der Datenbank eine dritte Datei mit dem Namen *FrogData3* und Standardwerten für alle weiteren Optionen hinzu.
 3. Verkleinern Sie die Datenbank um 20%.
 4. Leeren Sie die Transaktionsprotokolldatei *FrogLog2*.
 5. Entfernen Sie die Datei *FrogLog2*.
 6. Benennen Sie die Datenbank in *TreeFrog* um.
 7. Löschen Sie die Datenbank.



Sicherheit

Gestern haben Sie untersucht, wie man Datenbanken und die zugehörigen Dateien erstellt. Außerdem haben Sie erfahren, wie und wann man Dateigruppen mit SQL Server 2000 verwendet. Auch wenn Sie Dateigruppen selten brauchen, ist es gut, ihre grundlegende Arbeitsweise zu kennen. Jede Aktion, die Sie in SQL Server unternehmen, wird authentifiziert. Dazu gehört auch das Erstellen von Datenbanken und Dateien. Damit Sie mit SQL Server erfolgreich arbeiten können, ist das Verständnis des Sicherheitsmodells unerlässlich. In der heutigen Lektion untersuchen Sie, wie SQL Server die Verbindungen zu SQL Server und zu individuellen Datenbanken authentifiziert, und Sie lernen, wie die Windows 2000-Authentifizierung arbeitet. Die Lektion geht auch auf die Unterschiede der Sicherheitsmechanismen unter Windows 2000 und Windows 98 ein. Berechtigungen für einzelne Datenbankobjekte behandelt Tag 6.

5.1 Das SQL Server-Sicherheitsmodell

Bisher haben Sie die Verbindung zu SQL Server 2000 ganz einfach im gemischten Modus (den Sie bei der Installation gewählt haben) per Windows-Authentifizierung mit Ihren Zertifikaten als Mitglied der lokalen Administratorengruppe hergestellt. Wenn Sie die Verbindung herstellen, laufen einige Vorgänge ab, die auf den ersten Blick nicht zu erkennen sind.

Läuft SQL Server unter Windows 2000, wird die Sicherheit an drei verschiedenen Stellen überprüft, wenn Sie eine Verbindung zu SQL Server 2000 herstellen wollen (siehe Abbildung 5.1). Die Überprüfung findet durch Windows 2000, SQL Server selbst (in Form einer SQL Server-Anmeldung) und dann auf der Ebene der jeweiligen Datenbank (in Form des Benutzernamens für die Datenbank) statt.

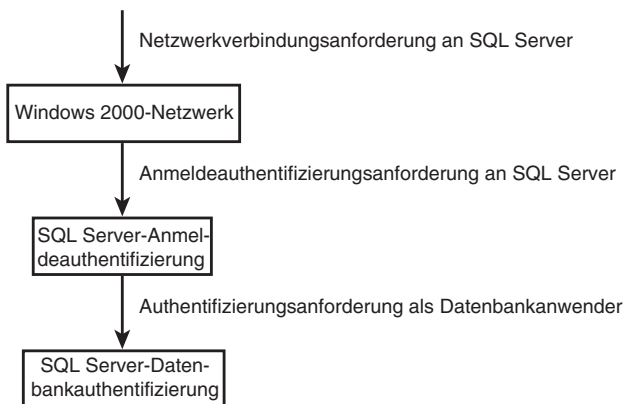


Abbildung 5.1:
Schichten der Netzwerk- und SQL
Server-Authentifizierung



Eine Anmeldung sagt noch nichts darüber aus, auf welche Datenbanken Sie zugreifen können – diese Rechte werden über den Datenbankbenutzernamen festgelegt. Außerdem haben Sie bisher noch nicht versucht, auf irgendwelche Datenbankobjekte zuzugreifen. Dafür sind Berechtigungen erforderlich, die Sie erst in der morgigen Lektion untersuchen.

Windows-Authentifizierung

Wenn Sie sich von Ihrem Clientcomputer mit einem Windows NT/2000-Computer verbinden, auf dem SQL Server 2000 läuft, kann Windows die Validierung der Netzwerkverbindung fordern oder nicht – das ist abhängig von der SQL Server-Netzwerkbibliothek. Wenn Sie Named Pipes oder Multiprotokoll als SQL Server-Netzwerkbibliothek verwenden, muss Ihre Anmeldung als autorisierte Windows 2000-Verbindung validiert werden, bevor Sie mit SQL Server arbeiten dürfen.

Wie Abbildung 5.2 zeigt, passieren sowohl Named Pipes als auch Multiprotokoll den Windows 2000-Serverdienst, der die Netzwerkvalidierung für die Verbindungsanforderung eines Benutzers durchführt. Somit müssen Sie über gültige Windows 2000-Sicherheitszertifikate verfügen, um sich mit dem Windows 2000-Servercomputer zu verbinden. Weil die TCP/IP-Socket-Netzwerkbibliothek den Serverdienst nicht durchläuft, brauchen Sie kein gültiges Windows-Konto, um sich mit SQL Server 2000 zu verbinden. Beginnend mit der Version 2000 von SQL Server ist TCP/IP-Sockets als Standardnetzwerkbibliothek eingerichtet, so dass dieses Problem nicht so kritisch wie bei vorherigen Versionen ist. Wenn Sie jedoch Clients mit SQL Server 7.0 (oder früher) betreiben, verwenden diese möglicherweise noch die Standardnetzwerkbibliothek Named Pipes.

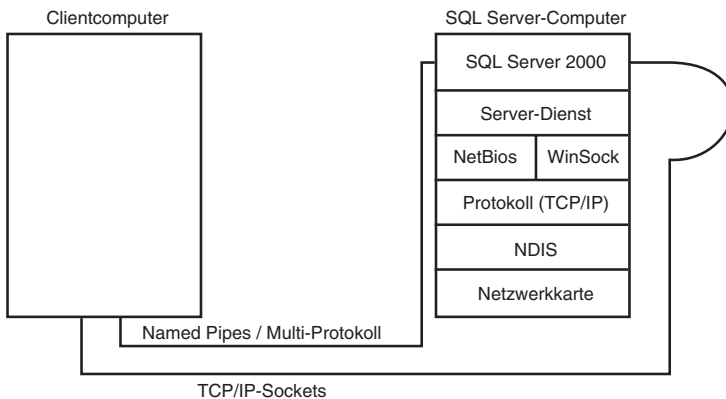


Abbildung 5.2:
SQL Server-Netzwerk-
kommunikation



Windows 98/ME autorisiert Netzwerkverbindungen nicht in der gleichen Weise wie Windows NT/2000; daher beginnen Sicherheitsüberprüfungen bei SQL Servern, die auf einem 9x Computer laufen, auf der Stufe der Anmeldung am SQL Server. Windows 9x unterstützt auch keine Named Pipes für Serveranwendungen, so dass Sie sich nicht mit einem Windows 9x-Server nach diesem Protokoll verbinden können.



Wenn Sie die Sicherheitsarchitektur verstanden haben, kann Ihnen das bei der Problembehebung behilflich sein. Erhalten Sie beispielsweise die bertichtigte Meldung »Verbindung mit Server kann nicht hergestellt werden; SQL Server existiert nicht oder Zugriff verweigert«, fehlen Ihnen möglicherweise die Berechtigungen, um die Verbindung zum Windows 2000-Computer herzustellen, auf dem SQL Server installiert ist. Um das Problem einzukreisen, erstellen Sie eine Freigabe auf dem SQL Server-Computer und versuchen, sich mit dem Server (oder mit einer eventuell vorhandenen Freigabe) zu verbinden. Funktioniert die Verbindung nicht oder werden Sie nach Benutzername/Kennwort gefragt, können Sie sich nicht mit SQL Server verbinden. Um sicher zu sein, dass es sich tatsächlich um ein derartiges Problem handelt, beenden Sie die Verbindung zur oben genannten Freigabe und versuchen dann, sich erneut mit SQL Server zu verbinden.

Funktioniert die Verbindung jetzt, müssen Sie die Sicherheitseinstellungen auf Ihrem Windows 2000-Computer verändern. Dazu haben Sie folgende Möglichkeiten:

- ▶ Sie aktivieren das Konto *Gäste*. Dieses Vorgehen ist nicht zu empfehlen, weil Sie dadurch die Windows 2000-Sicherheit bis zu einem bestimmten Grad aushebeln. Um das System sicher zu halten, erstellen Sie für jeden Benutzer ein eigenes Konto.
- ▶ Sie erstellen mit der Computerverwaltung ein lokales Windows 2000-Konto. Verwenden Sie dabei dasselbe Kennwort wie bei dem Konto, mit dem Sie momentan angemeldet sind. Diese Möglichkeit ist als letzter Ausweg anzusehen.
- ▶ Sie machen Ihren Server zum Mitglied einer Domäne, die entweder zur Domäne mit Ihrem Netzwerkkonto gehört bzw. dieser vertraut (eine bessere Möglichkeit) oder die sich in derselben Active Directory-Gesamtstruktur befindet. Das ist die beste verfügbare Option und Sie sollten sie nach Möglichkeit in Ihrem Netzwerk einsetzen. Konsultieren Sie Ihren Netzwerkadministrator oder die einschlägige Literatur, um sich detaillierter mit Domänen, dem Active Directory und der Sicherheit von Windows 2000 zu befassen.

Mit dem Übergang zur Standardnetzwerkbibliothek TCP/IP-Sockets sollte es kein Problem mehr darstellen, eine Verbindung zu SQL Server herzustellen. Wenn Sie aber aus bestimmten Gründen gezwungen sind, mit Named Pipes zu arbeiten, treffen die obigen Regeln zu.

SQL Server-Authentifizierung

Um sich mit SQL Server zu verbinden, müssen Sie einen gültigen SQL Server-Anmeldename und ein gültiges Kennwort bereitstellen (oder eine gültige integrierte Windows-Verbindung haben). In Kürze erfahren Sie Näheres zu diesem Thema. Wenn Ihre Anmeldenkennungen gültig sind, werden Sie mit SQL Server verbunden. Bei ungültigen Kennungen wird Ihnen der Zugriff verweigert – selbst wenn die Windows-Netzwerkauthentifizierung (Ihre Netzwerkverbindung zum Windows 2000-Server) erfolgreich war.

SQL Server-Datenbankbenutzername

Der Zugriff auf jede Datenbank des Systems muss explizit erlaubt sein. Auf eine Datenbank kann man nach verschiedenen Methoden zugreifen. Diese Methoden lernen Sie später in dieser Lektion kennen. Wenn Sie keinen Datenbankbenutzernamen haben, wird Ihnen der Zugriff auf die Datenbank verweigert.

Berechtigungen

Die letzte Ebene der Sicherheit sind Berechtigungen. Nachdem Sie sich erfolgreich bei SQL Server angemeldet und eine Datenbank aktiviert haben, brauchen Sie explizite Rechte, auf Datenbankobjekte zuzugreifen (entweder zum Lesen oder zum Bearbeiten). Auf Berechtigungen geht die morgige Lektion ausführlich ein.

5.2 Die Sicherheitsmodi von SQL Server

SQL Server 2000 erlaubt zwei unterschiedliche Sicherheitsmodi: Windows-Authentifizierungsmodus und gemischten Modus (sowohl Windows-Authentifizierung als auch SQL Server-Authentifizierung). Der Sicherheitsmodus bestimmt, ob nur Windows oder sowohl Windows als auch SQL Server für die Validierung der SQL Server-Verbindungsanforderungen zuständig sind. Diese Validierung ist völlig unabhängig von der Netzwerkauthentifizierung durch Windows, die Sie bereits kennen gelernt haben. Diese Unterschiede müssen Sie beherrschen, damit Sie SQL Server-Sicherheit richtig implementieren können.

Sicherheit im gemischten Modus

Im gemischten Authentifizierungsmodus von SQL Server kann sich ein Benutzer entweder mithilfe der Windows-Authentifizierung oder mithilfe der SQL Server-Authentifizierung zu einer Instanz von SQL Server verbinden. Der gemischte Modus stellt die beste Wahl für Abwärtskompatibilität dar und bietet die umfassendste Konnektivität mit Netzwerkcomputern, die nicht unter Windows laufen, beispielsweise für Benutzer von Novell NetWare.

SQL Server-Authentifizierung

Die SQL Server-Authentifizierung ist der Modus, in dem SQL Server ohne Windows-Unterstützung einen Anmeldenamen und ein Kennwort von einem Benutzer entgegennimmt und die Zertifikate überprüft. Auf Windows 9x-Computern wird dieses Verfahren immer verwendet, auf einem Windows 2000-Computer ist es optional. Dieser Modus ist nicht so sicher wie die Windows-Authentifizierung und er empfiehlt sich nicht für SQL Server, die vertrauliche Daten speichern. Anmeldeinformationen speichert SQL Server intern (in der Systemtabelle *sysxlogins* der Datenbank *master*). Alle Versionen vor SQL Server 7.0 verwenden einen als *Standardsicherheit* bezeichneten Modus der SQL Server-Authentifizierung, der dem SQL Server-Authentifizierungsmodus entspricht. Lassen Sie sich also nicht durch die unterschiedlichen Begriffe verwirren.

Wenn Sie die Verbindung über eine Anmeldung herstellen, verwenden Sie die SQL Server-Authentifizierung. Daher gibt es einen Eintrag in der Systemtabelle *sysxlogins* für das Benutzerkonto *sa* sowie ein Kennwort, falls Sie ein solches zugewiesen haben (die Standardinstallation verwendet ein leeres Kennwort, in den Beispielen des Buches wird *kennwort* verwendet). Nach der Installation von SQL Server 2000 gibt es nur das Benutzerkonto *sa*. Auf einem Windows 2000-Computer ist die lokale Administratorengruppe auch als Äquivalent von *sa* hinzugefügt worden (als Mitglieder der Sicherheitsrolle *sysadmin*).



Die Anmeldung *sa* wird standardmäßig hinzugefügt, selbst wenn Sie die Standardinstallation übernehmen und sich im integrierten Windows-Modus befinden.

Kennwörter

Die Kennwörter für die SQL Server-Authentifizierung speichert die Spalte *password* der Tabelle *sysxlogins* in der Datenbank *master*. Um sich die Einträge in der Tabelle *sysxlogins* anzusehen, starten Sie den SQL Server Query Analyzer und führen die folgende Abfrage aus (dazu müssen Sie als Mitglied der Rolle *sysadmin* angemeldet sein, mehr dazu später):



```
SELECT substring(name,1,25) AS Name,
       substring(password,1,20) as Kennwort,
       language as Sprache
FROM sysxlogins
```



Name	Kennwort	Sprache

-		
sa	0x0100E606C13F95ED7A68EE32B	Deutsch
SER\SQLService	NULL	Deutsch
VORDEFINIERT\Administrato	NULL	Deutsch
NULL	NULL	NULL

(4 row(s) affected)



Die Zeilen in der Ergebnismenge für die Windows 2000-Gruppe *VORDEFINIERT\Administratoren* sowie für das SQL Server-Dienstkonto (SQLService) sind – wie bereits erwähnt – bei einer SQL Server-Installation unter Windows 9x nicht vorhanden.

Die dritte Zeile in der oben gezeigten Ergebnismenge repräsentiert die lokale Windows-Gruppe *Administratoren* – mehr dazu in Kürze. Die zweite Zeile stellt das Dienstkonto dar, das Sie für SQL Server während des Setupprogramms gewählt haben. Die Anmeldung *sa* in der ersten Zeile ist mit dem während des Setups gewählten Kennwort – im Beispiel *kennwort* in der verschlüsselten Form – angegeben. In dieser Systemtabelle sind alle Anmeldungen und Kennwörter für die SQL Server-Authentifizierung enthalten. Ein leeres Kennwort ist durch den Wert NULL in der Spalte *Kennwort* (bzw. *password*) angegeben. Alle nichtleeren Kennwörter erscheinen in verschlüsselter Form als Hexadezimalzahl.

Bleibt hier die Sicherheit auf der Strecke, wenn sich Kennwörter ohne weiteres in einer Abfrage anzeigen lassen? Nein, und zwar aus folgenden Gründen:

- Nur ein Mitglied der Rolle *sysadmin* (einschließlich der Anmeldung *sa*) kann die Kennwortspalte sehen. Kein anderer Benutzer kann diese Spalte anzeigen, sofern er nicht ausdrücklich das Recht dafür erhalten hat (Tag 6 zeigt, wie man das macht).
- Die Verschlüsselung ist ein Einwegalgorithmus. Ein verschlüsseltes Kennwort lässt sich nicht entschlüsseln. Wenn Sie sich anmelden, wird das eingegebene Kennwort verschlüsselt und mit dem verschlüsselten Kennwort in der Tabelle *sysxlogins* verglichen. Stimmen die Kennwörter überein, wird der Zugriff auf den Server gewährt. Andernfalls erscheint die Fehlermeldung »Fehler bei der Anmeldung für den Benutzer 'xyz'« und Sie können keine Verbindung herstellen. Wenn Sie mit Windows-Authentifizierung arbeiten, speichert SQL Server überhaupt keine Kennwörter.



Falls Sie Sicherheitsbedenken – insbesondere in Bezug auf Kennwörter – haben, sollten Sie die Windows-Authentifizierung verwenden.

Benutzernamen bei SQL Server-Authentifizierung verwalten

Um den Zugriff auf den Server zu ermöglichen, müssen Sie als Erstes Benutzernamen einrichten. Das lässt sich mit der gespeicherten Systemprozedur `sp_addlogin` oder mit dem Enterprise Manager erledigen. Es sei noch einmal darauf hingewiesen, dass unter Windows 2000 die Windows-Authentifizierung als Sicherheitsmodus zu bevorzugen ist (mehr dazu später in dieser Lektion).

Die Syntax der gespeicherten Prozedur `sp_addlogin` sieht wie folgt aus:



```
sp_addlogin [@loginame =] 'login'
    [,[@passwd =] 'password'
    [,[@defdb =] 'database'
    [,[@deflanguage =] 'language'
    [,[@sid =] 'sid'
    [,[@encryptopt =] 'encryption_option']]]]]
```

Die Parameter haben folgende Bedeutung:

- *login* ist der Name, unter dem sich der Benutzer anmelden soll. Der Name muss ein gültiger SQL Server-Bezeichner sein (beginnend mit einem Buchstaben oder den Zeichen #, @, oder _, die übrigen Zeichen können diese Zeichen oder eine Kombination aus Buchstaben und Ziffern sein – bis zu 128 Unicode-Zeichen).
- *passwd* ist das Kennwort des Benutzernamens. Der Standard ist NULL.
- *database* ist die Standarddatenbank, mit der der Benutzername nach der Anmeldung verbunden ist. Der Standard ist *master*.
- *deflanguage* ist die Standardsprache, die dem Benutzer zugewiesen wird, wenn er sich bei SQL Server anmeldet. Der Standard ist *US_English*.
- *sid* ist die Sicherheits-ID, die Sie für einen Benutzer spezifizieren können (diese Option wird nicht empfohlen).
- *encryptopt* gibt an, ob das Kennwort in den Systemtabellen verschlüsselt zu speichern ist. Mit diesem Feature können Sie ein verschlüsseltes Kennwort aus einer früheren Version von SQL Server herausziehen und direkt in SQL Server 2000 einfügen, so dass der Benutzername ebenfalls mit diesem SQL Server arbeitet. Um die Verschlüsselung abzuschalten, geben Sie hier die literale Zeichenfolge `skip_encryption` an.

Um dem Server einen Benutzernamen hinzuzufügen, öffnen Sie den SQL Server Query Analyzer und melden sich an. Führen Sie den folgenden Transact-SQL-Befehl aus:



```
Exec sp_addlogin "IhrName", "IhrKennwort"
```



Für die Beispiele im Buch lautet diese Anweisung `sp_addlogin "frank", "kennwort"`. Wenn Sie alle Beispiele des Buches unverändert übernehmen möchten, sollten Sie diesen Namen und das Kennwort verwenden. Andernfalls ersetzen Sie einfach alle Stellen, an denen »frank« auftaucht, durch Ihren eigenen Namen. Da sich außerdem Ihre Standardinstanz im gemischten Modus befindet, müssen Sie diese Befehle für Ihre benannte Instanz (*HANDEL*) ausführen.

Wenn Sie die bereits weiter oben angegebene Abfrage bezüglich der Tabelle *sysxlogins* erneut ausführen, finden Sie eine neue Zeile mit Ihrem Namen und einem verschlüsselten Kennwort. Wenn Sie eine neue Verbindung zu SQL Server erstellen, können Sie sich mit dem eben hinzugefügten Namen und Kennwort anmelden.

Als Nächstes wollen Sie vielleicht Ihr Kennwort ändern. Auch hier lässt sich das entweder über den SQL Server Enterprise Manager oder eine gespeicherte Prozedur erledigen. Die gespeicherte Systemprozedur `sp_password` hat folgende Syntax:



```
sp_password [[@old =] 'old_password',]
            {[@new =] 'new_password'}
            [,[@loginame =] 'login']
```

Die Parameter haben folgende Bedeutung:

- *old_password* ist das alte Kennwort.
- *new_password* ist das neue Kennwort.
- *login* ist der Benutzername. Wenn Sie als Mitglied der Rollen *sysadmin* oder *securityadmin* angemeldet sind, können Sie die Kennwörter aller Benutzer ändern. Dabei brauchen Sie nicht einmal das alte Kennwort eines Benutzers zu kennen – Sie können das alte Kennwort einfach mit NULL überschreiben.



Die Mitglieder der Rolle *securityadmin* können die Kennwörter aller Benutzer ändern, *außer* die Kennwörter von Mitgliedern der Rolle *sysadmin*. Dagegen können Mitglieder der Rolle *sysadmin* ausnahmslos alle Kennwörter ändern.

Die gespeicherte Prozedur `sp_password` rufen Sie zum Beispiel wie folgt auf:

```
Exec sp_password NULL, 'newpass', 'frank'
```



Obwohl Sie Franks altes Kennwort nicht wissen, können Sie es ändern, weil Sie der Systemadministrator sind. Normale Benutzer können das aus Sicherheitsgründen nicht tun und müssen ihr altes Kennwort angeben, um es ändern zu können.

Es empfiehlt sich, Kennwörter regelmäßig zu ändern. Leider bietet SQL Server 2000 keine Möglichkeit, Kennworteinschränkungen und andere Sicherheitsmaßnahmen zu erzwingen. Deshalb sollten Sie mit integrierter Sicherheit arbeiten. Unter Windows 2000 lassen sich Regeln für die minimale Länge, die Häufigkeit der Änderung und die Komplexität von Kennwörtern festlegen – die so genannten *Kennwortrichtlinien*.

Für einen Benutzer möchten Sie vielleicht auch die Standarddatenbank oder die Standardsprache ändern. Das können Sie mit den gespeicherten Systemprozeduren `sp_defaultdb` und `sp_defaultlanguage` (und natürlich wieder mit dem Enterprise Manager) erledigen:



```
sp_defaultdb loginname, defdb  
sp_defaultlanguage loginname [, language]
```

Die Parameter sind die gleichen wie bei den weiter oben behandelten Prozeduren. Mit diesen Optionen können Sie einfach verschiedene Felder in der Tabelle `sysxlogins` ändern (die Standarddatenbank bzw. die Standardsprache).

Zur Verwaltung von Benutzernamen sind außerdem die gespeicherten Systemprozeduren `sp_helplogins` und `sp_droplogin` verfügbar. Mit der Prozedur `sp_helplogins` können Sie einen Bericht über die Benutzernamen erstellen, die für den Server eingerichtet sind. Abbildung 5.3 zeigt ein Beispiel für die Ausführung dieser gespeicherten Prozedur.

In Abbildung 5.3 fällt auf, dass erneut die Spalte *SID* erscheint. Diese Spalte wurde bereits im Zusammenhang mit der Prozedur `sp_addlogin` erläutert. Mit der *SID* (Sicherheits-ID) verfolgt SQL Server die Benutzernamen und Benutzer innerhalb von SQL Server. Bei integrierten Windows-Benutzern (mehr dazu in Kürze) ist das die global eindeutige Netzwerkidentifikationsnummer, die den Benutzer im Windows-Netzwerk identifiziert.

Die gespeicherte Systemprozedur `sp_droplogin` entfernt einen Benutzernamen aus der Tabelle `sysxlogins`. Danach kann sich der Benutzer nicht mehr unter diesem Namen bei SQL Server anmelden.



Die Syntax der gespeicherten Prozedur `sp_droplogin` lautet:

```
sp_droplogin login
```

Der Parameter *login* hat die gleiche Bedeutung wie bei den anderen gespeicherten Prozeduren dieser Gruppe.

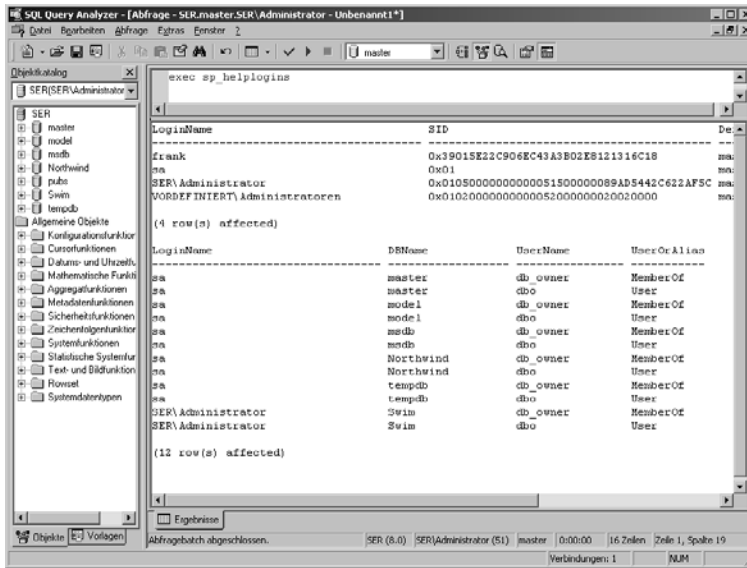


Abbildung 5.3:
Die Ergebnisse von
sp_helplogins

Sie wissen nun, wie man die Benutzernamen für die SQL Server-Authentifizierung mit gespeicherten Systemprozeduren verwaltet. Später in dieser Lektion erfahren Sie, wie man die gleichen Aufgaben mit dem Enterprise Manager realisiert.

Windows-Authentifizierung im gemischten Modus



Die *Windows-Authentifizierung* ist die andere Sicherheitsoption im gemischten Modus. Der nächste Abschnitt geht näher auf diese Option ein. In Bezug auf Windows-Verbindungen ist eine Bemerkung zur Terminologie angebracht: Eine über Windows-Authentifizierung hergestellte Verbindung bezeichnet man als *vertraute Verbindung*. Dabei handelt es sich um eine Verbindung, die Windows NT oder Windows 2000 authentifiziert hat.

Windows-Authentifizierungsmodus



Im Windows-Authentifizierungsmodus müssen Sie, nachdem Sie die Verbindung zu SQL Server hergestellt haben, dem SQL Server Ihre Windows-Sicherheitsinformationen (auch als *Zugriffstoken* bezeichnet) präsentieren. Diese Authentifizierungsinformationen erstellen Sie während der Anmeldung zu einem Windows 2000-Netzwerk und diese Sicherheitsinformationen werden stillschweigend übergeben, ohne dass Sie sich um die Weiterleitung kümmern



müssen. Den Zugriff auf SQL Server können Sie direkt über die Windows 2000-Sicherheitskonten oder indirekt über Windows 2000-Gruppen gewähren.

Das Beste am Windows-Authentifizierungsmodus ist, dass sich die Benutzer nicht separat an SQL Server anmelden müssen. Das entspricht dem Konzept, dass sich Benutzer nur einmal im Netzwerk anmelden und sich nur ein einziges Kennwort merken müssen. Darüber hinaus können Sie sich die Tatsache zunutze machen, dass die meisten Benutzer ohnehin bereits über Windows-Konten verfügen, was den Verwaltungsaufwand von Benutzerkonten für Ihren SQL Server verringert.

Windows 2000-Gruppen und -Benutzer einrichten

Wenn man die Sicherheit im Windows-Authentifizierungsmodus konfiguriert, beginnt man nicht bei SQL Server, sondern unternimmt einen Ausflug in das Dienstprogramm Computerverwaltung (oder Benutzermanager bei Windows NT-Computern). Zuerst erstellen Sie Windows 2000-Gruppen für Benutzer, legen die Benutzer an (falls sie noch nicht existieren) und fügen sie dann in die eben erstellten Gruppen hinzu. Schließlich geben Sie den Benutzern die Berechtigungen für die Anmeldung bei SQL Server. Wählen Sie dazu **START / PROGRAMME / VERWALTUNG / COMPUTERVERWALTUNG**, um das Programm *Computerverwaltung* zu starten (wenn Sie eine Domäne verwalten, verwenden Sie statt dessen das Programm *Active Directory-Benutzer und -Computer*). Erweitern Sie den Knoten *Lokale Benutzer und Gruppen*, wie es Abbildung 5.4 zeigt.

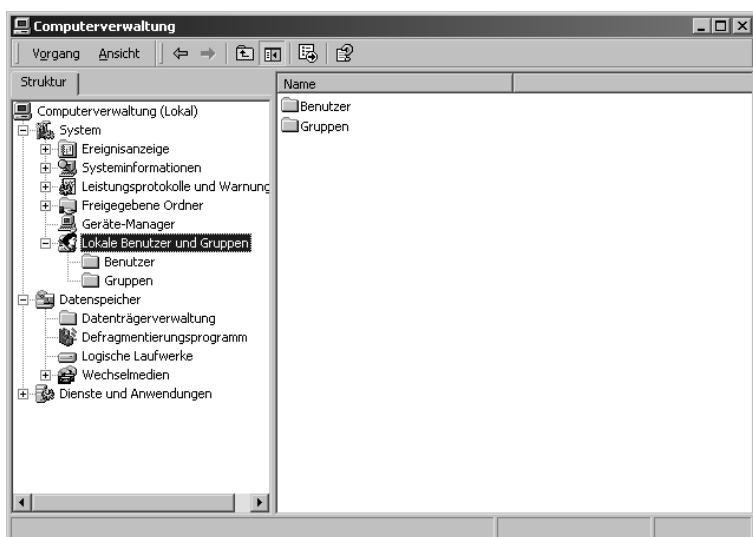


Abbildung 5.4:
Computerverwaltung: Lokale Benutzer und Gruppen



Bei Windows 9x-Installationen ist die *Computerverwaltung* nicht verfügbar und auf Windows NT-Computern heißt diese Anwendung *Benutzermanager* oder *Benutzermanager für Domänen*.

Auf dem Computer, auf dem SQL Server läuft, müssen Sie neue lokale Gruppen in die Sicherheitskontendatenbank hinzufügen. Dabei handelt es sich um eine Windows 2000-interne Datenbank und nicht um eine Datenbank von SQL Server. Möchten Sie statt des Computers, an dem Sie momentan arbeiten, einen Remotecomputer steuern, klicken Sie mit der rechten Maustaste auf den Knoten *Computerverwaltung (Lokal)*, wählen VERBINDUNG ZU ANDEREM COMPUTER HERSTELLEN aus dem Kontextmenü und geben dann den Namen des zu verwaltenden Computers ein. Wenn Sie eine Active Directory-Domäne verwalten, müssen Sie mit dem bereits erwähnten Programm *Active Directory-Benutzer und -Computer* arbeiten.

Wie bei den meisten Windows 2000-bezogenen Operationen im Buch müssen Sie am Computer als Mitglied der lokalen Administratorengruppe angemeldet sein. Andernfalls können Sie die folgenden Operationen nicht durchführen.

Erstellen Sie nun drei lokale Gruppen für die Nutzung in SQL Server: eine Gruppe für Benutzer, die sich als SQL Server-Administratoren anmelden können (solche Windows 2000-Benutzer, die SQL Server vollständig verwalten können), eine weitere Gruppe für das Verkaufspersonal und eine weitere für Marketing-Benutzer. Dann können Sie jeder Gruppe Benutzer zuweisen. Dabei lässt sich festlegen, welche Berechtigungen der jeweilige Benutzer hat, nachdem er sich angemeldet hat. Zu diesem Thema erfahren Sie später in dieser und der nächsten Lektion mehr.

Die erste Gruppe dient dem administrativen Zugriff auf SQL Server. Markieren Sie den Ordner *Gruppen*, klicken Sie mit der rechten Maustaste darauf und wählen Sie NEUE GRUPPE aus dem Kontextmenü. Füllen Sie Felder GRUPPENNAME und BESCHREIBUNG entsprechend Abbildung 5.5 aus.

Nehmen Sie nun Ihr Konto gemäß Abbildung 5.6 in die Gruppe auf. Klicken Sie auf die Schaltfläche HINZUFÜGEN, um eine Liste der verfügbaren Konten anzuzeigen. Die Liste enthält die globalen Gruppen und Konten Ihrer Standarddomäne oder nur die Konten, wenn Ihr SQL Server-Computer kein Mitglied einer Domäne ist.

Wenn Ihr Computernamen nicht in der Liste SUCHEN IN aufgeführt ist, wählen Sie ihn aus der Dropdown-Liste aus. Im Dialogfeld sehen Sie nun Ihren Computernamen, Ihre Standarddomäne und alle vertrauten Domänen. Wenn Sie eine Domäne verwenden, wählen Sie Ihr Domänenkonto aus, um es der Liste hinzuzufügen, statt ein doppeltes Windows 2000-Konto anzulegen. Wenn Sie nun die richtige Liste vor sich haben, fügen Sie Ihren Benutzernamen der Gruppe hinzu, indem Sie Ihren Kontonamen markieren und dann auf HINZUFÜGEN und OK klicken.



Abbildung 5.5:
Eine neue lokale Gruppe für Windows
2000-Sicherheit hinzufügen

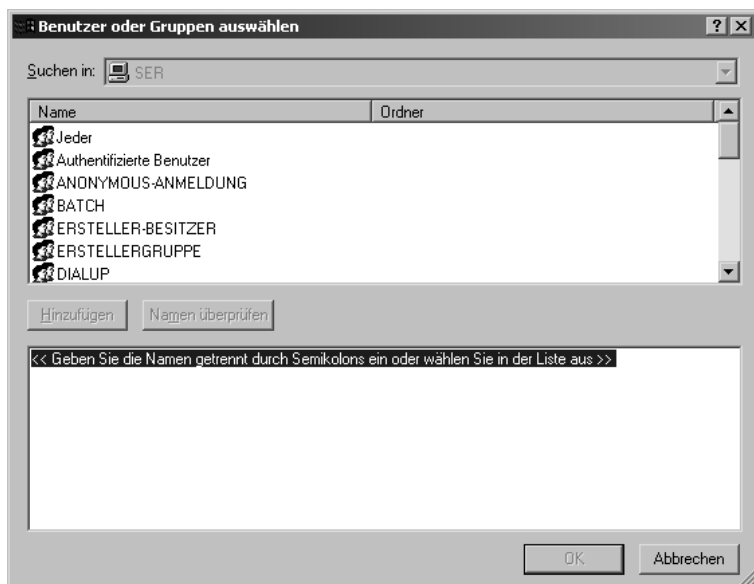


Abbildung 5.6:
Das Dialogfeld zum
Hinzufügen von
Benutzern und
Gruppen



Sie sollten auch das Konto hinzufügen, das Sie bei der Installation des SQL Server-Agent-Dienstes verwendet haben (SQLDienst, wenn Sie das Setup entsprechend den Schritten von Lektion 2 durchgeführt haben). Der SQL Server-Agent-Dienst benötigt für die korrekte Funktion eine vertraute Verbindung zu

SQL Server. Wenn Sie das Konto nicht hinzufügen und der SQL Server-Agent keine vertraute Verbindung zu SQL Server herstellen kann, steht die gesamte Funktionalität dieses Dienstes nicht zur Verfügung – einschließlich Aufträge, Warnungen und Replikation.

Haben Sie für diesen Dienst die Option *Konto 'Lokales System'* verwenden gewählt und kein Domänenbenutzerkonto angegeben, brauchen Sie hier für dieses Konto nichts Spezielles zu unternehmen. Das gilt auch, wenn Sie Windows 2000-Administratoren als SQL Server-Administratoren übernehmen möchten (das ist die Standardkonfiguration).

Die fertig gestellte Gruppe sollte nun Abbildung 5.7 entsprechen.



Abbildung 5.7:
Das vollständig ausgefüllte Dialogfeld
Neue Gruppe

Klicken Sie auf ERSTELLEN. Wiederholen Sie diesen Vorgang, um eine Gruppe *Verkauf* und eine Gruppe *Marketing* zu erstellen. Fügen Sie diesmal aber andere Benutzer hinzu. Auf dem Beispielsystem existieren zu diesem Zweck die Benutzer *Ann*, *Bob* und *Cat*, wobei *Ann* und *Bob* der Gruppe *Verkauf* sowie *Bob* und *Cat* der Gruppe *Marketing* zugeordnet wurden. *Bob* gehört absichtlich beiden Gruppen an. Um einer vorhandenen Windows 2000-Gruppe einen Benutzer hinzuzufügen, markieren Sie die Gruppe, klicken mit der rechten Maustaste darauf und wählen MITGLIED HINZUFÜGEN aus dem Kontextmenü. Im Dialogfeld EIGENSCHAFTEN VON *Gruppenname* klicken Sie auf Hinzufügen und wählen die in die Gruppe aufzunehmenden Benutzer aus, wie Sie es bereits weiter oben kennen gelernt haben. Falls Sie die genannten »Dummy-Benutzer« jetzt anlegen möchten,

denken Sie daran, das Kontrollkästchen **BENUTZER MUSS KENNWORT BEI DER NÄCHSTEN ANMELDUNG ÄNDERN** auszuschalten. Außerdem haben alle Benutzer für die Beispiele im Buch das Kennwort `kennwort` erhalten.

Windows 2000-Konten das Anmelden bei SQL Server ermöglichen

Nachdem Sie die Benutzer und Gruppen eingerichtet haben, ist diesen Gruppen der Zugriff auf SQL Server zu gewähren. Für diese Aufgabe stehen die gespeicherten Systemprozeduren `sp_grantlogin`, `sp_revokelogin` und `sp_denylogin` zur Verfügung, die funktionell den bereits behandelten Prozeduren `sp_addlogin` und `sp_droplogin` entsprechen.

Die Anmeldeberechtigung vergeben Sie zuerst an die Windows-Gruppen und dann je nach Bedarf an einzelne Benutzer. Diese Vorgehensweise erfordert den geringsten Aufwand und erlaubt dennoch, die Anmelderechte individuell zu kontrollieren. Die Berechtigung zur Anmeldung erteilen Sie mit der gespeicherten Systemprozedur `sp_grantlogin`:



```
sp_grantlogin [@loginame =] 'login'
```

Der Parameter *login* ist der Name des Windows 2000-Benutzers oder der Windows 2000-Gruppe, dem bzw. der Sie das Recht zum Anmelden bei SQL Server gewähren. Der Benutzername ist in der Form `SECURITY_DATABASE\Benutzername` anzugeben – beispielsweise `MeineDomäne\Frank`.

Mit der folgenden Anweisung gewähren Sie der Gruppe Verkauf die Berechtigungen zum Anmelden bei SQL Server:



```
Exec sp_grantlogin 'SER\Verkauf'
```

Ersetzen Sie *SER* durch Ihren Computernamen (oder Domänennamen, falls Sie SQL Server auf einem Domänencontroller ausführen). Zur Bestätigung erhalten Sie etwa folgende Meldung:



```
Für 'SER\Verkauf' wurde Anmeldezugriff erteilt.
```

Jetzt kann sich jeder Windows 2000-Benutzer, der Mitglied der Gruppe *Verkauf* ist, bei SQL Server anmelden. Probieren Sie das aus, indem Sie sich bei Windows 2000 als Ann oder Bob anmelden, den SQL Server Query Analyzer starten und die Option Windows-Authentifizierung wählen, um eine vertraute Verbindung zu SQL Server 2000 herzustellen.

len. In der Titelleiste des Abfragefensters erscheint Ihr Windows 2000-Benutzername. Beachten Sie, dass sich die Verbindung auf *Sie* bezieht, selbst wenn die Gruppe *Verkauf* die eigentliche Entität ist, der die Anmelderechte für SQL Server gewährt wurden. Eine derartige Verbindungsstrategie garantiert eine Überwachungsverfolgung und erlaubt später eine wesentlich bessere Kontrolle der Berechtigungen.



Falls Sie sich nicht als Ann oder Bob anmelden können und eine Fehlermeldung erhalten, dass es die lokalen Regeln des Systems nicht erlauben, sich interaktiv anzumelden, müssen Sie noch einige Zusatzschritte ausführen: Melden Sie sich an Ihrem Computer als Administrator an, wählen Sie **START / AUSFÜHREN** und geben Sie **GPedit.msc** ein. Erweitern Sie die Knoten *Windows-Einstellungen*, *Sicherheitseinstellungen*, *Lokale Richtlinien*, *Zuweisen von Benutzerrechten* und markieren Sie die Option *Lokal anmelden*. Klicken Sie mit der rechten Maustaste darauf, wählen Sie **SICHERHEITSEINSTELLUNGEN** im Kontextmenü und klicken Sie im Dialogfeld **LOKALE SICHERHEITSRICHTLINIE** auf **HINZUFÜGEN**, um die Gruppe *Verkauf* in die Liste aufzunehmen. Das Dialogfeld **LOKALE SICHERHEITSRICHTLINIE** sollte dann etwa Abbildung 5.8 entsprechen. Klicken Sie auf **OK**, und alle Benutzer können sich lokal am Computer anmelden.

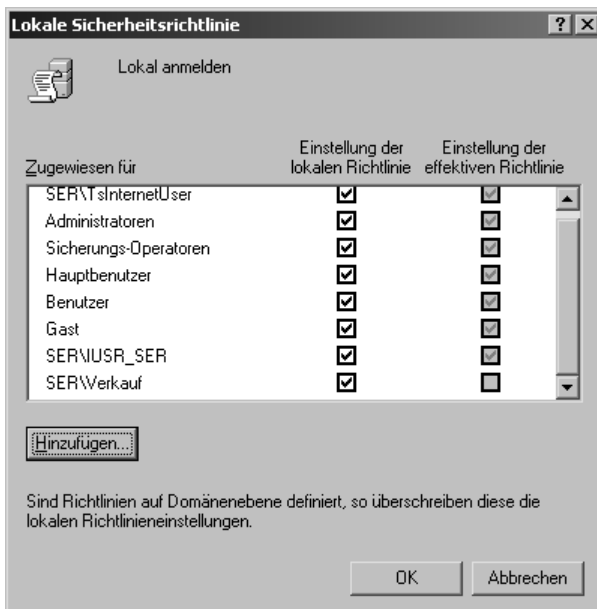


Abbildung 5.8:
Das Dialogfeld Lokale Sicherheitsrichtlinie



Einem Windows 2000-Benutzer oder einer Windows 2000-Gruppe können Sie auch die Rechte zum Anmelden an SQL Server entziehen. Das lässt sich mit der gespeicherten Systemprozedur `sp_revokelogin` realisieren:



```
sp_revokelogin [@loginame =] 'login'
```

Der Parameter *login* ist der Name der Windows 2000-Gruppe oder des Windows 2000-Benutzers, deren bzw. dessen Recht zum Anmelden an SQL Server Sie entziehen wollen.

Die Prozedur `sp_revokelogin` entfernt ein vorher gewährtes Recht. Die Anweisung im folgenden Beispiel nimmt deshalb allen Mitgliedern der Gruppe *Verkauf* die Möglichkeit, sich an SQL Server anzumelden, weil der Gruppe *Verkauf* dieses Recht vorher gewährt wurde:



```
Exec Sp_revokelogin 'SER\Verkauf'
```

Die nächste Anweisung ist dagegen wirkungslos, weil der Gruppe *Marketing* vorher keine Anmelderechte für SQL Server gewährt wurden:



```
Exec sp_revokelogin 'SER\Marketing'
```

Mit der Prozedur `sp_revokelogin` können Sie zwar einzelnen Benutzern das Recht zur Anmeldung entziehen, das hat aber keinen Einfluss auf die Rechte, die der Benutzer aus Gruppen erhält. Die folgende Anweisung entzieht zwar Ann das Recht zur Anmeldung, beeinflusst aber nicht die Rechte, die Ann aus Gruppen erhält:



```
Exec sp_revokelogin 'SER\Ann'
```

Hat also die Gruppe *Verkauf* das Recht zur Anmeldung und Sie führen die obige Anweisung aus, kann sich Ann trotzdem weiterhin bei SQL Server anmelden.

Wollen Sie festlegen, dass sich jeder Benutzer in der Gruppe Verkauf mit Ausnahme von Ann anmelden kann, müssen Sie die gespeicherte Systemprozedur `sp_denylogin` verwenden:



```
sp_denylogin [@loginame =] 'login'
```

Der Parameter *login* ist der Name der Windows 2000-Gruppe oder des Windows 2000-Benutzers, der bzw. dem Sie das Recht zum Anmelden an SQL Server verwehren wollen.

Probieren Sie den folgenden Code als Beispiel aus (ersetzen Sie wieder *SER* durch den Namen der entsprechenden Sicherheitsdatenbank):



```
Exec sp_grantlogin 'SER\Verkauf'  
Exec sp_denylogin 'SER\Ann'
```

Melden Sie sich nun an Ihrem Windows 2000-System als *Ann* an und versuchen Sie, sich mit SQL Server zu verbinden. Die Anmeldeberechtigungen werden Ihnen verweigert. Wenn Sie sich dagegen bei Windows 2000 als *Bob* anmelden und die Verbindung zu SQL Server herstellen, klappt alles wunderbar. Mit den obigen Anweisungen haben Sie zwar *Ann* die Berechtigungen entzogen, *Bob* hat aber als Mitglied der Gruppe *Verkauf* die Berechtigungen zum Anmelden erhalten. Das Verweigern von Rechten hat immer Vorrang gegenüber allen anderen zugelassenen Rechten.



SQL Server 2000 verwendet die Security Support Provider Interface (SSPI), um Aufrufe zu Windows 2000 auszuführen. Manchmal erhält man einen Fehler, der sich auf den SSPI-Kontext bezieht. Diese Nachricht bedeutet, dass irgendwas schiefgegangen ist, wenn SQL Server versucht hat, Windows 2000 aufzurufen; normalerweise heißt das, dass sich der Standort des Domänencontrollers nicht ermitteln lässt.

Sicherheitsmodus einstellen

Bislang haben Sie gelernt, wie Sie Benutzer für den SQL Server-Authentifizierungsmodus sowie Windows 2000-Benutzer und -Gruppen in SQL Server hinzufügen. Sie müssen aber auch wissen, welcher Sicherheitsmodus zu verwenden ist. Um festzustellen, welchen Sicherheitsmodus der Server derzeit verwendet, starten Sie SQL Server Enterprise Manager und klicken in der Konsolenstruktur mit der rechten Maustaste auf den Server. Wählen Sie **EIGENSCHAFTEN** aus dem Kontextmenü und gehen Sie auf die Registerkarte **SICHERHEIT**.

Wenn Sie an einem Windows 2000-Computer arbeiten, erscheint ein Dialogfeld wie in Abbildung 5.9. Hier können Sie für die Authentifizierung entweder *Nur Windows* (Standardeinstellung) oder *SQL Server und Windows* wählen. Auf einem Windows 9x-Computer sind die meisten Einstellungen in diesem Dialogfeld nicht verfügbar, weil es bei diesen

Systemen keine Windows-Authentifizierung gibt. Um die Sicherheitseinstellungen zu ändern, klicken Sie einfach auf die gewünschte Option. Die Veränderung wirkt sich erst aus, nachdem Sie den SQL Server-Dienst (*MSSQLServer* bei einer Standardinstanz, *MSSQL\$Instanzname* bei einer benannten Instanz) neu gestartet haben.

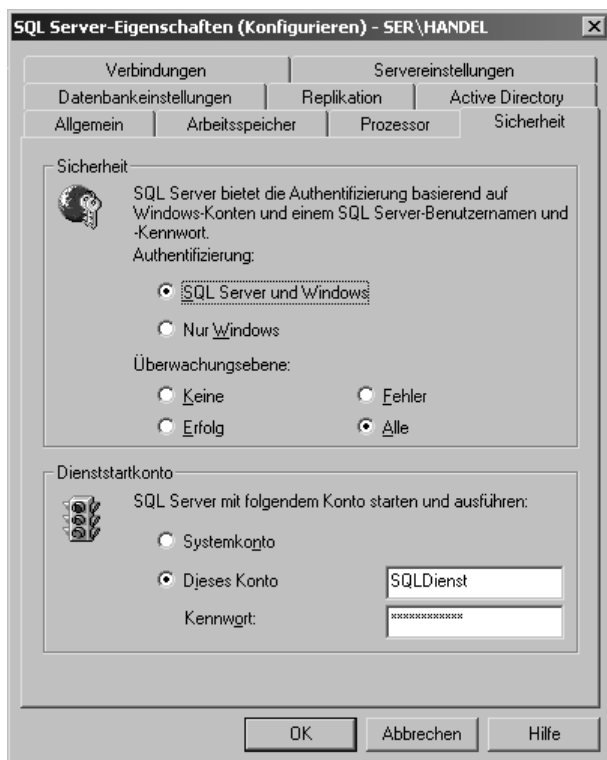


Abbildung 5.9:
Das Dialogfeld zur Sicherheitskonfiguration



Es empfiehlt sich, die Überwachung einzuschalten. In der Standardeinstellung ist keine Überwachung aktiviert. Als Überwachungsebene können Sie Fehler (gescheiterte Anmeldungen bei SQL Server), Erfolg (akzeptierte Anmeldungen an SQL Server) oder beide (die Option *Alle*) festlegen. Zu empfehlen ist beides, wie es Abbildung 5.9 zeigt. Diese Überwachungsvorgänge können Sie sich sowohl im Fehlerprotokoll von SQL Server als auch in der Ereignisanzeige von Windows 2000 ansehen. Damit die Überwachung beginnt, müssen Sie den SQL Server-Dienst anhalten und neu starten. SQL Server 2000 bietet noch wesentlich weitreichendere Sicherheitsüberwachungen, auf die Tag 20 im Detail eingeht.

Schließlich können Sie im Abschnitt DIENSTSTARTKONTO das Konto ändern, mit dem SQL Server gestartet und ausgeführt wird. Diese Option wurde bereits in Lektion 2 behandelt.

Nachdem Sie Änderungen in diesem Dialogfeld vorgenommen haben, müssen Sie SQL Server anhalten und neu starten, damit die Änderungen wirksam werden. Das sollte allerdings kein Problem darstellen, weil Sie derartige Änderungen kaum mehrmals durchführen.

Benutzernamen grafisch verwalten

An dieser Stelle können Sie wiederholen, was Sie in der heutigen Lektion gelernt haben. SQL Server unterscheidet zwei Arten von Benutzernamen:

- Windows NT/2000-Benutzernamen, entweder über Gruppen oder einzelne Benutzerkennungen
- SQL Server-Benutzernamen, die in der Systemtabelle `sysxlogins` in der Datenbank `master` gespeichert sind

Jede Option hat ihre Vorteile. SQL Server-Benutzernamen können Sie auf der Windows 9x-Plattform verwenden und müssen keine Windows 2000-Domänen auf Windows 2000-Systemen eingerichtet haben. Allerdings sind Windows 2000-Benutzernamen zu bevorzugen, wenn Sie bereits in einem entsprechend konfigurierten Netzwerk arbeiten, weil diese Benutzernamen bereits erstellt sind und jeden Benutzer in der Organisation eindeutig kennzeichnen. Sie wissen bereits, wie man Benutzernamen in beiden Optionen erstellt und wie man Anmeldungen an SQL Server gewährt. Diese Aufgaben haben Sie mit gespeicherten Systemprozeduren wie `sp_addlogin` und `sp_password` durchgeführt. Einfacher geht es mit dem SQL Server Enterprise Manager. In den bisherigen Abschnitten haben Sie sich aber die Grundlagen erarbeitet, die hinter diesen Abläufen stehen.

Abbildung 5.10 zeigt das Beispielsystem, nachdem die in der Lektion bisher vorgestellten Anweisungen ausgeführt wurden. Der Windows 2000-Gruppe Verkauf ist das Anmelde-recht für SQL Server erteilt und dem Windows 2000-Benutzer Ann explizit verweigert worden. Die Windows 2000-Gruppe `VODEFINIERT\Administratoren` und der Benutzername `sa` werden beim Setup hinzugefügt. Zusätzlich wurde der Benutzername *frank* als Standard (SQL Server) eingerichtet.

Um die Gruppe *Marketing* als gültigen Benutzernamen hinzuzufügen, klicken Sie an einer freien Stelle im Detailbereich des Enterprise Managers mit der rechten Maustaste und wählen NEUER BENUTZERNAME aus dem Kontextmenü. Daraufhin erscheint ein Dialogfeld entsprechend Abbildung 5.11. Füllen Sie das Dialogfeld wie in der Abbildung gezeigt aus, um die Gruppe *Marketing* hinzuzufügen.

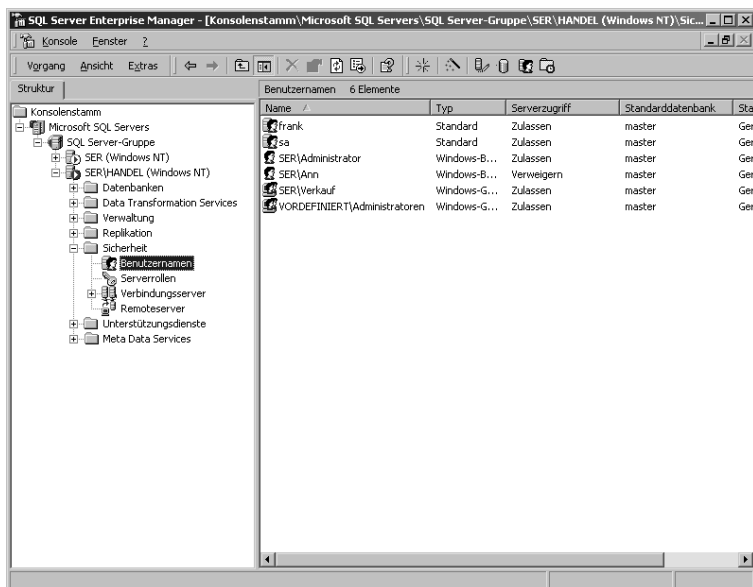


Abbildung 5.10:
Die Informationen
zu vorhandenen
Benutzernamen im
SQL Server Enter-
prise Manager

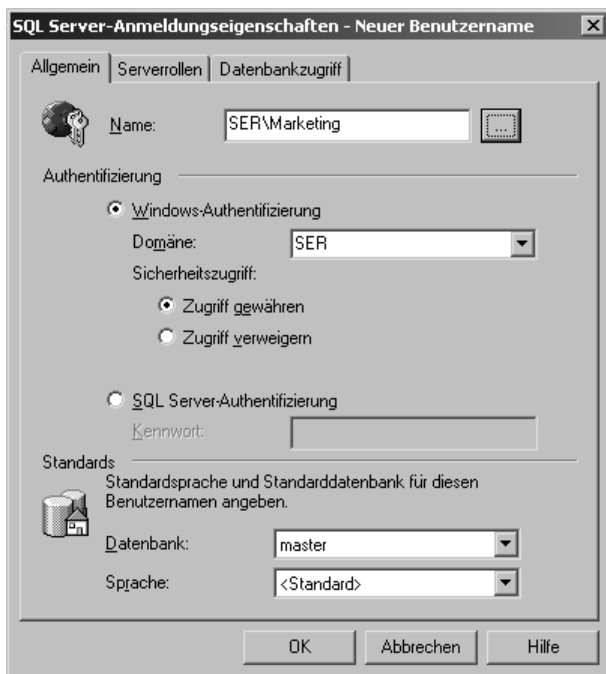


Abbildung 5.11:
Einen Windows 2000-Benutzerna-
men über den Enterprise Manager
hinzufügen

Klicken Sie auf OK, um den Benutzernamen zu übernehmen, und wiederholen Sie dann diesen Vorgang, um ein Konto für Bob hinzuzufügen. (Genaugenommen brauchen Sie kein Konto für Bob hinzuzufügen, weil sich Bob aufgrund der Mitgliedschaft in den Gruppen *Verkauf* und *Marketing* ohnehin anmelden kann.) Fügen Sie nun den SQL Server-Benutzernamen *Don* mit dem Kennwort *kennwort* hinzu. Abbildung 5.12 zeigt das Konfigurationsdialogfeld. Wenn Sie auf OK klicken, müssen Sie das eingegebene Kennwort bestätigen.

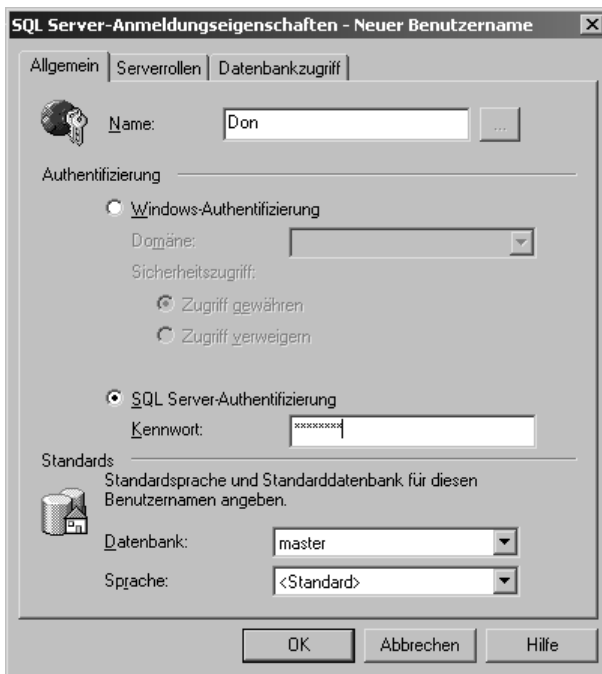


Abbildung 5.12:
Hinzufügen eines SQL Server-
Benutzernamens über den Enterprise
Manager

Um einem Benutzer den Zugriff zu verweigern, müssen Sie lediglich im Konfigurationsdialogfeld die Option *Zugriff verweigern* anklicken. Diese Option ist nur für Windows 2000-Benutzernamen sinnvoll. Einem SQL Server-Benutzernamen »verweigern« Sie die Anmeldung, indem Sie den Namen gar nicht erst erstellen.

Um einen vorhandenen Benutzernamen zu bearbeiten, markieren Sie den Ordner *Benutzernamen* in der Konsolenstruktur, klicken im Detailbereich mit der rechten Maustaste auf den Benutzernamen und wählen *EIGENSCHAFTEN* aus dem Kontextmenü. Es ist Ihnen mittlerweile sicher aufgefallen, dass es im Konfigurationsdialogfeld zwei weitere Registerkarten gibt: *SERVERROLLEN* und *DATENBANKZUGRIFF*. Darauf gehen die nächsten Abschnitte ein.



5.3 Datenbankbenutzer

Nachdem Sie die Anmeldesicherheit konfiguriert und Ihre Benutzernamen eingerichtet haben, können Sie sich der Konfiguration des Datenbankzugriffs widmen. Eine Anmeldung bei SQL Server ist noch kein Freibrief für den Zugriff auf irgendeine Datenbank im Server. Hierfür brauchen Sie erst einen Datenbankbenutzernamen.

Jede Datenbank hat einen separaten Zugriffsweg, der in der Systemtabelle `sysusers` in jeder Datenbank gespeichert ist. Die für die Anmeldung verwendeten Benutzernamen werden praktisch auf einen Benutzernamen in jeder Datenbank, auf die der Benutzer zugreifen muss, abgebildet. Mit der gespeicherten Systemprozedur `sp_grantdbaccess` oder mit dem SQL Server Enterprise Manager können Sie diese Abbildung oder einen Datenbankbenutzer in einer Datenbank erstellen.



Vielleicht finden Sie auch gespeicherte Systemprozeduren, um ähnliche Aufgaben durchzuführen. Dazu gehören `sp_adduser`, `sp_dropuser`, `sp_addgroup`, `sp_dropgroup` und `sp_changegroup`. Diese Legacy-Prozeduren wurden in SQL Server-Versionen vor 7.0 eingesetzt. Obwohl sie noch funktionieren, um Anmeldungen der SQL Server-Authentifizierung auf Benutzernamen abzubilden, ist die Funktion in Verbindung mit Benutzernamen und Gruppennamen bei Windows NT/2000-Authentifizierung nicht mehr garantiert. Verwenden Sie deshalb diese Prozeduren nur dann, wenn Sie eine Aktualisierung von SQL Server 6.5 ausgeführt und diese Prozeduren bereits in der alten Version verwendet haben. Es empfiehlt sich aber auch in diesem Fall, sobald als möglich auf die neuen gespeicherten Systemprozeduren überzugehen.

Einer Datenbank einen Benutzer hinzufügen

Mit der gespeicherten Systemprozedur `sp_grantdbaccess` lässt sich ein Benutzer einer Datenbank hinzufügen:



```
sp_grantdbaccess [@loginame =] 'login'  
[,[@name_in_db =] 'name_in_db'] OUTPUT
```

Die Parameter haben folgende Bedeutung:

- `login` ist der Benutzername, den sie bereits hinzugefügt haben (entweder als SQL Server-Benutzername oder als Windows NT/2000-Benutzername oder -Gruppe).
- `name_in_db` ist der Name für das Konto in dieser Datenbank (der Benutzername). Wenn Sie keinen Namen spezifizieren, wird er auf den Benutzernamen für die Anmeldung gesetzt.

Es empfiehlt sich, bei jeder Gelegenheit den Benutzernamen auf den Anmeldenamen zu setzen, so dass sich die Sicherheit von Anmeldungen zu Benutzern in jeder Datenbank einfacher verfolgen lässt. Diese Methode ist zwar keine Pflicht, aber ist es nicht verwirrend, wenn Sie sich zum Beispiel als *Richard* bei SQL Server anmelden, aber auf die Datenbank *Verkauf* vielleicht unter dem Namen *Bill* und auf die Datenbank *pubs* unter dem Namen *Johann* zugreifen? Streben Sie also einheitliche Namen an, um die Übersichtlichkeit zu wahren (was gerade bei einem so komplexen Produkt wie SQL Server von Vorteil ist). Wenn Sie Windows-Benutzer und -Gruppen verwenden, ist das Ganze natürlich kein Thema.

Soll beispielsweise Bob auf die Datenbank *pubs* auf Ihrem Server zugreifen können, führen sie die folgenden Anweisungen aus:



```
use pubs  
exec sp_grantdbaccess 'SER\Bob'
```

Das Ergebnis lautet:



Der Datenbankzugriff für 'SER\Bob' wurde erteilt.

Das Gleiche funktioniert auch mit einer Gruppe:



```
exec sp_grantdbaccess 'SER\Marketing'
```

Auch hier sollte die Erfolgsmeldung erscheinen. Nachdem Sie eine Gruppe hinzugefügt haben, kann jedes Mitglied der Windows-Gruppe auf die Datenbank zugreifen, aber nur die Gruppe hat einen Datenbankbenutzer – d.h. es gibt nur einen Eintrag für die Gruppe selbst und nicht für die einzelnen Benutzer, genau wie es bei Benutzernamen der Fall ist.

Um einem Benutzer den Datenbankzugriff zu entziehen, führen Sie die gespeicherte Systemprozedur `sp_revokedbaccess` aus.



```
sp_revokedbaccess [@name_in_db =] 'name_in_db']
```

Der Parameter *name_in_db* ist der Name des zu entfernenden Datenbankbenutzers.



Nur Mitglieder der Rollen `db_accessadmin` und `db_owner` (oder der festen Serverrolle `sys-admin`) können diese gespeicherten Systemprozeduren ausführen. Rollen lernen Sie später in der heutigen Lektion kennen.

Mit der gespeicherten Systemprozedur `sp_helpuser` können Sie ermitteln, welche Benutzer zu einer Datenbank gehören und welchen Benutzernamen sie zugeordnet sind:



```
sp_helpuser [[@name_in_db =] 'username']
```

Der Parameter *username* ist optional und gibt entweder einen Benutzernamen oder einen Rollennamen an.

Wenn Sie keinen Benutzernamen spezifizieren, erhalten Sie einen Bericht über alle Benutzer und Rollen. Andernfalls bezieht sich der Bericht nur auf einen bestimmten Benutzer bzw. die Rolle.

Wenn Sie eine Datenbank erstellen, verfügt diese bereits über zwei Benutzer. Einer dieser Benutzer heißt *dbo* (für DataBase Owner, Datenbankbesitzer). Der Benutzer *dbo* wird standardmäßig dem Benutzernamen *sa* zugeordnet. Bei der Installation von SQL Server gilt der Benutzername *sa* als Besitzer aller Datenbanken. Wenn ein anderer Benutzer eine Datenbank erstellt, ist dieser Benutzername der Besitzer der Datenbank. In der Datenbank selbst sind dem Benutzer *dbo* keinerlei Beschränkungen auferlegt. Dieser Benutzer hat die gleiche »Macht« wie die Mitglieder der Rolle *sysadmin* innerhalb jeder Datenbank. Allerdings haben nur Mitglieder der festen Serverrolle *sysadmin* bestimmte systemweite Privilegien (mehr dazu später).



In SQL Server 7.0 gibt es in jeder Datenbank zusätzlich noch den Benutzer `INFORMATION_SCHEMA`. Dieser Benutzername existiert als Besitzer von mehreren Ansichten, die die Informationen des Systemkatalogs entsprechend der ANSI-Spezifikation bereitstellen. In SQL Server 2000 existieren die Sichten, deren Besitzer `INFORMATION_SCHEMA` ist, nur in der Datenbank *master*; wenn man sie aber abfragt, zeigen sie immer die Informationen über die Datenbank, mit der man gerade arbeitet.

Versuchen Sie jetzt, einen Benutzer in der Datenbank *pubs* zu erstellen. Wenn Sie bisher alle Beispiele nachvollzogen haben, verfügen Sie in Ihrem SQL Server über den Benutzernamen Don. Diesen Benutzer haben Sie als SQL Server-Benutzername weiter vorn in dieser Lektion angelegt. Starten Sie den SQL Server Query Analyzer und führen Sie die folgenden Transact-SQL-Anweisungen aus:



use pubs
exec sp_grantdbaccess Don

Damit fügen Sie der Datenbank *pubs* den neuen Benutzer *Don* hinzu, der auf die Benutzer-ID von Don in der Tabelle *sysxlogins* der Datenbank *master* abgebildet wird. Um sich davon zu überzeugen, führen Sie die Prozedur *sp_helpuser* in der Datenbank *pubs* aus (siehe Abbildung 5.13).

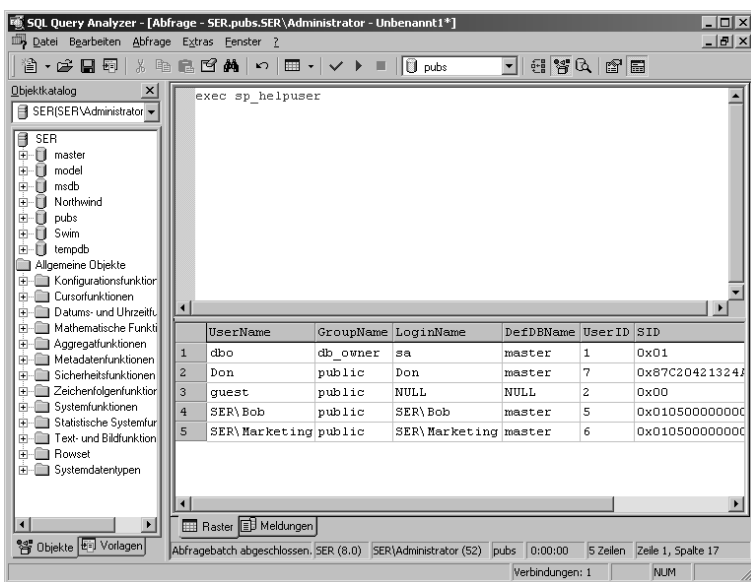


Abbildung 5.13:
Die Ergebnisse von
sp_helpuser

Wie Abbildung 5.13 zeigt, ist Don der Datenbank als Benutzer mit dem Benutzernamen (d.h. Anmeldenamen) Don hinzugefügt worden. Jetzt verstehen Sie sicherlich auch, warum es verwirrend wäre, unterschiedliche Namen zu verwenden. Außerdem ist zu sehen, dass die bereits früher für Windows 2000-Benutzer und -Gruppen eingerichteten Berechtigungen ebenfalls vorhanden sind, darüber hinaus noch *dbo* und *guest*.

Weiter vorn wurde festgestellt, dass Ihnen eine Anmeldung allein noch keinen Zugriff auf irgendwelche Datenbanken gestattet. Trotzdem können Sie nach der Anmeldung auf alle Systemdatenbanken und auch die Beispieldatenbanken *pubs* und *Northwind* zugreifen. Wie ist das möglich? Das Geheimnis liegt im Benutzernamen *guest* (Gast).

Wenn Sie in einer Datenbank die Anweisung `sp_grantdbaccess 'guest'` ausführen, wird der Datenbank ein spezielles Benutzerkonto namens *guest* hinzugefügt. Dieses Konto erscheint auch in der Ergebnismenge der gespeicherten Systemprozedur *sp_helpuser*, wie sie Abbildung 5.13 gezeigt hat. Außerdem ist zu erkennen, dass es nicht auf einen Benut-

zernamen abgebildet wird, sondern den Wert NULL als Benutzername (LoginName) erhält. Es handelt sich um einen speziellen Benutzer, der nicht den üblichen Regeln unterliegt. Wenn in einer Datenbank ein Benutzerkonto *guest* existiert, wird jeder angemeldete Benutzer, der auf die Datenbank zugreifen möchte und für den kein Benutzername in der Datenbank selbst existiert (weder über Gruppenmitgliedschaft noch über ein eigenes Konto), als Gast zugelassen. Während der Installation richtet deshalb das Setupprogramm ein Gastkonto in jeder Standarddatenbank auf dem Server ein.



Der Benutzername *guest* lässt sich nicht aus den Datenbanken *master* oder *tempdb* entfernen. Denn wo würde andernfalls ein Benutzer landen, wenn er zwar eine Anmeldung, aber keinen bestimmten Datenbankzugriff hat?

Der Benutzername guest

Wenn Sie sich unter einem Namen anmelden, für den sie in der Datenbank *pubs* keinen Eintrag erstellt haben, greifen sie unter dem Benutzernamen *guest* auf die Datenbank zu. Gibt es in dieser Datenbank auch keinen Benutzernamen *guest*, erhalten Sie eine Fehlermeldung. Nehmen wir an, Sie haben den Benutzernamen *guest* aus der Datenbank Northwind entfernt (mit `sp_revokedbaccess`) und möchten nun auf diese Datenbank zugreifen (indem Sie sie entweder im Dropdown-Listefeld DATENBANKEN des SQL Server Query Analyzers oder über den Transact-SQL-Befehl `use Northwind` auswählen). Dann erhalten Sie folgende Meldung:

```
Server: Nachr.-Nr. 916, Schweregrad 14, Status 1, Zeile 1
Der Serverbenutzer 'Don' ist kein gültiger Benutzer
in der Northwind-Datenbank.
```

Diese Fehlermeldung besagt, dass der Anmeldename nicht richtig auf einen Benutzer in der Datenbank Northwind abgebildet werden kann. Implizit geht daraus hervor, dass kein Konto *guest* existiert. Um dieses Problem zu beseitigen, fügen Sie entweder das Gastkonto mit der Anweisung



```
use Northwind
exec sp_grantdbaccess 'guest'
```

hinzufügen oder erstellen gleich eine geeignete Zuordnung des Benutzernamens zur Datenbank.



SQL Server 7.0 sieht standardmäßig in jeder Datenbank (außer *model*) ein Gastkonto vor, so dass es nicht notwendig ist, den Benutzer *guest* explizit hinzuzufügen. Nachdem Sie den Benutzer *guest* jedoch entfernt haben, können Sie das

geschilderte Problem nachvollziehen. Wenn der Benutzer *guest* in der Datenbank `model` vorhanden ist, verfügt jede neu angelegte Datenbank über den Benutzer *guest*.

Bisher haben Sie zwei Verfahren kennen gelernt, um auf eine Datenbank zuzugreifen: indem Ihr Anmeldename auf den Benutzernamen der Datenbank abgebildet oder der Benutzername *guest* verwendet wird. Außerdem können Sie der *dbo* (Datenbankeigentümer) sein, d.h. dass die Datenbank unter Ihrem Benutzernamen erstellt wurde. Aus historischen Gründen ist es darüber hinaus möglich, den Zugriff auf die Datenbank über einen Aliasnamen für einen anderen Benutzer zu erhalten.



Der folgende Abschnitt über Aliasnamen ist hier nur aus Gründen der Abwärtskompatibilität enthalten. In SQL Server 2000 sollten Sie generell auf Aliasnamen verzichten. Rollen stellen eine wesentlich elegantere Lösung dar, wie Sie in Kürze sehen werden.

Einen Alias hinzufügen



Mit der gespeicherten Systemprozedur `sp_addalias` können Sie einen Aliasnamen für den Datenbankbenutzer hinzufügen. Dieser Alias ermöglicht, dass der angemeldete Benutzer die Identität eines anderen Datenbankbenutzers annimmt, statt den Anmeldnamen auf seinen eigenen Benutzer- oder Gruppennamen abzubilden.



```
sp_addalias login_name, username
```

Die Parameter haben folgende Bedeutung:

- *login_name* ist der Benutzername, dem Sie einen Alias zuordnen wollen.
- *username* ist der Name des Datenbankbenutzers, dem der Benutzername zugeordnet wird.

Die Bildung von Aliasnamen soll die Sicherheit in einer Datenbank vereinfachen. Einem einzigen Datenbankbenutzernamen lassen sich beliebig viele Benutzernamen zuordnen. Wenn ein Benutzername bereits mit einem Benutzer in der Datenbank verbunden ist, kann dieser Benutzername nicht zusätzlich noch als Aliasname für einen Benutzer dienen. Rollen erlauben auch hier ein wesentlich sauberes und besseres Sicherheitsmodell.

Man verwendet diese Datenbankzugriffsoption normalerweise, um Benutzernamen auf den Benutzer *dbo* in einer Datenbank abzubilden. Allerdings kann es zu einem bestimm-



ten Zeitpunkt nur einen »echten« Datenbankbesitzer geben. Da viele Firmen mehr als einen Administrator haben, sollen wahrscheinlich auch mehrere Benutzer als Besitzer der Datenbank agieren können. Mit Rollen lässt sich dieses Problem lösen.

Eine Aliaszuordnung können Sie mit der gespeicherten Systemprozedur `sp_dropalias` wieder entfernen:



```
sp_dropalias login_name
```

Der Parameter *login_name* ist der Benutzername.

Aliasnamen sollten nur in einer Datenbank vorkommen, die von SQL Server 6.5 auf SQL Server 2000 aktualisiert wurde. Sobald es die Zeit und das Testen zulassen, sollte man sie entfernen, da sie in zukünftigen Versionen von Microsoft SQL Server höchstwahrscheinlich nicht mehr unterstützt werden.

Den Datenbankbesitzer ändern

Vielleicht möchten Sie den Besitzer einer vorhandenen Datenbank ändern, um die Verantwortlichkeit für die Datenbank einem speziellen Datenbankadministrator (DBA) zuzuweisen. Bei einer derartigen Änderung darf der Benutzername (d.h. der Anmeldename) nicht als Benutzer in der Datenbank existieren. Gegebenenfalls müssen Sie zunächst den Aliasnamen oder den Benutzer in der aktuellen Datenbank löschen.

Zur Änderung des Besitzers führen Sie die gespeicherte Systemprozedur `sp_changedbowner` aus.



```
sp_changedbowner [@loginame =] 'login' [,[@map =] drop_alias_flag]
```

Die Parameter haben folgende Bedeutung:

- *login* ist der Benutzername des neuen Besitzers der aktuellen Datenbank.
- *drop_alias_flag* ist ein optionaler Parameter, der anzeigt, ob vorhandene Aliasnamen für den alten Datenbankbesitzer (dbo) auf den neuen Besitzer der aktuellen Datenbank übergehen (TRUE) oder gelöscht (FALSE) werden sollen.

Insgesamt kennen Sie nun vier verschiedene Wege, um auf eine Datenbank zuzugreifen, nachdem Sie sich erfolgreich bei SQL Server angemeldet haben. Diese werden in der folgenden Reihenfolge geprüft:

- **sa:** Der *sa* (oder jedes Mitglied der Serverrolle *sysadmin*) kann immer auf eine Datenbank zugreifen und erscheint immer als *dbo*, selbst wenn Sie den Besitz der Datenbank mit *sp_changedbowner* einem anderen Benutzernamen zugeordnet haben.
- **Datenbankbenutzername:** Der »normale« Weg, auf eine Datenbank zuzugreifen, läuft über einen Datenbanknamen, der einem SQL Server-Anmeldenamen (entweder einem Windows 2000-Benutzernamen oder -Gruppennamen) zugeordnet ist. Das gilt auch für Windows 2000-Gruppen, für die Sie Zugriffsrechte gewährt haben.
- **Alias:** Der angemeldete Benutzer lässt sich über einen Aliasnamen einem gültigen Datenbankbenutzer zuordnen. In der Datenbank wird der andere Benutzer in Bezug auf Berechtigungen und Privilegien nachgebildet.
- **Guest:** Wenn alle anderen Überprüfungen fehlschlagen, prüft SQL Server, ob in der Datenbanksystemtabelle *sysusers* ein Gastkonto existiert. Ist ein derartiges Konto vorhanden, erhält der Benutzer als Gast Zugriff auf die Datenbank. Andernfalls wird ihm der Zugriff verweigert.



Vielleicht haben Sie bereits festgestellt, dass Windows NT/2000-Konten nicht auf Benutzernamen abgebildet werden. Erlangt ein Windows-Benutzer die Anmeldung und den Datenbankzugriff über die Mitgliedschaft in einer Windows 2000-Gruppe, wären weitere Zuordnungen mit einem individuellen Benutzernamen überflüssig. Normalerweise erkennt man eine derartige Situation und vermeidet sie – im Gegensatz zu SQL Server. Nehmen wir an, Sie fügen einer Datenbank eine Windows 2000-Gruppe hinzu und ein Benutzer erstellt dann eine Tabelle. Um den Benutzer der Tabelle zu verfolgen, erstellt SQL Server einen neuen Benutzer für den Windows 2000-Benutzer; der von SQL Server erstellte Benutzer dient aber nur der Verfolgung und gewährt keinen eigentlichen Datenbankzugriff.

5.4 Rollen

Das Thema Rollen kommt jetzt erst zur Sprache, weil Rollen ein universelles Bindemittel in SQL Server 2000 sind. Man kann sie sich als SQL Server-Gruppen vorstellen. Allerdings spricht man hier nicht von *Gruppen*, um dieses Instrument von SQL Server nicht mit Windows NT/2000-Gruppen zu verwechseln.

Mit SQL Server-Rollen lassen sich Datenbankbenutzernamen in Gruppierungen zusammenfassen. Dabei ist es egal, ob die Datenbankbenutzernamen von Windows 2000-Gruppen, Windows 2000-Benutzern oder SQL Server-Benutzernamen abgeleitet wurden. Rollen können sogar andere Rollen als Mitglieder enthalten.

Die öffentliche Rolle

SQL Server enthält in jeder Datenbank eine integrierte Rolle namens *public*. Alle Benutzer, Gruppen und Rollen sind Mitglieder der Rolle *public* und lassen sich nicht entfernen. Die Rolle *public* ist mit der Gruppe *Jeder* von Windows 2000 vergleichbar (aus technischer Sicht ist sie das Äquivalent zur Gruppe *Authentifizierte Benutzer*, weil die Benutzer zuerst durch SQL Server validiert werden). Es ist ein bequemes Kurzverfahren, auf alle Benutzer verweisen zu können, ohne sie explizit zu benennen. Diese Lösung lernen Sie am Tag 6 im Zusammenhang mit Berechtigungen näher kennen. Abbildung 5.13 weiter vorn in dieser Lektion hat für die meisten Benutzer die Rolle *public* angegeben.

Serverweite Rollen

Diese Lektion hat mehrfach darauf hingewiesen, dass der Benutzername *sa* allmächtig ist und dieser Benutzer alles unternehmen kann, was er in einer Instanz von SQL Server möchte. Genaugenommen hängt das damit zusammen, dass der Benutzername *sa* ein Mitglied der serverweiten Rolle *sysadmin* ist. SQL Server verfügt über acht serverweite Rollen. Einen Benutzernamen können Sie jederzeit zu einem Mitglied von einer oder mehreren dieser serverweiten Rollen machen. Allerdings können Sie in die Liste der serverweiten Rollen nichts hinzufügen und nichts daraus entfernen. Beispielsweise ist es nicht möglich, *sa* aus der serverweiten Rolle *sysadmin* zu entfernen.

Verfügbare serverweite Rollen

Die folgende Liste beschreibt den vollständigen Satz der verfügbaren serverweiten Rollen. Machen Sie sich eingehend mit ihnen vertraut, damit Sie sie in der richtigen Situation einsetzen können:

- **sysadmin:** Mitglieder dieser Rolle können alle Aktionen in SQL Server ausführen. Sie gelten als *dbo* jeder Datenbank (auch wenn sie es nicht sind). Praktisch umgehen sie die Berechtigungen und Sicherheitssysteme.
- **serveradmin:** Mitglieder dieser Rolle können Konfigurationsoptionen mit der gespeicherten Systemprozedur *sp_configure* einstellen und den Server herunterfahren. Außerdem können Sie den Volltext-Dienst konfigurieren. Für diese Rolle kommen insbesondere Serveroperatoren infrage.



Mitglieder der Rolle *serveradmin* können nur den Transact-SQL-Befehl *SHUTDOWN* auslösen, um den Server herunterzufahren. Ihre Berechtigungen, Dienste zu steuern, sind Windows 2000-Rechte und keine SQL Server-Rechte.

- **setupadmin:** Mitglieder dieser Rolle können Verbindungsserver installieren und konfigurieren sowie eine gespeicherte Prozedur markieren, die beim Startvorgang auszuführen ist.
- **securityadmin:** Mitglieder dieser Rolle können Serverbenutzernamen sowie Berechtigungen zum Erstellen von Datenbanken anlegen und steuern. Außerdem können sie die Sicherheitseinstellungen von Verbindungsservern konfigurieren und Kennwörter von SQL Server-Benutzernamen ändern (mit Ausnahme von Mitgliedern der Rolle *sysadmin*). Auch für diese Rolle kommen Operatoren und höchstwahrscheinlich das Personal für den Help-Desk infrage.
- **processadmin:** Mitglieder dieser Rolle können Prozesse steuern, die auf dem Datenbankserver laufen. Dazu gehört insbesondere auch das Beenden von Abfragen, die übermäßig lange laufen oder außer Kontrolle geraten sind.
- **dbcreator:** Mitglieder dieser Rolle können Datenbanken auf dem Server erstellen, ändern und löschen. Als Kandidaten für diese Rolle kommen DBAs infrage (falls Ihr DBA kein Mitglied der Rolle *sysadmin* sein soll).
- **bulkadmin:** Mitglieder dieser Rolle können die Anweisung `BULK INSERT` ausführen. Mehr zum Massenkopieren und zur Anweisung `BULK INSERT` erfahren Sie am Tag 19.
- **diskadmin:** Mitglieder dieser Rolle können Dateien und das Dateiwachstum auf dem Server verwalten. Allerdings stehen die verfügbaren Befehle hauptsächlich für die Abwärtskompatibilität zu SQL Server 6.5 bereit, so dass Sie diese Rolle wahrscheinlich nur selten benötigen. Als Kandidaten für diese Rolle kommen DBAs infrage (falls Ihr DBA kein Mitglied der Rolle *sysadmin* sein soll).

Einer Serverrolle einen Benutzernamen zuweisen

Einen Benutzernamen können Sie einer speziellen Serverrolle entweder mit dem SQL Server Enterprise Manager oder der gespeicherten Systemprozedur `sp_addsrvrolemember` zuweisen:



```
sp_addsrvrolemember [@loginame =] 'login' ,[@rolename =] 'role'
```

Die Parameter haben folgende Bedeutung:

- *login* ist der Benutzername, der der Rolle hinzugefügt wird.
- *role* ist der Name der festen Serverrolle, zu der der Benutzername hinzugefügt wird.

Ein Benutzername kann zu keiner, einer oder beliebig vielen Rollen gehören. Die Rolle *sysadmin* schließt allerdings alle anderen Rollen ein – sowohl serverweit als auch daten-



bankspezifisch. Wenn daher der Benutzername Mitglied der Rolle *sysadmin* ist, braucht man keine andere Rolle mehr festzulegen. Um einen Benutzernamen aus einer serverweiten Rolle zu entfernen, verwenden Sie die gespeicherte Systemprozedur `sp_droprsvrolemember`.



```
sp_droprsvrolemember [@loginame =] 'login' ,[@rolename =] 'role'
```

Die Parameter haben folgende Bedeutung:

- *login* ist der SQL Server-Benutzername, der aus der festen Serverrolle entfernt werden soll.
- *role* ist der Name der Serverrollen, aus der der Benutzername entfernt werden soll.

Das folgende Beispiel zeigt, wie man *SER\Bob* zu einem Mitglied der Rolle *securityadmin* macht:



```
exec sp_addsrvrolemember 'SER\Bob', 'securityadmin'
```

Zur Bestätigung erscheint die Meldung:



'SER\Bob' wurde zur Rolle 'securityadmin' hinzugefügt.

Aus dieser Rolle entfernen Sie Bob mit der folgenden Anweisung:



```
exec sp_droprsvrolemember 'SER\Bob', 'securityadmin'
```

Die Bestätigungsmeldung lautet:



'SER\Bob' wurde aus der Rolle 'securityadmin' gelöscht.

Die gleichen Änderungen können Sie auch im Enterprise Manager durchführen. In der Konsolenstruktur erweitern Sie dazu den Knoten für Ihren Server und darunter den Knoten *Sicherheit*. Klicken Sie auf den Eintrag *Serverrollen* (mit dem Schlüsselsymbol). Daraufhin erscheinen im Detailbereich die acht festen Serverrollen. Doppelklicken Sie auf

Security Administrators, um das Dialogfeld SERVERROLLE – EIGENSCHAFTEN – SECURITY-ADMIN anzuzeigen. Klicken Sie auf HINZUFÜGEN; es erscheint eine Liste der verfügbaren Benutzernamen. Markieren Sie SER\Bob (oder einen auf Ihrem Server vorhandenen Benutzernamen) und klicken Sie auf OK. Das Dialogfeld SERVERROLLE entspricht nun Abbildung 5.14. Wenn Sie hier noch auf OK klicken, haben Sie praktisch die gespeicherte Systemprozedur `sp_addsrvrolemember` ausgeführt, außer dass Sie in einer grafischen Umgebung arbeiten.

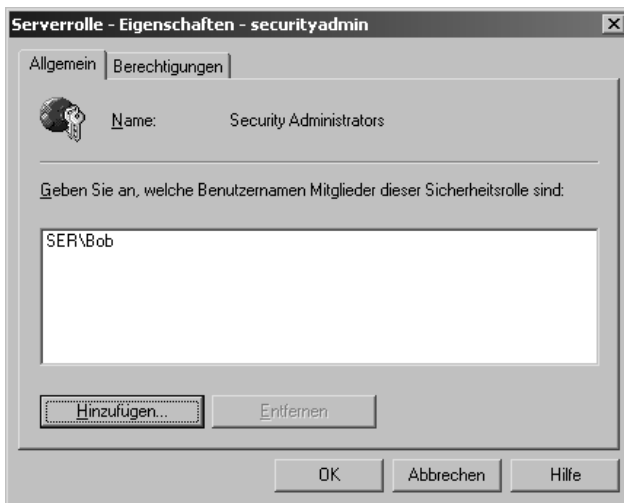


Abbildung 5.14:
Die Serverrolleneigenschaften für
die Rolle Security Administrators

Datenbankrollen

Jede Datenbank enthält auch Rollen. Einige dieser Rollen sind fest und Sie können auch eigene Rollen hinzufügen (im Gegensatz zu Serverrollen). Denken Sie immer daran, dass Datenbankrollen datenbankspezifisch sind – eine derartige Rolle kann sich also nicht auf mehrere Datenbanken gleichzeitig beziehen. Allerdings können Sie in jeder Datenbank die gleichen Rollen erstellen.

Feste Datenbankrollen

In jeder Datenbank existiert ein Satz fester Datenbankrollen, denen sich ein Datenbankbenutzer zuordnen lässt. Standardmäßig gibt es neun feste Datenbankrollen und diese sind immer vorhanden (keine davon lässt sich löschen). Genau wie die bereits behandelten Serverrollen weist jede Datenbankrolle benutzerspezifische Berechtigungen und Fähigkeiten zu. Mehr zu Berechtigungen erfahren Sie in der morgigen Lektion.

- **db_owner:** Mitglieder dieser Rolle können alle Aktionen ausführen – aber nur innerhalb ihrer Datenbank. Die Mitgliedschaft in der Rolle *db_owner* gibt einem Benutzer genau die gleichen Rechte und Berechtigungen wie dem Benutzer *dbo* einer Datenbank (dem Besitzer). Die Ausnahme bildet der Befehl *restore*; dazu erfahren Sie mehr am Tag 8.
- **db_accessadmin:** Mitglieder dieser Rolle können den Zugriff von Benutzern auf die Datenbank hinzufügen oder entfernen (beispielsweise die gespeicherte Systemprozedur *sp_grantdbaccess* ausführen).
- **db_securityadmin:** Mitglieder dieser Rolle können alle Berechtigungen, Rollen, Rollenmitgliedschaft und Besitzer von Objekten in der Datenbank kontrollieren.
- **db_ddladmin:** Mitglieder dieser Rolle können alle Datenbankobjekte erstellen, bearbeiten und löschen, jedoch keine sicherheitsbezogenen Befehle (*grant*, *revoke*, *deny*) ausführen. Die morgige Lektion geht im Detail auf diese Befehle ein.
- **db_dumpoperator:** Mitglieder dieser Rolle können ausgewählte DBCC-Befehle sowie die Prüfpunkt- und Sicherungsbefehle ausführen.
- **db_datareader:** Mitglieder dieser Rolle haben Auswahlberechtigungen für beliebige Tabellen, Sichten und Funktionen in der Datenbank.
- **db_datawriter:** Mitglieder dieser Rolle haben Änderungsberechtigungen (Einfügen, Aktualisieren, Löschen) für jede Tabelle und Sicht in der Datenbank.
- **db_denydatareader:** Mitglieder dieser Rolle können keine Daten aus den Tabellen und Sichten in der Datenbank auswählen.
- **db_denydatawriter:** Mitglieder dieser Rolle können keine Daten in den Tabellen und Sichten in der Datenbank mit den Anweisungen *insert*, *update* oder *delete* ändern.

Benutzerdefinierte Datenbankrollen

Zusätzlich zu den immer verfügbaren festen Datenbankrollen können Sie selbst Rollen erstellen und ihnen dann Benutzer oder Rollen zuweisen. In einer SQL Server-Datenbank erstellt man Rollen aus den gleichen Gründen wie eine Gruppe im Betriebssystem Windows 2000 – um Benutzer mit ähnlichen Funktionen komfortabel zusammenzufassen. Erstellen Sie so viele Rollen, wie es für die jeweilige Umgebung sinnvoll ist. Benutzer können Mitglied in beliebig vielen Rollen sein und Rollen können selbst als Mitglied zu anderen Rollen gehören.

Eine Rolle können Sie mit der gespeicherten Systemprozedur `sp_addrole` erstellen.



```
sp_addrole [@rolename =] 'role' [,[@ownername =] 'owner']
```

Die Parameter haben folgende Bedeutung:

- *role* ist der Name der neuen Rolle.
- *owner* ist der Besitzer der neuen Rolle (jeder Benutzer kann seine eigenen Rollen besitzen). Der Standard ist `dbo`.

Nur Mitglieder der serverweiten Rolle *sysadmin* oder der Datenbankrollen *db_owner* oder *db_securityadmin* können einer Datenbank eine neue Rolle hinzufügen. Dies trifft auch auf das Löschen von Rollen zu. Ungewöhnlich bei Rollen ist, dass Sie zwar den Besitzernamen angeben können, der Rollename in der Datenbank aber trotzdem eindeutig sein muss. Folglich brauchen Sie den Besitzer nicht zu kennen, wenn Sie eine Rolle löschen, weil der Name in einer Datenbank eindeutig ist.

Mit der gespeicherten Systemprozedur `sp_droprole` lässt sich eine Rolle aus einer Datenbank löschen:



```
sp_droprole [@rolename =] 'role'
```

Der Parameter *role* ist der Name der benutzerdefinierten Rolle, die Sie löschen möchten.

Eine Rolle, zu der Benutzer oder andere Rollen als Mitglieder gehören, können Sie nicht löschen. Auch wenn die Rolle Objekte besitzt, lässt sie sich nicht löschen. Mehr zu Objektbesitzern erfahren Sie in der morgigen Lektion.

Wie fügt man nun Benutzer zu einer Rolle hinzu? Mit der gespeicherten Prozedur `sp_addrolemember` lassen sich Benutzer zu benutzerdefinierten oder festen Datenbankrollen hinzufügen:



```
sp_addrolemember [@rolename =] 'role',  
                 [@membername =] 'database_account'
```

Die Parameter haben folgende Bedeutung:

- *role* ist der Name der Rolle, der sie einen Benutzer hinzufügen wollen.
- *database_account* ist der Benutzername, Gruppenname oder Rollename, den sie dieser Rolle hinzufügen möchten.



Um ein Mitglied aus einer Rolle zu entfernen, rufen Sie die gespeicherte Systemprozedur `sp_droprolemember` auf:



```
sp_droprolemember [@rolename =] 'role',  
                  [@membername =] 'database_account'
```

Die Parameter haben folgende Bedeutung:

- *role* ist der Name der Rolle, aus der Sie einen Benutzer entfernen wollen.
- *database_account* ist der Benutzername, Gruppenname oder Rollename, den sie aus dieser Rolle entfernen wollen.

Das folgende Codebeispiel zeigt, wie Sie diese gespeicherten Prozeduren einsetzen. Um der Datenbank *pubs* eine neue Rolle hinzuzufügen und ihr dann einen Benutzer zuzuweisen, führen Sie folgenden Code aus:



```
use pubs  
exec sp_addrole 'Management'  
exec sp_addrole 'Fertigung'  
exec sp_addrolemember 'Management', 'Don'  
exec sp_addrolemember 'Fertigung', 'SER\Marketing'
```

Als Ausgabe erscheinen folgende Meldungen:



```
Neue Rolle hinzugefügt  
Neue Rolle hinzugefügt  
'Don' wurde zur Rolle 'Management' hinzugefügt.  
'SER\Marketing' wurde zur Rolle 'Fertigung' hinzugefügt.
```

Wenn Sie versuchen, eine Rolle mit Mitgliedern zu löschen, erhalten Sie eine Fehlermeldung. Zum Beispiel liefert die Anweisung

```
exec sp_droprole 'Fertigung'
```

folgende Meldung:

```
Server: Nachr.-Nr. 15144, Schweregrad 16, Status 1,  
Prozedur sp_droprole, Zeile 53  
Die Rolle hat Mitglieder. Sie muss leer sein, bevor sie gelöscht  
werden kann.  
name  
-----  
SER\Marketing
```



Die obigen Ergebnisse erhalten Sie, wenn sich SQL Server Query Analyzer im Modus *Ergebnisse in Text* befindet. Wenn Sie im Standardmodus arbeiten (*Ergebnisse in Raster*), erscheint die Fehlermeldung auf der Registerkarte MELDUNGEN und die Liste der noch in der Rolle befindlichen Benutzer ist auf der Registerkarte RASTER zu sehen.

SQL Server gibt sogar darüber Auskunft, welche Mitglieder in der Rolle verbleiben. Um diese Mitglieder zu löschen, führen Sie die folgenden Anweisungen aus:



```
exec sp_droprolemember 'Fertigung', 'SER\Marketing'  
exec sp_droprole 'Fertigung'
```

Jetzt erscheint die Erfolgsmeldung:



SER\Marketing' wurde aus der Rolle 'Fertigung' gelöscht.
Rolle gelöscht.

Die Mitgliedschaft der einzelnen Rollen speichert SQL Server in einer Kombination der Systemtabellen *sysusers* und *systemmembers*. Die vorhandenen Rollen können Sie sich mit den gespeicherten Systemprozeduren *sp_helprole* oder *sp_helprolemember* anzeigen lassen. Beide Prozeduren übernehmen nur einen einzigen Parameter: den Rollennamen in Anführungszeichen.

Die gleichen Ergebnisse lassen sich auch mit dem SQL Server Enterprise Manager erreichen. Markieren Sie in der Konsolenstruktur den Ordner *Benutzernamen* für Ihren Server. Doppelklicken Sie dann auf einen Benutzernamen oder klicken Sie mit der rechten Maustaste darauf und wählen Sie EIGENSCHAFTEN aus dem Kontextmenü. Das folgende Beispiel bezieht sich auf *SER\Bob*. Serverrollen können Sie sowohl hier als auch auf der Registerkarte SERVERROLLEN konfigurieren. Zunächst aber konzentrieren wir uns auf die letzte Registerkarte: DATENBANKZUGRIFF. Markieren Sie die Datenbank *pubs*. Daraufhin zeigt der untere Teil des Dialogfelds eine Liste der Datenbankrollen an (siehe Abbildung 5.15).

Hier sind aber nicht nur alle festen Datenbankrollen, sondern auch alle benutzerdefinierten Rollen zu sehen. Blättern Sie die Liste bis zum Ende durch – dort finden Sie die Rolle *Management*. Um den Datenbankbenutzer zum Mitglied einer Rolle zu machen, schalten Sie einfach das Kontrollkästchen neben dem Rollennamen ein; um ihn aus der Rolle zu entfernen, schalten Sie das Kontrollkästchen aus. So einfach ist das!

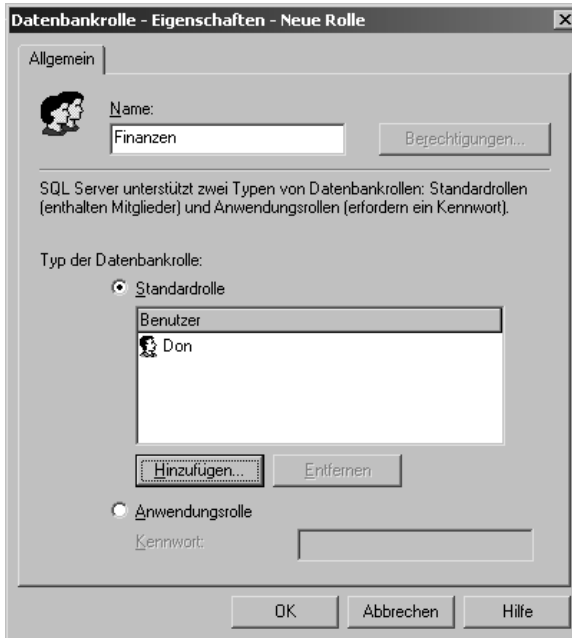


Abbildung 5.15:
Eine neue Datenbankrolle hinzufügen

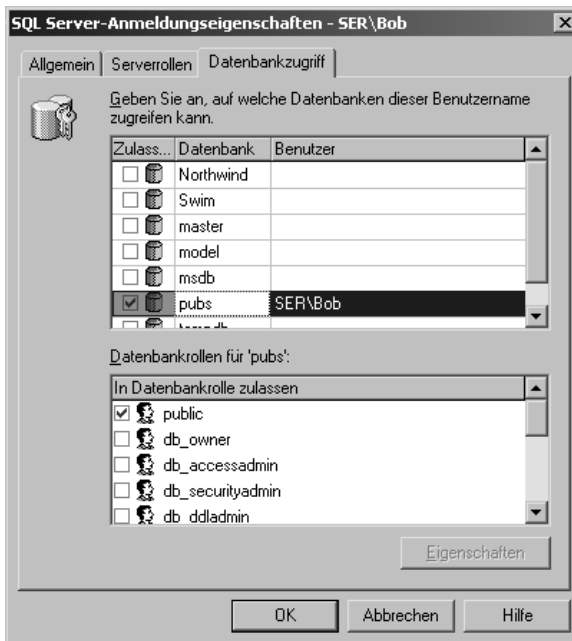


Abbildung 5.16:
Das Dialogfeld SQL Server-Anmel-
dungseigenschaften mit der Register-
karte Datenbankzugriff

Um mit dem SQL Server Enterprise Manager eine neue Rolle zu erstellen, erweitern Sie den Datenbankordner und dann die Datenbank, mit der Sie arbeiten wollen. Für dieses Beispiel erweitern Sie die Datenbank *pubs*. Wenn Sie den Ordner *Rollen* markieren, erscheint im Detailbereich eine Liste der Rollen. Um eine neue Rolle hinzuzufügen, klicken Sie mit der rechten Maustaste auf eine freie Stelle im Detailbereich und wählen NEUE DATENBANKROLLE aus dem Kontextmenü. Fügen Sie eine neue Rolle namens *Finanzen* mit Don als Mitglied der Rolle hinzu. Abbildung 5.16 zeigt das fertig ausgefüllte Dialogfeld. Klicken Sie auf OK, um das Erstellen der Rolle abzuschließen. Auf die Option *Anwendungsrolle* geht der nächste Abschnitt ein.

5.5 Anwendungsrollen

Anwendungsrollen sind ein sehr nützliches Merkmal von SQL Server 2000. Auch wenn Anwendungsrollen in gewisser Weise mit den bisher untersuchten Rollen vergleichbar sind, unterscheiden sie sich funktionell von anderen Rollen.

Anwendungsrollen dienen genau wie andere Rollen dazu, Benutzer zu gruppieren, so dass man Berechtigungen von einer höheren Ebene aus zuweisen kann und nicht jedem Benutzer einzeln zuweisen muss. Von anderen Rollen unterscheiden sich Anwendungsrollen dadurch, dass sie von einer Anwendung »eingeschaltet« werden können. Nachdem eine Anwendung eine Anwendungsrolle aktiviert hat, werden alle Berechtigungen der Benutzer vorübergehend aufgehoben und nur die Berechtigungen der Rolle sind wirksam. Natürlich ist ein Kennwort erforderlich, damit sich die Rolle aktivieren lässt (es sei denn, Sie verzichten explizit auf ein Kennwort).

Als Beispiel für den Einsatz von Anwendungsrollen bietet sich eine Buchhaltungsanwendung an. Alle Administratoren in der Buchhaltung müssen die Angaben zu Gehalts- und Prämienzahlungen an die Angestellten periodisch aktualisieren. Statt aber diese Aufgabe direkt über die SQL Server-Datenbank zu erledigen (was mit gewissen Risiken für den Datenbestand verbunden ist), sollen sie entsprechend einer Firmenvorschrift nur über die Buchhaltungsanwendung arbeiten. Beim Start der Anwendung können Sie die Benutzer unter ihrem eigenen Namen bei SQL Server anmelden lassen (entweder mit SQL Server-Benutzernamen oder vorzugsweise mit ihren Windows 2000-Anmeldeinformationen, so dass ihnen der Anmeldevorgang nicht einmal bewusst wird). Dann führen Sie den geeigneten Code (die gespeicherte Systemprozedur *sp_setapprole*) aus, um die Anwendungsrolle für die Buchhaltung zu aktivieren. Damit treten die Berechtigungen der Rolle in Kraft und die Berechtigungen der Benutzer werden abgeschaltet. Dieser Zustand bleibt bestehen, bis die Anwendung die Verbindung zur Datenbank trennt. Wenn nun die Buchhaltungsrolle über die Berechtigungen verfügt, die Lohn tabellen zu bearbeiten, die Administratoren der Buchhaltung jedoch nicht, dürfen die Administratoren trotzdem mit der Anwendung arbeiten. Vor allem können sie weder wissentlich (noch unwissentlich) die Sicherheits-



mechanismen umgehen, die Sie in der Anwendung eingerichtet haben. Darüber hinaus werden alle Aktivitäten mit den Anmeldungsinformationen der Benutzer überwacht.

Anwendungsrollen sind also ein vielversprechendes Instrument. Trotz ihrer Leistungsfähigkeit, sind sie aber relativ einfach zu implementieren. In der morgigen Lektion erfahren Sie, wie man Berechtigungen an Rollen zuweist.

Zunächst erstellen Sie mit der gespeicherten Systemprozedur `sp_addapprole` eine Anwendungsrolle:



```
sp_addapprole [@rolename =] 'role', [@password =] 'password'
```

Die Parameter haben folgende Bedeutung:

- *role* ist der Name der Rolle, die sie erstellen wollen.
- *password* ist das Kennwort, das die Anwendung übergeben muss, um die Rolle zu aktivieren.

Zum Löschen einer Anwendungsrolle führen Sie die gespeicherte Systemprozedur `sp_dropapprole` aus:



```
sp_dropapprole [@rolename =] 'role'
```

Der Parameter *role* ist der Name der Rolle, die Sie entfernen wollen.

Um die Rolle in Ihrer Anwendung einzusetzen, führen Sie die gespeicherte Systemprozedur `sp_setapprole` aus:



```
sp_setapprole [@rolename =] 'role' ,  
               [@password =] {Encrypt N 'password'} | 'password'  
               [,[@encrypt =] 'encrypt_style']
```

Die Parameter haben folgende Bedeutung:

- *role* ist der Name der Rolle, die sie aktivieren wollen.
- *password* ist das bei der Ausführung von `sp_addapprole` angegebene Kennwort.
- *Encrypt N 'password'* fordert die Kennwortverschlüsselung an, wenn das Kennwort über das Netzwerk geschickt wird (wenn Sie nur das Kennwort angeben, wird es ohne Verschlüsselung über das Netzwerk verschickt).

- *encrypt_style* spezifiziert den Verschlüsselungstyp für das Kennwort. Derzeit gibt es zwei verfügbare Werte: *none* und *odbc*. Den Wert ODBC (Open Database Connectivity) geben Sie an, wenn Sie einen ODBC-basierten Client verwenden. Bei dieser Option wird das Kennwort mit der ODBC-Funktion *Encrypt* verschlüsselt, bevor es an SQL Server gesendet wird.



Wenn Sie mit einem OLE DB-Client arbeiten, sind die Verschlüsselungsfähigkeiten dennoch verfügbar. Indem Sie einfach *odbc* spezifizieren, ist der gleiche Verschlüsselungstyp wirksam.

Um eine Anwendungsrolle zu erstellen und dann zu verwenden, können Sie das folgende Skript in SQL Server Query Analyzer (einem ODBC-basierten Dienstprogramm) ausführen:



```
use pubs
exec sp_addapprole 'Buchhaltung', 'kennwort'
go
exec sp_setapprole 'Buchhaltung', {Encrypt N 'kennwort'},'odbc'
```

Sie erhalten die folgenden Erfolgsmeldungen:



Neue Anwendungsrolle hinzugefügt
Die Anwendungsrolle 'Buchhaltung' ist jetzt aktiv.

Von diesem Moment an gelten für diese Verbindung zu SQL Server die Berechtigungen der Anwendungsrolle. In der Überwachung der durchgeführten Aktivitäten erscheinen dennoch die Anmeldeinformationen des einzelnen Benutzers – nicht die Anwendungsrolle. Somit können sie die Aktionen des jeweiligen Benutzers genau unterscheiden, selbst wenn er diese Gruppenfunktionalität aktiviert hat.



In Bezug auf Berechtigungen und Anwendungsrollen bestehen oftmals Unklarheiten. Zwar geht die morgige Lektion auf dieses Thema ein, dennoch ist bereits an dieser Stelle eine Anmerkung angebracht: Auch eine aktivierte Anwendungsrolle ist weiterhin Mitglied der »öffentlichen« Rolle; Berechtigungen, die der öffentlichen Rolle zugewiesen sind, funktionieren also auch, wenn eine Anwendungsrolle aktiviert ist.



5.6 Zusammenfassung

Um auf die Daten von SQL Server zugreifen zu können, müssen Sie mehrere Sicherheits-ebenen durchlaufen. In dieser Lektion haben Sie das Sicherheitsmodell von SQL Server kennen gelernt. Die Anmeldung bei SQL Server kann im Windows-Authentifizierungsmodus oder im gemischten Modus (Windows-Authentifizierung oder SQL Server-Authentifizierung) erfolgen.

Diese Lektion ist auf die Verwaltung von Benutzernamen, Benutzern und Gruppen eingegangen. Es wurde erläutert, was Datenbankbenutzer sind, was sich hinter dem Benutzer *guest* verbirgt und wann man Aliasnamen verwendet.

Das Thema Rollen hat einen weiteren Schwerpunkt in dieser Lektion gebildet. Sie haben gelernt, was die Rolle *public* bedeutet, welche Unterschiede zwischen Serverrollen und Datenbankrollen bestehen und was Anwendungsrollen sind.

5.7 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Worin liegt der Unterschied zwischen integrierter und gemischter Sicherheit?*
- A** Integrierte Sicherheit erlaubt nur vertraute Verbindungen, während gemischte Sicherheit auch Verbindungen per SQL Server-Authentifizierung zulässt.
- F** *Welcher Sicherheitsmodus ist zweckmäßig, wenn man vorwiegend Windows 2000-Clients, aber auch einige Unix-Computer betreibt, die Zugriff auf SQL Server benötigen?*
- A** Hier kommt nur der gemischte Modus infrage, weil die Unix-Computer höchstwahrscheinlich nicht in der Windows 2000-Domäne angemeldet sind.
- F** *Welche gespeicherte Prozedur aktiviert Anwendungsrollen?*
- A** Die gespeicherte Systemprozedur `sp_setapprole`.

F Wenn Sie sich unter einem Benutzernamen anmelden, aber keinen Datenbankbenutzernamen haben, wieso können Sie dann die Datenbank *pubs* verwenden?

A Weil es in der Datenbank *pubs* den Benutzernamen *guest* (Gast) gibt.

Kontrollfragen

1. Wie widerrufen Sie das Recht von Windows 2000-Administratoren, sich als Systemadministratoren (*sa*) an SQL Server anzumelden?
2. Wie gehen Sie vor, wenn ein Benutzer alle Rechte eines Datenbankbesitzers erhalten soll, aber bereits ein anderer Benutzer der *dbo* ist?
3. Wann müssen Sie einen individuellen SQL Server-Benutzernamen für einen Windows 2000-Benutzer erstellen, statt eine Windows 2000-Gruppe zu verwenden?

Übungen

1. Erstellen Sie die folgenden Benutzernamen in SQL Server. Fügen Sie außerdem jeden Benutzernamen in eine Benutzerdatenbank auf Ihrem Server hinzu.
 - ▶ George
 - ▶ Henry
 - ▶ Ida
 - ▶ John
2. Machen Sie John zum *dbo* der Datenbank, auf die Sie gerade verwiesen haben. Beheben Sie alle Fehler, um diese Änderung zu ermöglichen.



Berechtigungen



Die gestrige Lektion hat das Sicherheitsmodell von SQL Server 2000 behandelt. Bis jetzt ist es allerdings nur darum gegangen, wie man sich über das Netzwerk bei SQL Server anmeldet und auf eine Datenbank zugreift.

Nun brauchen Sie aber Berechtigungen für alle Aktionen, die Sie in der Datenbank unternehmen möchten. SQL Server ist ein inhärent sicheres System. Die heutige Lektion zeigt, wie Berechtigungen funktionieren, worin die Unterschiede zwischen Anweisungs- und Objektberechtigungen bestehen und wie Berechtigungen zwischen Rollen und Benutzerkonten kombiniert werden. Dabei erfahren Sie auch, was Besitzketten sind und warum Sie sich damit befassen müssen.

6.1 Warum Berechtigungen verwenden?

Bisher haben Sie alle Arbeiten in SQL Server unter dem Benutzernamen *sa* (Systemadministrator) oder Ihrem Windows-Benutzerkonto abgewickelt, wobei das Windows-Konto ein Mitglied der lokalen Administratorengruppe und somit ein Mitglied der festen Serverrolle *sysadmin* ist. Zu dieser Rolle gehört auch der Benutzername *sa*, wie Sie gestern gelernt haben. Wenn Sie daher die feste Serverrolle *sysadmin* untersuchen, wissen Sie auch, was dem Benutzernamen *sa* erlaubt ist. Mitglieder der festen Serverrolle *sysadmin* unterliegen keinen Einschränkungen hinsichtlich ihrer Aktionen in SQL Server. Das ist nicht nur bequem, sondern für viele Verwaltungsaufgaben auch notwendig. Normale Benutzer sollten sich jedoch nicht mit dem Benutzernamen *sa* oder als Mitglieder der festen Serverrolle *sysadmin* anmelden, weil sie damit alle Berechtigungen in SQL Server hätten – genug, um alle Datenbanken zu löschen und den Server herunterzufahren!

Wenn Sie einen guten Sicherheitsplan für SQL Server entwerfen und umsetzen, können Sie viele Probleme von vornherein vermeiden und die Zeit sinnvoller nutzen als mit der Suche, wo und wie Daten (oder SQL Server) beschädigt worden sind. So können Sie genau festlegen, welche Datenmodifikationen erlaubt sind und welche Daten der Benutzer sehen darf. Außerdem können Sie bestimmen, ob ein Benutzer Datenbanksicherungen durchführen oder Objekte in einer Datenbank erstellen oder manipulieren darf.

Durch die Möglichkeit, mehrere Benutzernamen, Benutzer und Berechtigungen zu aktivieren, lässt sich genau festlegen, was einzelnen Benutzern erlaubt ist. Man kann damit auch die Aktivitäten der jeweiligen Benutzer überwachen. Unerlässlich ist diese Fähigkeit, wenn Sie sich auf die Suche nach »rätselhaft verschwundenen« Daten machen müssen.

6.2 Implementieren von Datenbankberechtigungen

Von vornherein ist ein wichtiger Punkt zu beachten: Alle Berechtigungen in SQL Server werden an Datenbankbenutzer vergeben. Wenn Sie also Berechtigungen untersuchen, haben Sie es immer mit Berechtigungen eines Datenbankbenutzers zu tun und nicht mit Berechtigungen zum Anmelden. Das bedeutet, dass Berechtigungen datenbankspezifisch sind.



Zu jeder Regel gibt es eine Ausnahme. Feste Serverrollen beziehen sich auf Benutzernamen (d.h. Anmeldungen) und nicht auf Datenbankbenutzer. Diese Zuordnung ist sinnvoll, denn wenn Sie Mitglied einer dieser festen Serverrollen sind, erhalten Sie Berechtigungen, die sich auf den gesamten Server und nicht nur auf eine einzelne Datenbank erstrecken.

Nehmen wir als Beispiel an, dass Sue alle Berechtigungen für die Datenbank *Verkauf* hat. (Sie ist die Filialleiterin.) Für die Tabellen in der Datenbank *Einkauf* hat sie nur SELECT-(Lese-) Berechtigungen. (Sie kann die Artikel zwar auf der Bestellung einsehen, aber nur die Einkaufsabteilung kann neue Artikel in die Bestellung aufnehmen.) Schließlich hat Sue keinerlei Berechtigungen in der Datenbank *Buchhaltung*. (Nur die Buchhalter haben hierfür Berechtigungen.)

In diesem Beispiel muss sich Sue entweder mit einem Benutzernamen im SQL Server-Authentifizierungsmodus mit SQL Server verbinden oder ihre Windows-Kontoinformationen verwenden. In jeder Datenbank hat sie ein separates Benutzerkonto (oder kann ihr Windows-Benutzerkonto bzw. die Gruppenmitgliedschaft verwenden), um Zugriff auf die Datenbank zu erhalten. Wie Sie in der gestrigen Lektion gesehen haben, könnte sie auch Gastberechtigungen verwenden, beispielsweise in der Buchhaltungsdatenbank (falls ein Datenbankbenutzer *guest* existiert). Beachten Sie, dass es für jeden Benutzer in jeder Datenbank separate Berechtigungen gibt.



Die Systemtabelle *syspermissions* verfolgt die Sicherheit innerhalb einer Datenbank. Deshalb ist Sicherheit datenbankspezifisch. Aus Gründen der Abwärtskompatibilität zu SQL Server 6.5 gibt es auch eine Sicht namens *sysprotects*. Die Tabelle und die Sicht befinden sich in jeder SQL Server-Datenbank einschließlich *master*.

Berechtigungstypen

SQL Server unterscheidet drei Arten von Aktionen in Bezug auf Berechtigungen: GRANT (erteilen), DENY (verweigern) und REVOKE (entfernen). Diese Anweisungen lernen Sie später noch genauer kennen, fürs Erste soll ein kurzes Beispiel genügen, um die Terminologie zu erklären.



Damit ein Benutzer eine Aktion ausführen kann, muss man ihm eine bestimmte Art von Berechtigung erteilen. Um eine Aktion zu verhindern, verweigert man dem Benutzer das Recht zum Durchführen der Aktion. Und um eine vorher erteilte Berechtigung wieder aufzuheben, entfernt man sie.

Es gibt zwei Arten von Berechtigungen: Anweisungs- und Objektberechtigungen. Anweisungsberechtigungen ermöglichen einem Benutzer, einen bestimmten Transact-SQL-Befehl auszuführen, während Objektberechtigungen einem Benutzer erlauben, eine bestimmte Operation durchzuführen: Daten auswählen (SELECT), Daten einfügen (INSERT), Daten aktualisieren (UPDATE) oder Daten löschen (DELETE).

Vorrang von Berechtigungen

Damit man weiß, wann eine bestimmte Berechtigung wirksam ist, muss man wissen, auf welche Weise Berechtigungen angewandt werden. In SQL Server sind alle Berechtigungen kumulativ mit Ausnahme der Berechtigung DENY, die andere Berechtigungen überschreibt.

Wenn Sie die Berechtigung SELECT aus Ihrer Mitgliedschaft in Rolle1 und die Berechtigung INSERT aus Ihrer Mitgliedschaft in Rolle2 erhalten, verfügen Sie effektiv über die Berechtigungen SELECT und INSERT. Werden Ihnen die SELECT-Berechtigungen in diesen Rollen oder in Ihrem persönlichen Konto verweigert, verfügen Sie nicht mehr über die SELECT-Berechtigung. DENY überschreibt immer alle anderen Berechtigungen.

6.3 Spezielle Berechtigungen von SQL Server

SQL Server 2000 hat mehrere Berechtigungsebenen. Die meisten dieser Berechtigungen sind datenbankspezifisch. Wie bereits erwähnt, sind feste Serverrollen an Benutzernamen und nicht an Datenbankbenutzer gebunden. In Lektion 5 haben Sie gelernt, dass jede Rolle einen bestimmten Satz von Berechtigungen mitbringt. Auch die Mitgliedschaft in der Rolle *sysadmin* impliziert einen eigenen bestimmten Satz von Berechtigungen.

In jeder Datenbank gibt es feste Datenbankrollen, von denen jede mit einem speziellen Satz von Berechtigungen verbunden ist. Außerdem hat jede Datenbank einen speziellen Benutzer – den *dbo* (Datenbankbesitzer). SQL Server kennt auch ein Konzept des Datenbank-Objektbesitzers (kurz: Objektbesitzer), selbst wenn Sie damit nicht weiter in Berührung kommen, weil Objektbesitzer keine besonderen Benutzernamen oder Kennwörter haben. Der Ersteller eines Datenbankobjekts erhält implizit alle Berechtigungen.

Später in der heutigen Lektion untersuchen Sie die öffentliche Rolle (*public*) und welche Berechtigungen mit ihr verbunden sind. Im nächsten Abschnitt lernen Sie mehr über die festen Serverrollen und ihre Berechtigungen.

Berechtigungen fester Serverrollen

Jede Rolle verfügt über implizite Berechtigungen, die Sie sich mithilfe der gespeicherten Systemprozedur `sp_srvrolepermission` ansehen können:

```
sp_srvrolepermission [[@srvrolename =] 'role']
```

Das Argument *role* ist der Name der festen Serverrolle, für die Sie die Berechtigungen anzeigen möchten.

Das folgende Beispiel zeigt die Ergebnisse, die Sie bei Ausführung dieser gespeicherten Systemprozedur erhalten:



```
exec sp_srvrolepermission 'dbcreator'
```



ServerRole	Permission
dbcreator	Add member to dbcreator
dbcreator	ALTER DATABASE
dbcreator	CREATE DATABASE
dbcreator	DROP DATABASE
dbcreator	Extend database
dbcreator	RESTORE DATABASE
dbcreator	RESTORE LOG
dbcreator	sp_renamedb

(8 row(s) affected)

Die folgenden Abschnitte gehen näher auf diese Berechtigungen und Serverrollen ein.

sysadmin

Mitglieder der Serverrolle *sysadmin* können buchstäblich jede Aktion in SQL Server durchführen. Ihnen wird ein sehr umfangreicher Satz von Berechtigungen gewährt, mit denen man sorgsam umgehen sollte. Der Benutzername *sa* ist immer ein Mitglied dieser Rolle und lässt sich nicht aus ihr entfernen. Mitglieder der festen Serverrolle *sysadmin* gelten immer als Datenbankbesitzer der von ihnen verwendeten Datenbanken. Den Zugriff auf SQL Server-Datenbanken kann man Mitgliedern der Rolle *sysadmin* nicht verweigern.



Die Benutzeroberfläche liefert zwar eine Liste der Rechte für die Mitglieder von *sysadmin*, genau genommen ist diese aber überflüssig, weil Mitglieder dieser Rolle alles tun können. Dieser Tatsache sollten Sie sich immer bewusst sein, wenn Sie Benutzer zu Mitgliedern dieser Rolle machen wollen.

serveradmin

Für Serveradministratoren, die anderweitig keine Datenbanken oder andere Objekte zu verwalten haben, bietet sich die Rolle *serveradmin* an. Mitglieder dieser Rolle können die folgenden Operationen durchführen:

- Der festen Serverrolle *serveradmin* einen weiteren Benutzernamen hinzufügen.
- Die Anweisung `DBCC FREEPROCCACHE` ausführen.
- Die gespeicherte Systemprozedur `sp_configure` ausführen, um Systemoptionen zu ändern.
- Die Anweisung `RECONFIGURE` ausführen, um die mit `sp_configure` vorgenommenen Änderungen zu installieren.
- Die Anweisung `SHUTDOWN` ausführen, um SQL Server herunterzufahren.
- Die gespeicherte Systemprozedur `sp_fulltext_service` ausführen, um den Microsoft Search-Dienst (Volltextsuche) von SQL Server 2000 zu konfigurieren.

setupadmin

Mitglieder der Rolle *setupadmin* sind normalerweise Administratoren, die Remoteserver konfigurieren. Mitglieder dieser Rolle können die folgenden Operationen durchführen:

- Der festen Serverrolle *setupadmin* einen weiteren Benutzernamen hinzufügen.
- Verbindungsserver hinzufügen, löschen oder konfigurieren.
- Autostart-Prozeduren verwalten.

securityadmin

Mitglieder der Rolle *securityadmin* können jede Operation durchführen, die sich auf serverweite Sicherheit in SQL Server bezieht. Diese Rolle kommt für Helpdesk-Personal (Mitarbeiter, die neue Konten einrichten) infrage. Mitglieder dieser Rolle können die folgenden Operationen durchführen:

- Der festen Serverrolle *securityadmin* Mitglieder hinzufügen.
- Die Anweisungsberechtigung `CREATE DATABASE` gewähren, entziehen und verweigern.

- Das Fehlerprotokoll von SQL Server mithilfe der gespeicherten Systemprozedur `sp_readerrorlog` lesen.
- Ausführen von sicherheitsrelevanten gespeicherten Systemprozeduren. Dazu gehören die Prozeduren `sp_addlogin`, `sp_droplogin`, `sp_password` (für alle außer *sysadmins*), `sp_defaultdb`, `sp_defaultlanguage`, `sp_addlinkedsevrlogin`, `sp_droplinkedsevrlogin`, `sp_dropremotelogin`, `sp_grantlogin`, `sp_revokelogin`, `sp_denylogin`, `sp_grantdbaccess`, `sp_helpllogins` und `sp_remoteoption` (Aktualisierungsteil).



Als Mitglied von *securityadmin* haben Sie nicht automatisch Zugriff auf jede Datenbank. Deshalb erhalten Sie möglicherweise von einigen gespeicherten Prozeduren wie `sp_helpdb` oder `sp_droplogin` eine Fehlermeldung, wenn Sie auf eine Datenbank zugreifen, für die Ihnen die passenden Berechtigungen fehlen. Die Informationen, die Sie sehen dürfen, werden dennoch angezeigt; im Fall von `sp_droplogin` werden Benutzernamen trotzdem gelöscht.

processadmin

Mitglieder der Rolle *processadmin* können Prozesse steuern, die auf dem Datenbankserver laufen. Dazu gehört insbesondere auch das Beenden von Abfragen, die übermäßig lange laufen oder außer Kontrolle geraten sind. Hier kommt zum Beispiel das Helpdesk-Personal infrage. Mitglieder dieser Rolle können die folgenden Operationen ausführen:

- Hinzufügen von Mitgliedern zur festen Serverrolle *processadmin*.
- Ausführen der Anweisung `KILL`, um einen SQL Server-Prozess zu beenden.

dbcreator

Mitglieder der festen Rolle *dbcreator* können Operationen durchführen, die die Erstellung und Bearbeitung von Datenbanken betreffen. Für diese Rolle kommen vorrangig Chefadministratoren infrage. Die Berechtigungen für diese Rolle schließen die folgenden Operationen ein:

- Hinzufügen von Mitgliedern zur festen Datenbankrolle *dbcreator*.
- Ausführen der gespeicherten Systemprozedur `sp_renamedb`.
- Ausführen der Anweisungen `CREATE DATABASE`, `ALTER DATABASE` und `DROP DATABASE`.
- Eine Datenbank und ein Transaktionsprotokoll wiederherstellen.



diskadmin

Mitglieder der festen Serverrolle *diskadmin* können Dateien verwalten. Diese Rolle existiert hauptsächlich aus Gründen der Abwärtskompatibilität zu SQL Server 6.5. Im Allgemeinen sind die meisten Datenbankadministratoren mit der festen Serverrolle *dbcreator* besser bedient. Mitglieder der festen Serverrolle *diskadmin* können folgende Operationen durchführen:

- Mitglieder zur festen Serverrolle *diskadmin* hinzufügen.
- Die folgenden DISK-Anweisungen ausführen (für Abwärtskompatibilität): DISK INIT, DISK REINIT, DISK REFIT, DISK MIRROR und DISK REMIRROR.
- Die gespeicherten Systemprozeduren *sp_diskdefault* und *sp_dropdevice* ausführen.
- Die gespeicherte Systemprozedur *sp_addumpdevice* ausführen, um Datensicherungsmedien hinzuzufügen.

sa

Der Benutzername *sa* verdient besondere Erwähnung. In den SQL Server-Versionen vor 7.0 waren alle Berechtigungen zur serverweiten Verwaltung mit dem Benutzernamen *sa* und SQL Server-Authentifizierung verbunden. Es gab keine separaten Rollen, um diese Berechtigungen zu verteilen. Aus Gründen der Abwärtskompatibilität ist *sa* weiterhin vorhanden und hat immer noch alle Berechtigungen wie in früheren Versionen. Das hängt allerdings damit zusammen, dass der Benutzername *sa* Mitglied der festen Serverrolle *sysadmin* ist. Man kann *sa* weder aus dieser Rolle entfernen noch umbenennen.

Der Benutzername *sa* und alle Mitglieder der festen Serverrolle *sysadmin* bilden immer den Datenbankbesitzer (Benutzer *dbo*) in jeder Datenbank nach. Dieses Verhalten lässt sich nicht ändern. Das Nachbilden des Benutzers *dbo* ist nicht das Gleiche wie ein Mitglied der festen Datenbankrolle *db_owner* zu sein.



Mit SQL Server 2000 und dem Übergang zum Windows-Authentifizierungsmodus kommt dem Benutzernamen *sa* nicht mehr die gleiche Bedeutung zu wie in früheren Versionen.

Feste Datenbankrollen

Mitglieder von festen Datenbankrollen erhalten in jeder Datenbank spezielle Berechtigungen. Im Unterschied zu festen Serverrollen sind sie jedoch für jede Datenbank spezifisch. Ein Mitglied einer festen Datenbankrolle in einer Datenbank zu sein hat keine Auswir-

kung auf Berechtigungen in irgendeiner anderen Datenbank. Die Rollen können Sie sich mit der gespeicherten Systemprozedur `sp_dbfixedrolepermission` anzeigen:

```
sp_dbfixedrolepermission [[@rolename =] 'role']
```

Das Argument *role* ist der Name der festen Datenbankrolle, für die Sie die Berechtigungen anzeigen wollen.

Die nächsten Abschnitte beschreiben die neun festen Datenbankrollen und erläutern die Ergebnisse, die Sie bei Ausführung der gespeicherten Systemprozedur `sp_dbfixedrolepermission` erhalten.

db_owner

Mitglieder der festen Datenbankrolle *db_owner* sind die »Besitzer« einer Datenbank. Sie haben sehr umfassende Berechtigungen innerhalb einer Datenbank und können fast alles tun, was der eigentliche Datenbankbesitzer tun kann. Mitglieder der festen Datenbankrolle *db_owner* können in ihren Datenbanken folgende Operationen durchführen:

- Jeder festen Datenbankrolle mit Ausnahme von *db_owner* Mitglieder hinzufügen oder aus ihr entfernen.
- Beliebige Anweisungen der Datendefinitionssprache (DDL) ausführen.
- Die Anweisungen `BACKUP DATABASE` und `BACKUP LOG` ausführen.
- Die Anweisungen `RESTORE DATABASE` und `RESTORE LOG` ausführen.
- Auslösen einer CHECKPOINT-Anweisung in einer Datenbank.
- Die folgenden Befehlsoperationen des Datenbankkonsistenzprüfers (DBCC) ausführen: `DBCC CHECKALLOC`, `DBCC CHECKFILEGROUP`, `DBCC CHECKDB`, `DBCC CHECKIDENT`, `DBCC CLEANTABLE`, `DBCC DBREINDEX`, `DBCC PROCCACHE`, `DBCC SHOW_STATISTICS`, `DBCC SHOWCONTIG`, `DBCC SHRINKDATABASE`, `DBCC SHRINKFILE` und `DBCC UPDATEUSAGE`.
- Die Berechtigungen `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `REFERENCES` und `EXECUTE` für jedes Objekt (wie für jeden Objekttyp zweckmäßig) erteilen, entfernen und verweigern.
- Mit den folgenden gespeicherten Systemprozeduren Benutzer, Gruppen, Rollen oder Aliase einer Datenbank hinzufügen: `sp_addalias`, `sp_addapprole`, `sp_addgroup`, `sp_addrole`, `sp_addrolemember`, `sp_adduser`, `sp_approlepassword`, `sp_change_users_login`, `sp_changegroup`, `sp_dropalias`, `sp_dropapprole`, `sp_dropgroup`, `sp_droprole`, `sp_droprolemember`, `sp_dropuser`, `sp_grantdbaccess` und `sp_revokedbaccess`.
- Mit der gespeicherten Systemprozedur `sp_recompile` gespeicherte Prozeduren erneut kompilieren.



- Jedes Objekt mit der gespeicherten Systemprozedur `sp_rename` umbenennen.
- Bestimmte tabellenspezifische Optionen mit der gespeicherten Systemprozedur `sp_tableoption` bearbeiten.
- Den Besitzer eines Objekts mit der gespeicherten Systemprozedur `sp_changeobjectowner` ändern.
- Volltextdienst in der Datenbank mit den folgenden gespeicherten Systemprozeduren konfigurieren: `sp_fulltext_catalog`, `sp_fulltext_column`, `sp_fulltext_database` und `sp_fulltext_table`.



Die Berechtigungen der festen Datenbankrolle *db_owner* geben an, dass Mitglieder der Rolle »jede Anweisung der Datendefinitionssprache (DDL) mit Ausnahme von GRANT, REVOKE und DENY ausführen können«. Das ist etwas erklärungsbedürftig. Standardmäßig können Mitglieder der Rolle *db_owner* für jedes Objekt in der Datenbank Berechtigungen erteilen, entfernen und verweigern. Der Datenbankobjektbesitzer (DBOO), der später in der heutigen Lektion behandelt wird, kann jedoch die Möglichkeiten des *dbo* und der Mitglieder der festen Datenbankrolle *db_owner* beschneiden.



Es gibt Unterschiede, ob man der einzige Datenbankbesitzer (*dbo*) ist oder zur Datenbankrolle *db_owner* gehört. Der erste Unterschied betrifft die Objekterstellung: Die Objekte des *dbo* besitzt der Benutzer *dbo*, während sich die Mitglieder der Rolle *db_owner* mit ihren DB-Benutzernamen als Objektbesitzer zu erkennen geben. Der andere Unterschied ist folgender: Wenn eine Datenbank beschädigt ist oder aus einem anderen Grund wiederhergestellt werden muss, aber trotzdem noch existiert, verwendet SQL Server den Benutzer *dbo* wie er in der Systemtabelle *sysdatabases* der Datenbank *master* verzeichnet ist, um Ihr Recht zur Wiederherstellung der Datenbank zu ermitteln. Das geschieht deshalb auf diese Weise, weil SQL Server nicht auf die Datenbank zugreifen kann, um die Mitglieder der Rolle *db_owner* zu bestimmen. Die Wiederherstellung einschließlich der Sicherheitsanforderungen behandelt Tag 8.

db_accessadmin

Mitglieder der festen Datenbankrolle *db_accessadmin* legen fest, welche Benutzernamen auf eine Datenbank zugreifen können. Wie bei der Rolle *securityadmin* kommen vor allem die Helpdesk-Mitarbeiter für diese Rolle infrage. Die Mitglieder dieser Rolle können die folgenden Operationen durchführen:

- Ausführen der folgenden gespeicherten Systemprozeduren: `sp_addalias`, `sp_adduser`, `sp_dropalias`, `sp_dropuser`, `sp_grantdbaccess` und `sp_revokedbaccess`.

db_securityadmin

Mitglieder der festen Datenbankrolle *db_securityadmin* können die Sicherheit innerhalb einer Datenbank verwalten und die folgenden Operationen durchführen:

- Ausführen der Anweisungen GRANT, REVOKE und DENY.
- Ausführen der folgenden gespeicherten Systemprozeduren: sp_addapprole, sp_addgroup, sp_addrole, sp_addrolemember, sp_approlepassword, sp_changegroup, sp_changeobjectowner, sp_dropapprole, sp_droprole und sp_droprolemember.

db_ddladmin

Mitglieder der festen Datenbankrolle *db_ddladmin* können die folgenden Operationen durchführen:

- Ausführen aller DDL-Befehle außer GRANT, REVOKE und DENY.
- Erteilen der Berechtigung REFERENCES für jede Tabelle.
- Gespeicherte Prozeduren mit der gespeicherten Systemprozedur sp_recompile erneut kompilieren.
- Umbenennen eines beliebigen Objekts mittels der gespeicherten Systemprozedur sp_rename.
- Bearbeiten bestimmter tabellenspezifischer Optionen mittels der gespeicherten Systemprozedur sp_tableoption.
- Änderung des Besitzers eines beliebigen Objekts mittels der gespeicherten Systemprozedur sp_changeobjectowner.
- Ausführen der folgenden DBCC-Befehle: DBCC CLEANTABLE, DBCC SHOW_STATISTICS und DBCC SHOWCONTIG.
- Volltextdienste mit den gespeicherten Systemprozeduren sp_fulltext_column und sp_fulltext_table steuern.

db_backupoperator

Mitglieder der festen Datenbankrolle *db_backupoperator* können alle Operationen durchführen, die mit der Sicherung einer Datenbank verbunden sind. Dazu gehören unter anderem:

- Ausführen der Anweisungen BACKUP DATABASE und BACKUP LOG.
- Auslösen einer CHECKPOINT-Anweisung in einer Datenbank.



db_datareader

Mitglieder der festen Datenbankrolle *db_datareader* haben die Berechtigung `SELECT` für jede Tabelle oder Sicht in einer Datenbank. Sie können niemandem Berechtigungen erteilen oder entziehen.

db_datawriter

Mitglieder der festen Datenbankrolle *db_datawriter* haben die Berechtigungen `INSERT`, `UPDATE` und `DELETE` für alle Tabellen und Sichten in einer Datenbank. Sie können niemandem Berechtigungen erteilen oder entziehen.

db_denydatareader

Mitglieder der festen Datenbankrolle *db_denydatareader* können für keine Tabelle oder Sicht in der Datenbank die Anweisung `SELECT` ausführen. Diese Option ist nützlich, wenn der Datenbankadministrator (DBA) die Objekte zwar (als Mitglied der festen Datenbankrolle *db_ddladmin*) einrichten darf, aber nicht in der Lage sein soll, vertrauliche Daten in der Datenbank zu lesen.

db_denydatawriter

Mitglieder der festen Datenbankrolle *db_denydatawriter* können für keine Tabelle oder Sicht in der Datenbank die Anweisungen `INSERT`, `UPDATE` und `DELETE` ausführen.

Der Datenbankbesitzer (dbo)

Der Benutzer *dbo* verfügt über alle Rechte, die die Mitglieder der Rolle *db_owner* haben. Es kann nur einen einzigen *dbo* für jede Datenbank geben.

Wenn ein Benutzer der *dbo* ist und ein Objekt erzeugt, wird der Besitzer des Objekts auf *dbo* gesetzt und der Benutzer *dbo* wird zum Datenbankobjektbesitzer (*dboo*), wie man es erwartet. Dies trifft auf Mitglieder der festen Datenbankrolle *db_owner* (oder beliebige andere Benutzer der Datenbank) nicht zu. Der Name des Besitzers ist sein DB-Benutzername, sofern er nicht seinen Objektnamen mit dem *dbo*-Besitzernamen qualifiziert.



Wenn Sie eine `CREATE TABLE`-Anweisung ausführen, ist *dbo* der Besitzer der Tabelle. Sind Sie via Windows-Sicherheit angemeldet und ein Mitglied der Rolle *db_owner*, erscheinen Ihre Windows-Sicherheitsinformationen als Besitzer der Tabelle. So heißt die Tabelle zum Beispiel *dbo.mytable*, wenn Sie *dbo* sind, oder *[SER\frank].mytable*, wenn Sie als Mitglied von *db_owner* angemel-

det sind. Diese Tatsache ist deshalb wichtig, weil jede SQL Server-Anweisung standardmäßig immer nach dem Objekt unter dem Namen des aktuellen Benutzers sucht und dann nach dem Objekt, dessen Besitzer der *dbo* ist, wenn Sie keinen Objektbesitzer in der Anweisung spezifizieren. Haben Sie sich beispielsweise als *[SER\Joe]* angemeldet und lösen dann die Abfrage `select * from mytable` aus, teilen Sie SQL Server nicht mit, welche Tabelle Sie meinen. Deshalb sucht SQL Server zuerst nach der Tabelle *[SER\Joe].mytable* und dann nach der Tabelle *dbo.mytable*.

Die Moral von der Geschichte: Es ist immer besser, in allen SQL-Anweisungen den Besitzer anzugeben, wenn man sich auf Objekte bezieht.



Ersetzen Sie *SER* in allen Codebeispielen durch Ihren Computernamen.

In den folgenden vier Situationen gelten Sie als *dbo* einer Datenbank:

- Sie sind der Ersteller einer Datenbank. Der Benutzername, der eine Datenbank erstellt, ist der *dbo*. Standardmäßig ist der SQL Server-Benutzername *sa* der Besitzer jeder Datenbank, wenn Sie SQL Server 2000 installieren. Das gilt selbst dann, wenn Sie die Option *Windows-Authentifizierung* bei der Installation gewählt haben, weil SQL Server nur am Ende des Setupprogramms zur Windows-Authentifizierung wechselt.
- Sie werden als Datenbankbesitzer zugewiesen. Der Besitzer einer Datenbank lässt sich später mit der gespeicherten Systemprozedur `sp_changedbowner` zuweisen. Diese gespeicherte Prozedur haben Sie gestern kennen gelernt.
- Sie melden sich bei SQL Server als Mitglied der festen Serverrolle *sysadmin* an. Wenn Sie sich über den Benutzernamen *sa* oder als Mitglied der festen Serverrolle *sysadmin* bei SQL Server anmelden, haben Sie alle Berechtigungen in allen Datenbanken, weil Sie in jeder Datenbank die Identität des *dbo* annehmen.
- Sie verbinden sich zu einer Datenbank über einen Benutzernamen, der als Aliasname für den *dbo* einer Datenbank eingerichtet ist. Zu jedem Zeitpunkt kann nur ein einziger Benutzer der *dbo* sein. In früheren Versionen von SQL Server konnten Sie einen Benutzernamen als Alias für einen anderen Benutzer in einer Datenbank einrichten. Diese Technik ist seit SQL Server 7.0 überholt; SQL Server 2000 unterstützt Aliasnamen jedoch wegen der Abwärtskompatibilität. Falls Sie die Aktualisierung von SQL Server 6.5 aus durchgeführt haben und es Benutzernamen gibt, die als Aliasnamen auf den *dbo* verweisen, werden sie als *dbo* für Ihre Datenbank nachgebildet. In SQL Server sollten Sie Benutzer in die Rolle *db_owner* hinzufügen, statt Aliasnamen zu verwenden.



Berechtigungen des Datenbankobjektbesitzers

Ein Benutzer, der ein Datenbankobjekt erstellt, ist der Datenbankobjektbesitzer dieses Objekts (Database Object Owner, DBOO). Standardmäßig ist ein Benutzer, der ein Objekt erstellt, der Besitzer des Objekts. Mitglieder der festen Datenbankrollen *db_owner* und *db_ddladmin* können Objekte unter ihrem eigenen Namen erstellen oder den Objekt-namen mit *dbo* als Besitzer qualifizieren. Wenn Sie also die folgende create-Anweisung ausführen, während Sie als Joe angemeldet sind und der Datenbankbenutzername Joe lautet, ist der Besitzer Joe:

```
use pubs
create table meineTabelle (c1 int NOT NULL)
go
```

Um die Besitzverhältnisse dieses Objekts zu überprüfen, können Sie den folgenden Code ausführen:



```
exec sp_help meineTabelle
```



Als Ergebnis erhalten Sie:

Name	Owner	Type	Created_datetime
meineTabelle	joe	user table	2001-01-03 02:34:06.047
...			

Das Objekt besitzt keine Indizes.

Keine Einschränkungen wurden für dieses Objekt definiert.

Keine Fremdschlüssel verweisen auf diese Tabelle.

Keine Sichten mit Schemabindung verweisen auf diese Tabelle.

Beachten Sie, dass der Besitzer Joe ist.

Wenn Sie jedoch Mitglied der Rollen *db_owner* oder *db_ddladmin* sind, können Sie folgende Anweisungen ausführen:



```
drop table mytable
create table dbo.meineTabelle (c1 int)
go
exec sp_help meineTabelle
```



Die Ausgabe sieht jetzt folgendermaßen aus:

Name	Owner	Type	Created_datetime
meineTabelle	dbo	user table	2001-01-03 02:40:10.300

...

Das Objekt besitzt keine Indizes.

Keine Einschränkungen wurden für dieses Objekt definiert.

Keine Fremdschlüssel verweisen auf diese Tabelle.

Keine Sichten mit Schemabindung verweisen auf diese Tabelle.

Beachten Sie, dass die Objekte nicht mehr Joe, sondern wie erwartet dem *dbo* gehören, wovon Sie sich mit der Anweisung `sp_help meineTabelle` überzeugen können.

In Produktionsdatenbanken empfiehlt es sich, nur Objekte zu verwenden, die dem Benutzer *dbo* gehören. Den Grund dafür lernen Sie kennen, wenn Sie sich später in der heutigen Lektion mit Besitzketten beschäftigen. Ist der Besitzer als *dbo* angegeben, ist der Benutzer *dbo* der Besitzer des Objekts – nicht der Benutzer, der das Objekt tatsächlich erstellt hat. Somit ist der Benutzer, der ein Objekt unter Angabe des vollständigen Besitzernamens erstellt, nicht der *dbo* dieses Objekts.

Ein Benutzer, der ein Datenbankobjekt besitzt, erhält automatisch alle Berechtigungen für dieses Objekt. Die geeigneten Berechtigungen für jeden Objekttyp werden dem Besitzer gewährt. Erstellt ein Benutzer beispielsweise eine Tabelle, werden ihm die Berechtigungen `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `REFERENCES` und `BACKUP` für diese Tabelle erteilt. Der Objektbesitz lässt sich mit der gespeicherten Systemprozedur `sp_changobjectowner` ändern.

```
sp_changobjectowner [@objname =] 'object', [@newowner =] 'owner'
```

Die Argumente haben folgende Bedeutung:

- *object* ist der Name eines Datenbankobjekts (Tabelle, Sicht oder gespeicherte Prozedur). Falls erforderlich kann das Objekt mit dem Objektbesitzer gekennzeichnet werden (was sich immer empfiehlt).
- *owner* ist der Benutzername, der Rollenname oder der Windows NT-Benutzer bzw. die Windows NT-Gruppe, dem bzw. der das mit dem Objektparameter angegebene Objekt gehören soll.

Um den Objektbesitzer der Tabelle von *dbo* in Joe zu ändern, führen Sie die folgende Anweisung aus:

```
exec sp_changobjectowner 'dbo.meineTabelle', 'Joe'
```



Benutzerberechtigungen

Auf eine Datenbank greifen vorwiegend reguläre Benutzer zu. Der Datenbankbenutzer hat keine inhärenten Rechte oder Berechtigungen, die sich von den Berechtigungen der Rolle *public* unterscheiden (siehe den nächsten Abschnitt). Alle Rechte müssen den Benutzern, den Rollen des Benutzers oder der öffentlichen Rolle explizit erteilt oder zugewiesen werden.

Benutzern gewährte Berechtigungen können in Anweisungs- und Objektberechtigungen eingeteilt werden:

- *Anweisungsberechtigungen* erlauben Benutzern das Erstellen neuer Datenbanken, das Erstellen neuer Objekte in einer existierenden Datenbank sowie die Datensicherung der Datenbank und des Fehlerprotokolls. Anweisungsberechtigungen erlauben es, bestimmte Befehle oder Anweisungen auszuführen statt mit bestimmten Objekten umzugehen.
- *Objektberechtigungen* erlauben Benutzern die Durchführung von Aktionen mit individuellen Objekten. Beispielsweise können Benutzer die Möglichkeit haben, Daten aus einer Tabelle zu lesen (auszuwählen, `SELECT`), eine gespeicherte Prozedur auszuführen oder Daten in einer Tabelle oder Sicht zu bearbeiten (mit den Berechtigungen `INSERT`, `UPDATE` und `DELETE`).

Im Anschluss an den folgenden Abschnitt erfahren Sie mehr zu beiden Arten von Berechtigungen und wie man sie erteilt.

Die Rolle *public*

Die Rolle *public* gibt es in jeder Datenbank und sie kann nicht entfernt werden. Diese Rolle ist sehr nützlich; man muss aber auch verstehen, wie man sie einsetzt.

Alle Datenbankbenutzer, Rollen, Windows-Benutzer und Windows-Gruppen in einer Datenbank sind Mitglieder der Rolle *public* und können aus dieser nicht entfernt werden. Die *public*-Rolle ist daher hervorragend geeignet, wenn jeder eine Berechtigung haben soll.

SQL Server liefert dafür selbst ein Beispiel: Die `SELECT`-Berechtigung sowie die `EXECUTE`-Berechtigung bei vielen gespeicherten Systemprozeduren sind für alle Systemtabellen in jeder Datenbank der *public*-Rolle erteilt. Dies ist für erfolgreiches Arbeiten des SQL Servers erforderlich und sollte nicht geändert werden.



Seien Sie vorsichtig, wenn Sie der *public*-Rolle Berechtigungen erteilen. Denken Sie daran, dass jeder diese Berechtigungen erhält – auch Benutzer, die erst später zur Datenbank hinzukommen.

6.4 Anweisungsberechtigungen

Anweisungsberechtigungen erlauben einem Datenbankbenutzer, einer Datenbankrolle oder einem Windows-Benutzer bzw. einer Windows-Gruppe das Erstellen neuer Datenbanken, das Erstellen neuer Objekte in einer existierenden Datenbank sowie die Datensicherung der Datenbank und des Transaktionsprotokolls. Anweisungsberechtigungen erlauben es, bestimmte Befehle oder Anweisungen auszuführen statt mit bestimmten Objekten umzugehen.

Man sollte genau überlegen, welche Berechtigungen zur Erstellung eines Objekts man einem Benutzer oder einer Rolle erteilt. Wenn ein Benutzer ein Objekt erstellt, wird er der Besitzer dieses Objekts (sofern nicht der Ersteller den Besitzer als *dbo* beim Erstellen des Objekts spezifiziert) und hat alle Berechtigungen, die mit dem Besitz von Datenbankobjekten verbunden sind. Später werden Sie sehen, dass es zu kniffligen Situationen kommen kann, wenn verschiedene Objekte mit unterschiedlichen Besitzern vorhanden sind.

Anweisungsberechtigungen sollten nur gewährt werden, wenn es unbedingt notwendig ist. Das planlose Gewähren von Berechtigungen kann eine Datenbank mit unnötigen und sogar unbrauchbaren Objekten hinterlassen.

Sie können einzelnen Datenbankbenutzern, Windows-Benutzern/Gruppen oder Datenbankrollen einschließlich der *public*-Rolle Anweisungsberechtigungen erteilen. Zu den Anweisungsberechtigungen, die man erteilen, entfernen oder verweigern kann, gehören `CREATE DATABASE`, `CREATE TABLE`, `CREATE PROCEDURE`, `CREATE DEFAULT`, `CREATE RULE`, `CREATE VIEW`, `CREATE FUNCTION`, `BACKUP DATABASE` und `BACKUP LOG`.

Diese Berechtigungen lassen sich einzeln oder alle auf einmal (mit dem Schlüsselwort `ALL`) erteilen. Dabei ist allerdings auf Nebeneffekte zu achten, die bei den einzelnen Befehlen auftreten.

Die Berechtigung `CREATE DATABASE`

Die Berechtigung `CREATE DATABASE` erlaubt Benutzern, ihre eigenen Datenbanken zu erstellen und somit zum *dbo* dieser Datenbanken zu werden. Der Datenbankbesitz lässt sich später mit der gespeicherten Systemprozedur `sp_changedbowner` ändern. Nur Mitglieder der festen Serverrollen *sysadmin* und *dbcreator* dürfen einem Benutzer die `CREATE DATABASE`-Berechtigung gewähren. Weil Berechtigungen immer an Benutzer (und nie an Benutzernamen) erteilt werden, müssen Sie diese Berechtigung ausschließlich in der *master*-Datenbank gewähren. In anderen Datenbanken gibt es diese Anweisungsberechtigung nicht. Die Berechtigung `CREATE DATABASE` gewährt Ihnen auch das Recht, den Befehl `ALTER DATABASE` zu verwenden. Das heißt, Sie können die Anweisung `ALTER DATABASE` nicht verwenden, wenn sie nicht die Berechtigung `CREATE DATABASE` haben.



Es ist wesentlich besser, die feste Serverrolle *dbcreator* zu verwenden als die Anweisungsberechtigung `CREATE DATABASE` zu gewähren. In der Regel brauchen Sie ohnehin die anderen Rechte, die die Rolle *dbcreator* gewährt, und der Überblick über die Verteilung der Rechte ist leichter zu behalten, wenn man mit SQL Server-Rollen arbeitet.

Die Berechtigungen `CREATE TABLE`, `VIEW`, `FUNCTION`, `PROCEDURE`, `DEFAULT` und `RULE`

Die Berechtigungen `CREATE TABLE`, `VIEW`, `FUNCTION`, `PROCEDURE`, `DEFAULT` und `RULE` ermöglichen Benutzern die Ausführung der betreffenden Anweisungen, um Objekte in der Datenbank zu erstellen, in der die Berechtigungen erteilt wurden. Häufig erteilt man diese Berechtigungen an Programmierer, damit diese die von ihnen in einer Datenbank benötigten Ressourcen während der Entwicklung erstellen können.



Alle `CREATE`-Berechtigungen schließen das Recht ein, beliebige Objekte zu verändern oder zu löschen, die ein Benutzer erstellt hat. Wenn man diese Berechtigung erteilt, kann das zu ernststen Problemen in der Datenbank führen, weil ein Benutzer Objekte löschen kann, die er nicht mehr benötigt, aber andere Benutzer möglicherweise noch mit diesen Objekten arbeiten. Ein Benutzer kann ein Objekt auch ändern und es damit für andere Benutzer in der Datenbank unbrauchbar machen. Derartige Abhängigkeiten können Sie mit der gespeicherten Systemprozedur `sp_depends` ermitteln.

Die Anweisungsberechtigungen `BACKUP DATABASE` und `BACKUP LOG`

Die Berechtigungen `BACKUP DATABASE` und `BACKUP LOG` lassen sich auch einzelnen Benutzern, Windows-Benutzern/Gruppen und Rollen zuweisen. Obwohl die Datensicherung der Datenbank und der Transaktionsprotokolle normalerweise automatisch über geplante Aufträge erfolgt, die der Systemadministrator erstellt hat, ist es in bestimmten Umgebungen erforderlich, dass einzelne Benutzer die Möglichkeit zur Durchführung dieser Datensicherungen erhalten.



Auch hier ist es besser, die feste Datenbankrolle *db_backupoperator* zu verwenden, als die Anweisungsberechtigungen `BACKUP DATABASE` und `BACKUP LOG` zu gewähren. Normalerweise braucht man ohnehin die anderen Rechte, die man durch die Rollenmitgliedschaft erhält, und der Überblick über die Verteilung der Rechte ist leichter zu behalten, wenn man mit SQL Server-Rollen arbeitet.

Anweisungsberechtigungen zuweisen

Anweisungsberechtigungen können Sie mit Transact-SQL oder dem SQL Server Enterprise Manager erteilen, entfernen und verweigern.

Anweisungsberechtigungen mit GRANT erteilen

Anweisungsberechtigungen erteilt man einem Benutzer mit dem Befehl GRANT:

```
GRANT {ALL | statement_list} TO {account}
```

Die Argumente haben folgende Bedeutung:

- *ALL* steht für alle anwendbaren Anweisungsberechtigungen.
- *statement_list* ist eine aufgezählte Liste der Anweisungsberechtigungen, die ein Konto erhalten soll.
- *account* ist der Name eines Datenbankbenutzers, einer Datenbankrolle, eines Windows-Benutzers oder einer Windows-Gruppe.

Anweisungsberechtigungen mit REVOKE entfernen

Mit dem Befehl REVOKE entfernt man bereits gewährte Anweisungsberechtigungen.

```
REVOKE {ALL | statement_list} TO {account}
```

Die Argumente haben folgende Bedeutung:

- *ALL* steht für alle anwendbaren Anweisungsberechtigungen.
- *statement_list* ist eine aufgezählte Liste der Anweisungsberechtigungen, die Sie einem Konto entziehen möchten.
- *account* ist der Name eines Datenbankbenutzers, einer Datenbankrolle, eines Windows-Benutzers oder einer Windows-Gruppe.

Anweisungsberechtigungen mit DENY verweigern

Im Unterschied zum Befehl REVOKE verweigert DENY eine Anweisungsberechtigung explizit. Die Berechtigung muss einem Benutzer nicht vorher gewährt worden sein. Wenn beispielsweise Joe ein Mitglied einer Datenbankrolle ist und diese Rolle die Anweisungsberechtigung CREATE TABLE hat, kann Joe auch Tabellen erstellen. Wenn Sie jedoch nicht wollen, dass Joe Tabellen erstellen kann, selbst wenn er ein Mitglied einer Rolle ist, die



diese Berechtigung hat, können Sie Joe die Anweisungsberechtigung verweigern. Daher kann Joe die Anweisung `CREATE TABLE` nicht ausführen, selbst wenn seine Rolle ihm normalerweise das Recht geben würde, dies zu tun.

```
DENY {ALL | statement_list} TO {account}
```

Die Argumente haben folgende Bedeutung:

- *ALL* steht für alle anwendbaren Anweisungsberechtigungen.
- *statement_list* ist eine aufgezählte Liste der Anweisungsberechtigungen, die Sie einem Konto verweigern möchten.
- *account* ist der Name eines Datenbankbenutzers, einer Datenbankrolle, eines Windows-Benutzers oder einer Windows-Gruppe.

Transact-SQL-Beispiele mit Anweisungsberechtigungen

Anhand einiger Beispiele lässt sich am besten zeigen, wie man die behandelten Befehle einsetzt.

Die folgende Anweisung erteilt dem Windows-Benutzer Joe die Berechtigung zum Erstellen einer Sicht in einer Datenbank:



```
GRANT CREATE VIEW TO [SER\Joe]
```

Die nächste Anweisung entzieht Joe und Mary die Berechtigung zum Erstellen von Sichten und Tabellen:



```
REVOKE CREATE TABLE, CREATE VIEW FROM [SER\Mary], [SER\Joe]
```

Um Joe alle Berechtigungen in einer Datenbank zu erteilen, führen Sie die folgende Anweisung aus:



```
GRANT ALL TO [SER\Joe]
```




Wenn Sie den Befehl `GRANT ALL` in der Datenbank *master* ausführen, erhält der angegebene Benutzer alle Berechtigungen in dieser Datenbank. In jeder anderen Datenbank erhält der Benutzer alle Berechtigungen mit Ausnahme von `CREATE DATABASE`, weil diese spezielle Berechtigung nur in der Datenbank *master* gewährt werden kann.



Nehmen wir an, dass der Benutzer Bob ein Mitglied von *Rolle1* und *Rolle2* ist. Welche Berechtigungen hat er, nachdem Sie die folgenden Anweisungen ausgeführt haben?

```
exec sp_addrole 'Rolle1'
exec sp_addrole 'Rolle2'
go
GRANT CREATE TABLE TO Rolle1
GRANT CREATE VIEW TO Rolle2
GRANT CREATE DEFAULT TO [SER\Bob]
REVOKE ALL FROM Rolle1
DENY CREATE VIEW TO [SER\Bob]
```

Letztendlich kann Bob einen Standardwert (DEFAULT) erstellen, weiter nichts. Seine `CREATE TABLE`-Berechtigungen (erteilt an *Rolle1*) hebt der Befehl `REVOKE ALL FROM Rolle1` wieder auf. Die `CREATE VIEW`-Berechtigungen, die er durch seine Mitgliedschaft in *Rolle2* erhält, gehen verloren, wenn Bob die Berechtigung `CREATE VIEW` verweigert wird. Daher ist `CREATE DEFAULT` die einzige noch wirksame Berechtigung.

Anweisungsberechtigungen mit SQL Server Enterprise Manager verwalten

SQL Server Enterprise Manager bietet eine grafische Benutzeroberfläche, über die sich auch Anweisungsberechtigungen implementieren lassen. Um Anweisungsberechtigungen anzuzeigen oder zu bearbeiten, öffnen Sie den Datenbankordner für Ihren SQL Server. Klicken Sie mit der rechten Maustaste auf die Datenbank, die Sie anzeigen oder bearbeiten wollen, und wählen Sie **EIGENSCHAFTEN**. Gehen Sie auf die Registerkarte **BERECHTIGUNGEN**, um die Anweisungsberechtigungen für die Datenbank anzuzeigen. Abbildung 6.1 zeigt ein Beispiel für das Dialogfeld **PUBS-EIGENSCHAFTEN** mit der Registerkarte **BERECHTIGUNGEN**.

Ob eine Berechtigung erteilt, entfernt oder verweigert wird, zeigt ein Kästchen im jeweiligen Feld an:

- Ein Kontrollkästchen kennzeichnet eine erteilte Anweisungsberechtigung.
- Ein rotes X bezeichnet eine verweigerte Anweisungsberechtigung.
- Ein leeres Feld zeigt an, dass explizit keine Berechtigung zugewiesen ist.

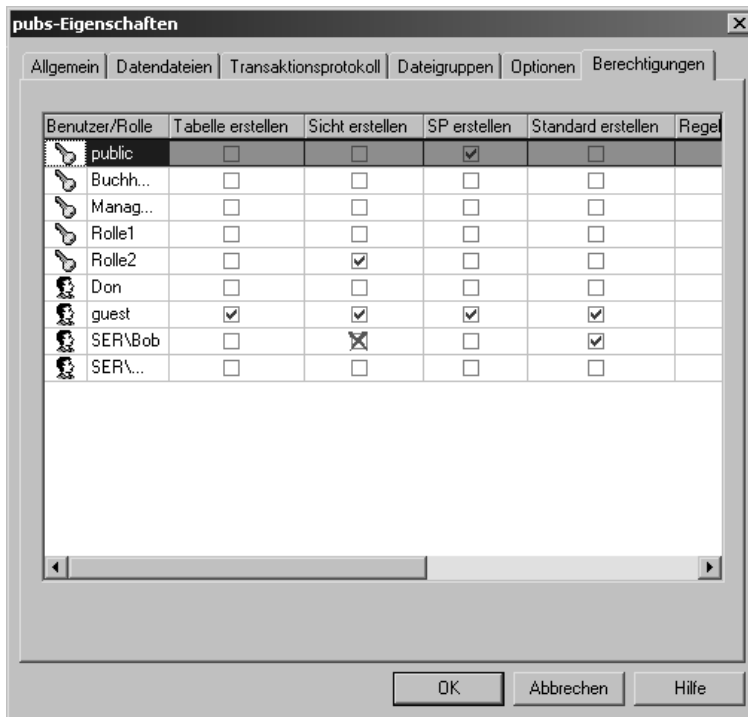


Abbildung 6.1:
Die Registerkarte
Berechtigungen für
die Datenbank pubs

Um eine Berechtigung zu gewähren, schalten Sie das betreffende Kästchen für ein Konto ein. Eine Berechtigung verweigern Sie, indem Sie zweimal auf das Kästchen klicken (bis das rote X erscheint). Wenn eine Berechtigung zuvor erteilt worden ist, führt das einmalige Klicken auf das Kästchen dazu, dass das rote X erscheint. Denken Sie daran, dass dies eine verweigte Berechtigung ist. Sie müssen noch einmal darauf klicken, um das Kontrollkästchen auszuschalten, damit SQL Server Enterprise Manager einen REVOKE-Befehl an Ihren Server schickt. Klicken Sie auf die Schaltfläche OK, um die Änderungen dauerhaft zu übernehmen.



Auf der Registerkarte **BERECHTIGUNGEN** erscheint die Anweisungsberechtigung **CREATE DATABASE** (Spalte *DB erstellen*) nur für die Datenbank *master*, weil sich die Berechtigung zum Erstellen einer Datenbank nur von der *master*-Datenbank zuweisen lässt. Abbildung 6.2 zeigt die Registerkarte **BERECHTIGUNGEN** für die Datenbank *master*.



Die Fähigkeit, Objekte in einer Datenbank zu erstellen, hat weitreichende Konsequenzen. Gewähren Sie die diesbezügliche Berechtigung nicht, sofern es nicht für die Aufgaben eines Benutzers unabdingbar ist.

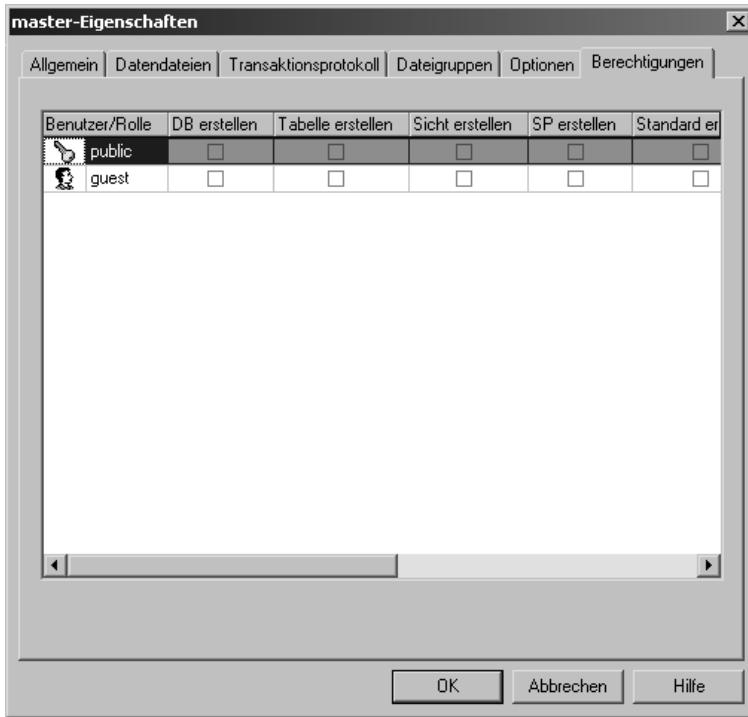


Abbildung 6.2:
Die Registerkarte
Berechtigungen für
die Datenbank mas-
ter

6.5 Objektberechtigungen

Objektberechtigungen erlauben einem Benutzer, einer Rolle oder einem Windows-Benutzer bzw. einer Windows-Gruppe bestimmte Aktionen mit einem Objekt in einer Datenbank durchzuführen. Die Berechtigungen beziehen sich nur auf das Objekt, das im Befehl zum Erteilen der Berechtigung genannt ist, und nicht auf alle Objekte, die in der gesamten Datenbank enthalten sind. Mithilfe von Objektberechtigungen können Benutzer einzelnen Benutzerkonten die Rechte geben, bestimmte Transact-SQL-Anweisungen für ein Objekt auszuführen. Objektberechtigungen sind die gebräuchlichsten Arten von gewährten Berechtigungen.

Es sind folgende Objektberechtigungen verfügbar:

- SELECT – Anzeigen von Daten in einer Tabelle, Sicht oder Spalte.
- INSERT – Hinzufügen von Daten zu einer Tabelle oder Sicht.
- UPDATE – Bearbeiten existierender Daten in einer Tabelle, Sicht oder Spalte.



- DELETE – Entfernen von Daten aus einer Tabelle oder Sicht.
- EXECUTE – Ausführen einer gespeicherten Prozedur.
- REFERENCES – Referenzieren einer Tabelle mit Fremdschlüsseln (siehe dazu Tag 14); Erstellen einer Funktion oder Sicht mit der Option WITH SCHEMABINDING, die auf ein Objekt verweist (siehe dazu Tag 15).



Die REFERENCES-Berechtigung – im Enterprise Manager mit DRI abgekürzt, was für *deklarative referenzielle Integrität* steht – ermöglicht es Benutzern (oder Anwendungen), einen Wert mit Werten in anderen Tabellen zu vergleichen, ohne die Daten in der anderen Tabelle tatsächlich zu sehen. Beispielsweise lässt sich mithilfe dieser Berechtigung erreichen, dass eine Anwendung eine übereinstimmende Sozialversicherungsnummer finden kann, ohne dass man der Anwendung die Rechte erteilen muss, die übrigen Sozialversicherungsnummern zu lesen.

Neu in SQL Server 2000 ist die Möglichkeit der so genannten *Schemabindung*, auf die Tag 15 näher eingeht. Damit lässt sich das Ändern von abhängigen Objekten verhindern, wenn man diese aus Sichten oder Funktionen heraus verwendet.

Objktberechtigungen zuweisen

Objktberechtigungen können Sie mit Transact-SQL oder dem SQL Server Enterprise Manager erteilen, entfernen und verweigern.

Objktberechtigungen mit GRANT erteilen

Der Befehl GRANT gibt jemandem eine oder mehrere Objktberechtigungen. Er entfernt auch eine DENY-Berechtigung:



```
GRANT {ALL [PRIVILEGES] | permission_list [...n]}  
{  
  [(column[...n])] ON {table | view}  
  | ON {table | view}[(column[...n])]  
  | ON {stored_procedure | extended_stored_procedure}  
  | ON {user_defined_function}  
}  
TO account[...n]  
[WITH GRANT OPTION]  
[AS {Gruppe | Rolle}]
```

Die Argumente haben folgende Bedeutung:

- *ALL* steht für alle anwendbaren Objektberechtigungen.
- *permission_list* ist eine aufgezählte Liste der Objektberechtigungen, die Sie einem Konto erteilen möchten.
- *column* ist die Ebene, bis zu der Objektberechtigungen gewährt werden können (nur für *SELECT* oder *UPDATE*).
- *account* ist der Name eines Datenbankbenutzers, einer Datenbankrolle, eines Windows-Benutzers bzw. einer Windows-Gruppe.
- *WITH GRANT OPTION* erlaubt dem Benutzer, der der Empfänger der gewährten Berechtigung ist, die ihm gewährte Berechtigung an andere Benutzer weiterzugeben.
- *AS {Gruppe | Rolle}* spezifiziert die Gruppe oder Rolle für das Erteilen einer bestimmten Berechtigung. Geben Sie diese Option an, wenn aufgrund mehrerer Rollen oder Windows-Gruppen mit Berechtigungskonflikten zu rechnen ist.

Objektberechtigungen mit REVOKE entfernen

Mit dem Befehl *REVOKE* entzieht man eine oder mehrere Objektberechtigungen, die bereits gewährt worden sind. Wenn Sie eine Berechtigung entziehen, die vorher noch nicht gewährt wurde, erhalten Sie keine Fehlermeldung; der Befehl ist in diesem Fall lediglich wirkungslos.



```
REVOKE [GRANT OPTION FOR]
{ALL [PRIVILEGES] | [...permission_list n]}
{
  [(column[...n])] ON {table | view}
  | ON {table | view}[(column[...n])]
  | {stored_procedure | extended_stored_procedure}
  | {user_defined_function}
}
FROM account[,...n]
[CASCADE]
[AS {Gruppe | Rolle}]
```

Die Argumente haben folgende Bedeutung:

- *ALL* steht für alle anwendbaren Objektberechtigungen.
- *permission_list* ist eine aufgezählte Liste der Objektberechtigungen, die Sie einem Konto entziehen möchten.
- *column* steht für die Spalte(n), der/denen Sie die Berechtigungen entziehen wollen. Objektberechtigungen auf Spaltenebene gelten nur für die Befehle *SELECT* oder *UPDATE*.



- *account* ist der Name eines Datenbankbenutzers, einer Datenbankrolle, eines Windows-Benutzers bzw. einer Windows-Gruppe.
- *CASCADE* entfernt Berechtigungen, die ein Benutzer gewährt hat, dem vorher die Berechtigung *WITH GRANT OPTION* erteilt worden ist.
- *AS {Gruppe | Rolle}* spezifiziert die Gruppe oder Rolle für das Erteilen einer bestimmten Berechtigung. Geben Sie diese Option an, wenn aufgrund mehrerer Rollen oder Windows-Gruppen mit Berechtigungskonflikten zu rechnen ist.

Objektberechtigungen mit DENY verweigern

Im Unterschied zu einem *REVOKE*-Befehl verwendet man *DENY*, um eine Objektberechtigung ausdrücklich zu verweigern. Die Berechtigung muss einem Benutzer nicht zuerst gewährt worden sein. Wenn Joe beispielsweise ein Mitglied einer Datenbankrolle ist und diese Rolle die Berechtigung *SELECT* für die Tabelle *authors* hat, kann Joe Daten in der Tabelle *authors* lesen. Soll Joe jedoch keine Daten aus der Tabelle *authors* lesen dürfen, selbst wenn er Mitglied einer Rolle ist, die diese Berechtigung hat, können Sie Joe die Berechtigung verweigern. Daher ist Joe nicht in der Lage, Daten aus der Tabelle auszuwählen, selbst wenn seine Rolle ihm normalerweise das Recht geben würde, dies zu tun.



```
DENY
{ALL [PRIVILEGES] | permission_list [...n]}
{
  [(column[...n])] ON {table | view}
  | ON {table | view}[(column[...n])]
  | ON {stored_procedure | extended_stored_procedure}
  | ON {user_defined_function}
}
TO account[,...n]
[CASCADE]
```

Die Argumente haben folgende Bedeutung:

- *ALL* steht für alle anwendbaren Objektberechtigungen.
- *permission_list* ist eine aufgezählte Liste der Objektberechtigungen, die Sie einem Konto verweigern möchten.
- *column* steht für die Spalte(n), auf die Sie den Zugriff verweigern wollen. Objektberechtigungen auf Spaltenebene gelten nur für die Befehle *SELECT* oder *UPDATE*.
- *account* ist der Name eines Datenbankbenutzers, einer Datenbankrolle, eines Windows-Benutzers bzw. einer Windows-Gruppe.

- *CASCADE* verweigert dem mit *account* spezifizierten Konto Berechtigungen und außerdem allen Konten, denen zuvor Berechtigungen von *account* mit *WITH GRANT OPTION* erteilt wurden.

Transact-SQL-Beispiele mit Objektberechtigungen

Anhand einiger Beispiele lässt sich am besten zeigen, wie man die behandelten Befehle einsetzt.

Um einem Benutzer namens Joe die Berechtigung zum Auswählen von Daten aus der Tabelle *sales* in der Datenbank *pubs* zu erteilen, führen Sie folgende Anweisung aus:



```
GRANT SELECT ON SALES TO [SER\Joe]
```

Um die Berechtigungen von Joe und Mary zum Auswählen von Daten aus der Tabelle *authors* und zum Einfügen von Daten in diese Tabelle zu entziehen, führen Sie folgende Anweisung aus:



```
REVOKE SELECT, INSERT ON AUTHORS FROM [SER\Mary], [SER\Joe]
```

Die nächste Anweisung gewährt Joe die Berechtigungen auf Spaltenebene, um Daten aus den Spalten *au_fname* und *au_lname* in der Tabelle *authors* der Datenbank *pubs* auszuwählen:



```
GRANT SELECT ON AUTHORS (AU_FNAME, AU_LNAME) TO [SER\Joe]
```

Diese Anweisung ist gleichbedeutend mit:

```
GRANT SELECT (AU_FNAME, AU_LNAME) on AUTHORS TO [SER\Joe]
```



Obwohl man Konten Berechtigungen auf Spaltenebene gewähren kann, ist es besser, den Zugriff auf Tabellen einzuschränken, indem man eine Sicht erstellt und dann dieser Sicht Berechtigungen erteilt. Eine *Sicht* ist eine vordefinierte Transact-SQL-Anweisung oder eine Gruppe von Anweisungen, die Daten zurückgeben kann. Für das vorhergehende Beispiel können Sie eine Sicht namens *viewAuthorName* erstellen, die die Felder *au_fname* und *au_lname* aus der Tabelle *authors* auswählt. Dann können Sie Joe Berechtigungen für diese Sicht erteilen. (Mehr über Sichten lernen Sie am Tag 15.)



Diese Lösung ist deshalb effizienter, weil jeder Benutzer, der diese Tabelle referenziert, hinsichtlich seiner Berechtigungen auf Spaltenebene zu überprüfen ist. Erstellen Sie dagegen eine Sicht, müssen die Berechtigungen nur einmal für die Sicht statt für jede Tabellenspalte einzeln überprüft werden.

Um Ann die Berechtigung zu erteilen, die Tabelle *publishers* auszuwählen und zu aktualisieren sowie diese Berechtigung an andere Benutzer weitergeben zu dürfen, führen Sie die folgende Anweisung aus:



```
GRANT SELECT, INSERT ON PUBLISHERS TO [SER\Ann] WITH GRANT OPTION
```

Diese Berechtigungen entziehen Sie Ann und jedem Benutzer, dem Ann die Berechtigungen erteilt hat, mit der folgenden Anweisung:



```
REVOKE SELECT, INSERT ON PUBLISHERS FROM [SER\Ann] CASCADE
```

Objektberechtigungen mit SQL Server Enterprise Manager verwalten

Objektberechtigungen sind ein Teil der Systemadministration. Das Erteilen und Entfernen dieser Berechtigungen gehört zu den allgemeinen Aufgaben, die der Administrator täglich durchzuführen hat.

SQL Server Enterprise Manager bietet eine komfortable Benutzeroberfläche, über die man Objektberechtigungen leicht und schnell verwalten kann. Berechtigungen lassen sich objektbezogen oder benutzerbezogen anzeigen. Diese Möglichkeit kann das Aufspüren von Fehlern sehr vereinfachen.

Berechtigungen für ein Objekt ansehen

Um Objektberechtigungen im SQL Server Enterprise Manager anzusehen oder zu bearbeiten, führen Sie die folgenden Schritte aus:

1. Erweitern Sie den Datenbankordner für die Datenbank, die Sie anzeigen oder bearbeiten wollen, und markieren Sie dann das Symbol für den zu verwaltenden Objekttyp.
2. Klicken Sie im Detailbereich mit der rechten Maustaste auf das gewünschte Objekt und wählen Sie EIGENSCHAFTEN aus dem Kontextmenü.
3. Klicken Sie auf die Schaltfläche BERECHTIGUNGEN. Abbildung 6.3 zeigt dieses Dialogfeld für die Tabelle *authors* der Datenbank *pubs*.

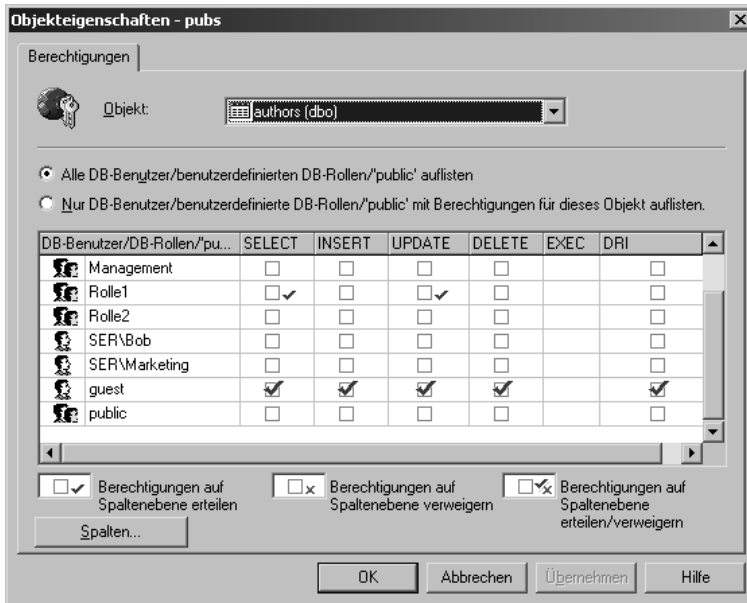


Abbildung 6.3:
Das Dialogfeld
Objekteigenschaften
mit den Berechtigungen für eine
Datenbank

Mit den Optionsfeldern im oberen Teil des Dialogfelds können Sie festlegen, ob Sie alle in der Datenbank verfügbaren Benutzer, Gruppen und Windows-Benutzer bzw. Windows-Gruppen oder lediglich eine Liste der Konten, denen bereits Berechtigungen an dem Objekt gegeben worden sind, anzeigen möchten. Beachten Sie, dass implizite Berechtigungen hier nicht erscheinen:

- DBOO-Berechtigungen werden nicht angezeigt. Wenn Joe eine Tabelle erstellt, zeigt SQL Server Enterprise Manager (und die help-Anweisungen in Transact-SQL) nicht die Berechtigungen SELECT, INSERT, UPDATE, DELETE und REFERENCES von Joe, selbst wenn sie wirksam sind. Dies liegt daran, dass diese Berechtigungen implizit zum Objektbesitz von Joe gehören.
- Wenn Ann ein Mitglied der Rolle *Buchhaltung* ist und diese Rolle INSERT-Berechtigungen für eine Tabelle hat, werden die Berechtigungen für Ann nicht gezeigt, weil auch hier die Berechtigungen implizit in der Rollenmitgliedschaft enthalten sind. Wenn Sie Ann jedoch explizit Berechtigungen erteilen, sind sie hier auch zu sehen.

Die Registerkarte **BERECHTIGUNGEN** funktioniert hier genau so, wie Sie es für Anweisungsberechtigungen kennen gelernt haben. Um eine Berechtigung zu erteilen, klicken Sie auf das Kontrollkästchen, bis ein grünes Häkchen zu sehen ist. Zur Verweigerung einer Berechtigung klicken Sie, bis ein rotes X erscheint. Wenn Sie die Berechtigung entziehen wollen, muss das entsprechende Kästchen leer sein. Nachdem Sie die gewünschten Änderungen vorgenommen haben, klicken Sie auf **ÜBERNEHMEN** oder **OK**, damit die Änderun-

gen wirksam werden. Um Berechtigungen auf Spaltenebene festzulegen, klicken Sie auf die Schaltfläche SPALTEN. Abbildung 6.4 zeigt diese Option für die Tabelle *authors* für den Benutzer [SER\Joe].

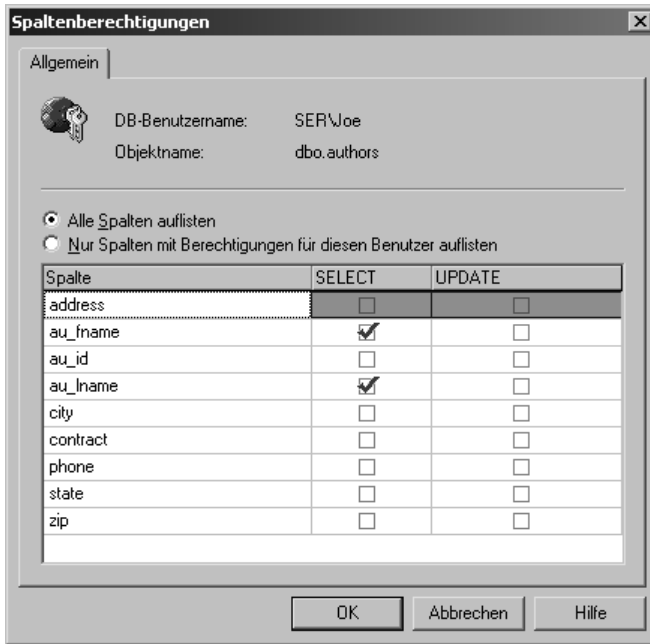


Abbildung 6.4:
Das Dialogfeld Spaltenberechtigungen für den Benutzer [SER\Joe]

Beachten Sie, dass SQL Server Enterprise Manager nur die auf den gewählten Objekttyp anwendbaren Berechtigungen anzeigt.

Berechtigungen für einen Benutzer oder eine Datenbankrolle anzeigen

Berechtigungen können Sie auch für einen Benutzer oder eine Rolle anzeigen. Um Berechtigungen im SQL Server Enterprise Manager benutzerbezogen oder rollenbezogen anzuzeigen oder zu bearbeiten, führen Sie die folgenden Schritte aus:

1. Erweitern Sie den Datenbankordner für die Datenbank, die Sie anzeigen oder bearbeiten möchten. Markieren Sie dann entweder das Symbol *Benutzer* oder das Symbol *Rollen*.
2. Klicken Sie im Detailbereich mit der rechten Maustaste auf den Benutzer oder die Rolle und wählen Sie EIGENSCHAFTEN aus dem Kontextmenü.
3. Klicken Sie auf die Schaltfläche BERECHTIGUNGEN. Abbildung 6.5 zeigt das Dialogfeld für die Datenbankrolle *Rolle1* in der Datenbank *pubs*.

4. Klicken Sie auf die entsprechenden Kästchen, um Objektberechtigungen zu erteilen oder zu entziehen. Wenn Sie die gewünschten Berechtigungen festgelegt haben, klicken Sie auf ÜBERNEHMEN oder OK, um die Änderungen dauerhaft zu übernehmen.

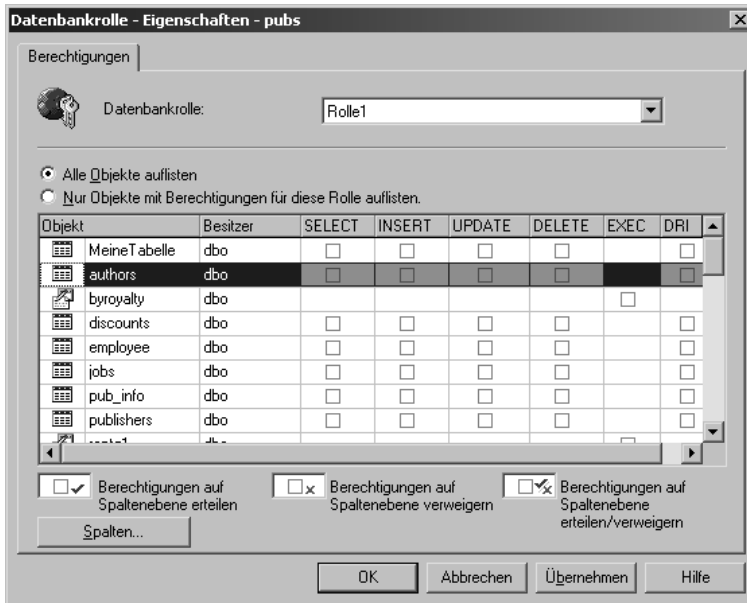


Abbildung 6.5:
Das Dialogfeld
Datenbankrolle –
Eigenschaften –
pubs

6.6 Berechtigungen für Sichten, gespeicherte Prozeduren und Funktionen

Mit Sichten, gespeicherten Prozeduren und Funktionen ist es einfacher, Berechtigungen zu verwalten, da man dann den zugrunde liegenden Datentabellen weniger Berechtigungen direkt zuweisen muss. Damit können Sie auch Berechtigungen auf Spaltenebene vermeiden, denn derartige Berechtigungen machen das Sicherheitsmodell nur übermäßig kompliziert. Auch wenn Sie an dieser Stelle noch nicht alle Einzelheiten zu Sichten und gespeicherten Prozeduren verstehen (Tag 15 widmet sich dann diesem Thema), sollte deutlich werden, wie sich mit Sichten, gespeicherten Prozeduren und Funktionen in einfacher Weise Berechtigungen einschränken lassen.

Berechtigungen für Sichten

Eine Sicht kann man sich am Einfachsten als gespeicherte Abfrage vorstellen, die den Benutzern als Tabelle erscheint. Die gespeicherte Abfrage ist eine `SELECT`-Anweisung. Um bestimmten Benutzern den Zugriff auf bestimmte Spalten oder Reihen einzuschränken, erstellen Sie eine Sicht, die sich nur auf ausgewählte Spalten einer Tabelle bezieht. Dann können Sie Berechtigungen für die Sicht an die betreffenden Benutzer erteilen und diese Benutzer haben keinerlei Rechte, die zugrunde liegende Tabelle zu sehen. Die Daten aus der Tabelle sind ihnen nur über die Sicht zugänglich. Abbildung 6.6 illustriert diesen Punkt. Hier gibt es eine Tabelle namens *Employees* (Angestellte) mit Spalten für Vornamen, Nachnamen, Adresse und Gehalt. Selbst wenn Mary die Adressen aller Mitarbeiter aktualisieren soll, hat sie keine Berechtigungen für die Gehaltsspalte *salary* (und sollte diese auch gar nicht haben). Diese Aufgabe lässt sich nach zwei Methoden lösen:

- Sie weisen Mary Berechtigungen auf Spaltenebene zu (eine ungünstige Lösung).
- Sie erstellen eine Sicht nur mit den Spalten, die Mary sehen und aktualisieren darf (*View_1* in Abbildung 6.6).

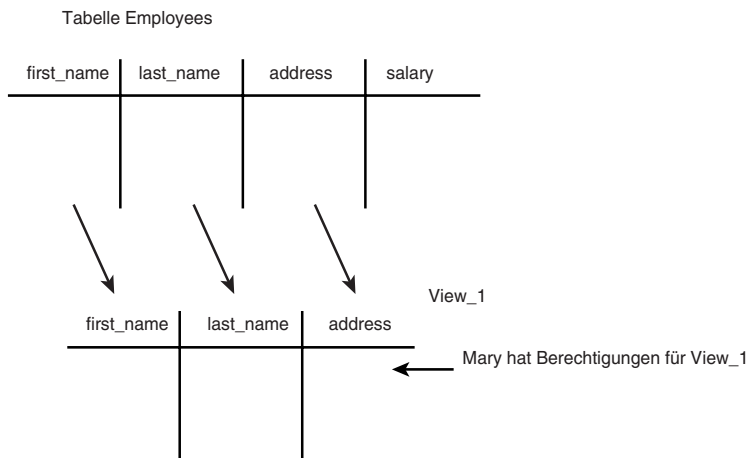


Abbildung 6.6:
Verwenden einer
Ansicht zur Bereit-
stellung von Berechti-
gungen auf Spalten-
ebene

Berechtigungen für gespeicherte Prozeduren

Vergleichbar mit den Berechtigungen an Sichten erlauben es Berechtigungen für gespeicherte Prozeduren, Benutzer von den Tabellen zu trennen und die Berechtigungen nicht direkt an die Tabellen zu erteilen. Im Unterschied zu Sichten können gespeicherte Prozeduren jedoch viele Anweisungen und Operationen enthalten. Man kann sie in der Tat als Miniprogramme betrachten, die Daten in mehreren Tabellen oder Sichten sehen und bearbeiten sowie Daten aus anderen Datenbanken übernehmen können. In SQL Server

2000 lassen sich sogar Daten aus anderen Datenquellen verarbeiten. Daher bilden gespeicherte Prozeduren einen hervorragenden Container, in dem man viele Operationen zusammenfassen kann. Wie Sie bereits weiter vorn in dieser Lektion gesehen haben, braucht ein Benutzer lediglich die EXECUTE-Berechtigung, um eine gespeicherte Prozedur auszuführen. Dabei spielt es keine Rolle, welche Aktionen die Prozedur ausführt – der Benutzer braucht einzig und allein diese Berechtigung. Dies ist einer von vielen Gründen, warum man gespeicherte Prozeduren in SQL Server ausgiebig nutzt. Tag 15 geht im Detail auf gespeicherte Prozeduren ein.

Berechtigungen für benutzerdefinierte Funktionen

Analog zu den Berechtigungen für gespeicherte Prozeduren erlauben es die Berechtigungen für Funktionen, Benutzer von den Tabellen und sogar Sichten zu trennen. Benutzerdefinierte Funktionen können fast so kompliziert wie gespeicherte Prozeduren sein.

Allerdings weisen Berechtigungen für benutzerdefinierte Funktionen einige Besonderheiten auf. Wenn eine Funktion einen Skalarwert (einen einzelnen Wert) zurückgibt, brauchen Sie EXECUTE-Berechtigungen, um die Funktion aufzurufen. Gibt die Funktion dagegen einen Tabellendatentyp zurück und verhält sich damit eher wie eine Sicht, brauchen Sie SELECT-Berechtigungen für die Funktion. Beide Optionen für Funktionen behandelt Tag 15 genauer.

6.7 Besitzketten

Jedem SQL Server-Objekt ist ein Besitzer zugeordnet. Obwohl es am besten ist, wenn alle Datenbankobjekte ausschließlich dem Benutzer *dbo* gehören, können auch normale Benutzer Objekte in Ihrer Datenbank besitzen, wenn Sie dies zulassen.



Beispielsweise kann ein Benutzer eine Sicht erstellen, die auf den Tabellen und Sichten eines anderen Benutzers basiert. Benutzer können auch gespeicherte Prozeduren erstellen, die Tabellen, Sichten und gespeicherte Prozeduren eines anderen Benutzers verwenden. Diese Typen von Objekten bezeichnet man als *abhängige Objekte*. In einer großen Datenbank kann es eine lange Reihe von Abhängigkeiten und Besitzern geben. Diese Folge der Abhängigkeiten ist die *Besitzkette*.

Sehen Sie sich folgendes Beispiel an: Dem *dbo* gehört eine Tabelle und er gibt Mary das Recht, aus seiner Tabelle auszuwählen. Mary erstellt eine Sicht, die auf der Originaltabelle basiert, und gibt Paul die Berechtigung, aus ihrer Sicht auszuwählen. Wenn Paul versucht,



Informationen aus Marys Sicht auszuwählen, kommen die Daten tatsächlich aus der Originaltabelle (die dem *dbo* gehört). Hat der *dbo* eigentlich Paul die Berechtigung zur Auswahl aus der Originaltabelle gegeben? Und darf Paul diese Daten überhaupt sehen?

Um diese Fälle zu behandeln, betrachtet SQL Server die Besitzketten der Objekte und wo die Zuweisung der Berechtigungen stattfindet.

Man unterscheidet zwei Arten von Besitzketten: Einzelbesitzerketten und unterbrochene Besitzketten, in denen Objekte in einer Abhängigkeitskette mehr als einem Benutzer gehören.

Einzelbesitzerkette

Eine Einzelbesitzerkette wird erstellt, wenn derselbe Benutzer alle abhängigen Objekte einer Kette besitzt. SQL Server überprüft in diesem Fall nur die Berechtigungen für das erste Objekt in der Kette, auf das der Zugriff erfolgt, und keine Berechtigungen an Objekten später in der Kette.

Im Beispiel von Abbildung 6.7 gehören dem *dbo* alle Objekte in der Kette. Der *dbo* erstellt *Sicht1* basierend auf den beiden Tabellen (die auch dem *dbo* gehören) und erstellt dann eine zweite Sicht (*Sicht2*) basierend auf *Sicht1* (die auf den Tabellen basiert). Wenn der *dbo* SELECT-Berechtigungen aus *Sicht2* an Melissa erteilt, werden die Berechtigungen nur einmal überprüft – wenn Melissa eine Auswahl (SELECT) aus *Sicht2* vornimmt. Um Objekte auf höheren Ebenen kümmert sich SQL Server 2000 überhaupt nicht, weil der Besitzer (in diesem Fall *dbo*) derselbe ist. Wenn der Besitzer derselbe ist, geht SQL Server davon aus, dass der ursprüngliche Besitzer die Berechtigungen gewährt haben würde, wenn SQL Server sie gefordert hätte.

Unterbrochene Besitzketten

Wenn ein Objekt von anderen Objekten abhängig ist, die unterschiedlichen Benutzern gehören, ist die Besitzkette unterbrochen. SQL Server prüft die Berechtigungen am ersten Objekt und jedem Objekt, bei dem sich der Besitz ändert. Im Beispiel von Abbildung 6.8 hat Melissa eine Sicht erstellt (*Sicht2*), die auf der *Sicht1* des Besitzers *dbo* basiert. Wenn Melissa die Berechtigung zur Auswahl aus ihrer *Sicht2* an Scott erteilt, werden die Berechtigungen für Scott an Melissas *Sicht2* und dann wieder an der *Sicht1* des *dbo* überprüft. Wenn Scott keine Berechtigungen zur Auswahl aus der *Sicht1* des *dbo* hat, kann er die SELECT-Anweisung nicht für die Sicht von Melissa ausführen.

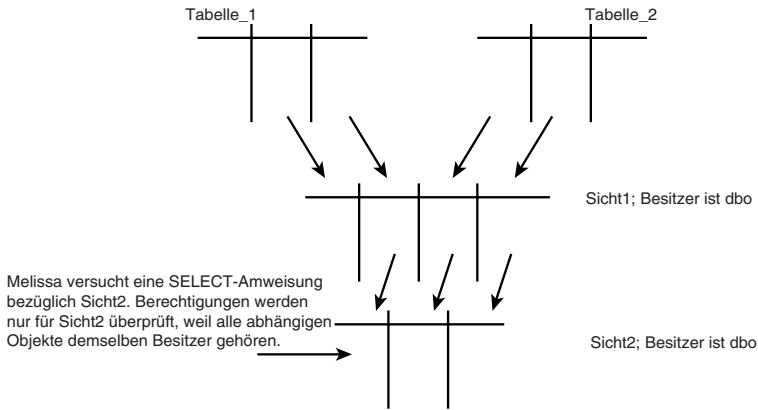


Abbildung 6.7:
Beispiel für eine
Einzelbesitzkette

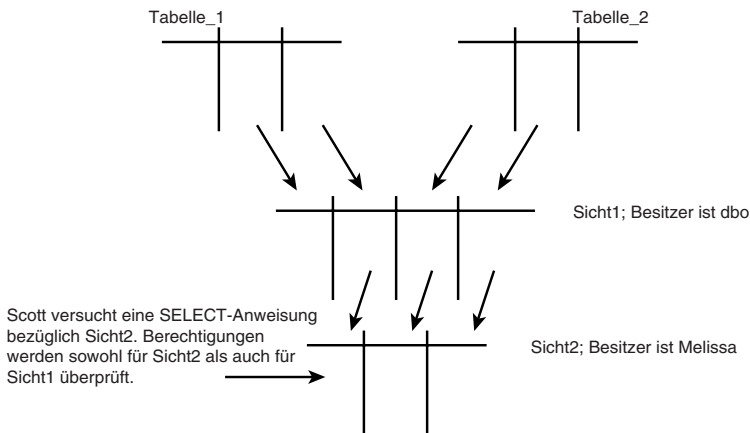


Abbildung 6.8:
Beispiel für eine
unterbrochene
Besitzkette



Unterbrochene Besitzketten können sehr schnell ziemlich komplex werden. Wenn Sie den Benutzern freie Hand lassen, Objekte zu erstellen, gleicht Ihre Datenbanksicherheit bald einer Spaghettischüssel. Deshalb sollten Objekte nur vom *dbo* erstellt werden oder von Benutzern, die Mitglieder der Rollen *db_owner* oder *db_ddladmin* sind und in ihren *CREATE*-Anweisungen ausdrücklich den Objektbesitzer auf *dbo* gesetzt haben. Diese Warnung gilt vor allem für Produktionsserver und nicht unbedingt für Entwicklungsserver (wenn Besitzketten kein Problem darstellen).

Stellen Sie sich jetzt eine Kette aus 10 oder mehr Objekten vor. Wenn Sie hier bestimmen müssen, welche Berechtigungen erforderlich sind, kann das schnell zu einem Albtraum werden. Bleiben Sie als nach Möglichkeit bei Einzelbesitzketten.



6.8 Berechtigungsstrategien entwerfen

Bislang hat sich die heutige Lektion darauf konzentriert, wie Sicherheit implementiert wird. Die folgenden Abschnitte beschäftigen sich mit Szenarios für Berechtigungen und geben eine Liste mit Richtlinien an.

Empfehlungen

Das Sicherheitsmodell von SQL Server ist sehr flexibel. Es kann deshalb aber auch schwierig sein, die beste Sicherheitsstrategie zu ermitteln. Es gibt aber mehrere Regeln und allgemeine Richtlinien für die Zuweisung von Berechtigungen. Natürlich dürfen Sie diese Regeln nicht kritiklos und starr anwenden, weil jede Situation eine spezielle Betrachtungsweise erfordert.

Wenn eine einzige Person für den gesamten SQL Server zuständig ist, muss sie sich als Mitglied der festen Serverrolle *sysadmin* oder unter dem Benutzernamen *sa* anmelden. Wenn ein Benutzer für eine einzige Datenbank zuständig ist, sollte er dieser Datenbank als *dbo* (oder als Mitglied der Rolle *db_owner*) zugewiesen werden. Benötigt ein Benutzer keine speziellen Berechtigungen, um seine Arbeit zu erledigen, sollte er als normaler Datenbankbenutzer behandelt werden und Berechtigungen aus der *public*-Rolle, einer oder mehreren Rollen, deren Mitglied er ist, oder von ihm direkt zugewiesenen Berechtigungen erhalten.

Wenn man Berechtigungen zuweist, lässt sich die Sicherheitsimplementierung einfacher verwalten und dokumentieren, wenn man folgende Regeln beherzigt:

- Berechtigungen, die alle Benutzer benötigen, weist man der öffentlichen Rolle zu.
- Berechtigungen, die alle Mitglieder einer Gruppe von Benutzern brauchen, weist man der betreffenden Windows-Gruppe zu oder erstellt eine Rolle und erteilt die Berechtigungen dieser Rolle.
- Einzelne Berechtigungen weist man Benutzern nur dann zu, wenn sich die betreffenden Berechtigungen keiner Rolle oder Windows-Gruppe zuweisen lassen.

Was man tun sollte und was nicht

Die allgemeinen Richtlinien in der folgenden Übersicht sollen Ihnen helfen, Ihre Sicherheit besser in den Griff zu bekommen. Im Kern sagen diese Richtlinien aus, dass Benutzer nur Berechtigungen erhalten sollen, die sie wirklich brauchen.

Was Sie tun sollten

ERTEILEN Sie Benutzern nur die Berechtigungen, die sie für ihre Arbeiten benötigen. Wenn beispielsweise alle Benutzer die Daten sehen müssen, erteilen Sie SELECT-Berechtigungen – wahrscheinlich der *public*-Rolle. Erteilen Sie keine Berechtigungen, wenn sie nicht notwendig sind. Benutzer verfügen gern über Berechtigungen, selbst wenn sie sie nicht brauchen. Wenn ein bestimmter Benutzer nur SELECT-Berechtigungen braucht, gewähren Sie ihm nur diese Berechtigung, selbst wenn der Benutzer alle Berechtigungen anfordern mag.

BEHALTEN Sie alle Berechtigungen im Auge, die sie gewährt haben. Führen Sie ein Protokoll, was Sie mit ihrem SQL Server machen. Eine andere Option ist die Erstellung von Skripten, die alle Objekte und Berechtigungen dokumentieren, die in der Datenbank enthalten sind. Später in der heutigen Lektion erfahren Sie, wie man Skripte erstellt.

... was nicht

Erteilen Sie Benutzern NICHT einfach alle Berechtigungen, um ein Problem zu beheben. Nehmen Sie sich die Zeit, genau herauszufinden, welche Berechtigungen sie wirklich brauchen, und gewähren Sie nur solche Berechtigungen. Beispielsweise lassen sich Probleme aufgrund fehlender Berechtigungen leicht lösen, wenn man Benutzer in die Rollen *sysadmin* oder *db_owner* aufnimmt. Damit behebt man zwar das ursprüngliche Sicherheitsproblem, schafft aber auch den Nährboden für neue kritischere Probleme, da der Benutzer zu viele Berechtigungen hat und die Datenbank (oder den gesamten Server) leicht beschädigen kann.

Erlauben Sie es normalen Benutzern NICHT, Datenbanken zu erstellen oder Objekte innerhalb von Datenbanken zu erstellen. Wenn Sie Benutzern das Erstellen von Datenbanken und Objekten erlauben, verlieren Sie nicht nur die Kontrolle darüber, was der SQL Server enthält und wo Datenbanken und Objekte liegen, Sie müssen sich auch mit unterbrochenen Besitzketten herumärgern. Alle Objekte in einer Datenbank sollte der *dbo* (oder Mitglieder der Rollen *db_owner* oder *db_ddladmin*, die den *dbo* als Objektbesitzer angeben) erstellen und dokumentieren.



Was Sie tun sollten	... was nicht
WEISEN Sie einen Benutzer einer Datenbank als <i>dbo</i> zu, wenn er für diese Datenbank verantwortlich ist. Wenn andere Benutzer Berechtigungen benötigen, die mit der Eigenschaft des <i>dbo</i> verbunden sind, müssen Sie die anderen Benutzer zu Mitgliedern der Rolle <i>db_owner</i> machen, weil es nur einen <i>dbo</i> geben kann.	

Wenn man Benutzern unnötige Berechtigungen erteilt, ist es später oft schwierig, Berechtigungen wieder zu entziehen.

In einer idealen Welt könnten sich alle Benutzer als Mitglieder der Rolle *sysadmin* anmelden und nur die Datenbank ändern, an der sie die Änderungen auch vornehmen sollen. Natürlich bleibt das eine Illusion und wenn man allen Benutzern die Berechtigungen der Rolle *sysadmin* erteilt, hat man früher oder später (meistens früher) mit größeren Schwierigkeiten zu rechnen. Trotzdem gibt es noch genügend Systeme in der Praxis, bei denen sich jeder Benutzer über den Benutzernamen *sa* und ohne Kennwort anmelden kann.

Auch wenn man Berechtigungen großzügig verteilt hat, kann alles wunderbar laufen – bis folgende Dinge passieren:

- Versehentliche Zerstörung von Datensätzen durch einen Benutzer
- Absichtliche Zerstörung von Datensätzen durch einen Benutzer
- Zerstörung von Datensätzen durch Fehlfunktionen eines Programms

Ein kluger Administrator schützt sich vor solchen Situationen mit wohldurchdachter Vergabe von Berechtigungen.

Sicherheitsskripts erstellen

SQL Server kann Datenbankobjekte und Sicherheit neu aufbauen. Man kann auch ein Skript generieren und zu einem späteren Zeitpunkt ausführen, um Objekte und Sicherheit wiederherzustellen. Um im Enterprise Manager ein Skript zu erstellen, klicken Sie mit der rechten Maustaste auf eine Datenbank und wählen ALLE TASKS / SQL-SKRIPT GENERIEREN, um das in Abbildung 6.9 gezeigte Dialogfeld zu öffnen. (Gegebenenfalls müssen Sie zuerst auf die Schaltfläche ALLE ANZEIGEN klicken, um die gleichen Optionen wie in der Abbildung zu sehen.)

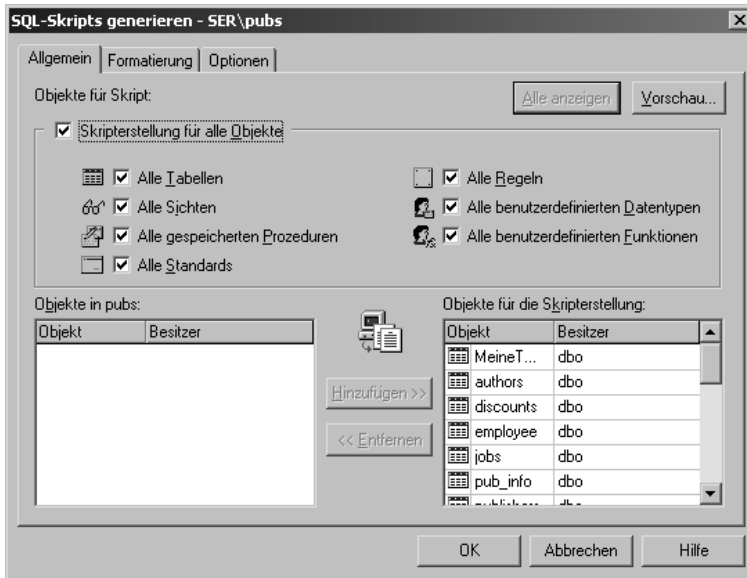


Abbildung 6.9:
Auswahl für die
Generierung eines
Skripts

Im Dialogfeld SQL-SKRIPTS GENERIEREN bestimmen Sie die Art des Skripts, das Sie erstellen möchten. Die Registerkarte ALLGEMEIN zeigt die Objekte, die Sie in das Skript aufnehmen können. Auf der Registerkarte FORMATIERUNG sehen Sie, dass standardmäßig ein Skript erstellt wird, das Objekte löscht (DROP) und erstellt (CREATE). Deaktivieren Sie diese beiden Kontrollkästchen und alle anderen Optionen auf dieser Registerkarte. Gehen Sie auf die Registerkarte OPTIONEN, um die passenden Optionen für die Sicherheitsskripterstellung auszuwählen (siehe Abbildung 6.10).

Um das Skript erstellen zu lassen, klicken Sie auf OK. Daraufhin erscheint das Standarddialogfeld DATEI SPEICHERN UNTER, in dem Sie einen Dateinamen und Standort für das Skript eingeben. Sie können auch auf der Registerkarte ALLGEMEIN auf die Schaltfläche VORSCHAU klicken, worauf das Skript erstellt und in einem Vorschaufenster angezeigt wird. Hier können Sie auf die Schaltfläche KOPIEREN klicken, um das gesamte Skript in die Zwischenablage zu kopieren (siehe Abbildung 6.11).

Beachten Sie, dass Sie abhängig vom gewünschten Ergebnis unterschiedliche Optionen aus dem Hauptbildschirm des Dialogfelds SQL-SKRIPTS GENERIEREN wählen können.

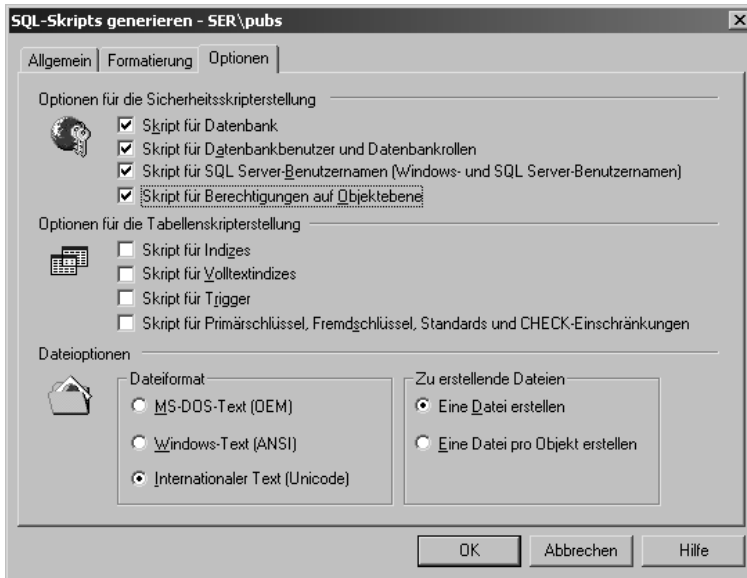


Abbildung 6.10:
Optionen für das Erstellen eines Skripts, mit dem sich die Sicherheit einer Datenbank neu aufbauen lässt.

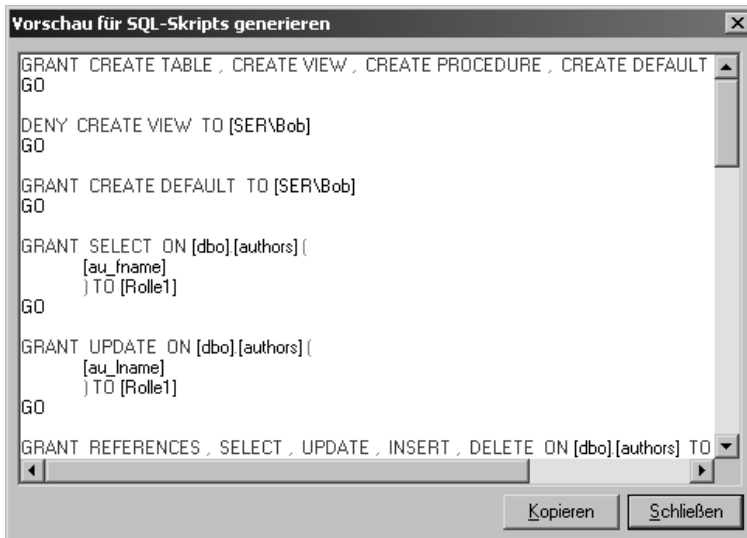


Abbildung 6.11:
Die Vorschau eines Skripts

6.9 Zusammenfassung

SQL Server stellt mit Berechtigungen ein Instrument für eine sichere Umgebung bereit. In dieser Lektion haben Sie den hierarchischen Aufbau der Sicherheit in SQL Server kennen gelernt. Durch die Kombination von Anweisungs- und Objektberechtigungen kann der Systemadministrator genau steuern, wer welche Aktionen in SQL Server unternehmen kann. Berechtigungen lassen sich mit Transact-SQL-Anweisungen oder im SQL Server Enterprise Manager erteilen, entfernen und verweigern.

Das Thema Besitzketten hat gezeigt, dass Probleme auftauchen können, wenn Sie verschiedenen Benutzern das Erstellen von Objekten in einer Datenbank erlauben. Berechtigungen müssen wohlgedacht implementiert werden, um die Sicherheit und Benutzbarkeit der Daten zu gewährleisten.

Am Ende der Lektion haben Sie gelernt, wie man Sicherheitsskripts generieren kann, um die Sicherheitsstruktur einer Datenbank bei Bedarf schnell rekonstruieren zu können.

6.10 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Das Einrichten von Datenbankberechtigungen scheint aufwändig zu sein. Sind Berechtigungen wirklich so wichtig?*
- A** Das hängt von der Umgebung und den Anforderungen ab. Je sicherer Sie Ihre Datenbank halten wollen, desto höher ist der Aufwand, ein hierarchisches Berechtigungssystem zu implementieren.
- F** *Sollte ich Rollen verwenden oder sollte ich Benutzern direkt Berechtigungen zuweisen?*
- A** Rollen und Windows-Gruppen sind viel leichter zu verwalten, weil Sie die Berechtigungen für Hunderte oder sogar Tausende von Benutzern mit einem einzigen Befehl oder Klick auf ein Kontrollkästchen steuern können.



- F *Muss ich die vordefinierten Rollen verwenden oder kann ich Berechtigungen auch ohne Rollen zuweisen?*
- A Sie sind nicht gezwungen, mit Rollen zu arbeiten. Allerdings erschweren Sie sich nur das Leben, wenn Sie auf Rollen verzichten. Mit Rollen lassen sich Berechtigungen für die üblichen Verwaltungsaufgaben komfortabel gruppieren.

Kontrollfragen

1. Wie würden Sie Mary die Berechtigung geben, Daten aus der Tabelle *MeineTabelle* zu lesen?
2. Mary hat eine Tabelle namens *MaryTabelle* erstellt und dafür an Joe *SELECT*-Berechtigungen erteilt. Joe hat eine Sicht namens *JoeSicht* erstellt. Joe möchte, dass Paul *SELECT*-Berechtigungen für diese Sicht erhält. Welche Berechtigungen sind hier erforderlich, wenn man davon ausgeht, dass die Benutzer ihre Objekte mit ihren eigenen Benutzerkennungen (nicht *dbo*) erstellt haben?
3. Joe erhält *SELECT*-Berechtigungen für *dbo.MEINETABELLE*. Der *public*-Rolle werden die *SELECT*-Berechtigungen für *MEINETABELLE* verweigert. Welche Berechtigungen sind wirksam?
4. Joe erhält *SELECT*-Berechtigungen für *dbo.MEINETABELLE*. Aus der *public*-Rolle werden die *SELECT*-Berechtigungen für *MEINETABELLE* entfernt. Welche Berechtigungen sind wirksam?
5. Was ist der bevorzugte Weg, um unterbrochene Besitzketten zu verhindern?
6. Sie sind der Besitzer einer Datenbank und wollen, dass alle Benutzer die Tabelle *MEINETABELLE* abfragen können. Welche Anweisung führen Sie zu diesem Zweck aus?
7. Sie führen die Anweisung *GRANT ALL TO JOE* in einer Benutzerdatenbank aus. Welche Berechtigungen hat Joe?



Sicherungen von Datenbanken



In der gestrigen Lektion haben Sie gelernt, wie sich eine Datenbank mithilfe von Berechtigungen sichern lässt. In dieser und der nächsten Lektion geht es um ein Thema, das vielleicht den wichtigsten aber wenig glanzvollen Teil der Wartung einer relationalen Datenbank darstellt: Sicherung und Wiederherstellung.

Unabhängig davon, wie viel Arbeit Sie investieren, um Ihre Datenbank sicher und Ihre Daten verfügbar zu machen – der Schutz der Daten ist einer der kritischsten Aspekte Ihrer Arbeit. Wenn der Server abstürzt und die Daten verloren gehen, sind Sie vielleicht auch Ihren Job los!

Die heutige Lektion erläutert den Sicherungsvorgang und die verschiedenen von SQL Server unterstützten Sicherungsarten. Bevor Sie sich mit der Sicherung selbst beschäftigen, lernen Sie zunächst einige Schutzmaßnahmen kennen, mit denen Sie dem Ausfall von Datenträgern begegnen und die Zuverlässigkeit Ihres SQL Server-Systems erhöhen können.

7.1 Daten schützen durch Spiegelung, Duplexing und Striping von Datenträgern

Die einführenden Themen beschäftigen sich mit dem fehlertoleranten Schutz für die Festplatten des Servers, erläutern die Spiegelung in SQL Server und wenden sich schließlich der Datensicherung und Wiederherstellung zu. Die Implementierung der Datensicherung und Wiederherstellung soll Datenverlusten vorbeugen, mit Fehlertoleranz für Festplatten und Geräte lässt sich zusätzliche Sicherheit realisieren. Das heißt jedoch nicht, dass Sie nach der Installation fehlertoleranter Verfahren auf die Sicherung Ihrer Datenbank verzichten können.



Am besten beginnen wir mit der Definition einiger Schlüsselbegriffe. Zwei Festplatten bezeichnet man als *gespiegelt*, wenn sie genau die gleichen Daten enthalten. Wenn sich Daten auf einer Festplatte ändern, werden sie auf der anderen Festplatte ebenfalls geändert (Abbildung 7.1). Das *Duplexing* impliziert die Spiegelung, wobei aber jede Festplatte von einem eigenen Festplattencontroller angesteuert wird (siehe Abbildung 7.2), während bei der reinen Spiegelung die Festplatten am selben Controller betrieben werden. Das *Striping* verteilt die Daten gleichmäßig über verschiedene Festplatten, wobei jede Festplatte im so genannten *Stripset* einen Teil eines logischen Datenblocks aufnimmt (siehe Abbildung 7.3).

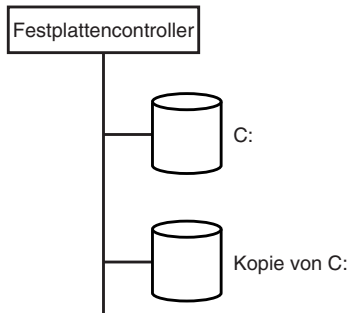


Abbildung 7.1:
Festplattenspiegelung

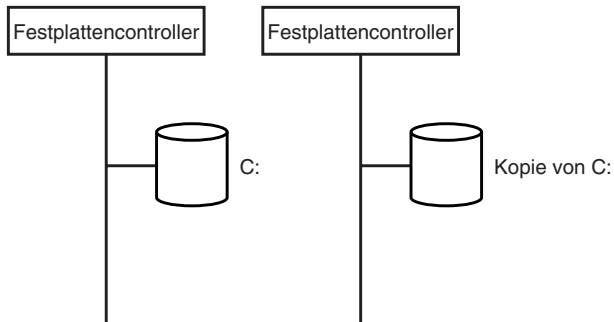


Abbildung 7.2:
Festplattenduplexing

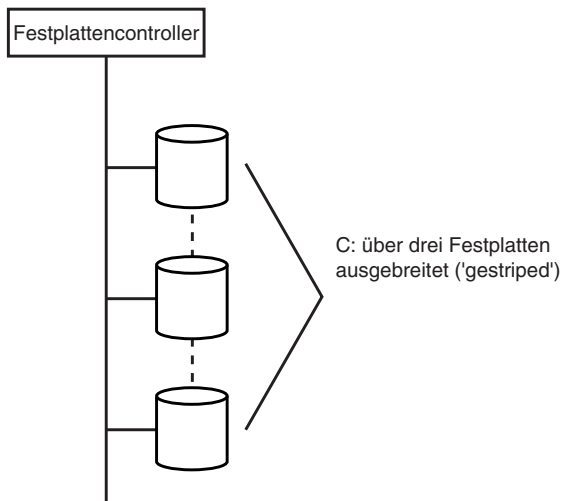


Abbildung 7.3:
Festplattenstriping

RAID

Spiegelung, Duplexing und Striping fügen sich in ein als RAID (Redundant Array of Inexpensive Disks) bekanntes Konzept ein. Man konfiguriert Festplatten in derartigen RAID-Arrays, um die auf den Festplatten enthaltenen Daten zu schützen und eine hohe Verfügbarkeit zu gewährleisten. Wenn sich Datenverluste vermeiden lassen, kann man die Benutzer vor den meisten Hardwarefehlern abschirmen. Man unterscheidet die folgenden sieben RAID-Stufen:

- RAID 0: Festplattenstriping ohne Paritätsinformationen
- RAID 1: Datenträgerspiegelung oder Duplexing
- RAID 2, 3, 4: Zwischenstufen auf dem Weg nach RAID 5 (selten eingesetzt)
- RAID 5: Festplattenstriping mit Parität, wobei die Paritätsdaten auf alle Datenträger geschrieben werden
- RAID 10: Festplattenspiegelung mit Striping, wobei ein Festplattenarray auf einen anderen Satz von verteilten Festplatten mit einem separaten Controller gespiegelt wird



In letzter Zeit setzt man immer häufiger RAID 10 ein, das aber in vielen Texten noch nicht in der Liste der RAID-Varianten auftaucht. Bei RAID 10 ist der Festplattenbedarf doppelt so hoch wie für ein RAID-5-System, weil RAID 10 nicht nur das Striping realisiert, sondern die Stripesets auch noch auf unterschiedlichen Controllern spiegelt.

RAID 0 bietet keine Fehlertoleranz an sich, verbessert aber die Lese-/Schreibleistung. RAID 5 (bzw. RAID 10) ist die bessere Wahl, da hier sowohl die Leistung gesteigert als auch Fehlertoleranz geboten wird.

RAID 1 bedeutet entweder die Spiegelung oder Duplexing der Festplatten. Fällt bei gespiegelten Festplatten die SCSI-Karte aus, die die gespiegelten Festplatten steuert, sind die Daten dieser Festplatten nicht mehr zugänglich. Falls aber nur eine der Festplatten ausfällt, können Sie weiter auf die Daten zugreifen. Falls Sie mit Festplattenduplexing arbeiten, können Sie auf Ihre Daten zugreifen, auch wenn eine Festplatte oder ein Festplattencontroller ausfällt. Aus diesem Grund stellt ist Duplexing im Sinne der Fehlertoleranz besser geeignet (kostet aber einen zusätzlichen Festplattencontroller).

Die RAID-Stufen 2, 3 und 4 sind Zwischenschritte zum Festplattenstriping mit Parität, die funktionell durch RAID 5 ersetzt wurden.

RAID 5 ist eine Konfiguration, die aus drei oder mehr logisch verknüpften Festplatten besteht. Jede Platte enthält einen entweder einen Abschnitt (»Stripe«) mit Daten oder mit Paritätsinformationen. RAID 5 verteilt die Paritätsinformationen für jeden Datenabschnitt

gleichmäßig auf das Festplattenarray. Abbildung 7.4 zeigt ein Beispiel für ein Stripeset, das aus drei Festplatten besteht.

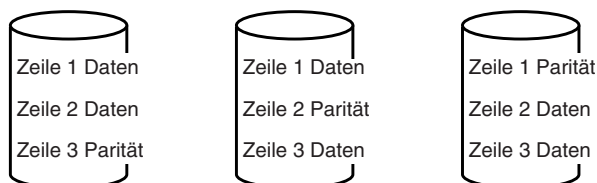


Abbildung 7.4:
RAID 5-Striping von Daten mit
Parität

Paritätsinformationen sind ein Satz berechneter Werte, die es ermöglichen, fehlende Daten wiederherzustellen. Heutige Paritätsinformationen erreichen eine Genauigkeit von 1 zu 2 Milliarden. Falls eine einzelne Festplatte ausfällt, während auf die Paritätsinformationen und die anderen Festplatten weiterhin zugegriffen werden kann, können die Daten der ausgefallenen Festplatte mithilfe der Paritätsinformationen restauriert werden. Falls die Platte mit den Paritätsinformationen ausfällt, haben Sie immer noch Zugriff auf die Festplatten mit den Daten. In beiden Fällen erleiden Sie bei Verwendung von RAID mit Parität keinen Datenverlust.

In Zeile 1 befindet sich die Hälfte der Daten auf Festplatte 1, die zweite Hälfte auf Festplatte 2 und die Paritätsinformationen (die in Kombination mit den Daten einer Festplatte ausreichend Informationen enthalten, um die restlichen Daten wiederherzustellen) befinden sich auf der dritten Festplatte. In Zeile 2 befinden sich die Daten auf Festplatte 1 und 3; die Paritätsinformationen sind auf Festplatte 2 untergebracht. Die dritte Zeile ist ähnlich unterteilt. Wenn Sie Ihrem Set mehr Festplatten hinzufügen, steigert sich die Leistungsfähigkeit, da der Overhead geringer ausfällt. Das Risiko, dass mehr als eine Festplatte ausfällt, steigt jedoch (da die Fehlerrate theoretisch für jede Festplatte gleich ist). Obwohl in diesem Beispiel nur drei Festplatten verwendet werden, können Sie so viele Festplatten einsetzen, wie es Ihre Hard- oder Software erlaubt.

RAID 5 wird generell als die kostengünstigste Version betrachtet, da hier der wenigste Overhead auftritt – insbesondere, wenn Sie die Anzahl der eingesetzten Festplatten erhöhen. RAID 10 wird vor allem für die Installation von Highend-Servern in Betracht gezogen. Spiegelung und Duplexing ist eine teurere Lösung, da Sie für die gleiche Datenmenge doppelt soviel Festplattenplatz benötigen.

Unser Beispiel nimmt an, dass Sie 27 GB Platz für die Daten des Datenbankservers brauchen. Falls Sie sich für die Einrichtung eines RAID 5-Arrays mit 9 GB SCSI-Festplatten entscheiden, brauchen Sie mindestens vier Festplatten, und zwar deshalb, weil Sie effektiv eine komplette Festplatte für die Speicherung der Paritätsdaten benötigen, womit nur noch drei Festplatten für die Unterbringung der Daten zur Verfügung stehen und drei Festplatten mal 9 GB = 27 GB. Um dieselbe Datenmenge auf einem gespiegelten System unterzubringen, wären sechs 9 GB Festplatten (und normalerweise eine dickere Brieftasche) erforderlich.



Hardwarebasierte RAID-Implementierung

Viele Computerhersteller implementieren hardwarebasierte RAID-Lösungen. Diese Methoden bieten eine höhere Leistung im Vergleich zu der in Windows NT enthaltenen softwarebasierten RAID-Lösung (die Sie gleich kennen lernen werden). Normalerweise werden hardwarebasierte RAID-Lösungen mit speziellen Controllern und teilweise auch mit eigenen Gehäusen (entweder eigenständig oder in so genannten Serverracks) installiert. Der Nachteil dieser Lösungen liegt darin, dass diese Controller (zumindest die guten) ziemlich teuer sind. Sie erhalten einen ganzen Stapel Festplatten für den Preis eines dieser Controller. Dennoch verkaufen sich diese Karten recht gut, so dass es offensichtlich triftige Gründe dafür gibt.

Softwarebasierte RAID-Lösungen unter Windows 2000 Server

Wenn Sie SQL Server 2000 unter Windows 2000 Server (oder Windows NT-Server) betreiben, können Sie die im Betriebssystem enthaltene Möglichkeit nutzen, RAID-Lösungen der Stufen 0, 1 und 5 mit normalen Festplatten zu implementieren.

RAID und andere Windows-Plattformen

Windows 98/ME, Windows NT Workstation und Windows 2000 Professional unterstützen diese Softwarefunktionen zwar nicht, können aber problemlos auf hardwarebasierten RAID-Lösungen betrieben werden (da das Betriebssystem eine solche Lösung nicht bemerkt).

Auswahl des Dateisystems

Eine weitere Überlegung im Hinblick auf RAID-Lösungen und Festplatten im Allgemeinen besteht darin, welches Dateisystem Sie für den Betrieb von SQL Server wählen. Die Betriebssysteme Windows 9x, Windows NT und Windows 2000 unterstützen derzeit drei Dateisysteme:

- *FAT* (auch als *FAT16* bezeichnet) ist das Dateisystem, das seit den frühen MS-DOS-Versionen allgegenwärtig ist. Dieses Dateisystem besitzt keine Sicherheitseinstellungen für Dateien und unterstützt keine erweiterten Funktionen wie z.B. Verschlüsselung. Es eignet sich im Allgemeinen nicht, um Dateien von SQL Server zu speichern.
- *FAT32* ist eine Weiterentwicklung des *FAT16*-Dateisystems. Die Effizienz der Datenspeicherung wurde verbessert, die Einschränkungen hinsichtlich der Sicherheit und der Verschlüsselung bestehen jedoch weiterhin. *FAT32* ist in späteren Versionen von Windows 95 (OSR2) implementiert und wird von Windows 98 und Windows 2000 vollständig unterstützt. Beachten Sie, dass Windows NT 4.0 dieses Dateisystem nicht unterstützt.

- NTFS (NT File System) unterstützt viele erweiterte Funktionen, so z. B. Datenkomprimierung und Verschlüsselung (unter Windows 2000), welche man jedoch in den meisten OLTP-Umgebungen (Online Transaction Processing) nicht einsetzen sollte. NTFS bietet aber vor allem die Möglichkeit, Zugriffsberechtigungen auf Datei- und Verzeichnisebene zu erteilen. Dadurch lässt sich der Zugriff auf die Datendateien von SQL Server nur auf die SQL Server-Dienst-Konten (und die Windows-Administratoren) beschränken.

7.2 Wie die Datensicherung funktioniert

SQL Server 2000 sichert Daten *dynamisch*. Das bedeutet, dass Sie die Datenbank während ihrer Sicherung nicht anhalten müssen. Dennoch empfiehlt es sich, die Sicherungen zu den Zeiten auszuführen, zu denen die Datenbank relativ wenig benutzt wird, da für die Sicherung doch einige Systemressourcen gebraucht werden. Benchmarks zeigten jedoch, dass eine laufende Sicherung wenig Einfluss auf den Datendurchsatz des SQL Servers in Bezug auf Abfragen und Änderungen hat.

SQL Server-Datenbanksicherungen sind in Bezug auf den *Abschluss* des Sicherungsvorgangs konsistent. Wenn also eine Sicherung um 18:00 abgeschlossen ist, verfügen Sie für diesen Zeitpunkt über ein vollständiges Abbild Ihrer Daten.



In allen Versionen von SQL Server vor 7.0 war eine Sicherung mit dem *Beginn* der Sicherung konsistent. Diese Änderung kann unbedeutend sein oder auch einen Einfluss auf Ihre bisher erstellten Sicherungskonzepte haben. In beiden Fällen ist es jedoch erwähnenswert.

Die Datenbank- und Transaktionsprotokoll-Sicherungen können Sie dynamisch in Dateien speichern oder Speicherplatz auf so genannten Sicherungsmedien reservieren. Sicherungsmedien können Dateien auf einer Festplatte, Bänder, Dateien auf Netzlaufwerken oder Named Pipes sein. Named Pipes werden normalerweise für Backup-Lösungen von Drittherstellern verwendet. In der Regel erstellt man Sicherungskopien auf Festplatten oder Bändern.

Mitglieder der festen Serverrolle *sysadmin* oder der Datenbankrollen *db_owner* und *db_backupoperator* können eine Sicherungskopie der Datenbank oder des Transaktionsprotokolls durchführen. Wie Sie in Lektion 6 gelernt haben, können Sie anderen Benutzern die Berechtigungen zum Erstellen von Sicherungskopien einer Datenbank und eines Transaktionsprotokolls erteilen.



SQL Server 2000-Sicherungskopien sind inkompatibel zu allen Vorgängerversionen von SQL Server. SQL Server 2000 kann zwar Sicherungen von SQL Server 7.0 lesen, jedoch keine Sicherungen von früheren SQL Server-Versionen. Auch wenn dieses Problem hauptsächlich die Aktualisierung von früheren Versionen betrifft, müssen Sie es als Administrator kennen. Eine Sicherungskopie von einer Version vor 7.0 können Sie nur restaurieren, indem Sie die alte Version installieren und dann bei einer SQL Server 6.5-Datenbank mit dem SQL Server-Aktualisierungs-Assistenten arbeiten oder bei einer früheren Datenbankversion die Daten exportieren/importieren.

Arten der Sicherung

Es gibt verschiedene Arten von Sicherungen, die Sie mit SQL Server 2000 durchführen können. Sie werden die vollständige Datenbanksicherung, die differenzielle Datenbanksicherung, die Sicherung von Dateien und Dateigruppen sowie die Sicherung von Transaktionsprotokollen kennen lernen. Im Rahmen der folgenden Erörterung werden wir alle Sicherungen außer der Sicherung von Transaktionsprotokollen als *Datenbanksicherungen* bezeichnen.



Aus historischen Gründen werden Backups manchmal als *Dumps* oder *Datenbank-Dumps* bezeichnet.

SQL Server 2000 erlaubt folgende Sicherungsarten:

- *Datenbanksicherung*: Sichert eine komplette Datenbank, einschließlich aller Tabellen, Indizes, Systemtabellen und Datenbankobjekte (die sich in den genannten Systemtabellen befinden). Eine Datenbanksicherung sichert gleichzeitig die Einträge im Transaktionsprotokoll. Es werden jedoch keine leeren Seiten gespeichert und es werden auch keine Einträge aus dem Transaktionsprotokoll entfernt.
- *Differenzielle Datenbanksicherung*: Hierbei werden alle Datenseiten, die seit der letzten vollständigen Datenbanksicherung geändert wurden, gesichert. Ein Grund für die Verwendung differenzieller Datenbanksicherung könnte für Sie darin bestehen, dass damit die Wiederherstellung schneller vor sich geht als bei der Verwendung von Transaktionsprotokollsicherungen. Den Wiederherstellungsvorgang werden Sie morgen näher kennen lernen.
- *Sicherung von Dateien und/oder Dateigruppen*: Hierbei werden statt der gesamten Datenbank nur ausgewählte Dateien oder Dateigruppen gesichert. Falls Sie eine oder mehrere Tabellen oder Indizes einer eigenen Dateigruppe zugeordnet haben, statt sie in der vorgegebenen Dateigruppe zu belassen, können Sie diese unabhängig von ande-

ren Daten sichern. Der Vorteil liegt darin, dass Sie beim Ausfall einer Festplatte, nur die betroffenen Dateien wiederherstellen müssen, wenn Sie eine Sicherungskopie der auf dieser Festplatte enthaltenen Dateien besitzen. Sie ersparen sich damit die Wiederherstellung der kompletten Datenbank und erhalten einen Zeitvorteil. Diese Vorgehensweise geht jedoch über die Aufgaben des durchschnittlichen SQL Server-Administrators hinaus. Hier sei auf die einschlägige Literatur verwiesen.

- *Sicherung des Transaktionsprotokolls:* Diese Sicherungsart enthält eine Kopie der Aufzeichnungen der im Laufe der Zeit an Ihrer Datenbank durchgeführten Änderungen. Darin sind sowohl die von den Benutzern ausgeführten Anweisungen als auch die im Hintergrund ablaufenden Systemaktivitäten, wie Speicherzuweisungen innerhalb der Datenbank, aufgezeichnet. Dieses Protokoll kann dann auf Ihre Datenbank angewendet werden, um alle darin aufgezeichneten Aktivitäten (Transaktionen) erneut durchzuführen. Eine weitere angenehme Eigenschaft des Transaktionsprotokolls ist es, dass Sie Ihre Transaktionen bis zu einem von Ihnen bestimmten Zeitpunkt wiederholen lassen können. Dazu erfahren Sie mehr in Lektion 8.

Transaktionen und Prüfpunkte

Transaktionen (auf die Tag 12 näher eingeht) sind eine logische Zusammenfassung von Operationen, die entweder alle zusammen erfolgreich abgeschlossen werden oder gemeinsam scheitern. Jede ausgeführte Anweisung zur Änderung von Daten in der Datenbank ist (standardmäßig) eine Transaktion und wird im Transaktionsprotokoll aufgezeichnet. Sie können aber auch mehrere Anweisungen explizit zu einer Transaktion zusammenfassen, so dass diese Kombination von Anweisungen wie eine einzelne Anweisung behandelt wird und entweder insgesamt erfolgreich ist oder scheitert.

Die im Transaktionsprotokoll aufgezeichneten Informationen können verwendet werden, um den Zustand einer Datenbank zu einem vom Zeitpunkt einer vollständigen Datenbanksicherung abweichenden Zeit wiederherzustellen. Andere Vorgänge, wie die Speicherplatzzuweisung, werden ebenfalls im Transaktionsprotokoll aufgezeichnet. Wenn Sie es nicht sichern oder löschen, wird mit der Zeit der Platz im Transaktionsprotokoll verbraucht (oder falls Sie keine maximale Größe für das automatische Wachstum angegeben haben, nimmt es den gesamten verfügbaren Speicherplatz der Festplatte ein, auf dem es sich befindet). Die Optionen für das automatische Größenwachstum hat Tag 4 besprochen. Mithilfe der Anweisung BACKUP LOG, zu der Sie heute noch Näheres erfahren, werden abgeschlossene Transaktionen aus dem Transaktionsprotokoll entfernt. Wenn eine Transaktion abgeschlossen wurde (d.h., alle Datenänderungen wurden erfolgreich durchgeführt), werden die Aufzeichnungen über diese Transaktion sofort auf die Festplatte geschrieben. Damit ist garantiert, dass Ihre Datenbank im Falle eines auftretenden Fehlers wiederhergestellt werden kann. Wie werden aber nun die Datenseiten auf die Festplatte geschrieben? Hierfür ist ein als Prüfpunkt bezeichneter Vorgang zuständig.



Prüfpunkte



Tag 12 erläutert umfassend das Prüfpunktkonzept, aber eine kurze Einführung erscheint uns hier bereits nötig. Ein *Prüfpunkt* ist eine wiederkehrende Operation, die geänderte Datenseiten aus dem Hauptspeicher auf die Festplatte überträgt. Es gibt noch weitere Ereignisse, die die Übertragung von Transaktionsprotokolleinträgen und geänderter Seiten auf die Festplatte auslösen. Prüfpunkte sind jedoch der Mechanismus, der hauptsächlich für den Abgleich zwischen im Hauptspeicher durchgeführten Änderungen und dem physikalisch auf der Festplatte vorhandenen Abbild Ihrer Datenbank sorgt. Prüfpunkte sind an sich *protokollierte Transaktionen* und werden somit ebenfalls im Transaktionsprotokoll festgehalten und auf der Festplatte gespeichert.

Alternative Sicherungsstrategien

Einige Administratoren glauben, dass eine weitere Möglichkeit, eine Sicherungskopie des SQL Servers anzulegen, darin besteht, den Server anzuhalten und alle zum SQL Server gehörenden Dateien (insbesondere die im Verzeichnis `MSSQL\DATA`) zu sichern. Dies ist jedoch bei weitem nicht so zuverlässig, wie es sich anhört. Insbesondere erlaubt es Ihnen diese Vorgehensweise nicht, einzelne Datenbanken wiederherzustellen, und die Möglichkeit, Datenbanken in einem Zustand wiederherzustellen, den sie zu einem bestimmten Zeitpunkt hatten. Deshalb wird dringend empfohlen, die Sicherungsfunktionen von SQL Server zu verwenden, statt die Dateien direkt zu sichern. Außerdem ist die »normale« Windows-Sicherung nicht in der Lage, die Gerätedateien für SQL Server zu sichern, während dieser läuft, da diese Dateien während des Betriebs des Servers geöffnet sind.

7.3 Terminologie



SQL Server 2000 verwendet verschiedene Spezialbegriffe in der Dokumentation und der Benutzeroberfläche, die sich auf Sicherungen beziehen. Ein *Sicherungssatz* ist die Ausgabe eines einzelnen Sicherungsvorgangs. Das ist die Sicherungskopie Ihrer Datenbank die sich auf einem Sicherungsmedium befindet (Ihrem Sicherungsmedium – mit dem wir uns im Verlauf des heutigen Tages noch beschäftigen werden). Wenn Sie eine Sicherung durchführen, erstellen Sie gleichzeitig einen *Mediensatz*. Ein Mediensatz enthält ein oder mehrere Sicherungsmedien, die für die Erstellung eines oder mehrerer Sicherungssätze verwendet wurden. Das ist vor allem praktisch, wenn Sie die Fähig-

keit von SQL Server 2000 in Betracht ziehen, Sicherungen auf mehrere Geräte gleichzeitig durchzuführen. Man spricht hier von *paralleler Sicherung und Wiederherstellung*.



Der nächste Begriff, den Sie kennen sollten, ist die *Medienfamilie*. Die Medienfamilie enthält alle für ein bestimmtes Sicherungsgerät benötigten Bänder. Falls Sie eine Sicherung durchführen, die fünf Bänder in Anspruch nimmt, bilden diese fünf Bänder zusammen die Medienfamilie für diese Sicherung. In diesem Falle ist die Medienfamilie prinzipiell dasselbe wie der Sicherungssatz. Falls Sie jedoch eine parallel auf mehrere Geräte verteilte Sicherung durchführen, wird jeder Satz von Bändern der einzelnen beteiligten Geräte als Medienfamilie bezeichnet.



Verwirrend? Vielleicht, aber diese Begriffe finden Sie überall in der von Microsoft zur Verfügung gestellten Dokumentation über Sicherung und Wiederherstellung. Die gute Nachricht ist, dass Sie viele Sicherungs- und Wiederherstellungsvorgänge durchführen können, ohne jemals diese Begriffe zu verwenden. Dennoch begegnen sie Ihnen von Zeit zu Zeit, so dass Sie wissen müssen, wovon die Rede ist.

7.4 Überlegungen zur Datensicherung

Es gibt viele Dinge, die Sie in Betracht ziehen müssen, wenn Sie Sicherungen durchführen. Wie oft sollten Sie Ihre Datenbank sichern? Welche Datenbanken sollten gesichert werden? Wohin sollten sie gesichert werden? Wie lange müssen die Sicherungskopien aufbewahrt werden? Wer ist für die Sicherungen verantwortlich, und wie stellen Sie sicher, dass die Sicherungskopien verwendbar sind? Wir werden diese Fragen jetzt nacheinander klären. Wir beginnen mit der Verantwortung für die Sicherungskopien.

Wer ist für die Sicherungen verantwortlich?

Ich pflege zu sagen: »Wenn sich niemand für eine Sache verantwortlich fühlt, wird sie nicht erledigt«. Das stimmt besonders dann, wenn es um Sicherungskopien geht. Sicherungen sind, nun ja, langweilig. Sie sind keine glanzvolle Aufgabe. Es bedeutet einen ziemlichen Arbeitsaufwand, den Sicherungsvorgang vernünftig zu planen und die Funktionsfähigkeit des Konzepts regelmäßig zu überprüfen. Die Verantwortung für die Durchführung der Datenbanksicherung kann durchaus beim Datenbank- oder Systemadministrator landen. Es ist genauso gut möglich, dass sich der Netzwerkadministrator dazu bereit erklärt, Ihnen bei der Durchführung der Sicherungen zu helfen. Auf jeden Fall ist es wichtig, dass irgend jemand für die Sicherungen verantwortlich ist und dass sie durchgeführt werden.

Wie stellen Sie fest, ob Ihre Datenbanksicherung brauchbar ist?

Es ist durchaus möglich, dass Sie eine Datenbank sichern können, obwohl Sie bereits beschädigt ist. Es kann sogar sein, dass Sie für die Beschädigung nicht einmal eine Fehlermeldung erhalten. Es gibt einige Überprüfungen, die Sie vor Beginn der Sicherung durchführen sollten (wir kommen später noch darauf zurück) und auch SQL Server selbst führt interne Prüfungen durch, um Beschädigungen zu entdecken (und eventuell sogar zu beheben). Aber falls Sie die Konsistenzprüfungen nicht durchführen oder die Ergebnisse nicht überprüfen, können Sie nicht wissen, ob Ihre Sicherungskopie brauchbar ist, bevor Sie versuchen, sie wiederherzustellen. Es empfiehlt sich also, die Sicherungskopien regelmäßig dadurch zu überprüfen, dass Sie sie auf einem Testserver wiederherstellen und sich davon überzeugen, dass alles erwartungsgemäß funktioniert. Danach wissen Sie, dass die Sicherungskopie brauchbar ist, und Sie haben zudem den Wiederherstellungsprozess überprüft. SQL Server 2000 hat die Fähigkeit, die Integrität einer Sicherungskopie zu überprüfen, nachdem sie erstellt wurde. Aber, wer hat je davon gehört, dass ein Datenbankadministrator entlassen wurde, weil er seine Sicherungs- und Wiederherstellungsstrategien zu gründlich oder zu vorsichtig geplant hat?

Welche Datenbanken sollten gesichert werden?

Es mag zwar selbstverständlich erscheinen, aber Sie sollten alle benutzerdefinierten Datenbanken sichern. Wenn es wichtig genug war, die Datenbank zu erstellen und auf dem Server einzurichten, dann sollte es auch wichtig genug sein, sie zu sichern. Was ist mit den Systemdatenbanken? Sie sollten auf jeden Fall die Datenbanken *master* und *msdb* sichern. Eine Sicherungskopie der *model*-Datenbank kann vielleicht hilfreich sein, *tempdb* muss nicht gesichert werden. Falls Sie mit Replikation arbeiten, sollten Sie auch die *distribution*-Datenbank regelmäßig sichern. Es empfiehlt sich, diese Datenbank wie jede andere regelmäßig geänderte benutzerdefinierte Datenbank zu behandeln.

Welche Medien sollten Sie für Ihre Datenbanksicherung verwenden?

Wie bereits erwähnt, können Sie Ihre Datenbanken auf Festplatte, Band, einem Netzwerklaufwerk oder einer Named Pipe sichern. Bänder sind das bevorzugte Sicherungsmedium; sie haben jedoch auch wesentliche Einschränkungen, was nicht zuletzt die Geschwindigkeit angeht. Sicherungen auf Festplatten sind normalerweise die schnellste Variante. Eine weitere Alternative ist es, die Sicherungskopie auf der Festplatte eines anderen Servers im Netz abzulegen. Sie sollten Ihre auf Festplatten erstellten Sicherungskopien dann auf Bändern sichern, gleichgültig ob sie lokal oder auf einem anderen Server vorliegen. Ein typisches Szenario ist z.B. die Durchführung der Datenbanksicherung über eine

schnelle Netzwerkanbindung (100 MB Ethernet, ATM o.ä.), um dann mit dem Sicherungsprogramm von Windows 2000 die erstellten Sicherungsdateien zu sichern. Sie können jedoch auch eine Backup-Lösung eines Drittherstellers (die eventuell Named Pipes unterstützt) kaufen. Die meisten führenden Hersteller von Backup-Programmen liefern ein Add-In zur Sicherung von SQL Server-Datenbanken mit.

Mit der Erhöhung der Bandgeschwindigkeit werden Bänder zum bevorzugten Sicherungsmedium. Wenn die Datensicherung unter SQL Server zu langsam ist, beruht diese Einschränkung normalerweise auf der Geschwindigkeit der Bandlaufwerke und nicht auf der des Servers.

Wie oft sollten Sie Ihre Datenbank sichern?

Die Antwort auf diese Frage hängt davon ab, wie groß Ihre Datenbank ist und wie schnell eine Wiederherstellung vor sich gehen soll. Datenbanken mit einer Größe von wenigen Gigabyte sollten täglich einmal vollständig gesichert werden; zusätzlich sollten im Laufe des Tages Sicherungen des Transaktionsprotokolls erfolgen. Als Mindestanforderung sollte die Datenbank einmal pro Woche gesichert werden, begleitet von täglichen Sicherungen des Transaktionsprotokolls. Für die Systemdatenbanken gelten besondere Überlegungen, die Sie bald kennen lernen werden.

Wenn Ihre Datenbank sehr viel größer wird, kann es sein, dass Sie nicht mehr genug Zeit haben, jeden Tag eine vollständige Sicherung durchzuführen. Diese Einschränkung mag jedoch wieder verschwinden, wenn schnellere Bandlaufwerke zur Verfügung stehen (oder Sie parallele Sicherungen auf mehreren Laufwerken durchführen). Auch hier gilt, dass die SQL Server-2000-Software normalerweise nicht für die Geschwindigkeitsbeschränkung der Sicherungsvorgänge verantwortlich ist.

Wie lange sollten Sie die Sicherungskopien aufbewahren?

Es kann sein, dass einige Sicherungskopien nicht mehr gebraucht werden, sobald die nächste Sicherung vorliegt. Trotzdem schadet es nichts, mehrere Sicherungskopien aufzubewahren, falls sich zum Beispiel die aktuelle Sicherungskopie als unbrauchbar erweist. Häufig werden die Sicherungskopien der letzten beiden Wochen als Sicherheitsreserve aufbewahrt. Es kann auch sein, dass es gesetzliche Bestimmungen über die Aufbewahrungsfrist für bestimmte Daten gibt, steuerlich relevante Daten müssen zum Teil über Jahre hinweg aufbewahrt werden. Sie sollten sich über diese Anforderungen informieren, bevor Sie die Aufbewahrungsstrategie für Ihre Sicherungskopien festlegen. Sie wären wahrscheinlich überhaupt nicht begeistert davon, deswegen mit dem Gesetz in Konflikt zu geraten, weil Sie eine Sicherungskopie nicht lange genug aufbewahrt haben.



Wie lange dauert die Wiederherstellung einer Sicherungskopie?

Es gibt wahrscheinlich nichts Schlimmeres als den Absturz eines Servers, davon abgesehen, dass Ihr Chef auf Sie zukommt und Sie fragt, »Wie lange dauert es, bis der Server wieder läuft?« und Sie nicht die geringste Ahnung haben. Deshalb wird Ihnen die regelmäßige Durchführung der probeweisen Wiederherstellung Ihrer Sicherungskopie zusätzlichen Ärger in einer ohnehin unangenehmen Situation ersparen und Ihnen den Wiederherstellungsprozess zur Routineaufgabe machen. Sie werden den Wiederherstellungsprozess in der morgigen Lektion näher untersuchen.

Haben Sie einen Katastrophenplan?

Eine der besten Errungenschaften der Mainframe-Ära war das Vorhandensein eines Katastrophenplans und die regelmäßige Durchführung von Ernstfallübungen. Ich habe einmal eine solche Übung mitgemacht. Wir schnappten uns unsere Bänder, stiegen ins Flugzeug und flogen zu unserer Ersatzinstallation, wo wir den Mainframe und unsere Datenbank hochfuhren. Falls Sie Ihre Sicherungsstrategie überprüfen und sicherstellen wollen, dass alles Nötige wirklich gesichert wurde, führen Sie eine solche Übung durch. Es ist erstaunlich, wie viele Dinge man vergessen kann. Es treten interessante Probleme auf. Hat Ihre Ersatzinstallation dieselben Bandlaufwerke? Haben Sie die erforderliche Software in die Sicherungskopien einbezogen?

Hier sind einige der Dinge, die Sie normalerweise brauchen werden:

- Windows NT/2000 auf CD.
- Das neueste Servicepack für Windows NT/2000.
- Eine SQL Server-2000-CD, einschließlich des neuesten Servicepacks.
- Alle anwendungsspezifische Software oder DLLs, die Sie dem SQL Server hinzugefügt haben. Dazu gehören auch die Backup-Programme von Drittherstellern (falls Sie diese verwenden).
- IHRE SICHERUNGSKOPIEN!

Außerdem sollten Sie noch folgende Fragen beantwortet können:

- Auf welchem Laufwerk ist SQL Server installiert?
- Welcher Zeichensatz und welche Sortierreihenfolge werden benutzt?
- Welche Netzwerkbibliotheken sind installiert?
- Welches Netzwerkkonto benutzt der SQL Server?

- Welche Laufwerke und welche Dateinamen werden für welche Datenbank verwendet?
- Welches Mailprofil wurde für die Automatisierung von E-Mails verwendet?

Das sollte zunächst reichen. Falls Sie mehr als einen SQL Server in Betrieb haben, kann das Problem schnell unlösbar werden, wenn Sie keinen Plan erstellt haben. Ein solcher Plan erfordert zwar noch einige gründliche Überlegungen, aber denken Sie daran, dass Katastrophen jederzeit auftreten können. Flut, Feuer, Schneestürme, Tornados, Hurrikans, Erdbeben, Vulkane, Sabotage und selbst hervorgerufene Probleme (z.B. eine schlecht geschriebene Transaktion) sind real existierende Probleme in verschiedenen Teilen der Welt (vielleicht sogar in Ihrer Gegend).

Wo sollten die Sicherungskopien aufbewahrt werden?

Liegen Ihre Sicherungsbänder auf Ihrem Schreibtisch? Auf dem Servergehäuse? Im Bandlaufwerk des Servers? Ich habe das alles schon gehört. Falls ein Erdbeben Ihr Bürogebäude dem Erdboden gleichgemacht hat (und vielleicht die umliegenden Gebäude gleich mit), können Sie dann Ihren Schreibtisch oder gar die Sicherungsbänder noch erreichen? Sie sollten eine externe Lagermöglichkeit für Ihre Sicherungskopien ausfindig machen. Einige Firmen lassen sie abwechselnd von Angestellten mit nach Hause nehmen. Für kleine Datenbanken ist ein Bankschließfach vielleicht die bessere Lösung. Für umfangreichere Sicherungskopien gibt es Aufbewahrungsfirmen, die sich auf die sichere Lagerung von Bändern spezialisiert haben. Die meisten dieser Firmen liefern Bänder und holen sie ab, im Notfall werden die benötigten Bänder sofort (gegen eine geringe Gebühr) zu Ihnen gebracht. Sie sollten sich die Lager ansehen und überprüfen, ob der Anbieter die nötigen Vorkehrungen (und Sicherheitsmaßnahmen) getroffen hat, um das wertvollste Gut Ihrer Firma – die Daten – zu schützen!

Einige Überlegungen zu Bändern

Wenn Sie sich dazu entschieden haben, Ihre Sicherungskopien auf Bändern zu erstellen, sollten Sie Folgendes beachten:

- SQL Server beschreibt Bänder mit ANSI-Bandbezeichnungen. Das bedeutet, dass SQL Server Bänder mit Windows NT/2000-Sicherungskopien gemeinsam nutzen kann.
- SQL Server benutzt das Dienstprogramm `console`, um dem Operator mitzuteilen, wann ein Band zu wechseln ist. (Das Dienstprogramm muss vor dem Sicherungsvorgang gestartet werden.)



- Wie Sie sicherlich erwartet haben, können Sie mehrere Sicherungskopien auf einem Band unterbringen.
- Falls Sie Gebrauch von irgendwelchen speziellen Leistungsmerkmalen eines bestimmten Bandlaufwerks machen, z.B. Hardware-Kompression, sollten Sie sicherstellen, dass dieses Modell auch dann zur Verfügung steht, wenn Sie Ihre Daten wiederherstellen wollen. Dies gilt insbesondere, falls Sie vorhaben, die Bänder über einen langen Zeitraum aufzubewahren. Sind Sie sicher, dass Sie sich dieses Bandlaufwerk auch noch in fünf Jahren besorgen können?

7.5 Benutzerdatenbanken sichern

Wie eben festgestellt, sollten benutzerdefinierte Datenbanken regelmäßig gesichert werden. Als Mindestmaß sollten hier eine vollständige Datenbanksicherung pro Woche sowie tägliche Sicherungen des Transaktionsprotokolls gelten. Besser wäre es jedoch, täglich eine vollständige Sicherung mit periodischer Sicherung des Transaktionsprotokolls vorzunehmen. Denken Sie daran, dass eine vollständige Datenbanksicherung keine Einträge aus dem Transaktionsprotokoll entfernt. Auch eine differenzielle Sicherung belässt das Transaktionsprotokoll unverändert.

Es könnten einige Gründe gegen eine regelmäßige Datenbanksicherung sprechen:

- Falls die Daten nur gelesen werden, reicht wahrscheinlich eine einzelne (überprüfte) Sicherung aus.
- Falls es sich um eine Testdatenbank handelt und die darin enthaltenen Daten unwichtig sind oder schnell neu erstellt werden können.
- Falls die Daten aus einer anderen Quelle schnell wiederhergestellt werden können.

Sie sollten sich jedoch darauf konzentrieren, wann es sinnvoll ist, Ihre Datenbank zu sichern. Falls es sich um eine große Datenbank (mit mehr als einigen Gigabyte) handelt, ist es vielleicht sinnvoll, sie direkt nach der Erstellung zu sichern. Es ist durchaus möglich, dass sie sich schneller wiederherstellen lässt, indem Sie sie mit der in Lektion 4 erwähnten Option `FOR LOAD` neu erstellen, und dann aus der Sicherungskopie wiederherstellen.

Datenbank-Wiederherstellungsmodelle



SQL Server 2000 führt *Wiederherstellungsmodelle* ein, mit denen Sie konfigurieren können, wie groß der Umfang der Transaktionsprotokollierung für eine Datenbank sein soll. Die Festlegung eines Wiederherstellungsmodells für Ihre

Datenbank ändert die Arten der zulässigen Sicherungen und Wiederherstellungen entscheidend. Allerdings müssen Sie sich damit beschäftigen, damit Sie wissen, welche Art von Sicherung auszuführen ist.

Der Grund für die unterschiedlichen Wiederherstellungsmodelle liegt darin, dass man gegebenenfalls verschiedene Stufen der Verfügbarkeit für die Wiederherstellung einer Datenbank benötigt. Möglicherweise bestehen auch unterschiedliche Anforderungen an die Anmeldung in Bezug auf Massenoperationen (in früheren Versionen auch als *nichtprotokollierte Operationen* bezeichnet). Operationen wie das Laden großer Datenmengen bedeuten, dass Sie Ihre Sicherung nur vor oder nach, nicht aber während der Ladeoperation erstellen müssen. Diese Art der Konfiguration wird voll über das massenprotokollierte Wiederherstellungsmodell unterstützt.

Eine weitere gebräuchliche Konfiguration ist es, die Wiederherstellung nur zu einer vollständigen Sicherung durchzuführen. Wenn sich Ihre Datenbank zum Beispiel nur einmal pro Tag ändert und in der übrigen Zeit schreibgeschützt ist, bietet sich das einfache Wiederherstellungsmodell an.

Vollständige Wiederherstellung

Das Modell der vollständigen Wiederherstellung werden Sie wahrscheinlich für die meisten Datenbanken ausführen. Es erlaubt alle Datenbanksicherungen (vollständig, Transaktionsprotokoll, differenziell, Datei-/Dateigruppe). Die Wiederherstellung können Sie zu einem bestimmten Zeitpunkt oder zu einer »markierten« Transaktion (mehr dazu später in der heutigen Lektion) vornehmen. Grundsätzlich gilt, dass Sie für einen 100%igen Schutz Ihrer Daten (und Ihres Jobs) die Datenbank für das Modell der vollständigen Wiederherstellung einrichten und in diesem Status belassen sollten. Es ist die Standardeinstellung für jede neue Datenbank, die Sie mit den SQL Server 2000-Editionen Standard und Enterprise erstellen.

Massenprotokollierte Wiederherstellung

Wenn Sie Auswahlbefehle wie SELECT INTO, BULK INSERT, CREATE INDEX, WRITETEXT und UPDATETEXT ausführen, schreiben Sie gewöhnlich große Datenmengen. Im Modell der vollständigen Wiederherstellung wird jede Zeile, die Sie SQL Server hinzugefügt haben, vollständig im Transaktionsprotokoll wiederhergestellt. Es liegt auf der Hand, dass die Daten hier praktisch zweimal geschrieben werden. Zwar bietet dieses Verfahren ein Maximum an Wiederherstellbarkeit, je nach den Anforderungen kann es aber auch nicht angemessen sein.

Im massenprotokollierten Modus werden die von den genannten Befehlen veränderten Daten nicht vollständig protokolliert, sondern nur Abbilder der modifizierten Seiten, nach-

dem die Seite geladen ist. Aufgrund der zur Protokollierung Ihrer Daten in der Datenbank verwendeten Methode geht die Möglichkeit der Wiederherstellung bis zu einem bestimmten Zeitpunkt verloren. Sie können nur bis zum Ende einer Datenbanksicherung oder bis zum Ende einer Transaktionsprotokollsicherung wiederherstellen.



Ein weiteres Problem bei massenprotokollierter Wiederherstellung ist die Frage, was beim Verlust einer Datendatei passiert. Wenn Sie auf eine vollständige Wiederherstellung eingerichtet sind, sichern Sie einfach das Transaktionsprotokoll und führen dann eine Datenbankwiederherstellung aus (mehr dazu in der morgigen Lektion). Allerdings erfordern Sicherungen des Transaktionsprotokolls, die im massenprotokollierten Modus durchgeführt wurden, den Zugriff auf die Datenbank. Wenn Ihre Datenbankdateien beschädigt sind, können Sie das Transaktionsprotokoll nicht sichern und verlieren gegebenenfalls Ihre Daten. Diese Situation stellt für Sie vielleicht kein Problem dar (weil Sie bereits eine Sicherung erstellt haben, bevor Sie mit dem Laden der Daten in die Datenbank begonnen haben), zweifellos müssen Sie die potenziellen Probleme aber kennen.

Einfache Wiederherstellung

Das Modell der einfachen Wiederherstellung weist die meisten Einschränkungen auf. Prinzipiell können Sie dieses Modell verwenden, wenn Sie lediglich zur letzten vollständigen oder eventuell einer differenziellen Sicherung zurückkehren wollen. Transaktionsprotokollsicherungen sind verfügbar und Ihr Transaktionsprotokoll wird regelmäßig gelöscht.



In SQL Server 7.0 und früheren Versionen haben Sie die Datenbankoptionen *Truncate Log on Checkpoint* (Protokoll bei Prüfpunkt abschneiden) und *Select Into/Bulkcopy* (Auswahl / Massenkopieren) gesetzt, um in den Modus zu gelangen, der die einfache Wiederherstellung verkörpert.

Wenn Ihre Datenbank so ausgelegt ist, dass Sie im Fehlerfall lediglich bis zu einem einzigen Zeitpunkt wiederherstellen müssen (das kann die letzte vollständige oder differenzielle Sicherung sein), bietet sich die einfache Protokollierung an. Dabei werden möglichst wenig Informationen im Transaktionsprotokoll aufgezeichnet, um den Umfang des Protokolls klein zu halten. Trotzdem finden einige Transaktionsaufzeichnungen statt, damit SQL Server selbst intern konsistent bleiben kann und sich auch Objekte unterstützen lassen, die das Transaktionsprotokoll benötigen (zum Beispiel Trigger).



Verzichten Sie grundsätzlich auf das Modell der einfachen Wiederherstellung bei Datenbanken, für die Sie Abfragen mit Datenänderungen ausführen wollen und dennoch die Wiederherstellung der Datenbank erwarten. Dieses Modell eignet sich für Testdatenbanken, statische Datenbanken, die sich nicht ändern,

oder Datenbanken, in denen sich problemlos alle ausgeführten Arbeiten wiederholen lassen. Für Produktionsdatenbanken, bei denen die Benutzer interaktiv Daten hinzufügen, ändern oder löschen, sollten Sie diesen Modus generell nicht verwenden.

Das derzeit festgelegte Wiederherstellungsmodell einer Datenbank lässt sich mit der folgenden Anweisung anzeigen:



```
select DATABASEPROPERTYEX('mydb', 'recovery')
```

Als Ergebnis erhalten Sie einen der Werte `FULL` (vollständig), `BULK_LOGGED` (massenprotokolliert) oder `SIMPLE` (einfach) zurück.



Bei den Datenbanken *pubs* und *Northwind* ist der Wiederherstellungsmodus auf `SIMPLE` eingestellt, bei der Datenbank *model* dagegen auf `FULL`. Wie Sie wissen, wird beim Erstellen einer Datenbank zuerst die Datenbank *model* kopiert. Deshalb erben Sie alle Einstellungen der Datenbank *model*, wenn Sie eine neue Datenbank erstellen.

Mit der Anweisung `ALTER DATABASE` lässt sich das Wiederherstellungsmodell ändern. Zum Beispiel ändern Sie mit der folgenden Anweisung den Wiederherstellungsmodus der Datenbank *pubs* in `FULL`:



```
alter database pubs set recovery full
```

7.6 Systemdatenbanken sichern

Als Nächstes beschäftigen Sie sich mit Systemdatenbanken und wie oft diese gesichert werden sollten. Die Datenbanken *master*, *model*, *msdb* und *tempdb* sind Systemdatenbanken. Die Sicherungsstrategie für diese Datenbanken ist anders als die für benutzerdefinierte Datenbanken.



Die master-Datenbank

Die *master*-Datenbank ist eine spezielle Systemdatenbank. Falls die *master*-Datenbank aus irgendeinem Grund verloren gehen sollte, funktioniert SQL Server 2000 nicht mehr. Es ist deshalb entscheidend, wie und wann Sie diese Datenbank sichern.



Anders als in den Versionen 6.x und früher von SQL Server ist die *master*-Datenbank in einem eigenen Satz von Dateien untergebracht, wobei das Transaktionsprotokoll eine separate Datei ist. Falls Sie eine Aktualisierung von SQL Server 6.5 oder früher vorgenommen haben, sollten Sie diese Änderungen beachten.

Microsoft empfiehlt, die *master*-Datenbank nach jeder an ihr vorgenommenen Änderung zu sichern. Praktisch ist das aber in den meisten Fällen nicht möglich. Denken Sie daran, was sich alles in der *master*-Datenbank befindet: Datenbanken, Benutzernamen, benutzerdefinierte und Systemfehlermeldungen etc. Zum Beispiel gehören zu den Benutzernamen auch Kennwörter. Es ist nicht immer möglich, die *master*-Datenbank jedes Mal zu sichern, wenn jemand sein Kennwort ändert. Es empfiehlt sich, die *master*-Datenbank täglich zu sichern. Das bedeutet, dass alle Kennwörter, die nach der letzten wiederhergestellten Sicherung geändert wurden, nicht richtig gesetzt werden. Das gilt jedoch nur für Benutzernamen mit SQL Server-Authentifizierung.

Es ist sehr wichtig, Ihre *master*-Datenbank nach jeder wesentlichen Änderung an Ihrem Server zu sichern. Wesentliche Änderungen sind z.B.: Konfigurationsänderungen, Erstellen, Ändern oder Löschen von Datenbanken, Hinzufügen von Remoteservern oder Aktivieren der Replikationsunterstützung.



Standardmäßig befindet sich die Datenbank *master* im Modus der einfachen Wiederherstellung. Sie sollten diese Option auch so belassen, da Sie für die *master*-Datenbank nur eine vollständige Datenbanksicherung durchführen können. Differenzielle, Datei- oder Dateigruppen-Sicherungen und Sicherungen des Transaktionsprotokolls sind für diese Datenbank nicht verfügbar.

Die msdb-Datenbank

Die *msdb*-Datenbank enthält Informationen für den SQL Server-Agent-Dienst und für Replikationsoperationen. Falls Sie den SQL Server-Agent-Dienst nutzen wollen (was ich annehme), sollten Sie diese Datenbank regelmäßig sichern.

Als Faustregel sollten Sie die *msdb*-Datenbank mindestens einmal pro Woche sichern, das Transaktionsprotokoll täglich. Wie bei den benutzerdefinierten Datenbanken kann es auch hier nötig sein, die Sicherungen häufiger auszuführen. Vielleicht wollen Sie die Sicherun-

gen der *msdb*-Datenbank auch mit den Sicherungen der *master*-Datenbank synchronisieren. Falls Sie die *master*-Datenbank wiederherstellen müssen (normalerweise nach einem Festplattenfehler), wird die *msdb*-Datenbank neu initialisiert und muss deshalb ebenfalls von einer Sicherungskopie wiederhergestellt werden.



Es ist nicht ganz klar, warum Microsoft die Datenbank *msdb* im einfachen Wiederherstellungsmodus ausliefert. Am besten ändern Sie den Modus in FULL oder führen – abweichend von den obigen Empfehlungen – täglich vollständige Sicherungen der Datenbank *msdb* durch.

Die model-Datenbank

Diese Datenbank ändert sich wahrscheinlich recht selten. Nachdem Sie an Ihre Bedürfnisse angepasste Tabellen, Ansichten, Sicherheitseinstellungen und gespeicherte Prozeduren einmal zur *model*-Datenbank hinzugefügt haben, bleiben sie dort normalerweise längere Zeit ohne große Änderungen. Da aber Paranoia für einen Administrator eine durchaus nützliche Eigenschaft sein kann, ist es gut, eine Sicherungskopie zu besitzen. Wenn Sie Ihre *master*-Datenbank wiederherstellen müssen, wird die *model*-Datenbank ebenfalls zurückgesetzt.

Die tempdb-Datenbank

Die *tempdb*-Datenbank wird bei jedem Start des MSSQLServer-Dienstes neu erstellt. Deshalb muss diese Datenbank nicht gesichert werden. Sie sollten sie im Wiederherstellungsmodus SIMPLE belassen, damit das Transaktionsprotokoll für diese Datenbank nicht unnötig wächst.

Die distribution-Datenbank

Wenn Sie Replikation einsetzen, werden in dieser Datenbank replizierte Transaktionen gespeichert. Einzelheiten zur Replikation erfahren Sie in Lektion 17. Die *distribution*-Datenbank sollten Sie genauso behandeln wie jede wichtige benutzerdefinierte Datenbank; tägliche Datenbanksicherung und gelegentliche Sicherungen des Transaktionsprotokolls sind empfehlenswert. Falls Zeit und Arbeitsaufwand es erlauben, sollten Sie häufigere Sicherungen vornehmen.



7.7 Vorbereitung der Datenbanksicherung

Nachdem Sie sich durch eine Menge Theorien und Ideen zur Datenbanksicherung gearbeitet haben, können Sie sich jetzt auf die physikalische Durchführung der Sicherungen konzentrieren. Sie werden die vollständige Datenbanksicherung, die differenzielle Datenbanksicherung sowie die Sicherung von Dateien und Dateigruppen kennen lernen. Weiterhin beschäftigen Sie sich mit der Sicherung von Transaktionsprotokollen. Schließlich beenden Sie diesen Abschnitt mit der Betrachtung einer realen Serverkonfiguration und der Erstellung eines Sicherungsplans für diesen Server.

Unterwegs werden noch einige weitere interessante Themen gestreift, die Sie kennen lernen sollten. Sie müssen Ihre Sicherungskopien irgendwo speichern, und Sie müssen die Brauchbarkeit Ihrer Sicherungen und Ihrer Datenbank sicherstellen. Hört sich das an wie ein Spaziergang? Gut! Fangen wir mit der Erstellung eines Sicherungsmediums an.

Erstellung eines Sicherungsmediums

Der erste Schritt zur Vorbereitung Ihrer Datenbanksicherung besteht darin, ein Sicherungsmedium zu erstellen. Es ist zwar technisch nicht notwendig, um eine Sicherung durchzuführen, erleichtert jedoch die Durchführung der Datenbanksicherung, da die Syntax der Backup-Anweisung vereinfacht wird, zudem hilft es dem geplagten Administrator, der das System zu betreuen hat, wenn Sie damit fertig sind.

Ein Sicherungsmedium ist eigentlich nur ein Verweis im SQL Server-Systemkatalog (die Systemtabelle *sysdevices* in der *master*-Datenbank), der den logischen Namen und den physikalischen Pfad zu einer Datei auf der lokalen Festplatte, einem lokalen Bandlaufwerk oder einer Datei auf einem anderen Computer enthält. Sie können Sicherungskopien nur auf Bandlaufwerken anlegen, die an Ihren lokalen Rechner angeschlossen sind. Wenn Sie die BACKUP-Anweisung ausführen, können Sie diesen logischen Namen anstelle des vollständigen Pfads und Dateinamens angeben. Das erspart eine Menge Tipparbeit und ist weniger fehleranfällig.

Nach der erstmaligen Installation von SQL Server 2000 sind noch keine Sicherungsmedien für Ihr System definiert. Einer Ihrer ersten Schritte wird es sein, diese Einträge zu erstellen. Falls Sie Ihre Sicherungen auf ein Bandlaufwerk durchführen wollen, erstellen Sie normalerweise nur ein Sicherungsmedium, das auf Ihr Bandlaufwerk verweist. Sie können jedoch trotzdem mehrere »logische« Geräte erstellen, die dann alle auf Ihr Bandlaufwerk verweisen. Wenn der Server über mehrere Bandlaufwerke verfügt, sollten Sie natürlich ein Gerät pro Bandlaufwerk erstellen.

Bei Sicherungen auf Festplatte (oder Netzwerklaufwerke) erstellen Sie normalerweise für jede Datenbank ein eigenes Sicherungsmedium. Sinnvollerweise beginnen Sie mit der Definition der Sicherungsmedien für die auf Ihrem Server vorhandenen Systemdatenbanken.

Sicherungsmedien mit Transact-SQL steuern

Sie können Sicherungsmedien mit der gespeicherten Systemprozedur `sp_addumpdevice` erstellen.



```
sp_addumpdevice [@devtype=] 'device_type',  
                [@logicalname =] 'logical_name',  
                [@physicalname =] 'physical_name'  
                [, ([@cntrltype =] cntrltype |  
                  [@devstatus =] 'device_status'))]
```

Die Parameter haben folgende Bedeutung:

- *device_type* ist der Gerätetyp, auf den sich das Sicherungsmedium beziehen soll. Folgende Angaben sind möglich:
 - ▶ *disk* – Ein Verweis auf eine Datei auf der lokalen Festplatte
 - ▶ *tape* – Ein Verweis auf ein lokales Bandlaufwerk
 - ▶ *pipe* – Ein Verweis auf eine Named Pipe
- *logical_name* ist der Name, den Sie in Ihren Sicherungs- und Wiederherstellungsanweisungen verwenden wollen, um auf das angegebene physikalische Sicherungsgerät zu verweisen.
- *physical_name* bezeichnet den physikalischen Ort des Sicherungsmediums. Für Festplatten ist dies der Pfad und der Dateiname. Für ein Bandlaufwerk ist dies ein Verweis auf Ihr Bandlaufwerk. Schließlich kann hier noch die Netzwerkadresse einer Named Pipe angegeben werden.
- *cntrltype* ist ein veralteter Parameter, der nur aus Gründen der Abwärtskompatibilität vorhanden ist.
- *device_status* bestimmt darüber, ob ANSI-Bandbezeichnungen beim Beschreiben des Sicherungsmediums ignoriert werden sollen. Falls Sie als Gerätetyp *tape* angeben, wird von SQL Server als Voreinstellung *noskip* verwendet. Das bedeutet, falls auf einem Band Kopfinformationen über eine frühere Sicherung vorhanden sind, ignoriert sie SQL Server nicht automatisch und initialisiert das Band nicht neu. Falls Sie dieses Verhalten wünschen (die automatische Initialisierung des Bandes, selbst wenn es Daten enthält), können Sie hier *skip* angeben.



Für den Parameter *physical_name* sind noch einige Details zu klären. Wenn Sie eine lokale Datei angeben, ist ein vollständiger Pfad zu spezifizieren. Also zum Beispiel:

```
F:\Programme\Microsoft SQL Server\mssql\backup\master_backup.bak
```

Eine Datei im Netzwerk muss mit ihrem UNC-Pfad angegeben werden, beispielsweise:

```
\\remoteserver\backups\sql\master_backup.bak.
```

Schließlich spezifiziert man Bandlaufwerke wie folgt:

```
\\.\tape#
```

Das erste Bandlaufwerk in Ihrem System hat dabei die Nummer 0. Falls Sie also nur ein Bandlaufwerk in Ihrem System haben, geben Sie es folgendermaßen an:

```
\\.\tape0
```

Die Anweisungen im folgenden Beispiel erstellen ein Sicherungsmediums, das für die Sicherung der *master*-Datenbank verwendet werden kann. Diese Anweisung fügt ein neues Festplattensicherungsmedium mit dem Namen *master_backup* am richtigen physikalischen Standort hinzu:

```
exec sp_addumpdevice 'disk', 'master_backup',  
    'F:\Programme\Microsoft SQL Server\mssql\backup\master_backup.dat'
```



Der Begriff *dump device* ist eine Hinterlassenschaft früherer Versionen von SQL Server. Er findet aus Gründen der Abwärtskompatibilität zu SQL Server 6.5 weiter Verwendung. Das Gleiche gilt für die gespeicherten Prozeduren, wie *sp_dropdevice*, die Sie später in dieser Lektion kennen lernen.

Um ein Sicherungsmedium zu entfernen, führen Sie die gespeicherte Systemprozedur *sp_dropdevice* aus:



```
sp_dropdevice [@logicalname =] 'logical_name', [@delfile =] 'delfile'
```

Die Parameter haben folgende Bedeutung:

- *logical_name* ist der Name, den Sie in Ihren Sicherungs- und Wiederherstellungsanweisungen verwenden wollen, um auf das angegebene physikalische Sicherungsgerät zu verweisen.
- *delfile* gibt optional an, dass die Sicherungsdatei (falls sie sich auf einer Festplatte befindet) physikalisch gelöscht werden soll.

Wollen Sie also das Sicherungsmedium *pubs_backup* entfernen, führen Sie folgende Anweisung aus:



```
exec sp_dropdevice 'pubs_backup'
```

Falls Sie (unter der Annahme, dass Sie die *pubs*-Datenbank auf Festplatte sichern) gleichzeitig auch die Sicherungsdatei loswerden wollen, führen Sie folgende Anweisung aus:



```
exec sp_dropdevice 'pubs_backup', 'delfile'
```

Die Anweisung sollte etwa folgende Ausgabe zurückliefern:



Die physische Datei 'F:\Programme\Microsoft SQL Server\mssql\backup\pubs_backup.bak' wurde erfolgreich gelöscht.
Medium gelöscht.

Um eine Liste aller Medien in Ihrem System zu erhalten, führen Sie die gespeicherte Systemprozedur *sp_helpdevice* aus:



```
sp_helpdevice [[@devname =] 'logical_name']
```

Das Argument *logical_name* ist der Name, den Sie in Ihren Sicherungs- und Wiederherstellungsanweisungen verwenden, um auf diesen physikalischen Sicherungsstandort zu verweisen. Wenn Sie zum Beispiel die Anweisung



```
exec sp_helpdevice
```

ausführen, erhalten Sie etwa folgende Ausgabe (die hier umgebrochen und gekürzt wurde, damit sie auf die Seite passt):

device_name	physical_name	
master	F:\Programme\...\MSSQL\DATA\MASTER.MDF	...
master_backup	F:\Programme\...\MSSQL\BACKUP\master_backup.BAK...	
mastlog	F:\Programme\...\MSSQL\DATA\MASTLOG.LDF	...
modeldev	F:\Programme\...\MSSQL\DATA\MODEL.MDF	...
modellog	F:\Programme\...\MSSQL\DATA\MODELLOG.LDF	...



```
tempdev      F:\Programme\...\MSSQL\DATA\TEMPDB.MDF    ...
templog      F:\Programme\...\MSSQL\DATA\TEMPLOG.LDF    ...
```

description	status	cntrltype	size
special, physical disk, 4 MB	2	0	512
disk, backup device	16	2	0
special, physical disk, 0.8 MB	2	0	96
special, physical disk, 0.6 MB	2	0	80
special, physical disk, 0.8 MB	2	0	96
special, physical disk, 2 MB	2	0	256
special, physical disk, 0.5 MB	2	0	64

Beachten Sie, dass hier bei einigen Beschreibungen `special, physical disk` angegeben ist. Bei einem Medium steht jedoch `disk, backup device`. Das ist Ihr Sicherungsmedium. Um Ihre Sicherungsmedien zu finden, müssen Sie die Ausgabe von `sp_helpdevice` durchsuchen.

Sicherungsmedien mit dem SQL Server Enterprise Manager steuern

Der SQL Server Enterprise Manager kann ebenfalls dazu verwendet werden, Sicherungsmedien zu erstellen oder zu entfernen. Öffnen Sie den Ordner *Verwaltung* für Ihren Server und markieren Sie den Ordner *Sicherung*. Klicken Sie mit der rechten Maustaste auf den Ordner (oder an einer freien Stelle im Detailbereich des SQL Server Enterprise Managers) und wählen Sie **NEUES SICHERUNGSMEDIUM** aus dem Kontextmenü. Sie sehen daraufhin ein Dialogfeld wie in Abbildung 7.5. Beachten Sie, dass nach der Installation von SQL Server 2000 zunächst keine Sicherungsmedien definiert sind. (Das Medium *master_backup*, das Sie mithilfe der oben angegebenen Transact-SQL-Anweisung erstellt haben, ist jedoch vorhanden.)

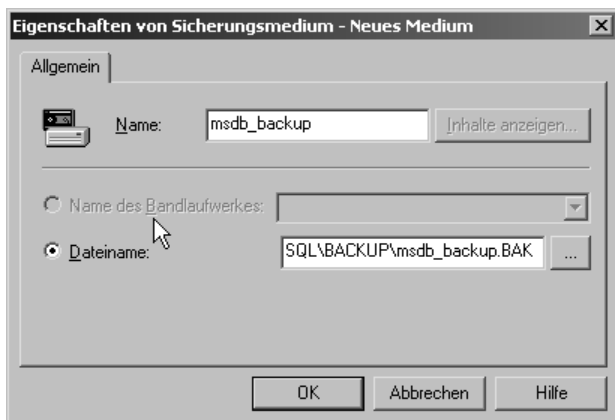


Abbildung 7.5:
Das Dialogfeld mit den Eigenschaften des Sicherungsmediums

Wie Sie es vorhin schon getan haben, können Sie auch hier ein Sicherungsmedium definieren, das Ihre Datenbanksicherungen aufnehmen soll, zum Beispiel ein Sicherungsmedium mit dem Namen *msdb_backup*, das die *msdb*-Datenbank aufnehmen soll. Beachten Sie, dass SQL Server Enterprise Manager automatisch den Dateinamen einfügt, wenn Sie den Namen des Sicherungsmediums eingeben, und dabei als Vorgabe den Namen des Sicherungsmediums übernimmt. Es empfiehlt sich in jedem Fall, die gleichen logischen und physikalischen Namen für ein Sicherungsmedium zu verwenden.

Nachdem Sie das Dialogfeld ausgefüllt haben, klicken Sie auf OK um das Erstellen des Mediums abzuschließen. Wenn Sie ein Medium auf einem Netzwerklaufwerk anlegen wollen, geben Sie den UNC-Pfad zum gewünschten Server in der Eingabezeile für den Dateinamen an. Verwenden Sie nicht die mit dem Netzlaufwerk verbundenen Laufwerkbuchstaben, da es ziemlich schwierig ist, mit diesen zu arbeiten. Erinnern Sie sich daran, dass Sie beim Erstellen eines Sicherungsmediums einfach nur einen Verweis in den Systemtabellen erzeugen. Zum jetzigen Zeitpunkt werden noch keine Daten auf das Medium geschrieben.

Um ein Sicherungsmedium mit dem SQL Server Enterprise Manager zu entfernen, klicken Sie einfach mit der rechten Maustaste auf das gewünschte Medium im Ordner *Sicherung* und wählen dann LÖSCHEN aus dem Kontextmenü. Bestätigen Sie den Löschvorgang, und das Medium wird aus den Systemtabellen von SQL Server entfernt. Falls bereits eine physikalische Datei erstellt worden sein sollte, wird sie durch den Löschvorgang im SQL Server Enterprise Manager nicht entfernt.

Überprüfung der Datenbankkonsistenz

Wenn Sie ein Sicherungsmedium zur Verfügung haben, können Sie schon fast mit der Datenbanksicherung beginnen. Es gibt jedoch noch einen wichtigen Schritt, der nötig sein könnte, bevor Sie mit der Sicherung beginnen. Wie bereits erwähnt, ist eine defekte Datenbanksicherung wertlos. Deshalb sollten Sie sicherstellen, dass Ihre Sicherungskopie brauchbar ist. Microsoft stellt zwei Optionen zur Verfügung, um die Integrität Ihrer Datenbank und Ihrer Sicherungskopie zu überprüfen. Die Anweisung *backup* besitzt eine Option zur Überprüfung der Sicherungskopie nach Durchführung der Sicherung. Außerdem gibt es einen Satz von Anweisungen zur Überprüfung der Datenbankintegrität.

Konzentrieren Sie sich im Moment auf die Möglichkeiten der Prüfung der Datenbankkonsistenz, die Sie innerhalb Ihrer SQL Server-Datenbank haben. Falls Ihre Datenbank vor der Sicherung in Ordnung ist, wird auch die Sicherungskopie brauchbar sein. Um die Konsistenz Ihrer Datenbank zu überprüfen, verwenden Sie Erweiterungen des Dienstprogramms DBCC (DataBase Consistency Checker). Es gibt verschiedene DBCC-Werkzeuge, die unterschiedliche Aspekte der Datenbankintegrität überprüfen. Höchstwahrscheinlich führen Sie vor allem die Anweisung DBCC CHECKDB aus:



```
DBCC CHECKDB ( 'database_name'
    [, NOINDEX |
      { REPAIR_ALLOW_DATA_LOSS | REPAIR_FAST | REPAIR_REBUILD } ]
    )
    [WITH { [ALL_ERRORMSGs | NO_INFOMSGs]
      [, TABLOCK]
      [, ESTIMATEONLY]
      [, PHYSICAL_ONLY]
      [, TABLERESULTS]
    }
  ]
```

Die Parameter haben folgende Bedeutung:

- *database_name* ist der Name der Datenbank, für die die Integritätsprüfung durchgeführt werden soll.
- *Noindex* legt fest, dass Sie nicht wollen, dass Ihre nicht gruppierten Indizes überprüft werden. (Auf Indizes geht Tag 13 ein.)
- *Repair_Allow_Data_Loss* gibt an, dass alle Reparaturmaßnahmen der Option *Repair_Rebuild* durchgeführt werden und repariert zusätzlich beschädigte Seiten sowie Text- und Bildobjektseiten ohne Rücksicht auf eventuellen Datenverlust. Die Datenbank muss sich für diese Option im Einzelbenutzermodus befinden.
- *Repair_Fast* repariert beschädigte Indizes, falls dies sicher möglich ist (ein Datenverlust ist mit dieser Option nicht möglich). Es werden nur Reparaturen ausgeführt, die schnell und einfach sind (etwa die Reparatur von beschädigten Indexschlüsseln). Die Datenbank muss sich für diese Option im Einzelbenutzermodus befinden.
- *Repair_Rebuild* repariert genau wie *Repair_Fast* beschädigte Indizes, führt aber auch zeitraubendere Reparaturmaßnahmen, wie die Wiederherstellung beschädigter Indizes, durch. Die Datenbank muss sich für diese Option im Einzelbenutzermodus befinden.
- *ALL_ERRORMSGs* liefert alle Fehlermeldungen der Anweisung zurück. Standardmäßig gibt SQL Server maximal 200 Fehlermeldungen zurück. Es ist zu hoffen, dass Sie diese Option niemals brauchen werden.
- *NO_INFOMSGs* gibt an, dass nur wichtige Fehlermeldungen zurückgeliefert werden; unterdrückt alle Informationsmeldungen. Diese Option werden Sie in den meisten Fällen verwenden wollen.
- *TABLOCK* gibt an, dass die Anweisung gemeinsame Tabellensperren anstelle der standardmäßigen Seitensperren verwendet. Diese Option beschleunigt den DBCC-Befehl, wenn Ihr System ausgelastet ist, bremst aber konkurrierende Benutzer, die durch den CHECKDB-Befehl blockiert werden.

- *ESTIMATEONLY* zeigt den *tempdb*-Speicherplatz an, der schätzungsweise erforderlich ist, um den Befehl auszuführen. Die Überprüfung an sich wird nicht durchgeführt.
- *PHYSICAL_ONLY* überprüft nur die physikalische Integrität der Datenbank (und nicht die logische Integrität). Diese Option läuft schneller und erkennt die gebräuchlichsten Arten von Beschädigungen in SQL Server (beispielsweise Probleme mit einer Festplatte, einem Festplattencontroller und zerrissenen Seiten).
- *TABLERESULTS* gibt die Ergebnisse des DBCC-Befehls im Format einer SELECT-Anweisung (als Ergebnismenge) zurück. Diese Option ist neu in SQL Server 2000 und erlaubt es, die Ergebnisse dieses Befehls (und aller anderen DBCC-Befehle) in einer Tabelle zu speichern, so dass man nicht mehr über Berichten brüten muss, um nach der eigentlich interessierenden Ausgabe zu suchen.

SQL Server 2000 kann Konsistenzprobleme beheben. Die Reparaturoptionen lernen Sie an Tag 8 kennen. Führen Sie fürs Erste den folgenden Beispielbefehl aus, um die Konsistenz und Integrität Ihrer Datenbank zu überprüfen:



DBCC CHECKDB ('pubs') With NO_INFOMSGS, TABLERESULTS

Als Ausgabe sollte erscheinen:



Befehl(e) erfolgreich abgeschlossen.

Wenn Sie die Option *With NO_INFOMSGS* in der Anweisung weglassen, erhalten Sie eine umfangreiche Ausgabe mit zwei Zeilen für jede Tabelle in der Datenbank (siehe Abbildung 7.6).

Es ist erheblich einfacher, die Ergebnisse der Anweisung zu interpretieren, wenn Sie die Option *NO_INFOMSGS* angeben. Sofern Sie keinen Bericht über die Zeilen und Seiten benötigen, gibt es keinen Grund, die Option *NO_INFOMSGS* wegzulassen.

Falls Sie andersartige Meldungen erhalten, ist Ihre Datenbank höchstwahrscheinlich beschädigt. Die Behandlung der Fehler, die die DBCC CHECK*-Anweisungen zurückliefern, ist im Rahmen dieses Buchs nicht möglich, sie können jedoch weitere Informationen dazu bei Microsoft (<http://support.microsoft.com>) erhalten. Falls Sie die Schäden an Ihrer Datenbank nicht beheben können, müssen Sie sie wahrscheinlich von Ihrer letzten Sicherungskopie wiederherstellen. Darauf geht Tag 8 näher ein.

Error	Level	State	MessageText	RepairLevel	Status	DbId	Id	IndId	File	Page
1	2593	10	1	Es sind 73 Zeilen in 1 Seit...	NULL	0	5	1	1	0
2	2593	10	1	Es sind 51 Zeilen in 3 Seit...	NULL	0	5	2	1	0
3	2593	10	1	Es sind 331 Zeilen in 5 Sei...	NULL	0	5	3	1	0
4	2593	10	1	Es sind 29 Zeilen in 1 Seit...	NULL	0	5	4	1	0
5	2593	10	1	Es sind 113 Zeilen in 5 Sei...	NULL	0	5	6	1	0
6	2593	10	1	Es sind 2 Zeilen in 1 Seite...	NULL	0	5	8	0	0
7	2593	10	1	Es sind 42 Zeilen in 1 Seit...	NULL	0	5	9	1	0
8	2593	10	1	Es sind 22 Zeilen in 1 Seit...	NULL	0	5	10	1	0
9	2593	10	1	Es sind 0 Zeilen in 0 Seite...	NULL	0	5	11	1	0
10	2593	10	1	Es sind 241 Zeilen in 1 Sei...	NULL	0	5	12	1	0
11	2593	10	1	Es sind 10 Zeilen in 1 Seit...	NULL	0	5	14	1	0
12	2593	10	1	Es sind 0 Zeilen in 0 Seite...	NULL	0	5	19	1	0
13	2593	10	1	Es sind 0 Zeilen in 0 Seite...	NULL	0	5	24	1	0
14	2593	10	1	Es sind 1 Zeilen in 1 Seite...	NULL	0	5	96	1	0
15	2593	10	1	Es sind 25 Zeilen in 1 Seit...	NULL	0	5	53575229	1	0
16	2593	10	1	Es sind 6 Zeilen in 1 Seite...	NULL	0	5	117575457	1	0
17	2593	10	1	Es sind 21 Zeilen in 1 Seit...	NULL	0	5	149575571	1	0
18	2593	10	1	Es sind 86 Zeilen in 1 Seit...	NULL	0	5	213575799	0	0
19	2593	10	1	Es sind 3 Zeilen in 1 Seite...	NULL	0	5	245575913	0	0
20	2593	10	1	Es sind 14 Zeilen in 1 Seit...	NULL	0	5	277576027	1	0

Abbildung 7.6:
Der DBCC
CHECKDB-Befehl
mit Ausgabe aller
Meldungen

Die Konsistenzprüfung der DBCC CHECKDB-Anweisung wird in 90% der Fälle mehr als ausreichend sein. Im Grunde genommen sollte es in SQL Server 2000 überhaupt nicht nötig sein, die DBCC-Anweisungen zu verwenden. Da Paranoia jedoch besonders dann hilfreich sein kann, wenn es um die Daten Ihrer Firma geht, ist eine tägliche Überprüfung mit DBCC gar keine schlechte Idee. Sie werden keine Probleme mit der Überprüfungsdauer der Datenbankintegrität bekommen, außer wenn Ihre Datenbank sehr groß (mehrere 100 GB) ist. Nur Mitglieder der Serverrolle *sysadmin* oder der Datenbankrolle *db_owner* können die Anweisung DBCC CHECKDB ausführen.

Andere Konsistenzprüfungen

Die nächsten drei Anweisungen – DBCC CHECKCATALOG, DBCC CHECKALLOC und DBCC CHECKTABLE – sind Untermengen von DBCC CHECKDB und werden ausgeführt, wenn Sie DBCC CHECKDB aufrufen. Deshalb brauchen Sie sie nicht selbst aufzurufen, wenn Sie DBCC CHECKDB verwenden können.

Für diese drei Befehle ist es erforderlich, dass Sie Mitglied der Serverrolle *sysadmin* oder der Datenbankrolle *db_owner* sind. Es gelten die gleichen Warnungen und Einsatzhinweise wie für DBCC CHECKDB.

DBCC CHECKCATALOG

Der Befehl DBCC CHECKCATALOG prüft die referenzielle Integrität der Systemtabellen. Mehr zur referenziellen Integrität lernen Sie am Tag 14.



```
DBCC CHECKCATALOG ( 'database_name' ) [WITH NO_INFOMSGS]
```

Der Parameter *database_name* ist der Name der Datenbank, für die Sie die Integritätsprüfung durchführen möchten.



```
DBCC CHECKCATALOG('pubs')
```



DBCC-Ergebnis für 'pubs'.
DBCC-Ausführung abgeschlossen. Falls DBCC Fehlermeldungen ausgegeben hat, wenden Sie sich an den Systemadministrator.

Beachten Sie, dass Sie nur zwei Ausgabezeilen erhalten. Bei diesem Befehl unterdrückt die Option NO_INFOMSGS nur die erste Zeile:

DBCC-Ergebnis für 'databasename'.

DBCC CHECKALLOC

Der Befehl DBCC CHECKALLOC überprüft, ob die Strukturen der Speicherplatzreservierung für eine bestimmte Datenbank korrekt sind.



```
DBCC CHECKALLOC ( 'database_name'
    [, NOINDEX |
      { REPAIR_ALLOW_DATA_LOSS | REPAIR_FAST | REPAIR_REBUILD } ]
    )
    [WITH { [ALL_ERRORMSG] | NO_INFOMSGS ]
        [, TABLOCK]
        [, ESTIMATEONLY]
        [, TABLERESULTS]
    ]
```

Alle Optionen in dieser Syntax entsprechen dem Befehl DBCC CHECKDB.

DBCC CHECKTABLE

Der Befehl DBCC CHECKTABLE führt die gleichen Operationen wie DBCC CHECKDB aus, ist aber auf eine einzelne Tabelle beschränkt. Das ist vor allem praktisch, wenn Sie eine große Tabelle überprüfen möchten, nicht aber die gesamte Datenbank.



```
DBCC CHECKTABLE ( 'table_name'
    [, NOINDEX | index_id
    | { REPAIR_ALLOW_DATA_LOSS | REPAIR_FAST | REPAIR_REBUILD } ]
    )
    [WITH { [ALL_ERRORMSGS | NO_INFOMSGS]
    [, TABLOCK]
    [, ESTIMATEONLY]
    [, PHYSICAL_ONLY]
    [, TABLERESULTS]
    }
    ]
```

In dieser Syntax entsprechen alle Optionen dem Befehl DBCC CHECKDB, mit Ausnahme der folgenden beiden Optionen:

- *table_name* ist der Name der Tabelle, für die die Integritätsprüfung durchzuführen ist.
- *index_id* gibt an, dass DBCC CHECKTABLE die Integrität nur für diesen Index überprüft. (Am Tag 13 erfahren Sie mehr zu Index-IDs.)

7.8 Sicherung einer Datenbank

Prima – Sie haben es geschafft, bis zur eigentlichen backup-Anweisung vorzudringen. Es gibt eine Menge zu tun, bevor Sie damit beginnen können, Ihre Datenbank zu sichern. Das ist selbst auf kleinen Systemen keine triviale Aufgabe. Nachdem Sie Ihre Sicherungsmedien erstellt und die Datenbankkonsistenz überprüft haben, wird es Zeit, alle Datenbanken zu sichern. Im ersten Schritt entscheiden Sie, welche Sicherungsmedien Sie verwenden wollen, und lernen, wie diese Sicherungsmedien zu verwenden sind.

Verwendung der Sicherungsmedien

Sie können Sicherungen auf ein einzelnes Sicherungsmedium oder auf mehrere Medien verteilt erstellen; Sie können ebenso mehrere Sicherungen auf dasselbe Sicherungsmedium durchführen. Überall, wo Sie in der BACKUP-Anweisung den Namen eines Sicherungsmediums angeben können, kann auch ein Dateiname angegeben werden. Sehen Sie sich die verschiedenen Möglichkeiten an.

Sicherung auf ein einzelnes Sicherungsmedium

Diese Art der Sicherung ist der Normalfall und die üblicherweise empfohlene Art, die Sicherung durchzuführen. Wenn jede Datenbank ihr zugehöriges Sicherungsmedium hat,



können Sie eine 1:1-Sicherungsstrategie anwenden. Bei der Sicherung auf Festplattensicherungsmedien ist dies empfehlenswert. Falls Sie aus irgendeinem Grund eine Datei verlieren, betrifft dies so nur eine einzelne Sicherung einer Datenbank, anderenfalls verlieren Sie mehrere Sicherungen auf einmal. Falls Sie Bänder zur Datensicherung verwenden, ist es schon eher sinnvoll (und billiger), mehrere Datenbanken auf ein Band (Sicherungsmedium) zu sichern.

Parallele verteilte Sicherungen

Eine parallele verteilte Sicherung ermöglicht es Ihnen, eine einzelne Datenbank auf mehrere Sicherungsmedien verteilt zu sichern. Sie geben einfach mehr als ein Sicherungsmedium in der Backup-Anweisung an. SQL Server 2000 startet für jedes Sicherungsmedium, das Sie in Ihrer parallelen verteilten Sicherung verwenden wollen, einen neuen Thread; dabei sind maximal 64 Medien (und somit Threads) möglich.

Für die Sicherung auf Bandlaufwerke bietet die parallele verteilte Sicherung einen großen Vorteil. Sie können mehrere Bandlaufwerke an einen Server anschließen, und eine Datenbank gleichzeitig auf mehrere Bänder sichern. Falls die Sicherung Ihrer Datenbank drei Stunden dauert und Sie zwei weitere Bandlaufwerke anschaffen (womit Sie dann drei Laufwerke zur Verfügung haben), wird die Sicherung Ihrer Datenbank wahrscheinlich nur noch eine Stunde dauern. Vor allem aber brauchen Sie für die Wiederherstellung Ihrer Datenbank nicht die gleiche Anzahl an Bandlaufwerken, die Sie für die Sicherung verwendet haben. Das ist besonders wichtig für den Fall einer Wiederherstellung nach einer Katastrophe. Es gibt keine Garantie dafür, dass der neue Server die gleiche Anzahl von Bandlaufwerken besitzt. Beachten Sie außerdem, dass Sie alle Bänder für die Wiederherstellung benötigen.



Wenn Sie ein Band einmal für eine parallele verteilte Sicherung verwendet haben, können Sie es nur noch für parallele verteilte Sicherungen mit der gleichen Anzahl von Bandlaufwerken verwenden, bis Sie es neu formatieren.

Mehrere Sicherungen auf einem einzelnen Medium

Sie können auch mehrere Sicherungen auf ein einzelnes Sicherungsmedium durchführen. Bei näherer Betrachtung ist das wahrscheinlich die Konfiguration, die Sie für die Sicherung auf Bändern wünschen. Sicherlich wollen Sie so viele Datenbanksicherungen wie möglich auf einem Band unterbringen. Sie können dies auch mit einem Festplattensicherungsmedium tun, was jedoch nicht zu empfehlen ist, weil Sie beim Verlust dieser Datei die Sicherungskopien aller Datenbanken auf diesem Sicherungsmedium verlieren.

Die Anweisung BACKUP DATABASE (für komplette Datenbanken)

Der nächste logische Schritt ist (schließlich) die Sicherung der Datenbank. Hierfür verwenden Sie die Anweisung BACKUP:



```
BACKUP DATABASE {database_name | @database_var}
TO <backup_device> [, ...n]
[WITH [BLOCKSIZE = {blocksize | @blocksize_variable}]
[[,] DESCRIPTION = {text | @text_variable}]
[[,] DIFFERENTIAL]
[[,] EXPIREDATE = {date | @date_var}
| RETAINDAYS = {days | @days_var}]
[[,] PASSWORD = {pwd | @pwd_var}
[[,] FORMAT | NOFORMAT]
[[,] {INIT | NOINIT}]
[[,] MEDIADESCRIPTION = {text | @text_variable}]
[[,] MEDIANAME = {media_name | @media_name_variable}]
[[,] MEDIAPASSWORD = {media_pwd | @media_pwd_variable}]
[[,] [NAME = {backup_set_name | @backup_set_name_var}]
[[,] {NOSKIP | SKIP}]
[[,] {NOUNLOAD | UNLOAD}]
[[,] {NOWIND | REWIND}]
[[,] [RESTART]
[[,] STATS [= percentage]]
]
```

Wobei gilt:

```
<backup_device> ::=
{
[backup_device_name | @backup_device_name_var]
|
{DISK | TAPE | PIPE} =
{'temp_backup_device' | @temp_backup_device_var}
}
```

Hier folgt jetzt eine Erklärung der Unzahl von Parametern. Sie werden bei der Erklärung der weiteren Backup-Anweisungen nicht noch einmal aufgeführt, wenn sie dort nicht Besonderheiten aufweisen. Falls sie Ihnen in den Anweisungen BACKUP LOG und BACKUP FILE/FILEGROUP noch einmal begegnen, müssen Sie ihre Bedeutung hier nachschlagen.

- *database_name* ist der Name der zu sichernden Datenbank.
- *@database_var* ist der Name der zu sichernden Datenbank – nur als Variable ausgedrückt.

- *backup_device* ist entweder der Name eines Sicherungsmediums oder ein variabler Platzhalter für ein Sicherungsmedium. Sie können auch einfach den Namen der Datei, des Bandes oder der Named Pipe angeben, der für die Wiederherstellung verwendet werden soll. Auch hierbei können Sie eine Variable einsetzen.
- *blocksize* gibt die Blockgröße an, die Sie für ein Band oder eine Named Pipe verwenden wollen. Empfohlene Blockgrößen finden Sie in der Dokumentation zu Windows 2000 oder Ihrem Bandlaufwerk.
- *DESCRIPTION* sind bis zu 255 Zeichen, die diese Sicherung näher beschreiben. Auch hier können Sie eine Variable einsetzen.
- *DIFFERENTIAL* verdient einen eigenen Abschnitt zur kompletten Beschreibung, der unten folgt.
- *EXPIREDATE* ist das Datum, ab dem das Band überschrieben werden darf. Falls Sie vor diesem Datum versuchen, das Band zu beschreiben, wird das ohne Angabe spezieller Parameter nicht möglich sein.
- *RETAIN_DAYS* hat den gleichen Effekt wie *EXPIREDATE*, nur geben Sie hier statt eines bestimmten Datums eine Anzahl von Tagen ein.
- *PASSWORD* ist das Kennwort, das der Benutzer eingeben muss, damit die Wiederherstellung von diesem Band möglich ist.



Die Sicherung ist lediglich durch ein Kennwort geschützt und nicht in irgendeiner Form verschlüsselt.

- *FORMAT* | *NOFORMAT* ermöglicht Ihnen, anzufordern, ob das Band neu formatiert werden soll. Auf dem Band vorhandene Daten oder ein Kennwortschutz werden ignoriert. Dieser Parameter hat die gleiche Wirkung wie eine Kombination von *INIT* und *SKIP*.
- *INIT* | *NOINIT* gibt an, ob Ihr Sicherungsmedium vor Beginn der Sicherung neu initialisiert werden soll. Vorgabewert ist *NOINIT*, was bedeutet, dass Ihre Sicherung an das Ende des Bandes oder der Datei angehängt wird, falls darauf bereits Daten einer anderen Sicherung vorhanden sind. Falls sich auf dem Medium noch keine Sicherungskopie befindet, wird es initialisiert, bevor der Schreibvorgang beginnt. Durch die Angabe von *INIT* werden bestehende Inhalte des Bandes oder der Datei gelöscht. Falls für ein Band das Verfallsdatum noch nicht erreicht ist, oder der *MEDIA_NAME* nicht übereinstimmt, wird das Band nicht initialisiert. Falls Sie das trotzdem wollen, müssen Sie zusätzlich *SKIP* angeben (dann können Sie aber auch gleich die Option *FORMAT* verwenden).

- *MEDIADESCRIPTION* ist ein weiteres Kommentarfeld mit einer Länge von 255 Zeichen.
- *MEDIANAME* ist die Beschreibung des Sicherungssatzes – maximal 128 Zeichen lang. Dieser Name wird bei der Überprüfung verwendet, ob ein Band überschrieben werden darf. Falls Sie auf einem Band sowohl Sicherungen von Windows 2000 als auch von SQL Server unterbringen wollen, müssen Sie den Parameter *MEDIANAME* angeben.
- *MEDIAPASSWORD* ist ein Kennwort, das Sie angeben können, wenn Sie Sicherungen als Teil dieses Mediensatzes erstellen. Dieses Kennwort wird abgefragt, wenn eine Sicherung von diesem Mediensatz wiederhergestellt wird.
- *NAME* ist ein weiterer Name für den Sicherungssatz. Auch hier können Sie eine Variable einsetzen.
- *NOSKIP* | *SKIP* gibt an, ob die Bandbezeichnung gelesen werden soll, bevor auf das Band geschrieben wird. Wenn Sie den Standardwert *NOSKIP* verwenden, wird die Bandbezeichnung gelesen und das Ablaufdatum und der Mediennamen überprüft, um ein versehentliches Überschreiben des Bandes zu verhindern.
- *NOUNLOAD* | *UNLOAD* gibt an, ob das Band nach Ende des Sicherungsvorganges aus dem Laufwerk ausgeworfen werden soll.
- *NOREWIND* | *REWIND* gibt an, ob das Band nach Ende des Sicherungsvorganges zurückgespult werden soll.
- *RESTART* gibt Ihnen die Möglichkeit, einen unterbrochenen Sicherungsvorgang neu zu starten. Dabei wird der Sicherungsvorgang genau an der Stelle fortgesetzt, an der unterbrochen wurde. Für den Neustart müssen Sie die gesamte *BACKUP*-Anweisung wiederholen und dabei diesen Parameter angeben. Der Parameter ist nur gültig, wenn sich Ihre Sicherung über mehrere Banddatenträger erstreckt.
- *STATS=percentage* gibt an, wie häufig Sie über den Fortschritt der Sicherung informiert werden wollen. Der Vorgabewert ist 10, was bedeutet, dass SQL Server nach jeweils 10% Fortschritt der Sicherung eine Meldung ausgibt, die Sie über den Stand der Sicherung informiert.

Die *BACKUP*-Anweisung wirkt auf den ersten Blick etwas abschreckend – wie wäre es mit einem Beispiel? Manchmal geht nichts über ein kleines Programm, das wieder Klarheit verschafft. Das folgende Skript erstellt ein neues Sicherungsmedium für die *pubs*-Datenbank, überprüft die Datenbankintegrität und führt dann die Datenbanksicherung durch.



```
exec sp_addumpdevice 'disk', 'pubs_backup',
    'F:\Programme\Microsoft SQL Server\MSSQL\BACKUP\pubs_backup.bak'
go
use pubs
dbcc checkdb ('pubs') with no_infomsgs
backup database pubs to pubs_backup with init
```



```
(1 row(s) affected)

'Disk'-Medium hinzugefügt.
208 Seiten für die pubs-Datenbank, Datei 'pubs' auf Datei 1
verarbeitet.
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 1 verarbeitet.
BACKUP DATABASE hat 209 Seiten in 0,626 Sekunden
erfolgreich verarbeitet (2,725 MB/Sekunde).
```

Es ist also alles halb so schlimm. Die nächste Anweisungsfolge sichert ebenfalls die *pubs*-Datenbank, diesmal jedoch auf das lokale Bandlaufwerk. Das Beispiel können Sie nur ausführen, wenn Sie ein Bandlaufwerk in dem Computer installiert haben, auf dem SQL Server 2000 läuft. Diesmal geben Sie an, dass das Band nach Ablauf von 30 Tagen verfällt, dass es formatiert werden soll und dass die Blockgröße für dieses Band 8192 Byte betragen soll. Der Name der Sicherungskopie lautet *Pubs Backup Tape*.



```
exec sp_addumpdevice 'tape', 'pubs_tape_backup', '\\.\tape0'
go
use pubs
dbcc checkdb ('pubs') With NO_INFOMSGS
BACKUP DATABASE pubs to pubs_backup WITH FORMAT,
Retain_days = 30, MediaName='Pubs Backup Tape',
Blocksize=8192
```

Die Anweisung DIFFERENTIAL BACKUP DATABASE

Wie Sie heute schon erfahren haben, kopiert eine differenzielle Datenbanksicherung alle geänderten Seiten einer Datenbank – d.h. die Seiten, die seit der letzten vollständigen Datenbanksicherung geändert wurden. Wenn nach einer differenziellen Sicherung eine weitere differenzielle Sicherung durchgeführt wird, enthält die zweite Sicherung alles, was bereits in der ersten gesichert wurde, sowie alle Seiten die nach der ersten differenziellen Sicherung noch geändert wurden. Sie sollten beachten, dass eine differenzielle Sicherung nur Sinn macht, wenn ihr eine vollständige Datenbanksicherung vorausgegangen ist.

Erstellen Sie jetzt eine differenzielle Sicherung der *pubs*-Datenbank. Zuerst erstellen Sie eine Tabelle und fügen ihr eine Zeile hinzu (damit etwas Neues zum Sichern vorhanden ist). Sie erstellen kein neues Sicherungsmedium, sondern hängen die differenzielle Siche-



zung an das Medium an, das schon die erste Datenbanksicherung enthält. Sie können (wie im Beispiel) zur Verdeutlichung den Parameter NOINIT angeben, obwohl das sowieso der Vorgabewert ist.



```
use pubs
create table backuptest (coll int not null)
insert backuptest values (1)
go
BACKUP DATABASE pubs to pubs_backup WITH differential, NOINIT
```

Die Anweisungen sollten etwa folgende Ausgabe zurückliefern:



```
(1 row(s) affected)

48 Seiten für die pubs-Datenbank, Datei 'pubs' auf Datei 2 verarbeitet.
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 2 verarbeitet.
BACKUP DATABASE WITH DIFFERENTIAL hat 49 Seiten in 0,284 Sekunden
erfolgreich verarbeitet (1,388 MB/Sekunde).
```

Sie werden noch eine Menge Informationen brauchen, um mit differenziellen Sicherungen richtig umgehen zu können. Das verschieben wir jedoch am besten auf Tag 8.



Die RESTART-Option ist für die Sicherung großer Datenbanken besonders interessant. Falls die Sicherung aus irgendeinem Grund scheitert, müssen Sie nicht ganz von vorn beginnen. Sie können dort fortsetzen, wo die Sicherung abgebrochen wurde. Beachten Sie, dass die Sicherung unabhängig von der Angabe dieses Parameters ganz von vorn beginnt, wenn Sie nur ein einzelnes Band verwenden oder sich noch auf dem ersten Band befanden, als die Sicherung unterbrochen wurde.

Einstellung der Aufbewahrungsdauer für Bandsicherungen

Falls Sie bei einer Sicherung auf ein Band weder die Option EXPIRE_DATE noch RETAIN_DAYS angeben, wird das Ablaufdatum für dieses Band auf den Wert gesetzt, der in der Serverkonfiguration für die Option `media retention` eingetragen ist. Sie können sich diesen Wert wie jede andere Serverkonfigurationsoption mithilfe der gespeicherten Systemprozedur `sp_configure` ansehen. Um den Vorgabewert für die Aufbewahrungsdauer der Bänder zum Beispiel auf 30 Tage zu setzen, können Sie folgende Anweisungen ausführen:



```
exec sp_configure 'media retention',30
RECONFIGURE WITH OVERRIDE
```



Sie müssen SQL Server 2000 anhalten und neu starten, damit diese Änderung in Kraft tritt. Der Vorgabewert für die Aufbewahrungsdauer beträgt 0 Tage (d.h., es gibt keine vorgeschriebene Aufbewahrungsdauer).

Die Anweisung BACKUP DATABASE (für Dateien und Dateigruppen)

Der Vollständigkeit halber geben wir hier noch die Syntax für die Sicherung von Dateien und Dateigruppen an. Sie sollten Dateien und Dateigruppen aber nur dann einzeln sichern, wenn sich eine vollständige Datenbanksicherung (oder eine Kombination von vollständiger und differenzieller Datenbanksicherung) nicht in einer annehmbaren Zeit durchführen lässt. Es gibt noch weitere legitime Gründe für die Sicherung von Dateien und Dateigruppen, aber diese fortgeschrittene Thematik geht über den Rahmen dieses Buches hinaus.

Im Wesentlichen erlaubt es Ihnen die Anweisung BACKUP DATABASE eine einzelne Datei oder eine als Dateigruppe bezeichnete Sammlung von Dateien (die Sie am Tag 4 kennen gelernt haben) zu sichern. Sie brauchen auf jeden Fall noch eine Sicherung des Transaktionsprotokolls, wenn Sie Sicherungskopien von Dateien oder Dateigruppen zur Wiederherstellung Ihrer Datenbank verwenden wollen.



```
BACKUP DATABASE {database_name | @database_name_var}
<file_or_filegroup> [, ...m]
TO <backup_device> [, ...n]
[WITH
[BLOCKSIZE = {blocksize | @blocksize_variable}]
[[,] DESCRIPTION = {text | @text_variable}]
[[,] EXPIREDATE = {date | @date_var}
| RETAINDAYS = {days | @days_var}]
[[,] PASSOWRD = {pwd | @pwd_var}
[[,] FORMAT | NOFORMAT]
[[,] {INIT | NOINIT}]
[[,] MEDIADESCRIPTION = {text | @text_variable}]
[[,] MEDIANAME = {media_name | @media_name_variable}]
[[,] MEDIAPASSWORD = {media_pwd | @media_pwd_variable}]
[[,] {NAME = {backup_set_name | @backup_set_name_var}}
[[,] {NOSKIP | SKIP}]
[[,] {NOUNLOAD | UNLOAD}]
[[,] {NOREWIND | REWIND}]
[[,] [RESTART]
[[, ] STATS [= percentage]]
]
```



Wobei gilt:

```
<file_or_filegroup> :: =  
{  
FILE = {logical_file_name | @logical_file_name_var}  
|  
FILEGROUP = {logical_filegroup_name | @logical_filegroup_name_var}  
}
```

Die meisten der angegebenen Parameter wurden bis auf folgende Ausnahmen bereits erklärt:

- *FILE* gibt den Namen einer einzelnen Datei (oder einer Variablen, die einen einzelnen Pfad und Dateinamen enthält) an, die gesichert werden soll.
- *FILEGROUP* gibt den logischen Namen der Dateigruppe an, die Sie sichern wollen. Alle Dateien dieser Dateigruppe werden gesichert.

Die Anweisung BACKUP LOG

Mit dem Befehl BACKUP LOG können Sie auch nur das Transaktionsprotokoll einer Datenbank sichern:



```
BACKUP LOG {database_name | @database_name_var}  
[ [WITH { NO_LOG | TRUNCATE_ONLY }] ]  
|  
[TO <backup_device> [, ...n]  
[WITH  
[BLOCKSIZE = {blocksize | @blocksize_variable}]  
[[,] DESCRIPTION = {text | @text_variable}]  
[[,] EXPIREDATE = {date | @date_var}  
| RETAINDAYS = {days | @days_var}]  
[[,] PASSOWRD = {pwd | @pwd_var}  
[[,] FORMAT | NOFORMAT]  
[[,] {INIT | NOINIT}]  
[[,] MEDIADESCRIPTION = {text | @text_variable}]  
[[,] MEDIANAME = {media_name | @media_name_variable}]  
[[,] MEDIAPASSWORD = {media_pwd | @media_pwd_variable}]  
[[,] [NAME = {backup_set_name | @backup_set_name_var}]  
[[,] NO_TRUNCATE]  
[[,] {NOSKIP | SKIP}]  
[[,] {NOUNLOAD | UNLOAD}]  
[[,] {NOREWIND | REWIND}]  
[[,] [RESTART]  
[[,] STATS [= percentage]]  
]  
]
```

Die meisten der angegebenen Parameter wurden bis auf folgende Ausnahmen bereits erklärt:

- `TRUNCATE_ONLY` und `NO_LOG` sind identisch und entfernen abgeschlossene Transaktionen aus dem Protokoll, ohne es wirklich zu sichern. Das Transaktionsprotokoll ist zum Beispiel nach der Sicherung einer Datenbank, in der nicht protokollierte Aktionen durchgeführt wurden, nicht mehr verwendbar. In diesem Fall würden Sie die Anweisung `BACKUP LOG WITH TRUNCATE_ONLY` gefolgt von einer vollständigen Datenbanksicherung ausführen. Die Leerung des Transaktionsprotokolls ist selbst eine protokollierte Aktion. Sie sollten außerdem beachten, dass diese Anweisung keine wirkliche Sicherung durchführt – deshalb ist die Angabe des Namens eines Sicherungsmediums hier unnötig. Natürlich können Sie auch das Wiederherstellungsmodell für Ihre Datenbank auf `SIMPLE` einstellen.
- Die Option `NO_TRUNCATE` bewirkt eine Sicherung des Transaktionsprotokolls (genau wie die Anweisung `BACKUP LOG` ohne Angabe weiterer Optionen). Wenn diese Option angegeben wird, können Sie jedoch im Gegensatz zur normalen Sicherung das Transaktionsprotokoll sichern, obwohl die zugehörige Datenbank nicht verfügbar ist. Nehmen wir an, dass Sie eine Datenbank haben, bei der sich die Datendatei auf der einen Festplatte befindet, das Transaktionsprotokoll jedoch auf einer anderen. Falls Sie die Datendatei aus irgendeinem Grund verlieren, können Sie die Anweisung `BACKUP LOG` mit der Option `NO_TRUNCATE` ausführen, um alle Transaktionen, die seit der letzten Sicherung des Transaktionsprotokolls hinzugekommen sind, zu sichern. Das erlaubt Ihnen, die Datenbank in den Zustand wiederherzustellen, den sie direkt vor dem Festplattenschaden hatte.

Sie können das Transaktionsprotokoll einer Datenbank nur sichern, wenn Sie vorher einmal eine vollständige Datenbanksicherung durchgeführt haben. Das Transaktionsprotokoll lässt sich nicht sichern, wenn eine der Datenbankoptionen *Truncate Log On Checkpoint* oder *Select Into/Bulkcopy* eingestellt ist.

Sicherungen von Transaktionsprotokollen sind eine Serie von Sicherungen. Anders als bei den differenziellen Sicherungen gibt es zwischen den Sicherungen von Transaktionsprotokollen keine Überschneidungen. Wenn Sie Transaktionsprotokolle wiederherstellen müssen, brauchen Sie jede Sicherung, die Sie seit der letzten vollständigen Datenbanksicherung durchgeführt haben. Die morgige Lektion geht näher auf die Wiederherstellung einer Datenbank unter Verwendung der Sicherungen von Transaktionsprotokollen ein.

Wie oft sollten Sie ihr Transaktionsprotokoll sichern? Ein typisches Szenario könnte (für eine Datenbank, die hauptsächlich während der Geschäftszeiten verwendet wird) so aussehen:



6:00	tägliche vollständige Datenbanksicherung mit FORMAT
6:05	Sicherung des Transaktionsprotokolls mit FORMAT
10:00	erneute Sicherung des Transaktionsprotokolls (NOINIT)
12:00	differenzielle Datenbanksicherung (NOINIT)
14:00	erneute Sicherung des Transaktionsprotokolls (NOINIT)
18:00	erneute Sicherung des Transaktionsprotokolls (NOINIT)
20:00	differenzielle Datenbanksicherung (NOINIT)
22:00	letzte Sicherung des Transaktionsprotokolls

Für die *pubs*-Datenbank würde das Skript wie in Listing 7.1 aussehen. Es setzt natürlich voraus, dass sich Ihre Datenbank im Wiederherstellungsmodus FULL befindet.



Die Datenbank *pubs* ist standardmäßig im Wiederherstellungsmodus SIMPLE. In diesem Modus lässt sich das Transaktionsprotokoll nicht sichern. Wenn Sie also den Code gemäß Listing 7.1 ausführen wollen, müssen Sie die Datenbank mit dem Befehl

```
alter database pubs SET RECOVERY FULL
```

in den Wiederherstellungsmodus FULL setzen.

Listing 7.1: Erstellen eines Zeitplans

```
--Skript startet um 6:00
use pubs
go
dbcc checkdb ('pubs') With NO_INFOMSGS, TABLERESULTS
go
BACKUP DATABASE pubs to pubs_backup WITH FORMAT,
Retain_days = 30, MediaName='Pubs Backup Tape',
Blocksize=8192
Go

-- 6:05
BACKUP LOG pubs to pubs_log_backup WITH INIT
Go
-- 10:00
BACKUP LOG pubs to pubs_log_backup
Go
-- 12:00
BACKUP DATABASE pubs to pubs_backup WITH DIFFERENTIAL,
NOINIT, NOSKIP, Retain_days = 30, MediaName='Pubs Backup Tape',
```




```
Blocksize=8192
Go

-- 14:00
BACKUP LOG pubs to pubs_log_backup
Go

-- 18:00
BACKUP LOG pubs to pubs_log_backup
Go

-- 20:00
BACKUP DATABASE pubs to pubs_backup WITH DIFFERENTIAL,
NOINIT, NOSKIP, Retain_days = 30, MediaName='Pubs Backup Tape',
Blocksize=8192
Go

-- 22:00
BACKUP LOG pubs to pubs_log_backup
Go
```

Mit diesem Skript können Sie Ihre Datenbank wiederherstellen, ohne Transaktionen zu verlieren (solange Sie Ihre Sicherungsmedien nicht verlieren). Das Beispiel geht von der Verwendung zweier Bandlaufwerke aus – eines für die vollständigen und differenziellen Datenbanksicherungen und eines für die Sicherungen der Transaktionsprotokolle.

Sicherungen mit Enterprise Manager

Erwartungsgemäß bietet auch der SQL Server Enterprise Manager alle für Sicherungskopien notwendigen Funktionen.

Zunächst erstellen Sie zwei weitere Sicherungsmedien, eines für die *pubs*-Datenbank (nennen Sie es *pubs_backup*) und ein weiteres für die *Northwind*-Datenbank (namens *northwind_backup*). Setzen Sie beide Datenbanken in den Wiederherstellungsmodus FULL.

Jetzt erweitern Sie in der Konsolenstruktur den Ordner *Datenbanken* und klicken entweder mit der rechten Maustaste auf die Datenbank, die Sie sichern wollen, und wählen ALLE TASKS / DATENBANK SICHERN. Daraufhin erscheint das in Abbildung 7.7 dargestellte Dialogfeld SQL SERVER-SICHERUNG – PUBS.

Nachdem Sie die Syntax der Transact-SQL-Anweisung backup kennen gelernt haben, sollte dieses Dialogfeld für Sie ein Kinderspiel darstellen. Geben Sie (falls gewünscht) eine Beschreibung für diese Sicherung ein und wählen Sie die gewünschte Sicherungsart. Denken Sie daran: Um das Transaktionsprotokoll zu sichern, müssen Sie zuerst eine vollstän-

dige Datenbanksicherung durchführen. Außerdem dürfen die Datenbankoptionen *Select Into/Bulkcopy* und *Truncate Log On Checkpoint* nicht aktiviert sein und die Datenbank darf sich auch nicht im Wiederherstellungsmodus SIMPLE befinden.



Abbildung 7.7:
Das Dialogfeld SQL Server-
Sicherung – pubs

Klicken Sie unter ZIEL auf HINZUFÜGEN, um das Dialogfeld für die Auswahl eines Sicherungsziels zu öffnen (siehe Abbildung 7.8). Hier können Sie einen Dateinamen, ein Bandlaufwerk, ein Netzwerklaufwerk, eine Named Pipe oder (die einfachste Alternative) ein Sicherungsmedium auswählen. Verwenden Sie die Option *Sicherungsmedium* und wählen Sie (wie in Abbildung 7.8) das Sicherungsmedium *pubs_backup* aus der Liste aus.

Klicken Sie auf OK, um die Auswahl des Sicherungsmediums zu beenden, und fahren Sie im Dialogfeld mit dem Abschnitt ÜBERSCHREIBEN fort. In der Standardeinstellung wird die Sicherungskopie an eventuell bereits auf dem Sicherungsmedium oder in der Sicherungsdatei vorhandene Sicherungen angehängt. Falls Sie wollen, dass der Inhalt des Sicherungsmediums vor der Speicherung der aktuellen Sicherung gelöscht wird, wählen Sie die Option *Vorhandene Medien überschreiben*.



Abbildung 7.8:
Das Dialogfeld Sicherungsziel auswählen

Der Abschnitt ZEITPLAN bietet Ihnen eine Schnittstelle zur Auftragsplanung des SQL Servers, die vom SQL Server-Agent-Dienst bereitgestellt wird. Markieren Sie dieses Kontrollkästchen, wenn Sie wollen, dass der SQL Server die Sicherungen als geplante Tasks zu von Ihnen gewünschten Zeiten durchführt. Normalerweise gehen Sie dabei wie folgt vor: Sie stellen in diesem Dialogfeld die Optionen für Ihre Sicherung wie gewünscht ein und markieren dann das Kontrollkästchen neben ZEITPLAN, um einen Zeitplan zu erstellen. Da die Planung von Sicherungen nur ein (zugegebenermaßen wichtiger) Teil der Auftragsplanungsmöglichkeiten von SQL Server ist, werden diese Möglichkeiten im Rahmen der Themen von Tag 18 detailliert behandelt.

Gehen Sie jetzt auf die Registerkarte OPTIONEN (siehe Abbildung 7.9). Beachten Sie, dass die Option *Sicherung bei Abschluss überprüfen* aktiviert ist. Dadurch wird Ihre Sicherungskopie auf Lesbarkeit und Fehler überprüft.



Die Option *Sicherung bei Abschluss überprüfen* ist nicht das Gleiche wie DBCC CHECKDB. Hier wird nicht die Datenbankintegrität geprüft, sondern nur, ob das Sicherungsmedium selbst nicht beschädigt ist.

Die grau dargestellten Optionen sind für die von Ihnen ausgewählte Sicherungsart – in diesem Fall eine vollständige Datenbanksicherung auf Festplatte, nicht verfügbar. Bei einer Transaktionsprotokollsicherung steht die Option *Inaktive Einträge aus dem Transaktionsprotokoll entfernen* zur Verfügung. Dies ist die Vorgabe für Sicherungen des Transaktionsprotokolls. Falls Sie ein Verhalten wünschen, wie es durch die Option NO_TRUNCATE der Backup-Anweisung erreicht wird, deaktivieren Sie dieses Kontrollkästchen.

Die restlichen Optionen auf dieser Registerkarte beziehen sich auf Bandlaufwerke und sind selbsterklärend. Um den Sicherungsvorgang zu beginnen, klicken Sie auf OK. Es erscheint ein Meldungsfeld wie in Abbildung 7.10, das Sie über den Fortschritt der Sicherung informiert. Falls Sie die Überprüfungsoption ausgewählt haben, erscheint auch noch das in Abbildung 7.11 gezeigte Meldungsfeld, das Sie über den Erfolg der Überprüfung informiert.

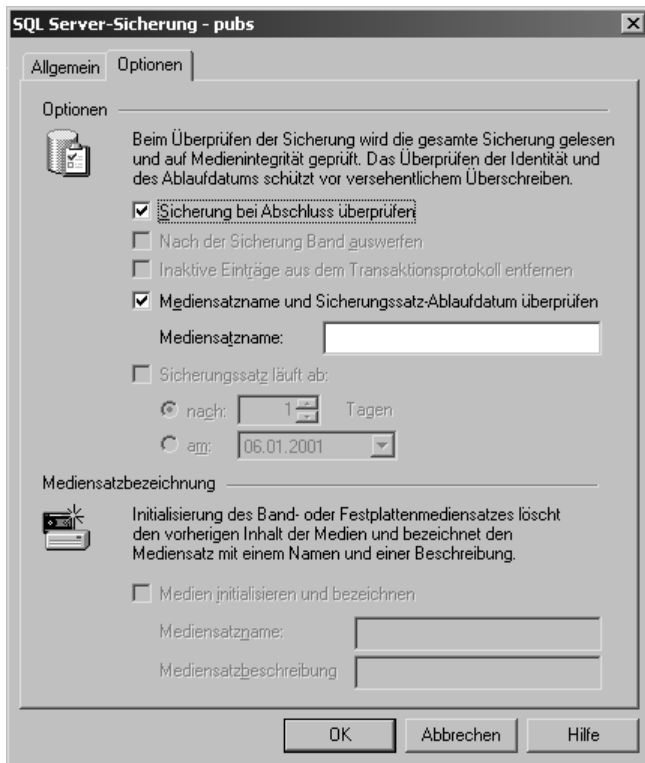


Abbildung 7.9:
Die Registerkarte Optionen des Dialogfelds SQL Server-Sicherung – pubs

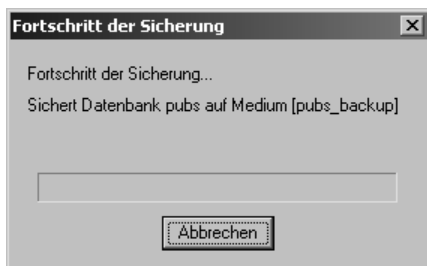


Abbildung 7.10:
Die Fortschrittsanzeige beim Erzeugen einer Sicherungskopie

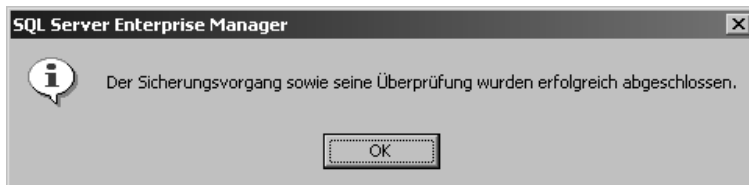


Abbildung 7.11:
SQL Server informiert Sie über den Erfolg der Überprüfung

Jetzt folgt noch das beste Argument für die Verwendung von Sicherungsmedien anstelle der Angabe von Dateien für die Sicherung. Erweitern Sie in der Konsolenstruktur den Ordner *Verwaltung*, klicken Sie auf *Sicherung* und doppelklicken Sie auf das Sicherungsmedium *pubs_backup* im Detailbereich. Klicken Sie auf INHALTE ANZEIGEN und Sie erhalten eine Liste aller auf diesem Sicherungsmedium enthaltenen Sicherungskopien – inklusive der Sicherungsart und dem Zeitpunkt der Sicherung (siehe Abbildung 7.12). Es ist recht hilfreich, diese Informationen bei der Hand zu haben. Wie Sie sehen, ist in Abbildung 7.12 sowohl eine vollständige Datenbanksicherung, als auch eine Sicherung des Transaktionsprotokolls auf dem Sicherungsmedium untergebracht.

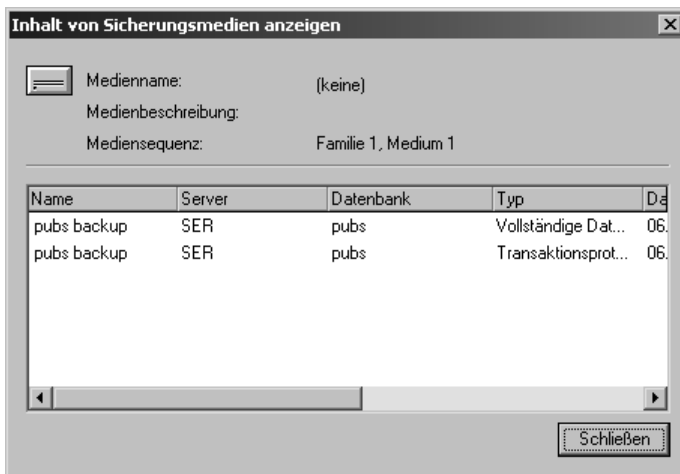


Abbildung 7.12:
Das Dialogfeld Inhalt von
Sicherungsmedien anzeigen

Das war's! Die Sicherung von Datenbanken und Transaktionsprotokollen mit dem SQL Server Enterprise Manager ist so einfach. Natürlich war es mit ein wenig Arbeit verbunden, um alle Abläufe und Optionen zu verstehen, aber sicherlich ist es den Aufwand wert gewesen.

7.9 Zusammenfassung

Sichern und Wiederherstellen sind wahrscheinlich die wichtigsten Aspekte der Wartung Ihres SQL Server-Systems. Die Sicherung einer Datenbank verlangt eine sorgfältige Planung. Sie müssen entscheiden, welche Sicherungsarten Sie benötigen (vollständige Datenbanksicherungen, differenzielle Datenbanksicherungen, Sicherungen der Transaktionsprotokolle oder Sicherungen von Dateien/Dateigruppen) und es gibt dafür keine allgemeingültige Antwort. Die Entscheidung hängt von der Größe Ihrer Datenbank und Ihren Ansprüchen an den Wiederherstellungsvorgang ab. Nur nach sorgfältiger Planung



und der Umsetzung der entstandenen Pläne können Sie sicher sein, dass Sie die richtigen Sicherungskopien haben. Sie sollten sich sowohl auf schwerwiegende Katastrophen, als auch auf das »versehentliche« Löschen wichtiger Daten, das üblicherweise kurz vor einem Feiertag auftritt, vorbereiten. Schützen Sie Ihren Server durch eine RAID-Konfiguration, dann müssen Sie Ihre Sicherungskopien weniger häufig einsetzen. Sie werden Ihre Sicherungen wahrscheinlich lieber zeitgesteuert durchführen lassen wollen, als sie jedes Mal interaktiv durchzuführen. Die Zeitplanung für Sicherungen (und andere Aktivitäten) erläutert Tag 18.

7.10 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Kann ich eine Datenbank dynamisch in eine neue Datei sichern?*
- A** Ja. Geben Sie einfach einen Dateinamen anstelle des Namens eines Sicherungsmediums an.
- F** *Welche Anweisung verwende ich zur Erstellung eines neuen Sicherungsmediums?*
- A** `sp_addumpdevice` (oder den SQL Server Enterprise Manager).
- F** *Schützt die RAID-Stufe 0 meine Daten im Fall eines Festplattenverlusts?*
- A** Nein. RAID 0 bedeutet lediglich die Verteilung der Daten auf mehrere Festplatten und stellt keine Fehlertoleranz zur Verfügung.
- F** *Wann ist es nicht möglich, eine Sicherung des Transaktionsprotokolls zu erstellen?*
- A** Wenn entweder die Datenbankoptionen `Select Into/Bulkcopy` oder `Truncate Log On Checkpoint` aktiv sind oder noch keine vollständige Datenbanksicherung durchgeführt worden ist.
- F** *Kann ich das Transaktionsprotokoll der master-Datenbank sichern?*
- A** Nein. Sie können für die master-Datenbank ausschließlich eine vollständige Datenbanksicherung durchführen.

Kontrollfragen

1. Welche RAID-Stufe ist für die Aufnahme von Transaktionsprotokollen vorzuziehen?
2. Können Sie eine Sicherungskopie auf ein Sicherungsmedium erstellen, dessen physikalischer Speicherort `\\myserver\sqlbackups\master_backup.dat` ist?
3. Die Option *Sicherung / Transaktionsprotokoll* ist im Dialogfeld SQL SERVER-SICHERUNG nicht verfügbar. Können Sie erklären, warum das der Fall sein könnte?
4. Wo sollten Sie Ihre Sicherungsbänder aufbewahren?
5. Hat ein Prüfpunkt Einfluss auf die Sicherung Ihres Transaktionsprotokolls?

Übungen

1. Erstellen Sie für jede Ihrer Datenbanken ein Sicherungsmedium und führen Sie für jede Datenbank eine Sicherung durch. Überprüfen Sie die eingestellten Datenbankoptionen, um festzustellen, für welche Datenbanken Sicherungen des Transaktionsprotokolls durchgeführt werden können.
2. Führen Sie eine differenzielle Datenbanksicherung für die *pubs*- und *Northwind*-Datenbank durch.
3. Sichern Sie das Transaktionsprotokoll der *pubs*-Datenbank.

Tag 1	Relationale Datenbanken und Microsoft SQL Server 2000	31
Tag 2	SQL Server 2000 installieren	45
Tag 3	SQL Server 2000 Tools und Utilities	95
Tag 4	Datenbanken und Datendateien erstellen und verwalten	151
Tag 5	Sicherheit	187
Tag 6	Berechtigungen	233
Tag 7	Sicherungen von Datenbanken	275

WOCHE

1

Tag 8	Wiederherstellen von Datenbanken	333
Tag 9	Datentypen und Tabellen	367
Tag 10	Daten mit SELECT abrufen	397
Tag 11	Daten bearbeiten	459
Tag 12	Transact-SQL-Erweiterungen	481
Tag 13	Indizes	523
Tag 14	Datenintegrität	573

WOCHE

2

Tag 15	Sichten, Trigger, gespeicherte Prozeduren und Funktionen	627
Tag 16	Replikation – Entwurf	665
Tag 17	Replikation – Implementierung	701
Tag 18	Aufträge mit dem SQL Server-Agent planen	759
Tag 19	Daten zwischen Servern migrieren	805
Tag 20	Leistungsüberwachung	829
Tag 21	XML und SQL Server 2000	865

WOCHE

3

Woche 2 – Vorschau

In Ihrer zweiten Woche werden Sie lernen, wie Sie Datenbanken wiederherstellen, Tabellen erstellen, Informationen im- und exportieren, Daten abfragen und verändern, Tabellen indizieren und wie Sie die Integrität Ihrer Daten sicherstellen.

Tag 8 wird Ihren Ausflug in die Welt des besorgten Datenbankadministrators vervollständigen. Sie werden lernen, wie man Datenbanken rettet, und Sie werden die Datenbank wiederherstellen, die Sie am siebten Tag gesichert haben. Obwohl der Vorgang nicht schwierig ist, müssen Sie unbedingt wissen, wann Sie Ihre Daten sichern sollten und wie Sie sie notfalls wiederherstellen können.

Am neunten Tag werden Sie lernen, welche Datentypen SQL Server 2000 Ihnen zur Verfügung stellt. Sie werden weiterhin lernen, wie Sie Ihre eigenen Datentypen definieren können. Nachdem Ihnen jetzt die Datentypen geläufig sind, können Sie diese zu Tabellen zusammensetzen.

Tag 10 behandelt die Grundlagen der Datenabfrage. Sie lernen die Syntax und Anwendung der SELECT-Anweisung kennen.

Auf dem hier vermittelten Wissen baut die zweite Hälfte von Tag 11 auf. Sie lernen, wie man leistungsfähigere Anweisungen verwendet, um genau die gewünschten Daten zusammenzutragen, Sie arbeiten mit Unterabfragen, Aggregatfunktionen und vielen Systemfunktionen. Diese Lektion ist recht lang, stellt aber einen ausgezeichneten Überblick über große Teile der Sprache Transact-SQL dar. Wenn Sie Tag 11 geschafft haben, sind Sie in der Lage, bestehende Daten in Ihrer Datenbank zu ändern; dies gilt sowohl für einzelne Datensätze, als auch für Gruppen von Datensätzen. Sie können Daten hinzufügen (INSERT), entfernen (DELETE) und ändern (UPDATE).

Tag 12 vermittelt Ihnen die Grundlagen der Programmierkonzepte mit Transact-SQL sowie einen Einstieg in Transaktionen und Sperren.

Alle diese Abfragen und Änderungen, die Sie bisher kennengelernt haben, lassen sich durch den Einsatz von Indizes optimieren. Der 13. Tag untersucht Indizes ausführlich und geht dabei sowohl auf zusammengesetzte als auch einfache Indizes ein. Außerdem erhalten Sie eine Einführung in die Volltextindizierung.

Datenabfragen und -änderungen bleiben sinnlos, wenn man den Daten in der Datenbank nicht vertrauen kann. Der 14. Tag beschäftigt sich mit der Datenintegrität. Dieses Kapitel versetzt Sie in die Lage, die Menge an ungültigen Daten, die in Ihre Datenbank gelangen kann, zu minimieren. Dabei lernen Sie auch den Nutzen der deklarativen referenziellen Integrität kennen.

Das hört sich nach einer langen Woche an, aber halten Sie durch! Sie werden einige der schwierigsten Themen im Zusammenhang mit Datenbanksystemen behandeln, aber am Ende haben Sie zwei Drittel Ihres Weges durch den 21-Tage-Kurs zurückgelegt und sich dabei solide Kenntnisse der behandelten Themen angeeignet. Führen Sie die Übungen am Ende jeder Lektion durch, da Sie bei der Entwicklung häufig auf Vorhandenem aufbauen müssen.



**Wiederherstellen
von Datenbanken**



In der gestrigen Lektion haben Sie Sicherungen in SQL Server untersucht. Heute schließen wir das Thema, das vielleicht zu den wichtigsten aber wenig glanzvollen Teilen der Wartung einer relationalen Datenbank gehört, mit der Wiederherstellung von Datenbanken ab.

Unabhängig davon, wie viel Arbeit Sie investieren, um Ihre Datenbank sicher und Ihre Daten verfügbar zu machen, ist der Schutz Ihrer Daten einer der kritischsten Aspekte Ihrer Arbeit. Wenn Ihr Server abstürzt und Ihre Daten verloren gehen, sind Sie vielleicht auch Ihren Job los! Heute untersuchen Sie den Wiederherstellungsvorgang und die verschiedenen von SQL Server unterstützten Szenarios. Dazu gehören die automatische und die manuelle Wiederherstellung. Der größte Teil der heutigen Lektion wird sich mit der manuellen Wiederherstellung beschäftigen – d.h. mit der Wiederherstellung einer Datenbank zu einem bestimmten Zeitpunkt.

8.1 Datenbanken wiederherstellen

Sicherungskopien sind zwar gut, aber nicht von vornherein sonderlich nützlich. Ihre wirkliche Bedeutung zeigen sie erst, wenn Sie Probleme mit Ihrem Server haben. Ob eine Festplatte ihre Funktion einstellt oder eine der Datenbankdateien beschädigt wird, jetzt müssen Sie jede Datenbank, die von den verlorenen Dateien betroffen ist, wiederherstellen.

Ein weiterer Grund eine Sicherungskopie wieder einzuspielen, ist es, den Zustand einer Datenbank zu einem bestimmten Zeitpunkt wiederherzustellen. Nehmen wir an, dass Ihr Chef um 17 Uhr alle Informationen über die an diesem Tag erfolgten Verkäufe gelöscht hat. Sie können dann mithilfe der letzten vollständigen Sicherung zunächst die Daten wiederherstellen und daraufhin die Sicherungskopien der Transaktionsprotokolle (oder weitere differenzielle Sicherungen und die Sicherungskopien der Transaktionsprotokolle) bis zu dem Zeitpunkt, bevor Ihr Chef die Daten löschte, darauf anwenden. Damit lässt sich mit einem minimalen Arbeitsaufwand der Datenbestand wiederherstellen.



SQL Server 2000 kann zwar Sicherungen von SQL Server 7.0 wiederherstellen, nicht jedoch Sicherungen aus allen früheren Versionen. Umgekehrt lässt sich eine mit SQL Server 2000 angelegte Sicherung mit keiner früheren Version von SQL Server – inklusive SQL Server 7.0 – einlesen.

Bevor Sie sich weiter mit der Wiederherstellung von Daten beschäftigen, müssen Sie die Arten der Wiederherstellung von SQL Server kennen: automatische und manuelle Wiederherstellung.

Automatische Wiederherstellung

Die automatische Wiederherstellung wird von SQL Server bei jedem Start des SQL Server-Dienstes durchgeführt. Sie können diesen Vorgang nicht abstellen und Sie müssen auch nichts tun, damit er ausgeführt wird (deshalb heißt er *automatisch*).

Bei jedem Neustart führt SQL Server eine bestimmte Abfolge von Schritten aus. Für alle nach dem letzten Prüfpunkt im Transaktionsprotokoll bestätigten Transaktionen wird zunächst ein *Rollforward* ausgeführt. Rollforward heißt, dass alle Transaktionen, für die ein COMMIT ausgeführt wurde, erneut auf die Datenbank angewendet werden. Danach führt SQL Server für alle Transaktionen im Transaktionsprotokoll, für die kein COMMIT ausgeführt wurde, einen *Rollback* durch. Rollback bedeutet, dass alle Transaktionen, die teilweise ausgeführt wurden, für die aber kein COMMIT ausgeführt wurde, wieder aus der Datenbank entfernt werden. Am Ende der automatischen Wiederherstellung liegen die Datenbanken wieder in einer logisch konsistenten Form vor und im Transaktionsprotokoll wird ein Prüfpunkt gesetzt. Mehr über Transaktionen erfahren Sie am Tag 12.

Die automatische Wiederherstellung garantiert, dass SQL Server unabhängig davon, weshalb er heruntergefahren wurde, in einem logisch konsistenten Zustand startet. Selbst wenn der Server unerwartet abstürzt, können Sie ihn problemlos wiederherstellen. Die Protokolleinträge aller ausgeführten Transaktionen werden sofort auf die Festplatte geschrieben (man bezeichnet dies als *synchrones Schreiben*). Deshalb befinden sich alle dauerhaften Änderungen an der Datenbank bereits auf der Festplatte und werden bei der automatischen Wiederherstellung berücksichtigt.

Die automatische Wiederherstellung bearbeitet jede Datenbank nach einer festgelegten Reihenfolge. Zunächst wird bestimmt, wo sich die Dateien der *master*-Datenbank (*master.mdf* und *mastlog.ldf*) befinden, indem der entsprechende Eintrag in der Windows-Registrierung ausgewertet wird. Sie finden diese Information im Registrierungsschlüssel `HKEY_LOCAL_MACHINE\Software\Microsoft\MSSQLServer\MSSQLServer\Parameters` für eine Standardinstanz bzw. im Schlüssel `HKEY_LOCAL_MACHINE\Software\Microsoft\Microsoft SQL Server\Instanzname\MSSQLServer\Parameters` für eine benannte Instanz (wobei Sie den Namen Ihrer Instanz anstelle von *Instanzname* einsetzen). Hier steht, wo SQL Server die Dateien Ihrer *master*-Datenbank vermutet. Abbildung 8.1 zeigt ein Beispiel für den relevanten Abschnitt der Windows-Registrierung.

Nachdem SQL Server die Dateien der *master*-Datenbank gefunden hat, wird die *master*-Datenbank geladen und wiederhergestellt. Sie müssen die *master*-Datenbank als erste laden und wiederherstellen, da sie Verweise auf die primären Datendateien (die *.mdf*-Dateien) der restlichen Datenbanken in der Systemtabelle *sysdatabases* in der Spalte *filename* enthält. Nachdem die *master*-Datenbank wiederhergestellt wurde, kann SQL Server damit beginnen, die restlichen Datenbanken zu suchen, zu öffnen und wiederherzustellen. Jede Datenbank zeichnet auf, welche Dateien zu ihr gehören, indem sie diese Informationen in den in jeder Datenbank vorhandenen Systemtabellen *sysfiles* und *sysfilegroups* speichert.

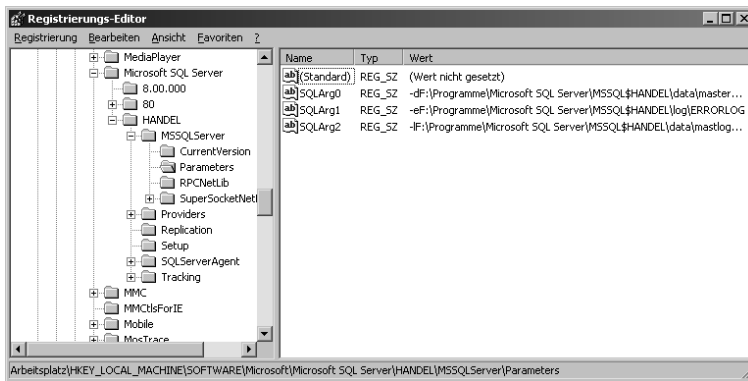


Abbildung 8.1:
Die SQL Server-
Registrierungs-
schlüssel für den
Dienststart

Bevor SQL Server irgendeine andere Datenbank wiederherstellen kann, muss zunächst die *model*-Datenbank wiederhergestellt werden. Wie Sie sich erinnern, ist die *model*-Datenbank das Modell für alle weiteren Datenbanken. Die *model*-Datenbank muss wiederhergestellt sein, bevor Sie fortfahren können, weil Ihr nächster Schritt die Erstellung der *tempdb*-Datenbank ist. Sie brauchen die *tempdb*-Datenbank für die Ausführung aller möglichen Abfragen und gespeicherten Prozeduren in SQL Server. Die *tempdb*-Datenbank wird bei jedem Start von SQL Server neu angelegt. Der Server erzeugt diese Datenbank, indem er eine Kopie der *model*-Datenbank anlegt und diese Kopie dann auf die von Ihnen vorher festgelegte Größe erweitert.

Als Nächstes werden die *msdb*-Datenbank, gefolgt von der *distribution*-Datenbank (falls vorhanden), dann die Datenbanken *pubs* und *Northwind* und schließlich die Benutzerdatenbanken (das sind die, die Ihnen wichtig sind) wiederhergestellt.

Sie können den gesamten Vorgang nachvollziehen, indem Sie sich das Fehlerprotokoll von SQL Server ansehen:

1. Falls der SQL Server Enterprise Manager noch nicht läuft, starten Sie ihn, stellen eine Verbindung zu Ihrem Server her, erweitern den Ordner *Verwaltung* und dann den Ordner *SQL Server-Protokolle*. Darunter finden Sie eine Liste mit dem aktuellen Fehlerprotokoll und normalerweise den sechs vorangehenden Protokollen.
2. Markieren Sie das Fehlerprotokoll, das Sie sich ansehen wollen. Es erscheint dann im Detailbereich des SQL Server Enterprise Managers (siehe Abbildung 8.2).



Sie können sich diese Dateien auch direkt mit einem Editor ansehen. Zu finden sind sie im Verzeichnis `\mssql\log` unter dem Namen `errorlog.*`, wobei * die Nummer der Sicherungskopie ist. Das aktuelle Fehlerprotokoll trägt den Namen `errorlog`. (mit einem Punkt, aber ohne Erweiterung). Die früheren Protokolle heißen `errorlog.1`, `errorlog.2` und so weiter.

3. Suchen Sie den Eintrag *Startet Datenbank 'master'*.



Das in Abbildung 8.2 dargestellte Fehlerprotokoll zeigt an, dass die Datenbank *master* (durch SQL Server) geöffnet und dann zu einem konsistenten logischen Punkt wiederhergestellt wurde.

Daran schließt sich die Wiederherstellung der Datenbanken *model*, *msdb* und der Benutzerdatenbanken (einschließlich *pubs* und *Northwind*) an. Als Nächstes wird *tempdb* gelöscht (durch eine Kopie von *model* überschrieben) und gestartet. Die erfolgreich abgeschlossene Wiederherstellung erkennen Sie am Eintrag *Wiederherstellung abgeschlossen* (in Abbildung 8.2 markiert).

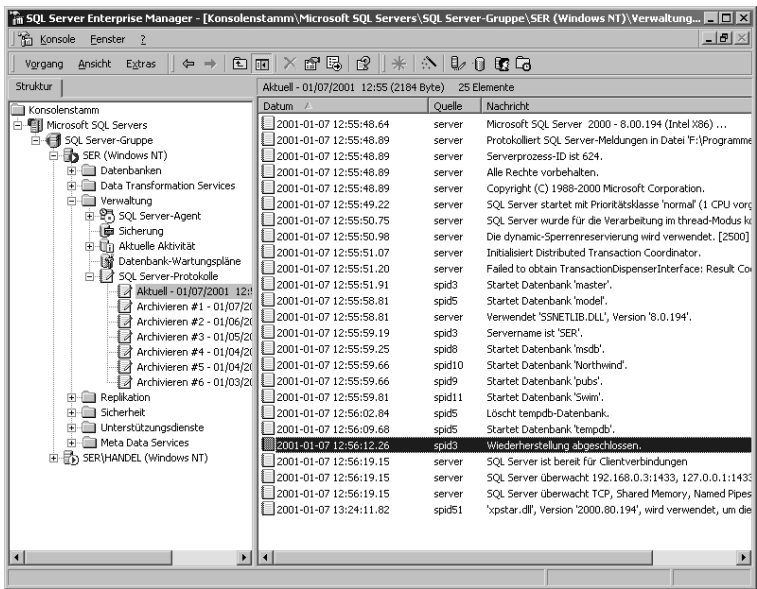


Abbildung 8.2:
Die Ansicht des Fehlerprotokolls
im Enterprise
Manager

Konfiguration der automatischen Wiederherstellung

Bei der Konfiguration der automatischen Wiederherstellung ist es vor allem wichtig, das Wiederherstellungsintervall einzustellen. Diese Option gibt die Zeit an, die der SQL Server maximal benötigt, um jede Datenbank wiederherzustellen. Der Standardwert beträgt 0 Minuten, d. h. der SQL Server entscheidet automatisch darüber, wie häufig ein Prüfpunkt gesetzt wird.

Wie bereits erwähnt, ist ein Prüfpunkt der Vorgang, der die geänderten Datenseiten und Protokolleinträge aus dem Speicher auf die Festplatte kopiert. Wenn dieser Vorgang abgeschlossen ist, muss für bestätigte Transaktionen kein Rollforward (Wiederherstellung) mehr durchgeführt werden.



Sie können das Wiederherstellungsintervall mit der gespeicherten Systemprozedur `sp_configure` einstellen. Um beispielsweise die Wiederherstellungszeit auf drei Minuten pro Datenbank zu begrenzen, führen Sie die folgende Anweisung aus:



```
EXEC sp_configure 'Recovery Interval', 3
GO
RECONFIGURE WITH OVERRIDE
GO
```

Sie sollten das Wiederherstellungsintervall auf dem vorgegebenen Wert für die automatische Einstellung belassen, solange Sie nicht auf Leistungseinbrüche durch zu häufige Prüfpunkte stoßen.



Das Wiederherstellungsintervall wird Ihnen nichts nützen, wenn Sie eine sehr lange dauernde Transaktion durchführen. Nehmen Sie einmal an, dass beispielsweise ein Systemprozess eine Transaktion startet, und mit einer Reihe von Änderungen beginnt, die Stunden dauern und Ihr Server in dieser Zeit abstürzt. Falls Ihr Speicher nicht ausreicht, um alle Änderungen aufzunehmen und einige der Änderungen deshalb bereits auf die Festplatte übertragen wurden, muss für diese ein Rollback durchgeführt werden. Selbst wenn Sie Ihr Wiederherstellungsintervall vorher (wie im vorangehenden Beispiel) auf drei Minuten gesetzt haben, wird es voraussichtlich länger als drei Minuten dauern, diese Datenbank wiederherzustellen. Um diesem Problem zu entgehen, begrenzen Sie die Größe Ihrer Transaktionen, um im Problemfall eine schnelle Wiederherstellung zu erreichen.

Manuelle Wiederherstellung

Der Rest der heutigen Lektion beschäftigt sich mit der manuellen Wiederherstellung von Datenbanken. Dies kann die Wiederherstellung von einer vollständigen oder differenziellen Sicherung oder die Wiederherstellung von einem oder mehreren Transaktionsprotokollen von Ihren Sicherungsgeräten sein. Die Einzelheiten des Wiederherstellungsprozesses hängen davon ab, aus welchem Grund eine Datenbank wiederhergestellt werden muss.

Sie können eine Datenbank wiederherstellen und dann die Transaktionsprotokolle anwenden, um eine verlorene Datenbank vollständig wiederherzustellen. Sie können ebenso den Zustand einer Datenbank zu einem bestimmten Zeitpunkt wiederherstellen, um Änderungen an der Datenbank rückgängig zu machen. Man kann sogar eine benannte (eine so genannte *markierte*) Transaktion wiederherstellen. Außerdem können Sie differenzielle Sicherungen verwenden (falls Sie welche erstellt haben), um die Anzahl der wiederherzustellenden Transaktionsprotokolle zu verringern. SQL Server 2000 verfügt darüber hinaus über eine Option, lediglich einen Satz von Dateigruppen wiederherzustellen (d.h. eine teilweise Wiederherstellung durchzuführen).

Die Wiederherstellung einer Datenbank unterliegt folgenden Einschränkungen:

- Die Datenbank darf nicht verwendet werden. Eine Datenbank wird verwendet, wenn ein Benutzer die Anweisung `USE` für diese Datenbank ausgeführt hat oder eine Abfrage in dieser Datenbank durchführt.
- Sie müssen ausreichende Benutzerrechte haben. Nur Mitglieder der festen Serverrollen *sysadmin* oder *dbcreator* oder der Benutzer *dbo* in einer Datenbank dürfen eine Datenbank wiederherstellen. Dieses Recht kann nicht an andere Benutzer vergeben werden, da es sich um ein sehr weitreichendes Recht handelt.
- Sie müssen die Transaktionsprotokolle in der Reihenfolge ihrer Entstehung wiederherstellen. Jedes Transaktionsprotokoll besitzt eine laufende Nummer. Sie müssen die Sicherungskopien der Transaktionsprotokolle in der richtigen Reihenfolge wiederherstellen, andernfalls wird die Wiederherstellung scheitern. In der heutigen Lektion lernen Sie noch weitere Aspekte kennen, die vor einer Wiederherstellung der Transaktionsprotokolle zu berücksichtigen sind.

Falls Sie die Datenbank wiederherstellen müssen, da eine Ihrer Festplatten ausgefallen ist, müssen Sie als Erstes die defekte Festplatte austauschen (und dabei die Spiegelung oder andere RAID-Konfigurationen erneut einrichten) und daraufhin alle betroffenen Datenbanken löschen. Sie können die Datenbanken dann neu erstellen oder sie im Laufe der Wiederherstellung neu erstellen lassen.

Falls Sie eine Datenbank wiederherstellen, die bereits existiert, werden standardmäßig alle Informationen der bestehenden Datenbank durch die aus der Sicherungskopie ersetzt (mit einigen wichtigen Ausnahmen, die Sie später erfahren werden).

Wiederherstellung aus den Dateien (erneutes Anfügen einer Datenbank)

Bevor Sie sich mit dem Wiederherstellungsprozess beschäftigen, sollten Sie sich mit einer weiteren Wiederherstellungsoption befassen. Sie können eine Datenbank erneut an den Server anfügen, wenn Sie eine Kopie aller dazugehörigen Dateien besitzen. SQL Server 2000 kann Datenbanken sowohl vom Server trennen als auch wieder anfügen. Zu diesem Zweck stehen gespeicherte Systemprozeduren und die Anweisung `CREATE DATABASE` mit der Option `FOR ATTACH` zur Verfügung.

Zunächst werden wir uns die Anweisung `CREATE DATABASE` ansehen. Dieses Programmbeispiel zeigt die Erzeugung einer Datenbank mit den beiden Dateien *test_data.mdf* und *test_log.ldf*:



Wiederherstellen von Datenbanken



```
create database test
on (filename =
    'F:\Programme\Microsoft SQL Server\MSSQL\data\test_data.mdf')
log on (filename =
    'F:\Programme\Microsoft SQL Server\MSSQL\data\test_log.ldf')
FOR ATTACH
```

Beachten Sie, dass Sie nur die Dateinamen angeben müssen. Wenn SQL Server die Dateien findet, können alle weiteren benötigten Informationen aus den Systemtabellen, die in der Datenbank gespeichert sind, abgerufen werden.

Sie können genau so gut auch die gespeicherte Systemprozedur `sp_attach_db` verwenden. Versuchen Sie es selbst, indem Sie eine Testdatenbank (wie oben) erzeugen, und dann die gespeicherte Systemprozedur `sp_detach_db` folgendermaßen anwenden:



```
Exec sp_detach_db 'test'
Go
```

Damit trennen Sie die Datenbank `test` vom Server. Sie können Ihre Datenbankdateien jetzt an jemanden verschicken, und der Empfänger kann folgende Anweisungen ausführen (Sie natürlich auch):

```
EXEC sp_attach_db @dbname = 'test',
@filename1='F:\Programme\Microsoft SQL Server\mssql\data\test.mdf',
@filename2='F:\Programme\Microsoft SQL Server\mssql\data\test_log.ldf'
```

Die gespeicherte Systemprozedur `sp_attach_db` führt im Wesentlichen die Anweisung `CREATE DATABASE FOR ATTACH` aus. Es gibt keinen funktionalen Unterschied zwischen beiden Optionen (d.h. zwischen der Anweisung `CREATE DATABASE` und der gespeicherten Systemprozedur `sp_attach_db`). Falls Sie im Codebeispiel einfach den Befehl

```
create database test
```

ausführen, wird die Datenbank mit den im Beispiel gezeigten Dateinamen erstellt. Wenn Sie verschiedene Dateinamen verwenden, müssen Sie die notwendigen Ersetzungen im Pfad zu den Dateien im Aufruf von `sp_attach_db` ersetzen.



Sie sollten beachten, dass dies keine gute Sicherungs- und Wiederherstellungsstrategie ist. Sie sollten vollständige und differenzielle Sicherungen durchführen und die Transaktionsprotokolle sichern, wie es Tag 7 beschrieben hat. Wie Sie noch sehen werden, unterliegen Sie einschneidenden Beschränkungen bei der Wiederherstellung von Datenbanken, wenn Sie die in SQL Server 2000 eingebauten Möglichkeiten zur Sicherung und Wiederherstellung nicht nutzen.

Den richtigen Sicherungssatz finden

Ihr erster Schritt besteht darin, den richtigen Sicherungssatz zu finden. Es gibt dafür zwei Wege: den einfachen und den komplizierten. Sie sollten (wie immer) mit dem komplizierten beginnen, weil Sie so lernen, was hierbei wirklich passiert. Es gibt drei Transact-SQL-Anweisungen, die Sie verwenden können, um (in unterschiedlicher Ausführlichkeit) herauszufinden, was sich auf Ihrem Sicherungsmedium befindet:

- **RESTORE LABELONLY** liefert Ihnen eine einzelne Zeile mit zusammenfassenden Informationen über den gesamten Sicherungssatz zurück.
- **RESTORE HEADERONLY** liefert Ihnen zusammenfassende Informationen für jedes Objekt eines Sicherungssatzes.
- **RESTORE FILELISTONLY** liefert Ihnen eine Liste aller auf dem Sicherungsmedium gespeicherten Sicherungen von Datenbanken und Transaktionsprotokollen.

RESTORE LABELONLY

Wenn Sie die Anweisung **RESTORE LABELONLY** ausführen, wird der Medienheader des Sicherungsgerätes (normalerweise verwenden Sie diese Anweisung für Bandlaufwerke) gelesen und eine Zusammenfassung der darin enthaltenen Informationen in einer Zeile zurückgeliefert. Diese Zeile enthält den Namen des Mediums, den Sie während der Sicherung vergeben haben, die Beschreibung, die Sie bei der Sicherung Ihrer Datenbank eingegeben haben sowie das Datum der letzten Sicherung auf dieses Medium. Es werden noch weitere Informationen zurückgeliefert, die Sie aber nicht weiter zu beachten brauchen, es sei denn, Sie wollten ein fortgeschrittenes Sicherungskonzept erarbeiten.



```
RESTORE LABELONLY FROM <backup_device>
[WITH (NOUNLOAD | UNLOAD )]
[[,] MEDIAPASSWORD = {mediapwd | @mediapwd_var}]
```

Die Argumente haben folgende Bedeutung:

- *backup_device* ist der Name eines Sicherungsgerätes oder eines variablen Platzhalters für das Sicherungsgerät. Sie können auch einfach den Namen der Datei, des Bandes oder der Named Pipe angeben, der für die Wiederherstellung verwendet werden soll. Auch hierbei können Sie eine Variable einsetzen. Sehen Sie sich noch einmal die Syntax der gestern erläuterten **BACKUP**-Anweisung an, die eine Beschreibung für den Platzhalter *backup_device* beinhaltet.
- **NOUNLOAD | UNLOAD** gibt an, ob das Band nach Abschluss der Anweisung **RESTORE LABELONLY** auszuwerfen ist.
- **MEDIAPASSWORD** ist das beim Erstellen der Sicherung festgelegte Kennwort (falls überhaupt angegeben). Wenn Sie nicht über das Kennwort verfügen und das Siche-

rungsmedium kennwortgeschützt ist, können Sie diesen Befehl nicht ausführen. Kennwörter für einzelne Sicherungen verhindern RESTORE LABELONLY nicht; nur wenn das Kennwort für das Medium selbst fehlt, lässt sich der Befehl nicht erfolgreich abschließen.

RESTORE HEADERONLY

RESTORE HEADERONLY liefert Informationen über jede Sicherung auf dem Sicherungsgerät. Es wird für jede vorhandene Sicherung eine Ausgabezeile zurückgeliefert. Das liegt daran, dass RESTORE HEADERONLY anders als RESTORE LABELONLY nacheinander die Informationen für jede auf dem Medium vorhandene Sicherung liest. Der Vorgang kann relativ zeitraubend sein, wenn Sie viele Sicherungen auf einem Band haben. Für Sicherungsdateien auf Festplatten wird die Anweisung jedoch normalerweise recht schnell ausgeführt.



```
RESTORE HEADERONLY FROM <backup_device>
    [WITH {NOUNLOAD | UNLOAD}][[,] FILE = file_num]
    [[,] PASSWORD = { pwd | @pwd_var }]
    [[,] MEDIAPASSWORD = {mediapwd | @mediapwd_var } ]
```

Diese Syntax verwendet die gleichen Optionen wie RESTORE LABELONLY, mit Ausnahme der beiden folgenden:

- *FILE* gibt an, welcher Sicherungssatz auf dem Band zu beschreiben ist. Wenn das Band zum Beispiel zwei Sicherungssätze enthält, können Sie entweder 1 (für den ersten Satz) oder 2 (für den zweiten Satz) spezifizieren.
- *PASSWORD* ist das Kennwort, dass Sie beim Erstellen des Sicherungssatzes festgelegt haben.

Das Resultat dieser Anweisung enthält die Namen (und die Beschreibung) der Sicherungskopien und ihr Verfallsdatum. Zusätzlich erhalten Sie Informationen darüber, wer die Sicherung angelegt hat, von welchem Server sie kam, welche Datenbank darin enthalten ist, wie groß die Sicherung ist, wann die Sicherung begann und wann sie endete. Weiterhin liefert diese Anweisung Angaben zu Codepage und Sortierreihenfolge (in SQL Server 7.0 verwendet), Sortierung (das SQL Server 2000-Äquivalent von Sortierreihenfolge / Codepage / Unicode-Gebietsschema-ID), Datenbankkompatibilitätsmodus und Versionsnummer des Servers für die Sicherung. Es gibt auch noch einige interne Informationen über Protokollsequenznummern (die aber über den Rahmen dieses Buches hinausgehen).

RESTORE FILELISTONLY

Die Anweisung RESTORE FILELISTONLY liefert Ihnen eine Liste der Datenbank- und Transaktionsprotokolldateien, die in einer bestimmten Sicherung enthalten sind. Sie können diese Informationen nur für jeweils eine Sicherung erhalten, deshalb müssen Sie angeben, über welche Sie Informationen erhalten wollen, wenn sich mehrere Sicherungen auf dem Sicherungsgerät befinden.

Diese Informationen sind für die Wiederherstellung nach einer Katastrophe besonders nützlich. Falls Sie eine Datenbank wiederherstellen müssen und nicht mehr wissen, wo die Dateien ursprünglich gespeichert waren, können Sie dies mithilfe der Anweisung **RESTORE FILELISTONLY** herausfinden.



```
RESTORE FILELISTONLY FROM <backup_device>
    [WITH {NOUNLOAD | UNLOAD}][[,] FILE = file_num]
    [[,] PASSWORD = { pwd | @pwd_var }]
    [[,] MEDIAPASSWORD = { mediapwd | @mediapwd_var } ]
```

Diese Syntax verwendet die gleichen Optionen wie **RESTORE HEADERONLY**.

Falls Sie die Dateinummer der Sicherung, die Sie untersuchen wollen, nicht kennen, führen Sie die Anweisung **RESTORE HEADERONLY** aus, um eine Liste der Dateinummern und der zugehörigen Sicherungen zu erhalten. Falls Sie den Parameter auslassen, wird angenommen, dass die erste Sicherung auf dem Sicherungsgerät verwendet werden soll.

Feststellen, ob der Sicherungssatz verwendbar ist

Nehmen wir für eine Minute an, dass Sie die erforderlichen DBCC-Überprüfungen vor der eigentlichen Datenbanksicherung vorgenommen haben. Nehmen wir weiterhin an, dass die Sicherung mit der Option *Sicherung bei Abschluss überprüfen* durchgeführt wurde. Heißt das, dass Ihre Sicherung auch wiederhergestellt werden kann?

Falls Sie darauf nicht sofort mit *Ja* antworten können, sind Sie nicht allein. Microsoft hatte scheinbar ein ähnliches Gefühl, deshalb gibt es die Anweisung **RESTORE VERIFYONLY**.

RESTORE VERIFYONLY

Die Anweisung **RESTORE VERIFYONLY** überprüft, ob alle Bänder oder Dateien für die Wiederherstellung zur Verfügung stehen und dass alle benötigten Informationen auch gelesen werden können. Die Anweisung führt keine DBCC-Überprüfungen für Ihre Sicherung durch – es wird nur getestet, ob die Dateien von Band oder Platte gelesen werden können.



```
RESTORE VERIFYONLY FROM <backup_device> [...n]
    [WITH [FILE = file_num] [[,] {NOUNLOAD | UNLOAD}]
    [[,] LOADHISTORY] [[,] {NOREWIND | REWIND} ]
    [[,] PASSWORD = { pwd | @pwd_var } ]
    [[,] MEDIAPASSWORD = { mediapwd | @mediapwd_var } ]
    ]
```

Diese Syntax verwendet die gleichen Optionen wie die bereits besprochenen Anweisungen, mit Ausnahme der folgenden Argumente:



- *LOADHISTORY* gibt an, dass die Informationen über die untersuchte Sicherung zu den Chroniktabellen in der *msdb*-Datenbank hinzugefügt werden sollen. Sie werden diese Tabellen sofort näher kennen lernen.
- *NOREWIND* | *REWIND* gibt an, ob das Band nach Abschluss von *RESTORE VERIFYONLY* zurückzuspulen ist.

Die msdb-Verlaufstabellen für Sicherungen

Bevor wir mit den eigentlichen Wiederherstellungsanweisungen fortsetzen, ist ein kleiner Exkurs angebracht. Jedes Mal, wenn Sie eine Sicherung durchführen, werden die Einzelheiten dieser Sicherung in der *msdb*-Datenbank in folgenden Tabellen gespeichert:

- *backupfile*: Enthält Informationen über Dateisicherungen.
- *backupmediafamily*: Enthält Informationen über Medienfamilien. Sie haben gestern eine kurze Einführung in das Konzept der Medienfamilien erhalten. Wie Sie sich erinnern werden, dienen Medienfamilien normalerweise der parallelen verteilten Sicherung auf verschiedene physikalische Medien.
- *backupmediaset*: Enthält Informationen über den Sicherungsmediensatz. Auch Mediensätze werden normalerweise für die Beschreibung paralleler verteilter Sicherungen verwendet, aber jede Sicherung gehört einem Mediensatz an.
- *backupset*: Enthält Informationen über jede von Ihnen erstellte Sicherung.
- *restorefile*: Enthält Informationen über jede wiederhergestellte Datei.
- *restorefilegroup*: Enthält Informationen über jede wiederhergestellte Dateigruppe.
- *restorehistory*: Enthält Informationen über jede von Ihnen durchgeführte Wiederherstellung.

Jedes Mal, wenn Sie irgendeine Sicherung oder eine Wiederherstellung durchführen, wird alles Notwendige in diesen Systemtabellen aufgezeichnet. Es spielt dabei keine Rolle, ob Sie die Sicherungs- und Wiederherstellungsvorgänge mit Transact-SQL-Anweisungen oder mit dem SQL Server Enterprise Manager durchführen. Daraus folgt aber auch, dass Sie Ihre *msdb*-Datenbank regelmäßig sichern müssen, um diese Informationen auch bei einem Verlust Ihrer *msdb*- oder *master*-Datenbank zu behalten.

Vollständige Wiederherstellung einer Datenbank

Die vollständige Wiederherstellung einer Datenbank verwendet eine vollständige Sicherung Ihrer Datenbank (man kann das als »normale« Sicherung betrachten) und legt diese wieder auf Ihrem Server ab. Danach wird das Transaktionsprotokoll, das mit der Sicherung der Datenbank gesichert wurde, verwendet, um für alle zum Zeitpunkt der Sicherung

bestätigten Transaktionen einen Rollforward durchzuführen, während für Transaktionen, die zum Zeitpunkt der Sicherung noch nicht beendet waren, ein Rollback ausgeführt wird. Wie bereits bei der Sicherung, bietet auch hier die Transact-SQL-Anweisung RESTORE mehr Möglichkeiten als der SQL Server Enterprise Manager. Die Dialogfelder des SQL Server Enterprise Managers sind zudem leichter zu verstehen, wenn Sie wissen, wie diese Anweisungen durch den SQL Server ausgeführt werden.

Sie haben bei der Wiederherstellung einer Datenbank verschiedene Optionen zur Auswahl. Sie können

- Dateien auf einen anderen Speicherplatz als zum Zeitpunkt der Datenbanksicherung verschieben,
- die Datenbankoption RESTRICTED_USER während der Wiederherstellung setzen, falls Sie noch nicht fertig sind oder den Zugriff auf die Datenbank einschränken möchten,
- wählen, ob Sie einen konsistenten Zustand zu einem bestimmten Zeitpunkt (abhängig davon, ob Sie Sicherungen der Transaktionsprotokolle einsetzen wollen oder nicht) oder zu einem markierten Transaktionseintrag wiederherstellen wollen,
- eine Rückgängigdatei erstellen. Eine derartige Datei erstellen Sie für den Fall, dass Sie weitere Sicherungen von Transaktionsprotokollen auf die Datenbank anwenden wollen und den Zustand der Datenbank im »schreibgeschützten« Modus zwischendurch kontrollieren wollen. Sie werden gleich mehr über diese Option erfahren,
- eine Wiederherstellung fortsetzen, die aus irgendeinem Grund abgebrochen wurde (z. B. weil während des Wiederherstellungsvorgangs ein Stromausfall auftrat).

Durchführung einer vollständigen Wiederherstellung einer Datenbank mithilfe der Transact-SQL-Anweisung RESTORE

Sie verwenden für die vollständige Wiederherstellung einer Datenbank aus einer vollständigen Sicherungskopie der Datenbank die Anweisung RESTORE. Sehen Sie sich zunächst die Syntax an:



```
RESTORE DATABASE databasename FROM <backup_device> [, ...n]
[WITH [RESTRICTED_USER] [[,] FILE = file_num]
[[,] PASSWORD = { pwd | @pwd_var } ]
[[,] MEDIANAME = { media_name | @media_name_variable}]
[[,] MEDIAPASSWORD = { mediapwd | @mediapwd_var } ]
[[,] MOVE 'logical_file_name' TO 'new_file_name'][,...n]
[[,] KEEP_REPLICATION]
[[,] {NORECOVERY | RECOVERY | STANDBY = undo_file_name}]
[[,] {NOUNLOAD | UNLOAD}] [[,] REPLACE] [[,] RESTART]
[[,] { NOREWIND | REWIND } ]
[[,] STATS [= percentage]]]
```



Die Parameter haben folgende Bedeutung:

- *dbname* ist der Name der wiederherzustellenden Datenbank.
- *backup_device* ist entweder der Name eines Sicherungsgerätes oder ein variabler Platzhalter für ein Sicherungsgerät. Sie können auch einfach den Namen der Datei, des Bandes oder der Named Pipe angeben, der für die Wiederherstellung verwendet werden soll. Auch hierbei können Sie eine Variable einsetzen.
- *RESTRICTED_USER* gibt an, dass nur Mitglieder der festen Serverrollen *sysadmin* oder *dbcreator* oder der festen Datenbankrolle *db_owner* auf die Datenbank zugreifen können, nachdem die Wiederherstellung abgeschlossen ist.
- *FILE* gibt an, welche Sicherung auf dem Sicherungsgerät Sie verwenden wollen.
- *PASSWORD* ist das Kennwort, das Sie beim Erstellen des Sicherungssatzes festgelegt haben.
- *MEDIANAME* gibt an, dass der angegebene Mediennamen mit dem in der Sicherung gespeicherten verglichen werden soll. Sind sie nicht gleich, scheitert die Wiederherstellung.
- *MEDIAPASSWORD* ist das beim Erstellen der Sicherung festgelegte Kennwort (falls überhaupt angegeben). Wenn Sie nicht über das Kennwort verfügen und das Sicherungsmedium kennwortgeschützt ist, können Sie diesen Befehl nicht ausführen.
- *MOVE* gibt an, dass für jeden von Ihnen angegebenen logischen Dateinamen die entsprechende logische Datei in der Sicherungskopie gesucht wird und diese unter dem von Ihnen angegebenen Pfad und Dateinamen wiederhergestellt werden soll.
- *KEEP_REPLICATION* weist den Wiederherstellungsvorgang an, die Replikationseinstellungen einer Datenbank beizubehalten. Diese Option ist für den Protokollversand verfügbar; auf diese erweiterte Technik geht das Buch nicht weiter ein.
- *RECOVERY* weist die Wiederherstellungsoperation an, für alle unbestätigten Transaktionen einen Rollback auszuführen, um die Datenbank wieder in einen konsistenten Zustand zu überführen. Nachdem Sie diese Option angewandt haben, können Sie keine weiteren Sicherungen von Transaktionsprotokollen oder weitere differenzielle Sicherungskopien auf die Datenbank anwenden. Diese Option ist der Vorgabewert, falls Sie keine andere Wiederherstellungsoption (*NORECOVERY* oder *STANDBY*) angeben.
- *NORECOVERY* weist die Wiederherstellungsoperation an, für keine der unbestätigten Transaktionen einen Rollback auszuführen, so dass die Datenbank nicht wieder in einen konsistenten Zustand überführt wird. Verwenden Sie diese Option während der Wiederherstellung Ihrer Datenbank, wenn Sie nach der vollständigen Wiederherstellung der Datenbank weitere Sicherungen von Transaktionsprotokollen oder weitere differenzielle Sicherungskopien auf die Datenbank anwenden wollen.

- **STANDBY** gibt den Pfad und den Dateinamen für eine Rückgängigdatei an. Diese Datei kann ausreichend Informationen aufnehmen, um Transaktionen, für die während der Wiederherstellung ein Rollback durchgeführt wurde, erneut zu starten. Verwenden Sie die Option **STANDBY**, wenn Sie Ihre Datenbank (im schreibgeschützten Modus) vor der Anwendung weiterer Sicherungen von Transaktionsprotokollen oder weiterer differenzieller Sicherungskopien überprüfen wollen. Falls die angegebene Datei nicht existiert, wird sie vom SQL Server erstellt.
- **NOUNLOAD** | **UNLOAD** gibt an, ob das Band nach der Ausführung der Anweisung ausgeworfen werden soll.
- **REPLACE** gibt an, dass die Wiederherstellung bereits existierende Datenbanken mit demselben Namen ersetzen soll. Eine Wiederherstellungsoperation ohne Angabe dieses Parameters führt normalerweise dazu, dass die Operation scheitert, wenn der angegebene Datenbankname von dem in der Sicherungskopie gespeicherten abweicht, oder der Satz von Dateien in der Sicherungskopie von dem der existierenden Datenbank abweicht. Es handelt sich hierbei um eine Sicherheitsüberprüfung.
- **RESTART** führt die Wiederherstellungsoperation an der Stelle fort, an der sie abgebrochen wurde. Wie bereits oben bemerkt, kann diese Option nur auf die Wiederherstellung einer auf mehrere Bänder verteilten Sicherungskopie angewandt werden, wenn Sie sich auf dem zweiten oder einem späteren Band befinden.
- **NOREWIND** | **REWIND** gibt an, ob das Band nach Abschluss von **RESTORE** zurückgespult werden soll.
- **STATS** gibt an, dass sie jeweils beim Erreichen des angegebenen prozentualen Fortschritts eine Statusmeldung erhalten möchten. Als Vorgabewert zeigt SQL Server (wie bei Sicherungen) nach jeweils 10 Prozent Fortschritt eine Meldung an.

Um die *pubs*-Datenbank wiederherzustellen, die auf das Gerät *pubs_backup* gesichert wurde, und alle Standardeinstellungen zu akzeptieren, führen Sie folgende Anweisung aus:



```
RESTORE DATABASE PUBS FROM PUBS_BACKUP
```

Falls Sie bereits drei Sicherungskopien auf dem Gerät *pubs_backup* angelegt haben und Sie die dritte dieser vollständigen Sicherungen wiederherstellen wollen, führen Sie zwei Sätze von Befehlen aus. Der erste Schritt dient der Überprüfung, dass es wirklich die dritte Sicherungskopie ist, die Sie wiederherstellen wollen:



```
RESTORE HEADERONLY from pubs_backup
```

Der zweite Schritt führt die eigentliche Wiederherstellung durch:



```
RESTORE DATABASE PUBS FROM PUBS_BACKUP WITH FILE = 3
```

Um die *pubs*-Datenbank wiederherzustellen, dabei aber die Datei *pubs.mdf* auf Laufwerk F: des Computers zu speichern, die Datenbank im Standbystatus zu belassen und den Wiederherstellungsfortschritt in 25%-Schritten anzuzeigen, führen Sie folgende Anweisung aus:



```
RESTORE DATABASE PUBS FROM PUBS_BACKUP  
WITH MOVE 'pubs' TO 'F:\pubs.mdf',  
FILE = 3,  
STANDBY = 'f:\standby.dat',  
STATS = 25
```

und Sie sollten etwa folgende Ausgabe erhalten:



```
26 Prozent restored.  
53 Prozent restored.  
76 Prozent restored.  
100 Prozent restored.  
208 Seiten für die pubs-Datenbank, Datei 'pubs' auf Datei 3  
verarbeitet.  
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf Datei 3  
verarbeitet.  
RESTORE DATABASE hat 209 Seiten in 0,464 Sekunden  
erfolgreich verarbeitet (3,674 MB/Sekunde).
```

Jetzt müssten Sie eine Sicherung des Transaktionsprotokolls oder eine differenzielle Sicherung auf die Datenbank anwenden, um die Wiederherstellung zu vervollständigen. Führen Sie beispielsweise folgende Anweisung aus:



```
restore database pubs from pubs_backup with file = 4, RECOVERY
```

wobei Datei Nummer 4 eine differenzielle Sicherung ist, die nach der vollständigen Sicherung angefertigt wurde. Die Anweisung sollte etwa folgende Ausgabe zurückliefern:



Die Datenbankdatei 'f:\standby.dat' wird gelöscht.
5 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 4 verarbeitet.
RESTORE LOG hat 5 Seiten in 0,048 Sekunden erfolgreich
verarbeitet (0,853 MB/Sekunde).

Beachten Sie, dass SQL Server Ihre Rückgängigdatei löscht, da sie jetzt nicht mehr erforderlich ist.

Die Option MOVE

Die Option MOVE ermöglicht es Ihnen, Daten auf einfache Weise von einem Server auf einen anderen zu übertragen. Falls der Produktionsserver beispielsweise die Laufwerke H und I besitzt und diese Laufwerke die Datenbanken und Transaktionsprotokolle enthalten, können Sie Sicherungskopien auf Ihrem Entwicklungsserver wiederherstellen. Das gilt auch, wenn Ihr Entwicklungsserver nur die Laufwerke C, D und E enthält. Sie können einfach die logischen Dateinamen der alten Sicherungskopie angeben und jeder dieser logischen Dateien einen neuen Platz auf dem neuen Server zuweisen.



Denken Sie daran, dass Sie die Liste der in einem Sicherungssatz enthaltenen Dateien mit der Anweisung RESTORE FILELISTONLY erhalten können.

Diese Option kann auch im Katastrophenfall ganz hilfreich sein. Es ist möglich, wenn auch nicht wahrscheinlich, dass der neue Server, mit dem Sie sich danach abfinden müssen, nicht mehr dieselben Laufwerkbuchstaben zur Verfügung hat. In diesem Fall ist es wichtig, die Option MOVE zu kennen.



Falls Sie glauben, dass ich hier etwas zu viel über Katastrophen erzähle, möchte ich Sie auf einen Zwischenfall im Internet hinweisen, der kürzlich auftrat. Ich besuchte eine Webseite, deren mangelnder Inhalt mit der Tatsache entschuldigt wurde, dass die Server bei einem Einbruch gestohlen wurden. Glauben Sie immer noch, dass Ihnen so etwas nie passieren könnte?

Die Option REPLACE

Die Option REPLACE ist nützlich, da Sie Ihnen etwas über die Überprüfungen verrät, die normalerweise während einer Wiederherstellung durchgeführt werden. Falls der Name der Datenbank in der Sicherungskopie (z.B. *Buchhaltung*) nicht derselbe ist, wie der, den Sie wiederherstellen wollen (sagen wir *Verkauf*), wird die normale Wiederherstellung nicht ausgeführt. Dies soll Sie davor bewahren, sich nach der Wiederherstellung wie Homer Simpson anzuhören, der einen verzweifelten Schrei von sich gibt.

Die Option ist auch hilfreich, wenn die Dateisätze nicht übereinstimmen. Wenn sowohl die gesicherte als auch die wiederherzustellende Datenbank *Buchhaltung* heißen, aber die

(gerade laufende) Produktionsversion aus 20 Dateien der Größe 1 GB besteht und die Sicherungskopie die falsche Datenbank (sagen wir Ihre 20 MB große Testdatenbank) enthält, bemerkt SQL Server das und verhindert die Wiederherstellung.

Sie können dieses Verhalten in beiden Fällen durch Angabe der Option REPLACE abstellen, was dem SQL Server im Prinzip mitteilt, dass er sich gefälligst rauszuhalten habe – Sie wüssten schon was Sie tun.



Seien Sie sich absolut sicher, bevor Sie die Option REPLACE verwenden. Falls Sie eine Fehlermeldung wegen einer der hier angesprochenen Bedingungen erhalten, sollten Sie besser einen Schritt zurücktreten, tief Luft holen und noch einmal überdenken, was Sie eigentlich tun. Es ist besser, eine Minute zu warten, als erklären zu müssen, warum Sie vorsätzlich die falsche Datenbank in die Produktionsumgebung zurückgespielt haben.

Wiederherstellung differenzieller Sicherungen

Die Wiederherstellung einer differenziellen Sicherung funktioniert genauso wie die Wiederherstellung einer vollständigen Sicherung. Es gibt keinen Unterschied in der Syntax. Geben Sie einfach die richtige Dateinummer aus Ihrem Sicherungssatz an.

Sie können eine differenzielle Sicherung erst wiederherstellen, nachdem Sie eine vollständige Sicherungskopie wiederhergestellt haben. Sie müssen bei der vollständigen Wiederherstellung entweder die Option STANDBY oder NORECOVERY angeben, um Ihre differenziellen Sicherungen anwenden zu können.

Die differenziellen Sicherungen sind kumulativ; falls Sie also nach Ihrer letzten vollständigen Sicherung drei differenzielle Sicherungen angelegt haben, müssen Sie nur die letzte davon wiederherstellen.

Wiederherstellung von Transaktionsprotokollen

Die Wiederherstellung von Transaktionsprotokollen ist relativ unkompliziert, solange Sie ein paar Regeln beachten. Sie können Transaktionsprotokolle nur in der Reihenfolge wiederherstellen, in der sie gesichert wurden. Anders als differenzielle Sicherungen sind die Sicherungskopien der Transaktionsprotokolle nicht kumulativ, deshalb brauchen Sie für eine komplette Wiederherstellung alle Sicherungen in einem brauchbaren Zustand.

Anders als bei vollständigen oder differenziellen Wiederherstellungen, bei denen die Wiederherstellung nur bis zu dem Zeitpunkt möglich ist, zu dem Sie die Sicherung durchgeführt haben, können Sie mit der Wiederherstellung der Transaktionsprotokolle den Zustand der Datenbank zu jedem beliebigen Zeitpunkt wiederherstellen.

Wiederherstellung von Transaktionsprotokollen mit Transact-SQL

Um ein Transaktionsprotokoll wiederherzustellen, verwenden Sie die Anweisung **RESTORE LOG**.



```
RESTORE LOG databasename FROM <backup_device> [, ...n]
[WITH [RESTRICTED_USER] [[,] FILE = file_num]
[[,] PASSWORD = { pwd | @pwd_var } ]
[[,] MEDIANAME = { media_name | @media_name_variable}]
[[,] MEDIAPASSWORD = { mediapwd | @mediapwd_var } ]
[[,] MOVE 'logical_file_name' TO 'new_file_name'][,...n]
[[,] KEEP_REPLICATION]
[[,] {NORECOVERY | RECOVERY | STANDBY = undo_file_name}]
[[,] {NOUNLOAD | UNLOAD}]
[[,] { NOREWIND | REWIND } ]
[[,] RESTART]
[[,] STATS [= percentage]
[[,] STOPAT = datetime]
| [,] STOPATMARK = 'markname' [AFTER datetime]
| [,] STOPBEFOREMARK = 'markname' [AFTER datetime]]]
```

Diese Syntax verwendet die gleichen Optionen wie für eine vollständige Datenbankwiederherstellung, außer den folgenden Argumenten:

- **STOPAT** gibt das Datum und die Zeit für den Zeitpunkt an, den Sie als Konsistenzpunkt für die Wiederherstellung verwenden wollen. Für alle Transaktionen, die zu diesem Zeitpunkt aktiv waren (für die kein COMMIT ausgeführt wurde), wird ein Rollback ausgeführt.
- **STOPATMARK** kennzeichnet eine benannte Markierung (spezifiziert den markierten Eintrittspunkt), bei der zu stoppen ist.
- **STOPBEFOREMARK** erlaubt es, unmittelbar vor der benannten Markierung anzuhalten.

Nachdem Sie eine vollständige Sicherung wiederhergestellt (und dabei entweder die Option **NORECOVERY** oder **STANDBY** angegeben) haben, können Sie ein Transaktionsprotokoll folgendermaßen darauf anwenden (vorausgesetzt, dass das Sicherungsgerät *pubs_log_backup* existiert und eine oder mehrere Sicherungskopien von Transaktionsprotokollen enthält):

```
RESTORE LOG pubs from pubs_log_backup
```

Das ist alles; Sie haben das Transaktionsprotokoll wiederhergestellt. Der komplette Wiederherstellungsvorgang sowohl der vollständigen Sicherung als auch des Transaktionsprotokolls sieht folgendermaßen aus:

```
RESTORE DATABASE pubs FROM pubs_backup WITH NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH RECOVERY
```

RECOVERY ist die Standardoption, die Sie eigentlich nicht angeben müssen und die hier nur zur Verdeutlichung genannt wurde.

Als Beispiel nehmen wir nun an, dass Sie eine Sicherungskopie Ihrer Datenbank erstellt und daraufhin drei Sicherungen von Transaktionsprotokollen vorgenommen haben (mit den Dateinummern 1, 2 und 3 auf dem Sicherungsgerät *pubs_log_backup*). Um die Datenbank und alle Transaktionsprotokolle wiederherzustellen, führen Sie folgende Anweisungen aus:



```
RESTORE DATABASE pubs FROM pubs_backup WITH NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH FILE = 1, NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH FILE = 2, NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH FILE = 3, RECOVERY
```

Als Ergebnis erhalten Sie etwa folgende Ausgabe:



```
208 Seiten für die pubs-Datenbank, Datei 'pubs' auf Datei 1
verarbeitet.
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 1 verarbeitet.
RESTORE LOG hat 209 Seiten in 0,744 Sekunden erfolgreich
verarbeitet (2,291 MB/Sekunde).
2 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 1 verarbeitet.
RESTORE LOG hat 2 Seiten in 0,066 Sekunden erfolgreich
verarbeitet (0,186 MB/Sekunde).
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 2 verarbeitet.
RESTORE LOG hat 1 Seiten in 0,110 Sekunden erfolgreich
verarbeitet (0,032 MB/Sekunde).
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 3 verarbeitet.
RESTORE LOG hat 1 Seiten in 0,053 Sekunden erfolgreich
verarbeitet (0,019 MB/Sekunde).
```

Wiederherstellung des Zustands zu einem bestimmten Zeitpunkt

Um den Zustand einer Datenbank wiederherzustellen, der zu einem bestimmten Zeitpunkt bestand, verfahren Sie wie im vorigen Beispiel, außer bei dem Transaktionsprotokoll, das den Zeitpunkt enthält, den Sie erreichen wollen. Sie können **RESTORE HEADER-ONLY** verwenden, um herauszufinden, welchen Zeitraum die einzelnen Sicherungskopien der Transaktionsprotokolle umfassen. Um also dasselbe zu erreichen wie im vorigen Beispiel, aber zu einem bestimmten Zeitpunkt im dritten Transaktionsprotokoll anzuhalten, geben Sie folgende Anweisungen ein:



```
RESTORE DATABASE pubs FROM pubs_backup WITH NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH FILE = 1, NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH FILE = 2, NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH FILE = 3, RECOVERY,
STOPAT = 'Jun 3, 2000 4:53 PM'
```

Dieser Vorgang versetzt Ihre Datenbank in den konsistenten Zustand, der genau um 16:53 Uhr am 3. Juni 2000 bestanden hat.

Wiederherstellung bis zu einem markierten Transaktionsprotokolleintrag

Wie bereits erwähnt, können Sie die Wiederherstellung auch bis zu einem markierten Punkt in Ihrem Transaktionsprotokoll durchführen. Der Trick hierbei besteht darin, dass Sie zum Transaktionsnamen und nicht zum Markennamen (der eigentlich nur ein Kommentar ist, damit Sie wissen, welche Transaktion später kommt) wiederherstellen. Sehen Sie sich dazu folgendes Beispiel an:

```
exec sp_dropdevice 'pubs_backup', 'DELFILE'
exec sp_dropdevice 'pubs_log_backup', 'DELFILE'
go
exec sp_addumpdevice 'disk', 'pubs_backup',
    'f:\programme\microsoft sql server\mssql\backup\pubs_backup.bak'
exec sp_addumpdevice 'disk', 'pubs_log_backup',
    'f:\programme\microsoft sql server\mssql\backup\pubs_log_backup.bak'
-- Sicherungsdatei #1 erzeugen (vollständige Datenbanksicherung)
backup database pubs to pubs_backup with init
go
use pubs
create table table3 (c1 int not null)
BEGIN TRANSACTION tran1 WITH MARK 'Erste Marke'
insert table3 values (1)
COMMIT TRANSACTION tran1
-- Sicherungsdatei #1 erzeugen (Transaktionsprotokollsicherung)
backup log pubs to pubs_log_backup with noinit
create table table4 (c1 int not null)
insert table4 values (1)
-- Sicherungsdatei #2 erzeugen (Transaktionsprotokollsicherung)
backup log pubs to pubs_log_backup with noinit
insert table3 values (2)
BEGIN TRANSACTION tran2 WITH MARK 'Zweite Marke'
insert table3 values (3)
insert table4 values (3)
COMMIT TRANSACTION tran2
-- Sicherungsdatei #3 erzeugen (Transaktionsprotokollsicherung)
backup log pubs to pubs_log_backup with noinit
```

Dieses Skript erstellt eine Sicherungsdatei für ein Transaktionsprotokoll und eine Datenbanksicherung, sichert die Datenbank, erstellt einige Objekte und erzeugt in der Mitte zwei »markierte« Transaktionen: *tran1* und *tran2*. Dann können Sie einen dieser markierten Punkte wie bisher wiederherstellen. Dazu folgendes Beispiel:



```
restore database pubs from pubs_backup with norecovery
restore log pubs from pubs_log_backup with file = 1, norecovery
restore log pubs from pubs_log_backup with file = 2, norecovery
restore log pubs from pubs_log_backup with file = 3, recovery,
stopatmark = 'tran2'
```

Als Ergebnis erhalten Sie etwa Folgendes:



```
208 Seiten für die pubs-Datenbank, Datei 'pubs' auf Datei 1
verarbeitet.
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 1 verarbeitet.
RESTORE DATABASE hat 209 Seiten in 0,435 Sekunden erfolgreich
verarbeitet (3,919 MB/Sekunde).
2 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 1 verarbeitet.
RESTORE LOG hat 2 Seiten in 0,021 Sekunden erfolgreich
verarbeitet (0,438 MB/Sekunde).
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 2 verarbeitet.
RESTORE LOG hat 1 Seiten in 0,024 Sekunden erfolgreich
verarbeitet (0,149 MB/Sekunde).
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 3 verarbeitet.
RESTORE LOG hat 1 Seiten in 0,028 Sekunden erfolgreich
verarbeitet (0,036 MB/Sekunde).
```

Wiederherstellung mit Transaktionsprotokollen und differenziellen Sicherungen

Wenn Sie die Wiederherstellung differenzieller Sicherungen und die Wiederherstellung von Transaktionsprotokollen verbinden wollen, müssen Sie nur beachten, dass Sie alle Sicherungskopien der Transaktionsprotokolle benötigen, die nach der differenziellen Sicherung erstellt wurden.

Falls Sie im vorigen Beispiel nach der zweiten Sicherungskopie des Transaktionsprotokolls noch eine differenzielle Sicherung der Datenbank vorgenommen hätten, könnten Sie folgende Anweisungen ausführen:



```
RESTORE DATABASE pubs FROM pubs_backup WITH FILE = 1, NORECOVERY
RESTORE DATABASE pubs FROM pubs_backup WITH FILE = 2, NORECOVERY
RESTORE LOG pubs from pubs_log_backup WITH FILE = 1, RECOVERY
```

Als Ergebnis erhalten sie etwa folgende Ausgabe:



```

144 Seiten für die pubs-Datenbank, Datei 'pubs' auf Datei 1
verarbeitet.
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 1 verarbeitet.
RESTORE DATABASE hat 145 Seiten in 0,599 Sekunden erfolgreich
verarbeitet (1,971 MB/Sekunde).
48 Seiten für die pubs-Datenbank, Datei 'pubs' auf
Datei 2 verarbeitet.
1 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 2 verarbeitet.
RESTORE DATABASE hat 49 Seiten in 0,312 Sekunden erfolgreich
verarbeitet (1,263 MB/Sekunde).
3 Seiten für die pubs-Datenbank, Datei 'pubs_log' auf
Datei 1 verarbeitet.
RESTORE LOG hat 3 Seiten in 0,070 Sekunden erfolgreich
verarbeitet (0,307 MB/Sekunde).
  
```

Die Datenbank wird nun mit der vollständigen Sicherung wiederhergestellt, dann wird die differenzielle Sicherung angewandt und schließlich kommt nur die Transaktionsprotokoll-sicherung zur Anwendung, die nach der differenziellen Sicherung durchgeführt wurde: Transaktionsprotokollsicherung Nummer 3.

Dateien und Dateigruppen wiederherstellen

Sie können auch einzelne Dateien oder Dateigruppen wiederherstellen. Genau wie die gestrige Lektion enthält auch die heutige Lektion der Vollständigkeit halber die dazu nötige Syntax und ein Beispiel. Obwohl die Möglichkeit, Dateien und Dateigruppen wiederherzustellen, eine der nützlichen Funktionen von SQL Server darstellt, ist mit einer ausgefeilten Planung zu gewährleisten, dass die wiederhergestellten Dateien dann auch erfolgreich verwendet werden können. Dieses Thema geht aber über den Rahmen dieses Buches hinaus.



```

RESTORE DATABASE databasename <file_or_filegroup> [, ...n]
[FROM <backup_device> [, ...n]]
[WITH [RESTRICTED_USER] [[,] FILE = file_num]
[[,] PASSWORD = { pwd | @pwd_var }]
[[,] MEDIANAME = { media_name | @media_name_variable}]
[[,] MEDIAPASSWORD = { mediapwd | @mediapwd_var }]
[[,] NORECOVERY] [[,] {NOUNLOAD | UNLOAD}]
[[,] {NOWIND | REWIND}]
[[,] REPLACE][[,] RESTART] [[,] STATS [= percentage]]]
  
```



Wobei gilt:

```
<file_or_filegroup> :: =  
{FILE = {logical_file_name | @logical_file_name_var} |  
FILEGROUP = {logical_filegroup_name | @logical_filegroup_name_var}}
```

Der einzige Unterschied zur Syntax der vollständigen Wiederherstellung besteht darin, dass Sie in dieser Anweisung den Namen entweder einer Datei oder Dateigruppe angeben können.

Falls Sie die Datei *test_data* einer Datenbank namens *test* auf das Sicherungsgerät *test_file_backup* gesichert haben, können Sie sie mit folgender Anweisung wiederherstellen:

```
RESTORE DATABASE test FILE = 'test_data'  
FROM test_file_backup WITH NORECOVERY  
RESTORE LOG test  
FROM test_log_backup WITH RECOVERY
```



Sie können eine Datei oder Dateigruppe nicht wiederherstellen, ohne danach eine Sicherungskopie des Transaktionsprotokolls darauf anzuwenden, um die Datenbank in einen verwendbaren Zustand zu überführen. Dies ist nur als Hinweis auf die Komplexität der Planung für die Wiederherstellung von Dateien oder Dateigruppen gedacht.

Wiederherstellung mit dem SQL Server Enterprise Manager

Sie haben es geschafft! Nachdem Sie nun wissen, was hinter den Kulissen vor sich geht, wird es Zeit, dass Sie sich ansehen, wie einfach die Wiederherstellung einer Datenbank mit dem SQL Server Enterprise Manager ist.

Erweitern Sie den Ordner *Datenbanken*, klicken Sie mit der rechten Maustaste auf die Datenbank, die Sie wiederherstellen möchten, und wählen Sie ALLE TASKS / DATENBANK WIEDERHERSTELLEN. Jetzt taucht sicherlich die Frage bei Ihnen auf: Was ist, wenn die wiederherzustellende Datenbank nicht existiert?

Nachdem die Verbindung zum Server hergestellt ist, wählen Sie aus dem Menü EXTRAS den Befehl DATENBANK WIEDERHERSTELLEN. In beiden Fällen gelangen Sie zum gleichen Dialogfeld – Datenbank wiederherstellen (siehe Abbildung 8.3 mit einem Beispiel für die Datenbank *pubs*).

Abbildung 8.3 dokumentiert, dass SQL Server Enterprise Manager die Wiederherstellung zu einer trivialen Aufgabe macht. Wie Sie bereits erfahren haben, führt SQL Server Buch über die von Ihnen erstellten Sicherungskopien, indem er die dafür nötigen Informationen in Systemtabellen in der *msdb*-Datenbank ablegt. Diese Informationen spiegeln sich im Wiederherstellungsdialogfeld wider. Für jedes markierte Kästchen stellt SQL Server die

entsprechende Sicherung wieder her. Abbildung 8.3 zeigt, dass eine vollständige Sicherungskopie der Datenbank wiederhergestellt wird (erstes Kontrollkästchen), sofort gefolgt von der Wiederherstellung einer Sicherungskopie des Transaktionsprotokolls (die beiden nächsten Kontrollkästchen). Die Wiederherstellung der vollständigen Sicherungskopie wird mit der Option NORECOVERY durchgeführt, da Sie als Nächstes eine Wiederherstellung des Transaktionsprotokolls durchführen. Diese Wiederherstellung erfolgt mit der Option RECOVERY; Ihre Datenbank ist dann wieder verfügbar.

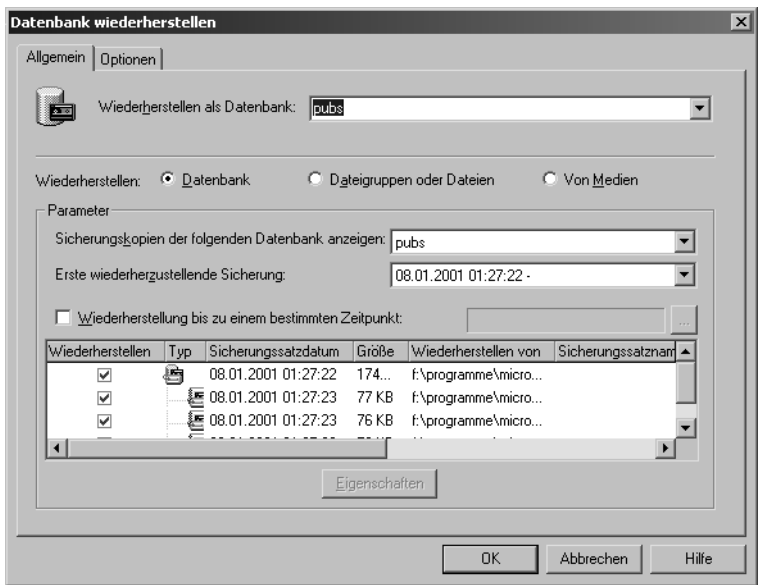


Abbildung 8.3:
Das Dialogfeld
Datenbank wieder-
herstellen

Um die Wiederherstellung einer Datei oder Dateigruppe durchzuführen, können Sie einfach die entsprechende Option anklicken. Um den Zustand einer Datei zu einem bestimmten Zeitpunkt (unter der Annahme, dass Sie ein oder mehrere Transaktionsprotokolle wiederherstellen wollen) wiederherzustellen, schalten Sie einfach das Kontrollkästchen WIEDERHERSTELLUNG BIS ZU EINEM BESTIMMTEN ZEITPUNKT ein.

Wenn Sie die Option Von Medien wählen, nimmt das Dialogfeld die in Abbildung 8.4 gezeigte Form an.

Um ein Sicherungsmedium auszuwählen, klicken Sie auf die Schaltfläche MEDIEN AUSWÄHLEN. Im Dialogfeld WIEDERHERSTELLUNGSMEDIEN WÄHLEN klicken Sie auf HINZUFÜGEN. Damit gelangen Sie zum Dialogfeld gemäß Abbildung 8.5. Wählen Sie eine Datei mit einer Datenbank- oder Transaktionsprotokollsicherung aus oder wählen Sie ein Sicherungsmedium.

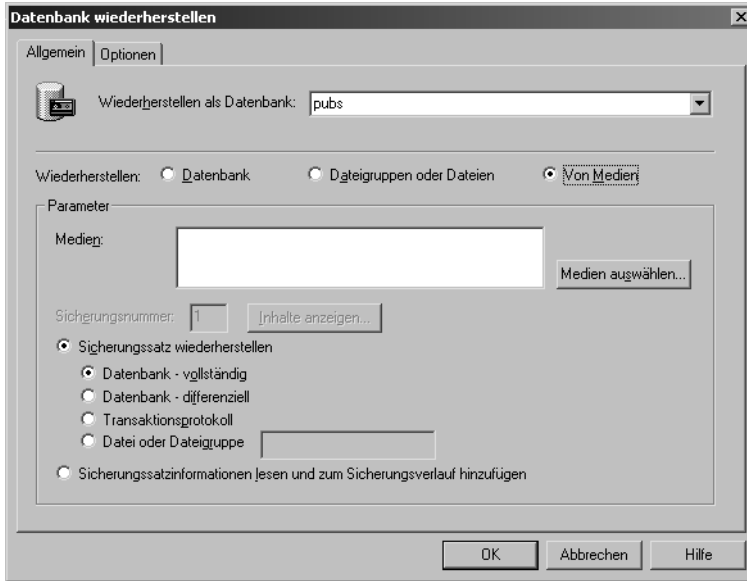


Abbildung 8.4:
Das Dialogfeld
Datenbank wieder-
herstellen, wenn
die Option Von
Medien ausge-
wählt ist.

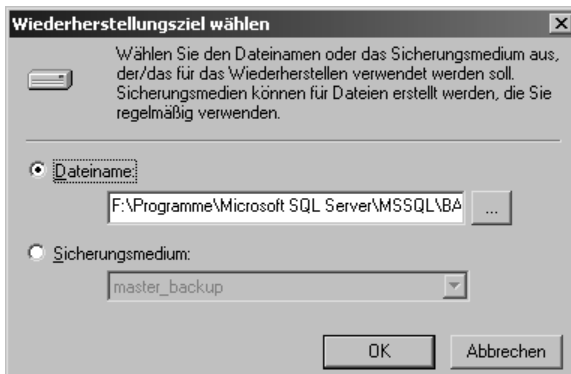


Abbildung 8.5:
Das Dialogfeld Wiederherstellungsziel
wählen

Nachdem Sie zum Dialogfeld DATENBANK WIEDERHERSTELLEN zurückgekehrt sind, geben Sie an, welche Sicherung Sie wiederherstellen wollen. Dann wählen Sie die Sicherungsnummer (entspricht dem Parameter FILE in der Anweisung RESTORE) aus. Eine komfortable Funktion bietet die Schaltfläche INHALTE ANZEIGEN. Dabei erhalten Sie eine Ausgabe ähnlich der LOAD HEADERONLY-Anweisung. Sie können hier jedoch, falls das Sicherungsmedium mehrere Sicherungen (zum Beispiel mehrere Sicherungen von Transaktionsprotokollen) enthält, die gewünschte Sicherung auswählen (siehe Abbildung 8.6).

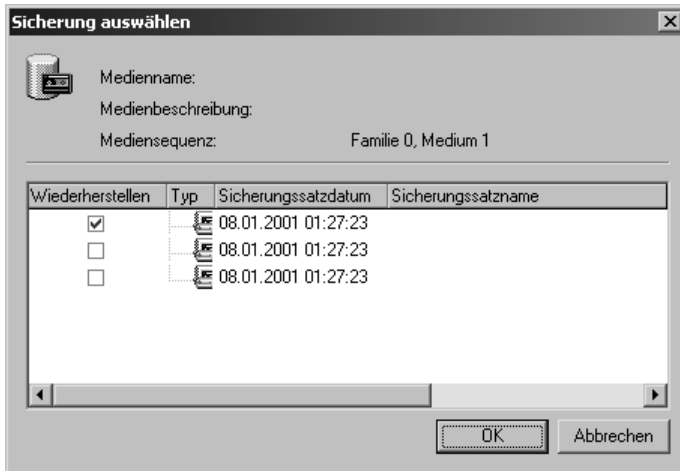


Abbildung 8.6:
Das Dialogfeld Sicherung
auswählen

Am unteren Rand der Registerkarte ALLGEMEIN des Dialogfelds DATENBANK WIEDERHERSTELLEN finden Sie die Option *Sicherungssatzinformationen lesen und zum Sicherungsverlauf hinzufügen*, mit der Sie die Sicherungssatzinformationen in der *msdb*-Datenbank von SQL Server aktualisieren können.

Um weitere Wiederherstellungsoptionen zu ändern (als ob Sie noch nicht genug davon gesehen hätten), gehen Sie auf die Registerkarte OPTIONEN (siehe Abbildung 8.7). Auf dieser Registerkarte sind folgende Optionen verfügbar:

- *Nach Wiederherstellen jeder einzelnen Sicherung gegebenenfalls Band auswerfen*: Diese Option macht genau das und wirft Ihre Bänder nach Beendigung der Wiederherstellung aus.
- *Bestätigung vor Wiederherstellen jeder einzelnen Sicherung*: Bei dieser Option erhalten Sie eine Benachrichtigung zwischen jeder Wiederherstellung, die Sie auf der Registerkarte ALLGEMEIN ausgewählt haben.
- *Wiederherstellung über vorhandene Datenbanken erzwingen*: Fügt die Option REPLACE zur Anweisung RESTORE DATABASE hinzu.
- *Datenbankdateien wiederherstellen als*: Hier können Sie jeder Datei aus Ihrem Sicherungssatz einen anderen Namen zuweisen. Dies entspricht der Option MOVE in der Anweisung RESTORE DATABASE.
- Der Abschnitt WIEDERHERSTELLUNGSABSCHLUSSSTATUS hat drei Optionen: Die erste, standardmäßig eingestellte, belässt Ihre Datenbank in einem ausführbaren Zustand (das entspricht der Angabe der Option RECOVERY beim letzten Wiederherstellungsvorgang).

- Falls Sie weitere differenzielle Sicherungen oder Transaktionsprotokolle auf die Datenbank anwenden wollen, wählen sie die mittlere Option. Sie entspricht der Angabe der Option NORECOVERY bei der Wiederherstellung des letzten Transaktionsprotokolls (oder der letzten vollständigen/differenziellen Wiederherstellung falls Sie keine Transaktionsprotokolle anwenden wollen).
- Die letzte Option entspricht der Angabe der Option STANDBY in der Wiederherstellungsanweisung. Wie bei der entsprechenden Transact-SQL-Anweisung müssen Sie auch hier den Namen und Speicherort für die Rückgängigdatei angeben.

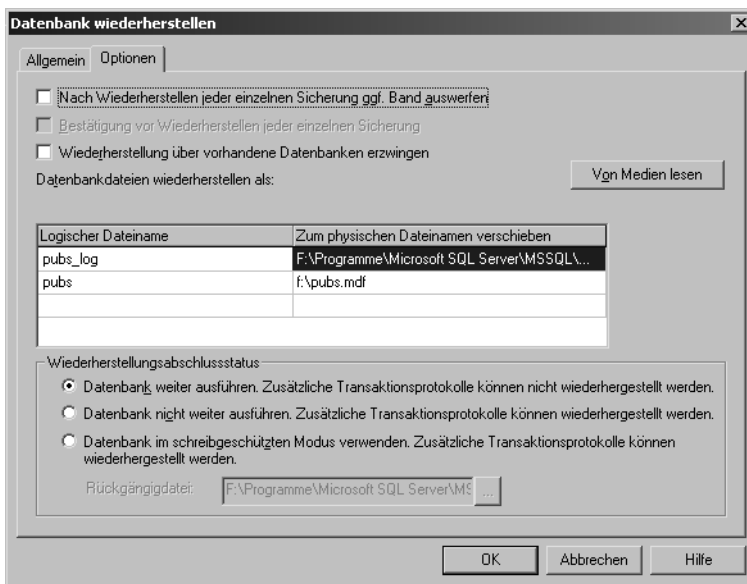


Abbildung 8.7:
Die Registerkarte
Optionen des Dia-
logfelds Datenbank
wiederherstellen

Wenn Sie bereit sind, mit der Wiederherstellung zu beginnen, klicken Sie auf OK. Der SQL Server Enterprise Manager wird daraufhin eine oder mehrere RESTORE-Anweisungen so ausführen, wie Sie sie in der grafischen Benutzeroberfläche festgelegt haben.

8.2 Wiederherstellungsszenarien

Zu Ihren wichtigsten Aufgaben gehört es, sich mit den verschiedenen Sicherungs- und Wiederherstellungstechniken vertraut zu machen, die Sie in den letzten beiden Lektionen kennen gelernt haben.

Wiederherstellung nach dem Ausfall einer Festplatte

Zu den gebräuchlichsten Wiederherstellungsszenarios gehört die Wiederherstellung einer Datenbank nach dem Ausfall einer Festplatte auf dem Server. Um Ihre Datenbank wiederherstellen zu können, führen Sie folgende Schritte aus:

1. Stellen Sie sicher, dass Sie alle zur Wiederherstellung Ihrer Geräte und Datenbanken benötigten Informationen haben.
2. Erstellen Sie eine Sicherungskopie Ihres Transaktionsprotokolls, wenn Sie darauf noch zugreifen können.
3. Löschen Sie die betroffenen Datenbanken.
4. Ersetzen Sie die defekte Festplatte.
5. Stellen Sie die entsprechenden Datenbanken und die Transaktionsprotokolle wieder her.

Sehen Sie sich die einzelnen Schritte einmal genauer an:

Stellen Sie sicher, dass Sie alle zur Wiederherstellung Ihrer Geräte und Datenbanken benötigten Informationen haben

Sie sollten die zur Wiederherstellung aller Datenbanken auf der betroffenen Festplatte benötigten CREATE DATABASE-Anweisungen kennen. Die gute Nachricht ist, dass Sie unter SQL Server 2000 diese Informationen mit den heute besprochenen Anweisungen (RESTORE HEADERONLY, RESTORE FILELISTONLY) erhalten können, solange Sie Ihre Sicherungskopien (oder Bänder) finden können. In SQL Server 2000 sind jetzt alle Informationen, die Sie zur Wiederherstellung Ihrer Sicherungskopien benötigen, in diesen Sicherungskopien enthalten.

Erstellen Sie eine Sicherungskopie Ihres Transaktionsprotokolls, wenn Sie darauf noch zugreifen können

Falls Ihre Datenbankdateien aus irgendeinem Grund unbrauchbar geworden sind, die Transaktionsprotokolle jedoch (auf einer anderen Festplatte) noch verfügbar sind, können Sie die Anweisung BACKUP LOG mit der Option NO_TRUNCATE ausführen. Dem liegt das Konzept zugrunde, eine Sicherungskopie der seit dem letzten Backup aufgetretenen Einträge zu erhalten, so dass Sie diese Protokolleinträge wiederherstellen können, wenn Sie für die Wiederherstellung Ihrer Datenbank bereit sind.



Löschen Sie die Datenbank

Da die Datenbank fehlerverdächtig oder teilweise verloren ist, können Sie sie löschen und dann für die Wiederherstellung vorbereiten. Verwenden Sie die Anweisung `DROP DATABASE`, um die Datenbank und alle zugehörigen Dateien vom Server zu löschen. Beachten Sie, dass `DROP DATABASE` ausgeführt werden kann, obwohl die Datenbank als fehlerverdächtig markiert ist.

Ersetzen Sie die defekte Festplatte

Dieser Schritt ist wohl selbstverständlich. Sie müssen jedoch alle RAID-Einstellungen zurücksetzen, bevor Sie Windows 2000 neu starten. Falls Sie Windows 2000-RAID verwenden, müssen Sie die RAID-Einstellungen in der Datenträgerverwaltung neu festlegen, bevor Sie den nächsten Schritt ausführen. Weitere Informationen zu Windows 2000-RAID finden Sie in der Windows 2000-Dokumentation; zu anderen Verfahren informieren Sie sich bitte in Ihrer Hardwaredokumentation.

Erstellen Sie die verlorene Datenbank neu (optional)

Falls Sie es wollen, können Sie die Datenbank neu erstellen. Dieser Schritt kann auch weggelassen werden. Der Wiederherstellungsprozess kann die Datenbank für Sie neu erstellen.

Führen Sie die Wiederherstellung einer vollständigen Sicherung der Datenbank, der vorhandenen differenziellen Sicherung und der benötigten Sicherungen der Transaktionsprotokolle durch

Nach allem, was Sie in dieser Woche gelernt haben, sollte dieser Schritt ebenfalls selbstverständlich sein. Stellen Sie Ihre vollständige Datenbanksicherung sowie Ihre letzte differenzielle Sicherung wieder her und wenden Sie dann Ihre Transaktionsprotokolle (einschließlich der letzten Sicherung, die Sie durchgeführt haben, nachdem die Festplatte ausgefallen war) darauf an.

Führen Sie `DBCC CHECKDB` aus, um eine funktionsfähige Wiederherstellung sicherzustellen (optional)

Dieser Schritt kann zwar weggelassen, dennoch ist er zu empfehlen. Führen Sie `DBCC CHECKDB` für jede wiederhergestellte Datenbank aus, um sicherzugehen, dass die Wiederherstellungsoperationen einen funktionsfähigen Stand der Datenbanken hinterlassen haben. Falls dies aus irgendeinem Grund nicht der Fall sein sollte, müssen Sie mit einer früheren Sicherungskopie von vorn beginnen.



Wie Sie sehen können, ist das Wiederherstellen von Datenbanken zwar nicht übermäßig kompliziert, aber auch keine triviale Aufgabe. Deshalb sollten Sie die Wiederherstellung von Zeit zu Zeit üben, so dass Sie mit jedem Schritt vertraut sind, wenn statt des Testservers Ihr Produktionsserver ein Problem hat.

Wiederherstellung nach dem Verlust der master-Datenbank

Die Anweisung `RESTORE` funktioniert hervorragend, solange Sie Ihre *master*-Datenbank nicht verlieren. Was dann? Es gibt eigentlich zwei Arten, wie Sie vom Verlust Ihrer *master*-Datenbank betroffen werden können. Davon hängt dann ab, was Sie zur Behebung des Fehlers tun können.

Falls Ihre Datenbank oder Ihr Transaktionsprotokoll beschädigt wurden oder ein logisches Problem besteht (dass z.B. jemand alle Benutzerkonten gelöscht hat), SQL Server sich aber noch starten lässt, können Sie die *master*-Datenbank einfach wiederherstellen. Das andere – und ehrlich gesagt häufigere – Szenario besteht darin, dass Sie mit dem Verlust der *master*-Datenbank auch Ihren Server verlieren. Deshalb ist es wirklich empfehlenswert (wie es Tag 20 noch demonstriert), die Festplatten mit der *master*-Datenbank und den Transaktionsprotokollen zu spiegeln.

Wiederherstellung der master-Datenbank

Zunächst das einfache Szenario: Um Ihre *master*-Datenbank von einer vollständigen Sicherungskopie wiederherzustellen, starten Sie SQL Server im Einzelbenutzermodus. Am einfachsten lässt sich das von der Eingabeaufforderung des Computers erledigen, auf dem SQL Server läuft. Öffnen Sie eine Eingabeaufforderung und führen Sie die nachstehende Sequenz von Befehlen aus:



```
NET STOP MSSQLSERVER
```

Dieser Befehl fährt den MSSQLServer-Dienst herunter (für eine Standardinstanz). Um eine benannte Instanz von SQL Server herunterzufahren, geben Sie `MSSQL$Instanzname` an (zum Beispiel `MSSQL$HANDEL` für die Beispielinstallation »HANDEL«).

Als Nächstes starten Sie SQL Server mit dem Schalter für den Einzelbenutzermodus (`-m`):



```
SQLSERV.EXE -m -SHANDEL
```

Damit wird SQL Server als Anwendung gestartet, die in diesem Kommandozeilenfenster läuft. Schließen Sie dieses Fenster nicht und geben Sie hier auch nichts ein. Das ist jetzt Ihr SQL Server. Wenn der SQL Server aufhört, Text auszuspuken (der nicht nur zufällig den Fehlerprotokollen ähnelt, die Sie in dieser Lektion bereits kennen gelernt haben), ist er bereit, eingesetzt zu werden.

Um eine benannte Instanz zu starten, müssen Sie sie mit der Option `-s` starten, wie es Tag 2 erläutert hat. Die Instanz `HANDEL` starten Sie dann folgendermaßen:



```
SQLSERVER.EXE -m -SHANDEL -SHANDEL
```

Als Nächstes starten Sie Ihr bevorzugtes Abfrageprogramm (oder den SQL Server Enterprise Manager) und stellen Ihre *master*-Datenbank wieder her, wie Sie es auch mit jeder anderen vollständigen Sicherung durchführen würden. Nach Abschluss des Wiederherstellungsvorgangs beendet sich SQL Server selbsttätig. Starten Sie den Dienst einfach neu, um zur normalen Arbeitsweise zurückzukehren. Sie können den Dienst folgendermaßen neu starten:



```
NET START MSSQLServer
```

oder



```
NET START MSSQL$HANDEL
```



Die *master*-Datenbank lässt sich nur im Einzelbenutzermodus wiederherstellen. Andernfalls werden der SQL Server Enterprise Manager und die Abfragewerkzeuge Sie daran hindern und Sie erhalten etwa folgende Fehlermeldung:

```
RESTORE DATABASE muss im Einzelbenutzermodus verwendet werden, um die
master-Datenbank wiederherzustellen.
```

```
RESTORE DATABASE wird fehlerbedingt beendet.
```

Sie müssen die Wiederherstellung der *master*-Datenbank beenden und SQL Server normal neu starten, bevor Sie irgend eine andere Datenbank wiederherstellen können.

Alle Änderungen, die Sie an Ihrer master-Datenbank nach der letzten vollständigen Sicherungskopie vorgenommen haben, müssen Sie jetzt erneut vornehmen – ein weiterer guter Grund regelmäßige Sicherungen Ihrer master-Datenbank durchzuführen.

Wiederherstellung der master-Datenbank

Die andere Möglichkeit ist, dass Ihre *master*-Datenbank nicht verfügbar ist, nicht funktioniert und Sie es buchstäblich mit einem toten Server zu tun haben. Wie einer meiner Lieblingsautoren zu sagen pflegt: »Keine Panik!« Sie kommen da durch, und werden vielleicht sogar der Held der Abteilung.

Wenn SQL Server nicht hochfährt, weil Ihre *master*-Datenbank gestorben ist, heißt das üblicherweise, dass die Festplatte (oder Festplatten), auf denen sich die Dateien der *master*-Datenbank befanden, defekt ist. Der erste Schritt besteht darin, das Dienstprogramm Rebuild Master (*rebuilddm.exe*) auszuführen, das Sie im Verzeichnis `\Programme\Microsoft SQL Server\80\Tools\binn` finden.

Dieses Dienstprogramm geht davon aus, dass SQL Server nicht läuft. Beim Start erscheint das Dialogfeld gemäß Abbildung 8.8.

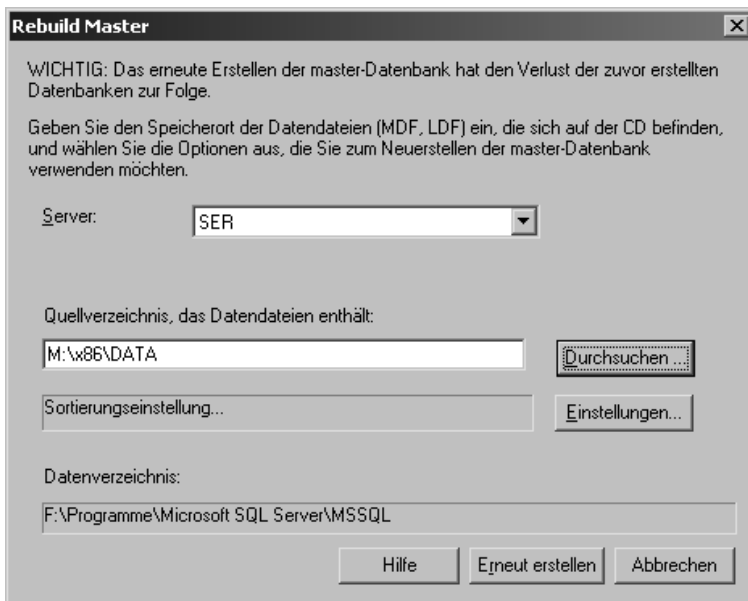


Abbildung 8.8:
Das Dienstpro-
gramm Rebuild
Master

Klicken Sie auf DURCHSUCHEN, um zum Verzeichnis `\data` der Installations-CD zu navigieren. Klicken Sie dann auf Einstellungen, um die Sortierungseinstellung zu ändern, falls Sie bei der Installation von SQL Server nicht die Standardeinstellungen übernommen haben (siehe Abbildung 8.9).

Nachdem Sie die Optionen eingestellt haben, klicken Sie auf OK. Im Dialogfeld REBUILD MASTER klicken Sie auf ERNEUT ERSTELLEN, damit SQL Server die Originaldatenbank von Ihrer CD kopiert. Dabei werden alle Dateien der Datenbanken *master*, *msdb*, *model*, *pubs* und *northwind* kopiert. Daraufhin wird SQL Server gestartet, der Ihre *tempdb*-Datenbank erneut erstellt und alle erforderlichen Umwandlungen vornimmt, die den festgelegten Sortierungseinstellungen entsprechen. Zum Abschluss wird Ihnen mitgeteilt, dass das erneute Erstellen beendet ist.

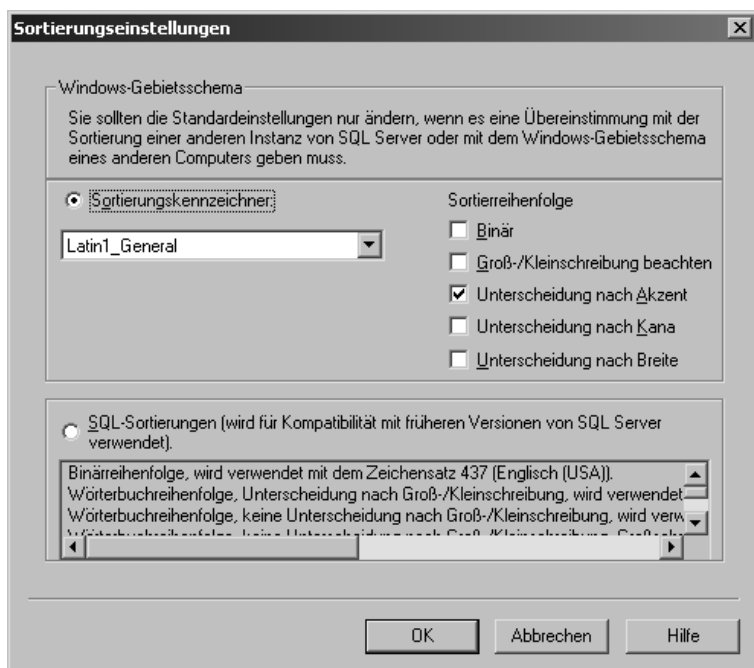


Abbildung 8.9:
Auswahl der
gewünschten
Wiederher-
stellungsoptionen

Wiederherstellung der master- und der anderen Systemdatenbanken

Nachdem Ihr Server jetzt wieder läuft, stellen Sie wie bereits beschrieben Ihre *master*-Datenbank aus den Sicherungskopien wieder her. Danach müssen Sie die *model*-Datenbank (falls Sie Änderungen an ihr vorgenommen hatten), die *msdb*-Datenbank und, falls Ihr Server zur Replikation verwendet wird, die distribution-Datenbank wiederherstellen.

Anhängen oder Wiederherstellen der übrigen Datenbanken (falls nötig)

Zuletzt stellen Sie alle anderen betroffenen Datenbanken wieder her. Falls die Datenbankdateien noch existieren, können Sie die Anweisung `CREATE DATABASE FOR ATTACH` oder die gespeicherte Systemprozedur `sp_attach_db` verwenden, um die Datenbanken erneut an SQL Server anzuhängen.

Durchführung von DBCC-Überprüfungen für den kompletten Server

Zuletzt müssen Sie nach einem solchen Absturz `DBCC CHECKDB` für jede Datenbank ausführen, um sicher sein zu können, dass Ihr Server wieder fehlerfrei läuft.

8.3 Zusammenfassung

Sicherung und Wiederherstellung von Daten sind wahrscheinlich die wichtigsten Aufgaben bei der Wartung Ihres SQL Server-Systems. Wie Sie in der heutigen Lektion gesehen haben, kann die Wiederherstellung recht unkompliziert sein, wenn Sie sie mit dem SQL Server Enterprise Manager durchführen, oder etwas komplizierter, wenn es darum geht, Ihre *master*-Datenbank wiederherzustellen. Der einzige Rat, den ich Ihnen geben kann, ist es, Sicherungen und Wiederherstellungen regelmäßig auszuprobieren. Das ist die einzige Möglichkeit, um mit diesen Vorgängen vertraut zu werden.

8.4 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.



Fragen & Antworten

- F** *Wie heißt die Anweisung für die Wiederherstellung einer Datenbank von einer Sicherung?*
- A** RESTORE DATABASE.
- F** *Welche Anweisung benutze ich, um eine differenzielle Sicherung der master-Datenbank wiederherzustellen?*
- A** Sie können keine anlegen, deshalb können Sie auch keine wiederherstellen. Sie können nur vollständige Sicherungskopien der master-Datenbank anlegen.
- F** *Welche Anweisung stellt ein Transaktionsprotokoll wieder her, ohne die Datenbank in einen ausführbaren Stand zu versetzen?*
- A** Die Anweisung RESTORE LOG WITH NORECOVERY.

Kontrollfragen

1. Welche weiteren Datenbanken müssen Sie wiederherstellen, wenn Sie Ihre *master*-Datenbank neu erzeugen?
2. Was erhalten Sie als Ergebnis der Anweisung `RESTORE FILELISTONLY`?
3. Können Sie eine mit mehreren Bandlaufwerken erstellte Sicherung auf einem nur mit einem Bandlaufwerk ausgestatteten Server wiederherstellen?
4. Welche Konfigurationsoption verwenden Sie, um die automatische Wiederherstellung zu beeinflussen?
5. Wie würden Sie die Lohntabelle wiederherstellen, falls Ihr Vorgesetzter sie versehentlich gelöscht hat?

Übungen

1. Erstellen Sie eine Testdatenbank, sichern Sie sie und stellen Sie die Datenbank wieder her. Führen Sie dabei eine vollständige Datenbanksicherung durch und versuchen Sie auch, die Transaktionsprotokolle wiederherzustellen. Führen Sie zwischen den einzelnen Sicherungen der Transaktionsprotokolle Änderungen an Ihrer Datenbank durch; erstellen Sie zum Beispiel neue Tabellen. Wenden Sie bei der Wiederherstellung die einzelnen Transaktionsprotokolle an, und überprüfen Sie, ob bei jeder Wiederherstellung nur die richtigen Änderungen durchgeführt werden.

2. Sichern Sie Ihre Datenbank, führen Sie einige Änderungen aus und erstellen Sie dann eine differenzielle Sicherung. Erstellen Sie schließlich nach einigen weiteren Änderungen eine Sicherung des Transaktionsprotokolls. Stellen Sie die Sicherungen unter ausschließlicher Verwendung von Transact-SQL-Anweisungen wieder her und überprüfen Sie, ob die Wiederherstellung erfolgreich war.
3. Wiederholen Sie Übung 2. Halten Sie aber nach der Wiederherstellung der vollständigen Sicherungskopie bei jeder weiteren Wiederherstellung an, indem Sie die `STANDBY` Option verwenden. Untersuchen Sie den Zustand Ihrer Datenbank, damit Sie die Abläufe verstehen und überprüfen können, ob für Ihre Änderungen tatsächlich Rollforwards und Rollbacks ausgeführt werden. Bei der Wiederherstellung des letzten Transaktionsprotokolls verwenden Sie natürlich die Option `WITH RECOVERY`.



Datentypen und Tabellen



In der gestrigen Lektion haben Sie gelernt, wie Sie Ihre Datenbanken aus Datensicherungen wiederherstellen können. Jetzt können Sie sich endlich damit beschäftigen, wie Sie Tabellen erstellen und Daten in Ihre Datenbank einfügen.

Als Erstes erhalten Sie einen allgemeinen Überblick über Tabellen und lernen die unterschiedlichen Datentypen kennen. Anschließend lernen Sie, wie Sie diese Elemente einsetzen, um Tabellen in einer Datenbank zu erstellen und zu ändern. Dabei erfahren Sie auch, welche Möglichkeiten der grafischen Codeerstellung SQL Server mit dem Enterprise Manager bietet.

9.1 Tabellen

Tabellen enthalten sämtliche in einer Datenbank gespeicherten Daten und sind in Form von Zeilen und Spalten (bzw. Datensätzen und Feldern) organisiert. Jede Spalte in einer Tabelle kann einen ganz bestimmten Typ von Information bzw. einen Datentyp speichern.

Jede Tabelle bildet eine Entität in der Datenbank, wobei jede einzelne Zeile in der Tabelle ein Auftreten dieser Entität repräsentiert. Die Spalten in der Tabelle beschreiben dabei die Attribute dieser logischen Entität. Nehmen wir zum Beispiel an, dass Sie eine Tabelle für Mitarbeiter erstellen: Die einzelnen Mitarbeiter bilden dann eine logische Entität, während eine einzelne Zeile in der Tabelle *Mitarbeiter* eine einzelne Instanz eines Mitarbeiters darstellt. Die Spalten, aus denen die Tabelle besteht, beschreiben den jeweiligen Mitarbeiter. Zu den Spalten könnten beispielsweise die Mitarbeiter-ID, der Vorname, der Nachname, die Sozialversicherungsnummer oder Ähnliches gehören.

9.2 Spalten

Jeder Spalte (oder jedem Feld) in einer Tabelle ist ein Name, ein Datentyp, eine Länge (optional) und ein Nullwertstatus zuzuweisen. Die Spalten können Sie in Ihrer Tabellendefinition in beliebiger Reihenfolge anordnen. Jede Spalte muss weiterhin innerhalb der Tabelle einen eindeutigen Namen besitzen und den Regeln für SQL Server-Bezeichner entsprechen (siehe den folgenden Abschnitt).

Eine einzelne Datenbank kann bis zu 2 Milliarden Tabellen aufnehmen, von denen jede einzelne bis zu 1024 Spalten enthalten darf. Die maximale Länge jeder Tabellenzeile beträgt 8192 Byte. Davon ist ein Bereich abzuziehen, der für die Verwaltung der gespeicherten Zeilen erforderlich ist. Die Größe dieses Bereiches hängt von den verschiedensten Gegebenheiten ab, deren Erörterung allerdings den Rahmen dieses Buches überschreiten würde. Letztendlich bleiben 8060 Bytes pro Zeile für Daten übrig.

Die maximale Länge einer einzelnen Spalte beträgt 8000 Bytes, wobei Sie bis zu 1024 Spalten in eine Zeile aufnehmen können. Eine Zeile darf allerdings nicht die Gesamtmenge einer Datenseite (8192 Byte) für Daten und den erforderlichen Overhead übersteigen. Eine Ausnahme hiervon bilden nur Text- und Grafikdatentypen, die einen Speicherplatz von bis zu 2 GB für Text- oder binäre Daten möglich machen. Weil diese Datentypen physikalisch nicht als Teil einer Tabellenzeile gespeichert werden, unterliegen sie auch nicht der einschränkenden Richtlinie, dass eine Datenzeile als Ganzes in eine einzelne Datenseite passen muss. Später in der heutigen Lektion erhalten Sie weitere Informationen zu Spaltenlängen und dem Verwaltungsaufwand für Tabellen und Zeilen.



In einer Datenseite lassen sich maximal 8060 Datenbytes speichern. Mit anderen Worten beträgt der minimale Overhead für jede Datenseite 132 Bytes.

9.3 SQL Server-Bezeichner

Sämtliche Tabellennamen sowie die Spaltennamen und überhaupt die Namen aller Objekte müssen den nachfolgend aufgeführten Regeln für SQL Server-Bezeichner entsprechen:

- Bezeichner können aus 1 bis 128 Unicode-Zeichen einschließlich Buchstaben, Symbolen und Zahlen bestehen.
- Das erste Zeichen muss ein Buchstabe oder eines der folgenden Symbole sein: @, # oder _ (Unterstrich). Die Zeichen @ und # besitzen in SQL Server eine besondere Bedeutung.
- Nach dem ersten Zeichen können Sie eines der folgenden Symbole verwenden: #, \$ und _.
- Bezeichner, die mit dem Symbol @ beginnen, werden als lokale Variablen behandelt. Beachten Sie, dass das Symbol @ nur als das erste Zeichen in einem Bezeichner auftreten darf.
- Bezeichner, die mit dem Symbol # beginnen, weisen darauf hin, dass das Objekt ein temporäres Objekt ist. Der Benutzer kann dieses Objekt während der Sitzung referenzieren.
- Bezeichner, die mit den Symbolen ## beginnen, weisen darauf hin, dass das Objekt ein globales temporäres Objekt darstellt. Alle Benutzer der Datenbank können dieses Objekt während der Sitzung referenzieren.



- Innerhalb Ihrer Bezeichner können Sie zwar Leerzeichen verwenden, müssen dann aber bei der Referenzierung dieser Objekte doppelte Anführungszeichen oder eckige Klammern verwenden. So müssen Sie beispielsweise auf eine Tabelle namens *Mitarbeiter Rente* in der Datenbank entweder als »Mitarbeiter Rente« oder [Mitarbeiter Rente] Bezug nehmen. Wenn Sie durchaus nicht auf Leerzeichen verzichten wollen, sollten Sie die Schreibweise mit eckigen Klammern wählen, damit Sie SQL Server-Bezeichner nicht mit Zeichenfolgen für Konstanzen verwechseln.



Obwohl Symbole und Leerzeichen in SQL Server-Bezeichnern erlaubt sind, wird deren Einsatz von den meisten Entwicklern vermieden, da irreführende SQL-Anweisungen die Folge sein können.

Bezeichner	Bemerkung
1001ArabianNights	Ungültiger Bezeichner. Bezeichner müssen mit einem Buchstaben beginnen.
@Addresses	Dieser Bezeichner ist nur für das Erstellen einer Variablen gültig.
Table@Address	Ungültiger Bezeichner. Das Symbol @ darf nur als erstes Zeichen in einem Bezeichner auftauchen.
#tblCities	Dieser Bezeichner ist nur dann gültig, wenn er zum Erstellen eines temporären Objekts verwendet wird, das dem Benutzer zur Verfügung steht, der es erstellt hat.
##tblCities	Dieser Bezeichner ist nur dann gültig, wenn Sie ein temporäres Objekt erstellen, das allen Benutzern zur Verfügung steht.
tblEmployee	Gültiger Bezeichner. Ein Präfix für Bezeichner (wie <i>tbl</i> für <i>Table</i>) ist zwar nicht erforderlich, macht Ihre SQL-Anweisungen aber verständlicher, weil das Präfix darauf hinweist, wofür der Bezeichner steht. Andere übliche Bezeichner dieser Art sind beispielsweise <i>qry</i> für <i>Query</i> (Abfrage), <i>tr</i> für <i>Trigger</i> , <i>sp_</i> für gespeicherte Systemprozeduren (stored procedure).
Titles	Dies ist ein gültiger Bezeichner.
Author_Review	Dies ist ein gültiger Bezeichner.
AuthorReview	Dies ist ein gültiger Bezeichner.

Tabelle 9.1: Beispiele für gültige und ungültige Bezeichner

Datentypen

Der Datentyp legt fest, welche Art Information (numerisch, Zeichen usw.) eine bestimmte Spalte enthält und wie viel Speicherplatz diese Information beansprucht. Manche Datentypen besitzen eine variable Länge, während andere wiederum von fester Länge sind. Die

Datentypen von SQL Server können in die folgenden Gruppen von Spalten eingeordnet werden: Zeichenfolge, Unicode-Daten, Binärzeichenfolgen, ganze Zahlen, ungefähre bzw. genaue numerische Werte, besondere, Datum und Zeit, Währung, automatisch generierte Zahlen, Synonyme, benutzerdefinierte Datentypen und berechnete Spalte. Tabelle 9.2 zeigt die verschiedenen Datentypen, die in SQL Server 2000 zulässig sind.

Gruppe	Datentyp	Bemerkung
Zeichenfolge	char(n), varchar(n)	Speichert Zeichenfolgen.
Binärzeichenfolgen	binary(n), varbinary(n)	Speichert binäre Daten in 2-Byte-Paaren.
Ganze Zahlen	int, smallint, bigint, tinyint	Speichert ganzzahlige Werte.
Ungefähre numerische Werte	float, real	Speichert ungefähre numerische Werte.
Genaue numerische Werte	decimal, numeric	Speichert genaue numerische Werte.
Besondere	bit, text, image, sql_variant, table	Speichert ein einzelnes Bit, Zeichendaten größer als 8000 Byte oder Bilddaten. sql_variant nimmt alle Datentypen auf und table speichert Tabellen.
Datum und Zeit	datetime, smalldatetime	Speichert Datums- und Zeitwerte.
Währungswerte	money, smallmoney	Speichert Währungswerte.
Automatisch generierte Zahlen	rowversion (vormals timestamp)	Speichert Werte von Datentypen, die automatisch vom SQL Server erstellt oder inkrementiert werden.
Synonyme	(Siehe Tabelle 9.9)	Nimmt die Zuordnung zwischen ANSI-Datentypen und SQL Server-Datentypen vor.
Benutzerdefinierte Datentypen		Datentypen, die Sie zur Speicherung von Daten selbst definieren können.
Berechnete Spalten		Speichert den Ausdruck, der zur Berechnung der Spalte verwendet wird. Beachten Sie, dass hier nicht die Ergebnisdaten, sondern lediglich die Ausdrücke gespeichert werden, die die Ergebnisdaten erzeugen.
Unicode	nchar, ntext, nvarchar	Speichert Daten im Unicode-Format (ein Doppelbyte pro gespeichertem Zeichen).

Tabelle 9.2: Unterstützte Datentypen des SQL Servers 2000

NULL-Zulässigkeit

Die NULL-Zulässigkeit einer Spalte gibt an, ob für diese Spalte ein Eingabewert erforderlich ist. Wenn Sie eine Spalte zulassen möchten, deren Wert unbekannt ist, dann spezifizieren Sie an dieser Stelle `NULL`. Wenn Sie dagegen wünschen, dass in jeder Zeile in dieser Spalte ein Wert enthalten sein muss, spezifizieren Sie `NOT NULL`. Falls Sie weder `NULL` noch `NOT NULL` angeben, wird der Standardwert der Datenbank (bzw. der jeweiligen SQL Server-Verbindung) verwendet. Bei der erstmaligen Installation von SQL Server wird als Standardwert jeder Datenbank festgelegt, dass alle Spalten mit der Option `NOT NULL` erstellt werden. Wie die meisten SQL Server-Optionen lässt sich auch dieser Standardwert ändern, sogar getrennt nach Verbindung. Es ist daher empfehlenswert, immer vorher anzugeben, ob eine Spalte das Attribut `NULL` oder `NOT NULL` haben soll.

Sortierung

Man kann auch die Sortierung für eine einzelne Datenspalte in einer Tabelle spezifizieren. Wenn Sie keine Sortierung angeben, wird der Spalte die Standardsortierung der Datenbank zugewiesen.

9.4 Zeichenfolgen

Zeichenfolgen enthalten Zeichendaten, die aus Buchstaben, Zahlen und Symbolen bestehen können. Zeichendaten speichern Sie unter Verwendung der Schlüsselwörter `char(n)` oder `varchar(n)` mit fester bzw. variabler Länge. In diesen Datentypen können 8000 Bytes gespeichert werden.

Beim Erstellen eines Feldes mit fester Länge legen Sie fest, dass dieses Feld immer nur n Bytes an Daten enthalten soll. Sind die in das Feld eingegebenen Daten kleiner als n , wird es mit Leerzeichen aufgefüllt, so dass es immer diese n Bytes enthält. Wenn Sie versuchen, mehr als diese n Bytes in das Feld einzugeben, werden die Daten abgeschnitten. Tabelle 9.3 zeigt einige Beispiele für die Eingabe von Daten in ein Feld, das als `Nname char(8)` definiert ist (im Beispiel weist das Zeichen `*` auf ein Leerzeichen hin).

Eingegebene Daten	Nname enthält jetzt:
Lawrence	Lawrence
Mark Anthony	Mark Ant
Denise	Denise**

Tabelle 9.3: Zeichenfelder fester Länge

Bei Verwendung von Feldern variabler Länge spezifizieren Sie die maximale Länge des Feldes. Anders als bei den Feldern fester Länge werden jedoch Felder mit variabler Länge nicht mit Leerzeichen aufgefüllt, was Ihre Datenbank zwar hinsichtlich des verwendeten Speicherplatzes effizienter macht, aber einen gewissen Preis hinsichtlich der Leistung kostet. Ein Feld, das als variabel lang definiert wurde, muss von SQL Server daraufhin untersucht werden, wo das Feld endet und das nächste Feld beginnt. Wegen der zusätzlichen Bytes, die den Zeilen und der Tabelle hinzugefügt werden müssen, hat dies einen erhöhten Verwaltungsaufwand zur Folge. Der Datentyp `varchar` ist insbesondere dann nützlich, wenn Sie davon ausgehen müssen, dass die Daten im Feld in der Länge stark variieren oder wenn Sie `NULL`-Werte in Ihrem Feld zulassen wollen.

Wenn Sie Zeichendaten in SQL Server eingeben, müssen Sie die Daten in einfache oder doppelte Anführungszeichen einschließen. Zu empfehlen sind an dieser Stelle einfache Anführungszeichen, damit es keine Verwechslungen zwischen Zeichenfolgenkonstanten und Bezeichnern von SQL Server gibt. Zur Eingabe von `NULL` in ein Feld von SQL Server verwenden Sie das Schlüsselwort `NULL` ohne Anführungszeichen.

Listing 9.1 enthält ein Beispiel für die Erstellung einer Tabelle unter Verwendung der Schlüsselwörter `char` und `varchar`.

Listing 9.1: `char` und `varchar` verwenden

```
CREATE TABLE tblKunden
(
  KundID char(8) NOT NULL,
  KundName varchar(30) NOT NULL,
  Email varchar(50) NULL
)
```

9.5 Binäre Daten

Der binäre Datentyp speichert binäre Daten, die als Folge von Einsen und Nullen abgelegt werden. Diese werden dann bei der Ein- und Ausgabe als hexadezimale Paarwerte dargestellt. Die hexadezimalen Paarwerte bestehen aus den Zeichen 0 bis 9 und A bis F. Wenn Sie zum Beispiel das Feld *EinigeDaten binary(20)* erstellen, legen Sie damit fest, dass das Feld 20 Bytes Daten enthalten soll.

Wie auch bei den Datentypen für Zeichenfolgen können Sie für die Datentypen `binary(n)` und `varbinary(n)` maximal 8000 Bytes spezifizieren. Falls Sie den Datentyp `binary(n)` verwenden, werden die von Ihnen eingegebenen Daten mit Leerzeichen (hexadezimal `0x20`)



aufgefüllt. Der Datentyp `varbinary` wird dagegen nicht aufgefüllt. Falls Sie versuchen, Daten einzugeben, die die festgelegte maximale Länge überschreiten, werden die Daten abgeschnitten.

Zur Eingabe von Daten in ein Feld eines binären Datentyps müssen Sie vor die Zeichenfolge ein `0x` setzen. Beispielsweise geben Sie den Wert 10 in ein binäres Feld als `0x10` ein.

Dazu noch einige Beispiele:

```
MeineIcons varbinary(255)
MeineCursor binary(200)
KleinesWav varbinary(255)
```

9.6 Ganze Zahlen

Es gibt vier Datentypen für ganzzahlige Werte: `bigint`, `int`, `smallint` und `tinyint`, die genaue Skalarwerte speichern können. Die einzelnen Typen für ganzzahlige Werte unterscheiden sich hinsichtlich Speicherplatz und darstellbarem Wertebereich. Tabelle 9.4 gibt die Wertebereiche für ganzzahlige Datentypen an.

Datentyp	Länge	Wertebereich
<code>tinyint</code>	1	0 – 255
<code>smallint</code>	2	+/-32.767
<code>int</code>	4	+/-2.147.483.647
<code>bigint</code>	8	+/-2 ⁶³

Tabelle 9.4: Ganzzahlige Datentypen

Die folgenden Beispiele zeigen Deklarationen für ganzzahlige Typen:

```
MitarbeiterAlter tinyint NULL
MitarbeiterID smallint NOT NULL
KundenID int NOT NULL
```



Ganzzahldaten schneiden hinsichtlich der Speicherung, des Abrufs und mathematischer Berechnungen grundsätzlich besser als jeder andere Datentyp ab. Verwenden Sie also nach Möglichkeit ganzzahlige Werte. SQL Server unterstützt eine standardmäßige maximale Genauigkeit von 28 Ziffern. Hierdurch wird die Kompatibilität mit Entwicklungswerkzeugen wie Microsoft Visual Basic oder PowerBuilder gewährleistet. Wenn Sie SQL Server 2000 mit der Option `/p` starten, wird eine standardmäßige Genauigkeit von 38 verwendet.



9.7 Numerische Datentypen

SQL Server erlaubt sowohl die ungefähren Datentypen wie `float` und `real` als auch genaue numerische Typen wie `decimal` und `numeric`. Bei der Deklaration von ungefähren Datentypen legen Sie die Genauigkeit in Form der maximalen Anzahl von Ziffern fest, die auf beiden Seiten des Dezimaltrennzeichens erlaubt sind. Bei der Deklaration eines genauen Datentyps müssen Sie darüber hinaus die Anzahl der Dezimalstellen festlegen, die auf der rechten Seite des Dezimaltrennzeichens erlaubt sind.

Tabelle 9.5 gibt die Genauigkeiten für ungefähre und exakte Datentypen sowie die Anzahl der erforderlichen Speicherbytes an.

Genauigkeit	Speicherbedarf
1 – 9	5
10 – 19	9
20 – 28	13
30 – 38	17

Tabelle 9.5: Genauigkeiten und Speicherbedarf

Ungefährer numerische Datentypen

Zu den ungefähren Datentypen (Gleitkommatypen) zählen `float` und `real`. Die Zahlenwerte, die in diesen Datentypen gespeichert werden können, bestehen aus zwei Teilen, nämlich der Mantisse und dem Exponenten. Der Algorithmus, der zur Erzeugung dieser beiden Teile verwendet wird, arbeitet nicht hochgenau, was zur Folge hat, dass Sie nicht unbedingt exakt den Wert zurückerhalten werden, den Sie eingegeben haben. Zu einem Problem wird dies jedoch nur dann, wenn die Genauigkeit der gespeicherten Zahl an die Genauigkeit des Datentyps heranreicht. Die für `float` zugelassene Genauigkeit beträgt beispielsweise bis zu 38 Ziffern. Die Datentypen `float` und `real` sind insbesondere für wissenschaftliche und statistische Daten sinnvoll, die zwar keine absolute Genauigkeit erfordern, jedoch Wertebereiche aufweisen, die zwischen extrem kleinen und extrem großen Zahlen schwanken können.

`real`-Werte weisen eine Genauigkeit von 7 Ziffern auf und erfordern 4 Bytes Speicherplatz. Wenn Sie einen `float`-Wert deklarieren und dabei eine Genauigkeit von weniger als 7 festlegen, erzeugen Sie damit in Wirklichkeit einen `real`-Datentyp. `float`-Werte können eine Genauigkeit von 1 bis 38 aufweisen und erhalten standardmäßig eine Genauigkeit von 15 Ziffern, sofern kein anderslautender Wert festgelegt worden ist. Für `float`-Werte können Sie sämtliche Rechenarten mit der Ausnahme der Modulo-Operation (liefert den Rest einer ganzzahligen Division) durchführen.



Erstellen Sie beispielsweise den folgenden Datentyp:

```
Einige_Daten real
```

Sie können nun die Zahlen 188.445,2 oder 1.884,452 speichern, aber nicht die Werte 188.445,27 oder 1.884,4527, weil diese die standardmäßige Genauigkeit von sieben Ziffern überschreiten. Um diese größeren Werte zu speichern, müssen Sie eine `float`-Variable mit einer Genauigkeit erzeugen, die groß genug für sämtliche Ziffern ist. Da es insgesamt acht Ziffern gibt, müssen Sie also den folgenden Datentyp deklarieren:

```
Einige_Daten float(8)
```

Genaue numerische Datentypen

Genaue numerische Datentypen sind die Typen `decimal` und `numeric`. Die Exaktheit der Werte wird bis auf die letzte signifikante Ziffer bewahrt. Bei der Deklaration eines genauen numerischen Datentyps sollten Sie immer eine Genauigkeit und die Anzahl der Dezimalstellen angeben.



Falls Sie die Genauigkeit und die Anzahl der Dezimalstellen nicht angeben, verwendet der SQL Server die Standardwerte 18 und 0. Interessant ist hier Folgendes: Falls Sie eine Anzahl Dezimalstellen von 0 festlegen, erstellen Sie damit das Äquivalent eines Ganzzahldatentyps, da 0 Ziffern rechts vom Dezimaltrennzeichen ja möglich sind. Von Interesse ist dieser Sachverhalt deswegen, weil Sie nun sehr große und sehr genaue numerische Werte haben können, die größer als 2 Milliarden sind. Die Eigenschaft `IDENTITY`, die automatisch einen neuen Wert generiert, wenn ein neuer Datensatz hinzugefügt wird, erfordert einen ganzzahligen Datentyp. Sollten Sie für Ihre Datenbank mit mehr als 2 Milliarden Datensätzen rechnen, können Sie einen genauen numerischen Datentyp mit der Genauigkeit 0 verwenden.

Ist eine Spalte als `decimal(7,2)` deklariert, kann sie die Zahlenwerte 1000,55 und 11000,55 enthalten, jedoch nicht die Zahlenwerte 11110000,55 oder 1100,5678. Wenn Sie versuchen, eine Zahl unterzubringen, die größer als die erlaubte Genauigkeit und die Anzahl der Dezimalstellen nach dem Trennzeichen ist, wird die Zahl abgeschnitten.

Listing 9.2 enthält ein Beispiel für die Verwendung von genauen numerischen Datentypen.



Listing 9.2: Numerische Datentypen verwenden

```
CREATE TABLE tblGold
(
  AtomGewicht decimal(8,4),
  Grammoekuel numeric(12,6),
  Dichte numeric(5,4)
)
```



Im Allgemeinen sollten Sie die Verwendung der Datentypen `float` und `real` vermeiden. Es spricht zwar nichts dagegen, wenn Sie Ihre Daten in den dezimalen und numerischen Datentypen speichern, aber Sie sollten in Betracht ziehen, dass Berechnungen mit den Datentypen `float` und `real` zu unerwarteten oder irritierenden Ergebnissen führen können.

9.8 Besondere Datentypen

Es gibt mehrere Datentypen, die sich keiner bestimmten Kategorie zuordnen lassen und die deshalb in diesen Abschnitt aufgenommen wurden. Sie erfahren nun mehr über den Datentyp `bit`, die zwei BLOB-Typen (BLOB; binary large object) `text` und `image` sowie den Datentyp `RowGUID`.

`bit`

Der Datentyp `bit` ist ein logischer Typ, mit dem man boolesche Werte speichert. Boolesche Datentypen verwendet man als Flags, um Zustände wie ein/aus, wahr/falsch oder ja/nein zu kennzeichnen. Die hier gespeicherten Werte lauten entweder 0 oder 1.

`bit`-Spalten können `NULL` (unbekannter Wert) sein und lassen sich nicht indizieren. Die `bit`-Datentypen erfordern ein einziges Speicherbyte. Falls Sie in einer Tabelle mehrere `bit`-Spalten definiert haben, fasst SQL Server automatisch diese in Felder zu je acht Bit zusammen, die jeweils ein einzelnes Byte Speicherplatz verbrauchen. Die folgenden Anweisungen zeigen Beispiele für den Datentyp `bit`:

```
Geschlecht bit NOT NULL
bezahlt bit NULL
gedruckt bit NOT NULL
```



Text und Image

Die Datentypen `text` und `image` werden dann verwendet, wenn die Speicherplatzanforderungen die Begrenzung von 8000 Bytes pro Spalte übersteigen. Diese Datentypen nennt man auch BLOBs (binary large objects). Die Typen `image` und `text` können pro Deklaration bis zu 2 GB binäre oder Textdaten speichern.

Wenn Sie einen Datentyp `text` oder `image` deklarieren, wird der Zeile ein 16-Bit-Zeiger hinzugefügt. Dieser 16-Bit-Zeiger verweist auf eine besondere Datenseite mit 8 KB, in der die zusätzlichen Daten gespeichert werden. Überschreiten die Daten die Datenseite mit 8 KB, werden Zeigerkonstruktionen erstellt, die auf die zusätzlichen Seiten des BLOBs verweisen.

Die Speicherung und das Abrufen von Text- und Grafikdaten kann sich hinderlich auf die Leistung der Datenbank auswirken, da nun während der Einfüge-, Aktualisierungs- und Löschvorgänge größere Mengen von Daten in die Transaktionsprotokolle geschrieben werden müssen. Dieses Problem können Sie umgehen, indem Sie den Befehl `WRITETEXT` verwenden, der Änderungen der Daten ermöglicht, ohne dass damit in Zusammenhang stehende Einträge ins Transaktionsprotokoll vorgenommen werden.



Falls Sie nicht protokollierte Operationen ausführen, sollten Sie Ihre Datenbank immer unverzüglich sichern, da andernfalls die Wiederherstellbarkeit der Datenbank gefährdet ist (siehe dazu Tag 7).

Als Alternative zu den großen Speicheranforderungen der Datentypen `text` und `image` können Sie diese Objekte in der Form von Betriebssystemdateien ablegen und nur deren Pfadnamen in der Datenbank speichern.

Beispiele für die Deklaration der Datentypen `text` und `image` sind:

```
MitarbeiterFoto image  
GescannteVertraege image  
Beschreibung text  
Kommentar text
```

RowGUID

Bei Einsatz der Mergereplikation (die Tag 16 und Tag 17 näher behandeln) muss jede Spalte in den replizierten Tabellen einen eindeutigen Bezeichner besitzen. Das lässt sich erreichen, indem man in jeder replizierten Tabelle eine Spalte mit dem Datentyp `uniqueidentifier` erstellt. Dieser Datentyp besitzt eine Eigenschaft namens `ROWGUIDCOL`. Bei Aktivierung der Eigenschaft `ROWGUIDCOL` kann der Spalte ein globaler eindeutiger Bezeichner (GUID) zugewiesen werden. Damit wird erreicht, dass die Spalten

einer Version der replizierten Tabelle dieselbe GUID wie in den anderen Versionen derselben Tabelle erhalten. Wenn Sie Änderungen an einer Zeile in einer replizierten Tabelle mit ROWGUIDCOL vornehmen, ändert SQL Server den Wert der Eigenschaft ROWGUIDCOL. Replizierte Zeilen aus zwei Datenbanken können so getrennt voneinander identifiziert werden.

GUIDs lassen sich nach folgenden beiden Methoden initialisieren:

- Sie rufen die Funktion NEWID auf.
- Sie wandeln eine Zeichenfolgenkonstante in eine hexadezimale Ziffer der Form xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx um. Ein Beispiel für eine gültige RowGUID lautet: 8FE17A24-B1AA-23DA-C790-2749A3E09AA2.

Auf den Datentyp uniqueidentifier können Sie die folgenden Vergleichsoperatoren anwenden: =, <>, IS NULL und IS NOT NULL.

sql_variant

Der Datentyp sql_variant ist dem Visual Basic-Datentyp variant ähnlich. Der Typ sql_variant speichert Werte in anderen von SQL Server unterstützten Datentypen, mit Ausnahme von text, ntext, timestamp und sql_variant.

Diese Datentypen können Sie in Spalten, als Variablen in einer DECLARE-Anweisung, als Parameter und als Ergebnisse von benutzerdefinierten Funktionen verwenden. (Auf benutzerdefinierte Funktionen geht Tag 15 näher ein.)

Für Vergleiche von sql_variant-Datentypen gelten spezielle Regeln. Die Datentypen von SQL Server sind in Datentypfamilien unterteilt, die Tabelle 9.6 zeigt.

Datentyphierarchie	Datentypfamilie
sql_variant	sql_variant
datetime, smalldatetime	datetime
float, real	ungefähre Zahl
bigint, bit, decimal, int, money, smallint, smallmoney, tinyint	exakte Zahl
char, nchar, nvarchar, varchar	Unicode
binary, varbinary	Binär
uniqueidentifier	uniqueidentifier

Tabelle 9.6: Datentypfamilien

Für Vergleiche von sql_variant-Daten ist vor allem die folgende Regel zu beachten: Wenn man zwei sql_variant-Datentypen aus verschiedenen Familien vergleicht, gilt der



Wert als der größere, dessen Datentypfamilie weiter oben in der Datentyphierarchie (siehe Tabelle 9.6) aufgeführt ist. Haben Sie beispielsweise einen `float` in einem `sql_variant`-Typ und einen Binärwert in einem anderen gespeichert, liefert der `float` bei einem Vergleich immer einen höheren Wert als der binäre Typ.

Viele Richtlinien und Details zum Datentyp `sql_variant` gehen über den Rahmen dieses Buches hinaus. Hier sei auf die einschlägige Literatur verwiesen.

table

Der Datentyp `table` dient im Allgemeinen dazu, Ergebnismengen für eine spätere Verwendung zu speichern. Man kann das mit einer temporären Tabelle vergleichen. Die eigentliche Leistung des Datentyps `table` liegt darin, dass man ihn wie eine normale Tabelle verwenden kann; er lässt sich wie eine lokale Variable in Funktionen, Stapeln und gespeicherten Prozeduren einsetzen. Tag 15 kommt auf diesen neuen Datentyp noch einmal zurück.

9.9 Die Datentypen Datum und Zeit

Datums- und Zeitdaten lassen sich in den Datentypen `datetime` und `smalldatetime` speichern. Bemerkenswert dabei ist, dass Datum und Zeit immer zusammen in einem einzelnen Wert abgelegt werden.

Datum und Zeit können mehrere verschiedene Formate annehmen. Den Monat geben Sie über den vollständigen Namen oder eine Abkürzung an. Die Groß- und Kleinschreibung wird ignoriert und Kommas sind optional.

Beispiele für das alphanumerische Format des 15. Aprils 2001 sind:

```
"Apr 15 2001"  
"Apr 15 01"  
"Apr 01 15"  
"15 Apr 01"  
"2001 April 15"  
"2001 15 April"
```



Sie können für den Monat auch einen *Ordinalwert* angeben. Ein Ordinalwert gibt die Position eines Elements in einer Liste dieser Elemente an. Da in den oben genannten Beispielen der April der vierte Monat des Jahres ist, können Sie also die Zahl 4 als Festlegung dieses Monats verwenden.



Es folgen einige Beispiele für die Verwendung des Ordinalwertes für den 15. April 2001:

4/15/01 (mtj)
4-15-01 (mtj)
15.4.01 (tmj)
4/01/15 (mjt)
15/01/04 (tjm)
01/15/04 (jmt)

Auch für die Zeit gibt es mehrere verschiedene Formate. Beispiele hierfür lauten:

16:30 (4 Stunden, 30 Minuten)
16:30:20:999 (4 Stunden, 30 Minuten, 22 Sekunden, 999 Millisekunden)
4:30 PM

Ein Datum, das im Datentyp `datetime` gespeichert ist, wird auf Millisekunden genau in der Form von insgesamt 8 Bytes (4 für die Anzahl der Tage seit dem 1. Januar 1900 und 4 weitere für die Anzahl der Sekunden seit Mitternacht) gespeichert. Ein Datum vor diesem Zeitpunkt wird als negativer Wert gespeichert. Insgesamt ergibt sich ein Wertebereich vom 1.1.1753 bis zum 31.12.9999. Die Genauigkeit dieser Datums-/Zeitwerte bewegt sich dabei innerhalb von 3,33 Millisekunden.



Verwenden Sie nicht den Datentyp `datetime`, um Teile von Datumswerten, also Monat, Tag oder Jahr zu speichern. Wenn Sie lediglich das Jahr benötigen, sind die Typen `smallint` oder `tinyint` effizienter. Speichern Sie das Datum nicht in dieser Weise, müssen Sie die Datumswerte selbst analysieren, wenn Sie derartige Informationen einfügen, aktualisieren oder anderweitig damit arbeiten möchten.

Der Datentyp `smalldatetime` benötigt insgesamt 4 Bytes. Ein Datum, das mit diesem Typ gespeichert ist, wird auf die Minute genau gespeichert. Intern werden dazu zwei Bytes (`smallint`) benutzt, von denen ein Byte die Anzahl der Tage seit dem 1. Januar 1900 und das andere Byte (`smallint`) die Anzahl der Sekunden seit Mitternacht festhält. Der Wertebereich für `smalldatetime` erstreckt sich vom 1.1.1900 bis zum 6.6.2079.



Verwenden Sie in Datenbanken `smalldatetime` hauptsächlich für aktuelle Datumsangaben, die sich häufiger ändern; beispielsweise Datumsangaben, die nur für wenige Jahre benötigt werden.

Tabelle 9.7 gibt Speicherbedarf und Wertebereich für die `datetime`-Datentypen an.



Datentyp	Speicherplatz	Wertebereich
datetime	8	1.1.1753 – 31.12.9999
smalldatetime	4	1.1.1900 – 6.6.2079

Tabelle 9.7: Datums-/Zeit-Datentypen

9.10 Währungsdaten

An Währungsdattentypen stehen Ihnen `money` und `smallmoney` zur Verfügung. Beide Typen können vier Dezimalstellen nach dem Dezimaltrennzeichen speichern und sind in der Lage, internationale Währungswerte aufzunehmen. Leider gibt es aber in SQL Server keinerlei Währungsumrechnungsfunktionen. Bei der Eingabe von Währungsdaten sollten Sie diesen ein Dollarzeichen als Präfix mitgeben. Tabelle 9.8 zeigt die Währungsdattentypen mit Speicherplatzbedarf und Wertebereich.

Datentyp	Speicherplatz	Wertebereich
money	8	922.337.203.685.447,5808
smallmoney	4	214.748.3647

Tabelle 9.8: Währungsdattentypen

Wie Sie Tabelle 9.8 entnehmen können, nimmt der Typ `smallmoney` bis zu 10 Ziffern mit vier Stellen nach dem Dezimaltrennzeichen auf. Der Datentyp `money` ist immerhin groß genug, um Hunderte von Trillionen speichern zu können.

Es folgen Beispiele für Deklarationen mit Währungsdattentypen:

```
KontenEinnahmen money  
KontenAusgaben smallmoney
```

9.11 Der Datentyp rowversion (timestamp)

SQL Server 2000 führt die Bezeichnung `rowversion` für den bisherigen Datentyp `timestamp` ein. Timestamp-Werte werden weiterhin in SQL Server unterstützt und sind mit dem Standard ANSI SQL-99 kompatibel. Immer dann, wenn Sie einer Tabelle mit einem `timestamp`-Feld einen neuen Datensatz hinzufügen, werden automatisch Zeitwerte hinzugefügt. Bei jeder Aktualisierung einer Zeile aktualisiert sich der `rowversion`-Typ automatisch selbst.

Über den Datentyp `rowversion` wird ein vom SQL Server generierter, eindeutiger und automatisch aktualisierter Wert erzeugt. Obwohl `rowversion` dem Datentyp `datetime` ähnelt, ist es ein gänzlich anderer Typ. Werte des Typs `rowversion` werden bei Spalten mit dem Attribut `NOT NULL` als `binary(8)` und bei Spalten, die Nullwerte erlauben, als `varbinary(8)` gespeichert. Pro Zeile ist nur eine Spalte des Typs `timestamp` möglich.



Der Datentyp `timestamp` bezieht sich nicht auf die Systemzeit, sondern stellt einfach nur einen fortlaufend inkrementierten Zählerwert dar.

Über den Datentyp `timestamp` können Sie verfolgen, in welcher Reihenfolge die Einträge in Ihren Tabellen hinzugefügt und geändert werden. Es folgen Beispiele für den Einsatz des Datentyps `timestamp`:

```
ZuletztGeändert timestamp NOT NULL
Telefonanruf timestamp NOT NULL
```

9.12 ANSI und Unicode

Um sicherzustellen, dass die Datentypen von SQL Server dem ANSI-Standard entsprechen, können Sie anstelle der SQL Server-Datentypen auch die ANSI-Typen verwenden. Tabelle 9.9 stellt die ANSI-Datentypen und die äquivalenten SQL Server-Datentypen gegenüber.

ANSI-Datentyp	SQL Server-Datentyp
character	char
character(n)	char(n)
char varying	varchar
character varying(n)	varchar(n)
binary varying	varbinary
dec	decimal
double precision	float
float(n) für n = 1 bis 7	real
float(n) für n = 8 bis 15	float
rowversion	timestamp
integer	int

Tabelle 9.9: Synonyme von SQL Server

Daten im Unicode-Format werden über den Unicode-Zeichensatz UCS-2 dargestellt, der aus mehreren Bytes besteht. Bei gewöhnlichen ANSI-Zeichen wird zur Speicherung eines gegebenen Zeichens immer ein Byte verwendet. Unicode ist dagegen als »weiter« bzw. Mehr-Byte-Zeichensatz bekannt. Im UCS-2-Unicode-Zeichensatz werden zur Darstellung eines Zeichens zwei Byte verwendet, was sich insbesondere dann als nützlich erweist, wenn Sie es mit Datenbanken zu tun haben, in denen mit verschiedenen nationalen Sprachen gearbeitet wird. Denken Sie beispielsweise daran, dass das Alphabet der Sprachen Englisch und Spanisch zwar noch problemlos mit einem Ein-Byte-Zeichensatz dargestellt werden kann, dies aber schon nicht mehr für Sprachen wie etwa Japanisch gilt (schon das Standard-Japanisch Kana enthält mehr als 1000 Zeichen). Ein normales Byte mit 8 Bit kann nur 256 Zeichen darstellen, während das Zeichen eines Mehr-Byte-Unicode-Zeichensatzes 65.536 Zeichen darstellen kann.

Zur Darstellung Ihrer Unicode-Daten können Sie die Datentypen `nchar`, `nvarchar` und `ntext` einsetzen. `nchar` und `nvarchar` weisen eine Obergrenze von 8000 Byte bzw. 4000 Zeichen auf. So wäre zum Beispiel `nchar(4000)` gültig, während `nchar(6000)` nicht zulässig ist, da es 12.000 Zeichen darstellt und somit nicht in eine einzelne Datenseite passt. Einen derartigen Wert müssten Sie deshalb entweder in zwei getrennte `nchar`-Datentypen aufteilen oder über einen einzelnen `ntext`-Typ darstellen. Der Typ `ntext` unterstützt bis zu 2,14 GB Daten.

Unicode-Daten sind ein besonders leistungsfähiges Merkmal von SQL Server 2000. Die Speicherung der Daten im Unicode-Format ist eine hervorragende Lösung für den Fall, dass Sie einen internationalen Einsatz Ihres SQL Servers in Ländern planen, in denen kein US-amerikanisches Englisch benutzt wird. Unicode wird sich wahrscheinlich mit der Verbreitung von Windows NT/2000 und SQL Server als international einsetzbare Lösung als standardmäßiges Speicherformat durchsetzen.

9.13 Benutzerdefinierte Datentypen

Benutzerdefinierte Datentypen können Sie für eine bestimmte Datenbank spezifizieren oder auch in der *model*-Datenbank platzieren, die bekanntlich als Muster für neu zu erstellende Datenbanken fungiert. Auf diese Weise stehen Ihnen Ihre neuen benutzerdefinierten Datentypen in allen später erzeugten Datenbanken sofort zur Verfügung.

Einen benutzerdefinierten Datentyp müssen Sie ausschließlich mit den vom System angebotenen Datentypen erstellen. Beispielsweise können Sie zwar einen neuen Datentyp namens `MitarbeiterID` erzeugen und mit dem Datentyp `character` oder `integer` definieren, jedoch keinen nicht existenten Datentyp wie etwa `column_id` benutzen.

Bevor Sie den benutzerdefinierten Datentyp der Tabelle hinzufügen, müssen Sie ihn zunächst erzeugen, was Sie über den SQL Server Enterprise Manager oder die gespeicherte Systemprozedur `sp_addtype` erledigen. Auf benutzerdefinierte Datentypen geht Tag 14 näher ein.

Zum Erstellen benutzerdefinierter Datentypen über SQL Server-Query Analyzer benutzen Sie die gespeicherte Systemprozedur `sp_addtype`. Zum Beispiel erstellen die folgenden Anweisungen drei benutzerdefinierte Datentypen in der Datenbank *pubs*:



```
sp_addtype mitarbid, 'char(9)', 'NULL'  
sp_addtype id, 'varchar(11)', 'NULL'  
sp_addtype tid, 'varchar(6)', 'NULL'
```

Nach der Deklaration eines benutzerdefinierten Datentyps können Sie ihn beliebig oft in Ihrer Datenbank verwenden. Benutzen Sie dazu beispielsweise die folgende CREATE TABLE-Anweisung:



```
CREATE TABLE tblMitarbeiter  
(  
  MitarbeiterId empid NOT NULL,  
  Vname char(15) NOT NULL,  
  Nname char(20) NOT NULL,  
  PensionsPlan id NOT NULL  
)
```

Zum Hinzufügen benutzerdefinierter Datentypen über den SQL Server Enterprise Manager führen Sie die folgenden Arbeitsschritte aus:

1. Öffnen Sie den SQL Server Enterprise Manager.
2. Erweitern Sie die Konsolenstruktur bis zur Datenbank *pubs*.
3. Wechseln Sie zum Ordner *Benutzerdefinierte Datentypen*.
4. Klicken Sie mit der rechten Maustaste auf den Ordner und wählen Sie im Kontextmenü NEUER BENUTZERDEFINIERTER DATENTYP.
5. Geben Sie den Namen, den Datentyp, die Länge und eventuelle Regeln oder Standardwerte ein, die für diesen Typ gelten sollen.
6. Klicken Sie zum Beenden auf OK.

Zum Löschen benutzerdefinierter Datentypen können Sie den SQL Server Enterprise Manager oder die gespeicherte Systemprozedur `sp_droptype` verwenden.



Führen Sie dazu im SQL Server Enterprise Manager die folgenden Arbeitsschritte aus:

1. Erweitern Sie Ihre Datenbank und markieren Sie den Ordner *Benutzerdefinierte Datentypen*.
2. Klicken Sie im Detailbereich mit der rechten Maustaste auf den benutzerdefinierten Datentyp, den Sie löschen möchten, und wählen Sie LÖSCHEN.

Damit ist die Sache erledigt. In Transact-SQL können Sie die gespeicherte Systemprozedur `sp_droptype` ausführen:



`sp_droptype mitarbid`



Ein benutzerdefinierter Datentyp kann nicht gelöscht werden, falls er in einer Tabelle noch verwendet wird oder einen an ihn gebundenen Standardwert besitzt. Was diese Einschränkung bedeutet, erfahren Sie am Tag 14.

9.14 Berechnete Spalten



Eine *berechnete Spalte* stellt ein interessantes Leistungsmerkmal von SQL Server 2000 dar. Eine berechnete Spalte speichert keine berechneten Werte (wie Sie vielleicht angenommen haben), sondern statt dessen den Ausdruck, der zur Berechnung des Wertes eingesetzt wird. So können Sie beispielsweise für Ihre Tabelle namens *Summe* eine berechnete Spalte mit dem Ausdruck `Total AS Preis * Menge` erstellen.

Die Ausdrücke, die Sie speichern können, dürfen aus nicht berechneten Spalten in derselben Zeile, aus Konstanten, Funktionen, Variablen und sogar Namen bestehen. Die berechnete Spalte beginnt automatisch mit der Berechnung der Werte, sobald sie in den SELECT-, WHERE- und ORDER BY-Klauseln einer Abfrage (diese werden ausführlich am 10. Tag erörtert) aufgerufen wird. Darüber hinaus kann sie in normalen Ausdrücken verwendet werden.

Beim Umgang mit berechneten Spalten gilt es einige Regeln zu beachten:

- Die im Ausdruck einer berechneten Spalte referenzierten Spalten müssen sich in derselben Tabelle befinden.
- Der Ausdruck einer berechneten Spalte darf keine Unterabfragen enthalten.



- Berechnete Spalten dürfen nicht als Teil von Schlüsseln oder Indizes eingesetzt werden; diese Einschränkung gilt auch für Primärschlüssel, Fremdschlüssel und eindeutige Indizes.
- Eine berechnete Spalte darf keine angehängte DEFAULT-Einschränkung haben.
- Auf berechnete Spalten darf keine INSERT- oder UPDATE-Anweisung angewendet werden.

Es folgen nun einige Beispiele zum Einsatz von berechneten Spalten:

```
CREATE TABLE tblBestellung (  
BestID int NOT NULL,  
Preis money NOT NULL,  
Menge smallint NOT NULL,  
Summe AS Preis * Menge  
)
```

Die erste Tabelle besitzt eine berechnete Spalte namens *Summe*, die wiederum aus den Feldern *Preis* und *Menge* besteht.

```
CREATE TABLE tblRechnungDrucken (  
RechnungsID int NOT NULL,  
RechDatum datetime NOT NULL,  
DruckDatum AS DateAdd(day,30, RechDatum)  
)
```

Dieses Beispiel enthält eine berechnete Spalte namens *DruckDatum*, die 30 Tage mit der Funktion *DateAdd* zur Spalte *RechDatum* addiert.



Falls nur die letzten beiden Ziffern des Jahres angegeben sind, interpretiert SQL Server Werte kleiner als 50 als 20jj, während Werte größer oder gleich 50 als 19jj interpretiert werden. Beispielsweise wird April 15 03 als 15. April 2003 interpretiert.

9.15 Tabellen erstellen

Sie kennen nun sämtliche Datentypen von SQL Server 2000. Es ist also an der Zeit, alles Gelernte im Rahmen der Anweisung *CREATE TABLE* anzuwenden. Wie bei den meisten Dingen in SQL Server gibt es auch hier zwei Methoden zum Erstellen von Tabellen: Entweder setzen Sie dazu den SQL Server Enterprise Manager oder aber ein Transact-SQL-Skript ein. Als Erstes lernen Sie die Anweisung *CREATE TABLE* kennen.



Die Anweisung CREATE TABLE

Die Syntax der Anweisung CREATE TABLE lautet:



```
CREATE TABLE [Datenbank.[Besitzer.]]Tabellenname  
(  
  Spaltenname Datentyp  
    [Identität | Einschränkung | NULL | NOT NULL | Sortierung]  
  [...]  
)
```

Die Argumente haben folgende Bedeutung:

- *Tabellenname* ist der Name der neuen Tabelle. Er muss den Regeln für Bezeichner entsprechen und innerhalb der Datenbank des Besitzers eindeutig sein. Wenn also zwei Benutzer innerhalb einer Datenbank die Berechtigung zum Erstellen von Tabellen haben, können die Tabellen an sich zwar den gleichen Namen tragen, sie gelten aber dennoch als eindeutig, weil der Name des Besitzers einen Bestandteil des Tabellennamens bildet.
- *Spaltenname* ist der Name der Spalte und muss den Regeln für Bezeichner entsprechen.
- *Datentyp* ist der Datentyp der Spalte.
- Der letzte Teil ist optional. Hier können Sie die IDENTITY-Eigenschaft, Feldeinschränkungen und die NULL-Zulässigkeit festlegen.
- *Sortierung* ist ebenfalls optional. Mit dieser Eigenschaft können Sie die Standardsortierung der Datenbank für diese Spalte überschreiben.



Die vorhergegangene Anweisung CREATE TABLE ist gegenüber den Beispielen, die sich in der SQL Server-Online-Dokumentation finden, stark vereinfacht worden. Viele der zusätzlichen Optionen beziehen sich auf ANSI-Einschränkungen, die Sie in Lektion 14 im Detail untersuchen.

Listing 9.3 zeigt ein Beispiel für die Anweisung CREATE TABLE.

Listing 9.3: Erstellen einer Tabelle

```
CREATE TABLE Mitarbeiter  
(  
  mitarb_id tinyint IDENTITY NOT NULL,  
  Vname char(15),  
  Nname char(20) NOT NULL,  
  Adress1 varchar(30),
```



```

Adresse2 varchar(30),
Stadt varchar(30),
Bundesland char(2),
Plz char(10),
Anfangsdatum datetime
)

```

Erstellen von Tabellen über den SQL Server Enterprise Manager

Bei Verwendung des SQL Server Enterprise Managers können Sie Ihre Tabellen grafisch erstellen. Führen Sie im SQL Server Enterprise Manager dazu die folgenden Arbeitsschritte aus:

1. Verbinden Sie sich mit Ihrem SQL Server, erweitern Sie Ihren Datenbankordner und die Datenbank, mit der Sie arbeiten möchten. Markieren Sie dann den Ordner *Tabellen*.
2. Klicken Sie im Detailbereich mit der rechten Maustaste und wählen Sie aus dem Kontextmenü **NEUE TABELLE**, wie es Abbildung 9.1 zeigt.

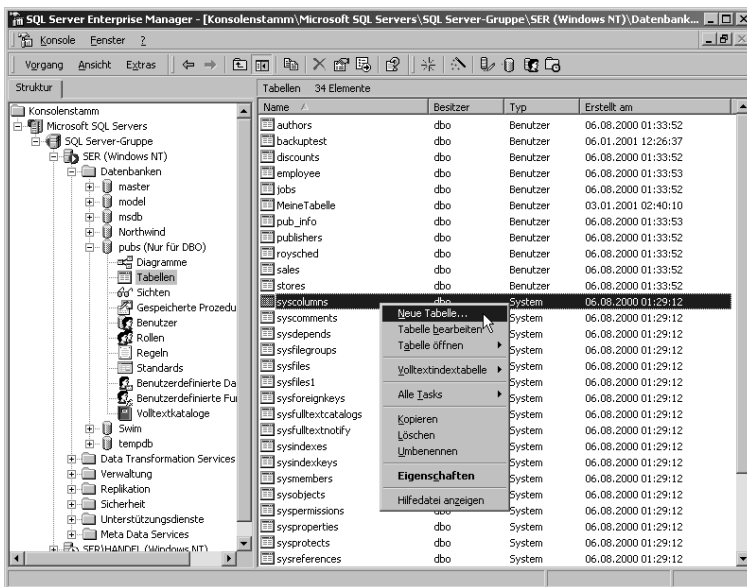


Abbildung 9.1:
Markieren Sie den
Ordner *Tabellen* in
der Datenbank, in
die Sie eine neue
Tabelle hinzufügen
möchten.



3. Tragen Sie die Angaben unter *Spaltenname*, *Datentyp*, *Größe* und *NULL zulassen* ein. In der unteren Fensterhälfte können Sie Spalteneigenschaften festlegen, wie zum Beispiel *Beschreibung* und *Standardwert*. Falls Sie ein Identitätsfeld erstellen möchten, setzen Sie die *Identität*-Eigenschaft auf der Registerkarte SPALTEN auf Ja und wählen dann einen *ID-Startwert* und eine *ID-Schrittweite*. Mehr zu den Eigenschaften *Standardwert* und *Ist RowGuid* erfahren Sie am Tag 14. Abbildung 9.2 zeigt ein Beispiel für eine Tabelle, in der mehrere Spalten hinzugefügt wurden.
4. Wenn Sie die Tabellendefinition fertig gestellt haben, klicken Sie auf das Symbol **SPEICHERN**. Geben Sie der Tabelle einen Namen und schließen Sie danach das Dialogfeld.

Tabellen löschen

Tabellen können Sie über den SQL Server Enterprise Manager auf einfache Weise löschen, indem Sie die Tabelle mit der rechten Maustaste anklicken und **LÖSCHEN** aus dem Kontextmenü wählen. Zum Löschen einer Tabelle über Transact-SQL führen Sie die Anweisung **DROP TABLE** aus. Beispielsweise lautet die Anweisung zum Löschen der Tabelle *tblMitarbeiter*:

```
DROP TABLE tblMitarbeiter
```

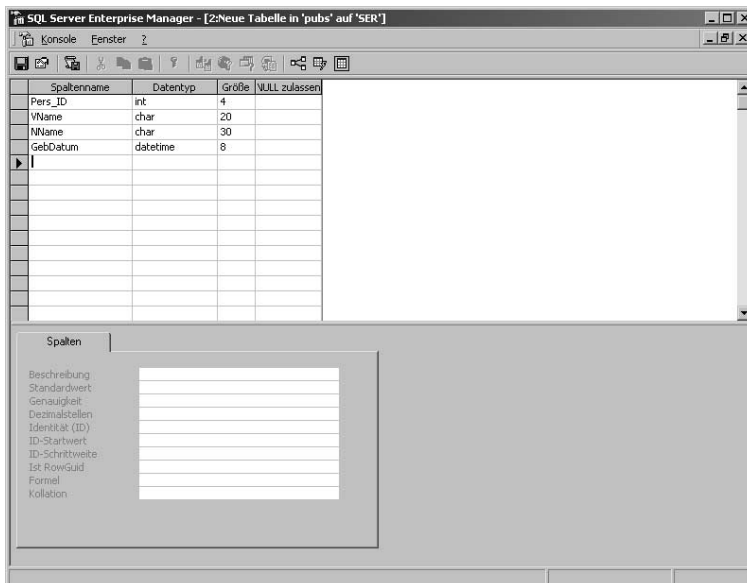


Abbildung 9.2:
Im Raster können
Sie die Felder und
Eigenschaften kom-
fortabel festlegen,
ohne Code schrei-
ben zu müssen.



Systemtabellen können nicht gelöscht werden.

Zum Löschen einer Tabelle über den SQL Server Enterprise Manager führen Sie die folgenden Schritte aus:

1. Markieren Sie in der Konsolenstruktur den Ordner *Tabellen* unter der gewünschten Datenbank und klicken Sie mit der rechten Maustaste im Detailbereich auf die Tabelle, die Sie löschen möchten.
2. Wählen Sie LÖSCHEN aus dem Kontextmenü. Es erscheint nun ein Dialogfeld wie in Abbildung 9.3 gezeigt.

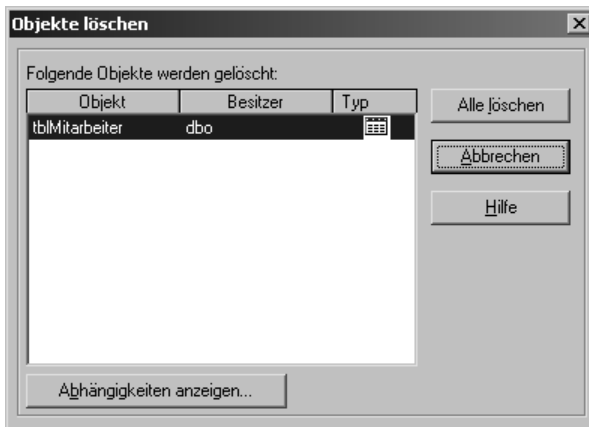


Abbildung 9.3:
Sie können die hier aufgelisteten
Tabellen löschen oder nach Abhän-
gigkeiten prüfen

3. Falls Sie die Abhängigkeiten für diese Tabelle sehen möchten, klicken Sie auf die Schaltfläche ABHÄNGIGKEITEN ANZEIGEN. Sofern die Tabelle nicht mehr Teil einer Primärschlüssel-/Fremdschlüsselbeziehung ist, können Sie auf die Schaltfläche ALLE LÖSCHEN klicken, um die Tabelle zu entfernen. (Auch auf diese Beziehungen geht Tag 14 näher ein.)

9.16 Zusammenfassung

Der SQL Server besitzt viele eigene Datentypen, mit denen Sie die Spalten für Ihre Tabellen anlegen können. Den konkreten Datentyp müssen Sie mit Umsicht wählen, da er Auswirkungen auf die Leistung und den Speicherplatzbedarf haben kann. Außerdem können Sie einen Datentyp nur ändern, indem Sie die Tabelle löschen und neu erstellen.



Darüber hinaus erlaubt es Ihnen SQL Server, benutzerdefinierte Datentypen zu erstellen. Zu diesem Zweck können Sie die gespeicherte Systemprozedur `sp_addtype` oder den SQL Server Enterprise Manager einsetzen. Benutzerdefinierte Datentypen müssen auf systemeigenen Typen von SQL Server aufbauen und vor der Verwendung in einer Tabelle definiert werden.

Tabellen erstellen Sie mit dem Tabelleneditor des SQL Server Enterprise Managers oder über die Anweisung `CREATE TABLE`.

9.17 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F *Braucht man eigentlich benutzerdefinierte Datentypen?*

A Nein. Benutzerdefinierte Datentypen sind zwar nicht erforderlich, machen aber Ihre Spalten und die `CREATE TABLE`-Anweisungen verständlicher. Außerdem bieten sie die Möglichkeit, Datentypen anzulegen, die sich an mehreren Stellen wiederverwenden lassen.

F *Welche Regeln sollten für den Umgang mit numerischen Werten beachtet werden?*

A Ganzzahlige Datentypen verarbeitet SQL Server am schnellsten; außerdem haben sie den geringsten Speicherbedarf. Falls Sie Ihre Daten in einem Typ `int`, `tinyint` oder `smallint` unterbringen können, sollten Sie das auch unbedingt tun. Bei größeren Zahlen oder Werten, die ein Dezimaltrennzeichen benötigen, sollten Sie anstelle der Datentypen `float` und `real` besser die genauen numerischen Typen `decimal` und `numeric` einsetzen. Die Typen `float` und `real` besitzen nicht die Genauigkeit der genauen numerischen Typen.

F *Welche allgemeinen Regeln gibt es für den Umgang mit Zeichendaten?*

A Dies hängt gänzlich vom Typ der Zeichendaten ab, mit denen Sie umgehen. Falls Sie Spalten haben, deren Länge nur wenig variiert (5 bis 10 Zeichen), ist die Wahl einer Spalte mit fester Länge vermutlich vorteilhafter als die einer Spalte mit variabler Länge. Bei Feldern mit variabler Länge wird durch Hinzufügen

von 2 Bytes pro Zeile zu Ihrer Datenbank ein zusätzlicher Verwaltungsaufwand erforderlich. Diese Felder sind außerdem langsamer bei der Verarbeitung als ein Feld mit fester Länge. Falls Ihre Felder mit fester Länge stark variieren (20 bis 100 Zeichen), ist ein Feld mit variabler Länge die bessere Wahl. Sie müssen bei dieser Frage die Geschwindigkeit des einen oder anderen Datentyps gegenüber dem Speicherplatz abwägen, der bei Daten mit fester Länge verbraucht wird.

- F** *Sie haben erwähnt, dass ich bis zu 2 Milliarden Tabellen in meiner Datenbank unterbringen kann. Wie viele Tabellen sollten es in der Praxis maximal sein?*
- A** Dies hängt einzig und allein von der geschäftlichen Umgebung ab, die Sie abzubilden versuchen. Falls Sie in die Größenordnung von etwa 100.000 Tabellen kommen, sollten Sie über einen neuen Entwurf der Datenbank nachdenken! Eine typische Datenbank mit 5 bis 10 GB hat in der Regel zwischen 20 bis 75 Tabellen. Größere Systeme können auch mehr haben.
- F** *Welche Möglichkeiten, mit BLOBs umzugehen, gibt es eigentlich noch, wenn man in Betracht zieht, dass die Datentypen text und image viel Speicherplatz beanspruchen und beim Hinzufügen, Löschen und Ändern nicht protokolliert werden?*
- A** Es gibt viele Unternehmen, die es als sinnvoll ansehen, BLOBs in ein eigenes Verzeichnis auf einem schnellen Dateiserver im Netzwerk zu legen. Anschließend speichert man den Pfadnamen zur Datei als varchar in der Datenbank von SQL Server. Diese Lösung hat ihre Vor- und Nachteile, wobei der Hauptnachteil die Konsistenz ist. Es stellt sich nämlich die Frage, wie SQL Server darüber informiert werden kann, falls jemand diese Datei löscht oder verschiebt. Auf der Plusseite wiederum ist die Tatsache zu verbuchen, dass Sie für Tausende von Dokumenten oder Grafiken nun keinen wertvollen Speicherplatz in Ihrer Datenbank mehr benötigen.
- F** *Was muss man beim Erstellen von Tabellen unbedingt beachten?*
- A** Die drei Regeln des Grundstücksgeschäfts heißen: Lage, Lage und nochmals Lage! Die drei Regeln des Datenbankentwurfs lauten: Entwurf, Entwurf und nochmals Entwurf! Wenn man direkt von der Programmierung kommt, ist man manchmal dazu verführt, blind in die Sache hinzuspringen und sofort mit der Codierung des Projektes anzufangen. Auf lange Sicht wird hierdurch aber nur Spaghetticode und eine Menge von Tabellen in der Datenbank erzeugt; außerdem wächst exponentiell dazu der Mangel an Effizienz. Widmen Sie sich daher ausgiebig dem Entwurf Ihrer Tabellen, Beziehungen, benutzerdefinierten Datentypen und Einschränkungen, bevor Sie mit ihrer Implementierung beginnen. Sorgen Sie auch für eine gedruckte Dokumentation Ihres Entwurfs, die sich einmal als ein wertvolles Werkzeug für jeden herausstellen wird, der Ihr SQL Server-System verwalten muss.



Kontrollfragen

1. Wann sollten Sie den Datentyp `char` gegenüber dem Datentyp `varchar` bevorzugen?
2. Wann sollten Sie den Datentyp `varchar` gegenüber dem Datentyp `char` bevorzugen?
3. Welche Alternativen stehen Ihnen zum Datentyp `text` zur Verfügung? Wann sollten Sie jede einzelne von ihnen einsetzen?
4. Welche sind Ihrer Meinung nach die besten Verfahrensweisen zum Einsatz von Datentypen im SQL Server?
5. Welche Datentypen würden Sie für die folgenden Daten benutzen? Geben Sie auch zusätzliche Charakteristika wie etwa die Anzahl der Elemente der Zeichenfolge oder die Genauigkeit und die Dezimalstellen an.
 - a. Postleitzahl
 - b. Geburtsdatum
 - c. Baujahr eines Autos
 - d. Fahrgestellnummer eines Autos
 - e. Artikelnummer, wobei 95% der Artikel eine zehnstellige alphanumerische Eingabe erfordern. Die verbleibenden 5% variieren in der Größe, verlangen aber nicht mehr als 10 Ziffern. Der Firmenbesitzer erwägt, ob nicht sämtliche Artikel besser eine zehnstellige Artikelnummer haben sollten.
 - f. Firmenname, wobei der Name in der Länge zwischen 10 und 50 Zeichen variieren kann.
 - g. Ein Datumswert in einer nicht langfristig angelegten Datenbank. Die Daten bewegen sich in einem Wertebereich von 10 Jahren.
 - h. Währungsdaten, wobei Sie sechs Dezimalstellen nach dem Trennzeichen in Rechnung stellen müssen.
6. Sie haben mit der Anweisung `CREATE TABLE` die folgende Tabelle erstellt. Welches ist die maximale Größe einer Zeile in dieser Tabelle?

```
CREATE TABLE tabelle1
(
  id char(10),
  vname char(15) NULL,
  nname char(15) NOT NULL,
  bemerkung varchar(255)
)
```

7. Ist die folgende CREATE TABLE-Anweisung implementierbar? Wenn ja, warum; wenn nein, warum nicht? Welches ist die maximale Zeilengröße?

```
CREATE TABLE phooey  
(id char(10) NOT NULL,  
vname char(15) NULL,  
nname char(15) NOT NULL,  
bemerkung varchar(255),  
notizen text,  
anweisungen varchar(255),  
haus_bild image)
```

Übungen

Schreiben Sie als Erstes die Antworten auf die folgenden Fragen auf. Wenn Sie sich sicher sind, dass die Antworten stimmen, erstellen Sie die folgenden Objekte in der Datenbank:

1. Wie würden Sie die folgenden benutzerdefinierten Datentypen erstellen?

```
plz  
telefon_nummer  
artikel_id  
fax  
email
```

2. Wie müssen SQL-Anweisungen aussehen, mit denen eine Tabelle erstellt werden kann, die Daten über Artikel enthält? Die Daten sollten aus der Artikelbezeichnung, dem Namen, der Adresse, der Stadt, dem Bundesland, der Postleitzahl, dem Besitzer, einer Kontaktinformation, der Faxnummer und der E-Mail-Adresse bestehen.
3. Wie müssen SQL-Anweisungen aussehen, mit denen eine Tabelle erstellt werden kann, die Daten über Artikelverkäufe enthält? Die Daten sollten aus der Artikelbezeichnung, dem Verkaufsdatum, den Umsätzen, den Gewinnen und den Einlagen bestehen.



Daten mit
SELECT abrufen

Sobald Sie Daten in Ihrer Datenbank haben, müssen Sie wissen, wie Sie die gewünschten Daten abrufen können und wie sich Daten bearbeiten lassen. In der heutigen Lektion beginnen Sie mit einer grundlegenden Anweisung und erfahren, wie Sie ganz bestimmte Spalten auswählen können. Anschließend erweitern Sie diese einfache SELECT-Anweisung mit Techniken zur Datenmanipulation und -konversion. Sie erfahren weiterhin, wie Sie die Daten in bestimmten Zeilen in Tabellen auswählen und doppelt vorkommende Informationen ausschalten können. Die Lektion endet dann mit fortgeschrittenen Themen zu Abfragetechniken wie etwa Unterabfragen, Verknüpfungen und Datenkorrelationen.

Diese Lektion ist zwar recht lang, aber auch sehr wichtig – SQL ist eine grundlegende Sprache, die von nahezu jedem vorhandenen relationalen Datenbanksystem eingesetzt wird. Die meisten der hier genannten Anweisungen können Sie daher ohne oder mit nur geringfügigen Änderungen für nahezu jede Datenbank verwenden. Nehmen Sie sich also die nötige Zeit und beginnen Sie nun mit der grundlegenden SELECT-Anweisung.



SQL Server 2000 unterstützt sowohl die Syntax nach dem ANSI-Standard SQL-92 als auch die eigene SQL Server-Version von Microsoft, der Sie das gesamte Buch hindurch als Transact-SQL begegnen.

10.1 Einfache SELECT-Anweisungen

Abfragen zum Abrufen von Informationen aus Ihrer Datenbank erstellen Sie mit dem SQL Server Query Analyzer innerhalb oder außerhalb des SQL Server Enterprise Managers sowie mit Befehlszeilenprogrammen wie `osql`. Es gibt auch noch weitere Programme, die Sie zu diesem Zweck einsetzen können, beispielsweise MSQuery und das Werkzeug SQL Server English Query. Darüber hinaus stehen noch Programme und Werkzeuge von Drittanbietern zur Verfügung. In diesem Buch konzentrieren wir uns ausschließlich auf den SQL Server Query Analyzer.

Die SELECT-Anweisung besteht aus drei Basiskomponenten: SELECT, FROM und WHERE. Die grundlegende Syntax lautet:



```
SELECT <Spaltenliste>
FROM <Tabellenliste>
WHERE <Suchkriterien>
```

Der SELECT-Teil der Anweisung legt die Spalten fest, die Sie abrufen möchten. Die FROM-Klausel gibt die Tabellen an, deren Spalten abgerufen werden sollen, während die WHERE-Klausel dazu da ist, die von Ihrer Abfrage zurückzuliefernden Zeilen auszuwählen.

Die vollständige Syntax der SELECT-Anweisung lautet:

```
SELECT [ ALL | DISTINCT ] [ TOP n [PERCENT] [ WITH TIES] ]
<Auswahlliste>
[ INTO neue_Tabelle ]
[ FROM <Tabellenquellen> ]
[ WHERE < Suchkriterien> ]
[ GROUP BY [ALL] group_by_Ausdruck [...n]
[ WITH { CUBE | ROLLUP } ]]
[ HAVING < Suchkriterien> ]
[ ORDER BY { Spaltenname[ ASC | DESC ] } [...n] ]
[ COMPUTE { { AVG | COUNT | MAX | MIN | SUM } (Ausdruck) }
[,...n] [ BY Ausdruck [...n] ]
[ FOR BROWSE ] [ OPTION (<Abfragesteuerung> [...n] )
```

SELECT * FROM Tabellename stellt wohl die Basis aller Anfragen dar. Wenn Sie für *Spaltenliste* einen Stern (*) einsetzen, werden sämtliche Spalten der Tabelle zurückgeliefert. Intern wandelt SQL Server den * in eine Liste sämtlicher Spalten der Tabelle.

Sie können nun für die Datenbank *pubs* die folgende Abfrage zur Auswahl sämtlicher Spalten und Zeilen der Tabelle *employee* ausführen:



```
SELECT *
FROM employee
```



emp_id	fname	minit	lname	job_id	job_lvl	...
PMA42628M	Paolo	M	Accorti	13	...	
PSA89086M	Pedro	S	Afonso	14	...	
VPA30890F	Victoria	P	Ashworth	6	...	
H-B39728F	Helen		Bennett	12	...	
L-B31947F	Lesley		Brown	7	...	
F-C16315M	Francisco		Chang	4	...	
...	...	...	...	...	...	
GHT50241M	Gary	H	Thomas	9	...	
DBT39435M	Daniel	B	Tonini	11	...	

(43 row(s) affected)

Zur Auswahl bestimmter Spalten muss jede vorkommende Spalte mit einem Komma (,) getrennt werden. Nach dem letzten Spaltennamen dürfen Sie kein Komma setzen.



```
SELECT Spaltenname [, Spaltenname...]
FROM Tabellename
```

Die folgende Abfrage wählt den Vornamen, den Nachnamen und die Mitarbeiter-ID für jeden Mitarbeiter aus der Tabelle *employee* aus:



```
SELECT fname, lname, emp_id
FROM employee
```



fname	lname	emp_id
-----	-----	-----
Paolo	Accorti	PMA42628M
Pedro	Afonso	PSA89086M
Victoria	Ashworth	VPA30890F
Helen	Bennett	H-B39728F
Lesley	Brown	L-B31947F
Francisco	Chang	F-C16315M
. . .	. . .	
. . .	. . .	
. . .	. . .	
Gary	Thomas	GHT50241M
Daniel	Tonini	DBT39435M

(43 row(s) affected)

Wenn Sie eine Abfrage mit der Anweisung `SELECT *` ausführen, stimmt die Spaltenanordnung mit der in der `CREATE TABLE`-Anweisung festgelegten Anordnung der Spalten überein. Bei der Auswahl von Spalten aus einer Tabelle muss die Anordnung in *Spaltenliste* aber nicht unbedingt der Anordnung in der Tabelle entsprechen. Durch eine Neuordnung der Spalten in *Spaltenliste* sind Sie ohne weiteres in der Lage, diese Anordnung in Ihrer Abfrage entsprechend abzuändern.

Nehmen Sie dazu die oben stehende Abfrage und ordnen Sie die Spaltenfolge neu an. Es werden immer noch dieselben Informationen zurückgeliefert, die aber diesmal in einer anderen Spaltenfolge angezeigt werden:



```
SELECT emp_id, lname, fname
FROM employee
```



emp_id	lname	fname
-----	-----	-----
PMA42628M	Accorti	Paolo
PSA89086M	Afonso	Pedro
VPA30890F	Ashworth	Victoria
H-B39728F	Bennett	Helen

```

. . .      . . .
. . .      . . .
. . .      . . .
GHT50241M   Thomas   Gary
DBT39435M   Tonini   Daniel

```

(43 row(s) affected)

Spaltenüberschriften ändern

Bei der Anzeige der Ergebnisse einer Abfrage entsprechen die Spaltenüberschriften den Namen, die auch in *Spaltenliste* verwendet worden sind. Anstelle von Überschriften wie `lname` oder `fname` ist es aber sinnvoller, aussagekräftigere Spaltenüberschriften wie etwa `FirstName` (bzw. Vorname) oder `LastName` (bzw. Nachname) zu verwenden. Sie erreichen dies, indem Sie einen Alias für die Spaltenüberschriften einsetzen. Einen Alias für die Spaltenüberschriften vergeben Sie entweder über die Syntax von SQL Server 2000 oder über die ANSI SQL-Syntax.

Einen Alias für Spalten in SQL Server vergeben Sie auf zweierlei Art und Weise:



```

SELECT Spaltenüberschrift = Spaltenname
FROM Tabellename

```

Die ANSI-Syntax sieht folgendermaßen aus:

```

SELECT Spaltenname 'Spaltenüberschrift'
FROM Tabellename

```



Falls der von Ihnen verwendete Alias Leerzeichen enthält oder ein Schlüsselwort von SQL Server darstellt, müssen Sie ihn in einfache Anführungszeichen oder die Trennzeichen für SQL Server-Bezeichner (`[ ]`) einschließen. Das folgende Beispiel verwendet Leerzeichen und eckige Klammern:

```

SELECT lname AS 'Last Name', fname AS [First Name]
FROM employee

```

Das nächste Beispiel arbeitet mit einem SQL-Schlüsselwort:

```

SELECT 'count' = Count(*)
FROM employee

```



Ihre Abfrage können Sie mit der SQL Server 2000-Syntax neu formulieren:

```
SELECT EmployeeID = emp_id, LastName = lname, FirstName = fname
FROM employee
```

Sie können aber auch die ANSI SQL-Syntax einsetzen:

```
SELECT emp_id AS EmployeeID, lname AS LastName, fname AS FirstName
FROM employee
```



Das Schlüsselwort **AS** ist nicht erforderlich. Die folgende Anweisung liefert die gleichen Ergebnisse wie bei der obigen Abfrage zurück:

```
SELECT empID EmployeeID, lname LastName, fname FirstName FROM employee.
```

Beide Abfragen liefern die gleichen Ergebnisse:

EmployeeID	LastName	FirstName
PMA42628M	Accorti	Paolo
PSA89086M	Afonso	Pedro
VPA30890F	Ashworth	Victoria
H-B39728F	Bennett	Helen
L-B31947F	Brown	Lesley
F-C16315M	Chang	Francisco
. . .		. . .
. . .	. . .	
. . .	. . .	
GHT50241M	Thomas	Gary
DBT39435M	Tonini	Daniel

43 row(s) affected)

Verwenden von Literalen



Mit Literalen gestalten Sie Ihre Ausgaben noch übersichtlicher. Ein *Literal* ist eine von einfachen oder doppelten Anführungszeichen eingeschlossene Zeichenfolge, die in der *Spaltenliste* enthalten ist und als zusätzliche Spalte im Abfrageergebnis angezeigt wird. Im Abfrageergebnis erscheint dann neben der eigentlichen Spalte ein Label für diese Spalte.

Die Syntax zum Einbinden von literalen Werten lautet:



```
SELECT 'literal' [, 'literal'...]
```

Die folgende Abfrage gibt Vorname, Nachname, eine Spalte mit der literalen Zeichenfolge Employee ID: und die Mitarbeiter-IDs für alle Mitarbeiter aus der Tabelle *employee* zurück:



```
SELECT fname, lname, 'Employee ID:', emp_id
FROM employee
```



fname	lname	emp_id	
Paolo	Accorti	Employee ID:	PMA42628M
Pedro	Afonso	Employee ID:	PSA89086M
Victoria	Ashworth	Employee ID:	VPA30890F
Helen	Bennett	Employee ID:	H-B39728F
. . .	. . .		
. . .	. . .		
. . .	. . .		
Gary	Thomas	Employee ID:	GHT50241M
Daniel	Tonini	Employee ID:	DBT39435M

(43 row(s) affected)

10.2 Daten manipulieren

Sie können die Daten in Ihren Abfrageergebnissen so manipulieren, dass neue Spalten entstehen, die berechnete Werte, neue Zeichenfolgen, ein umgewandeltes Datum und anderes mehr enthalten. Weiterhin haben Sie die Möglichkeit, Ihre Abfrageergebnisse unter Verwendung von arithmetischen Operatoren, mathematischen Funktionen, Zeichenfolgenfunktionen, Funktionen für Datum/Zeit-Umwandlungen und Systemfunktionen noch weitgehender zu bearbeiten. Für eine noch einfachere Datenmanipulation können Sie die Funktion `CONVERT` einsetzen, mit deren Hilfe Sie einen gegebenen Datentyp in einen anderen umwandeln.

Arithmetische Operatoren

Arithmetische Operatoren können Sie auf die folgenden Datentypen anwenden: `bigint`, `int`, `smallint`, `tinyint`, `numeric`, `decimal`, `float`, `real`, `money` und `smallmoney`. Tabelle 10.1 zeigt die arithmetischen Operatoren und die Datentypen, die Sie zusammen mit den Operatoren verwenden können.

Datentyp	Addition	Subtraktion	Division	Multiplikation	Modulo
	+	-	/	*	%
<code>bigint</code>	ja	ja	ja	ja	ja
<code>decimal</code>	ja	ja	ja	ja	nein
<code>float</code>	ja	ja	ja	ja	nein
<code>int</code>	ja	ja	ja	ja	ja
<code>money</code>	ja	ja	ja	ja	nein
<code>numeric</code>	ja	ja	ja	ja	nein
<code>real</code>	ja	ja	ja	ja	nein
<code>smallint</code>	ja	ja	ja	ja	ja
<code>smallmoney</code>	ja	ja	ja	ja	nein
<code>tinyint</code>	ja	ja	ja	ja	ja

Tabelle 10.1: Datentypen und arithmetische Operationen

Rangfolge der Operatoren



Bei arithmetischen Operationen gibt es zwei Ebenen der Rangfolge: Die *Rangfolge der Datentypen* und die *Rangfolge der Operatoren*.

- Die Rangfolge der Datentypen wird verwendet, wenn arithmetische Operationen auf verschiedene Datentypen angewendet werden. Bei Verwendung von verschiedenen Datentypen wird dabei der niedrigere Datentyp in den höheren Datentyp umgewandelt. Wenn Sie beispielsweise eine Multiplikation eines Typs `smallint` mit einem Typ `int` durchführen, wird das Ergebnis vom Typ `int` sein. Die einzige Ausnahme hiervon bildet die Verwendung des Datentyps `money`, die immer zum Datentyp `money` führt.
- Die Rangfolge der Operatoren wird verwendet, wenn mehrere Operatoren eingesetzt werden. Die Operatoren folgen immer den normalen Regeln für die Operatorrangfolge, wobei Modulo zuerst ausgewertet wird. Anschließend folgt die Multiplikation und Division, danach die Addition und Subtraktion – von links nach rechts gelesen.

Wie bei arithmetischen Operationen üblich, können Sie die Rangfolge verändern, indem Sie Ausdrücke in Klammern einschließen. Dabei wird der ganz innen stehende (am tiefsten geschachtelte) Ausdruck zuerst ausgewertet. Durch Klammern kann ein arithmetischer Ausdruck darüber hinaus übersichtlicher gestaltet werden.

Dazu ein Beispiel:

$5 + 5 * 5 = 30$ (die Multiplikation wird zuerst ausgeführt)

aber:

$(5 + 5) * 5 = 50$ (zuerst wird der eingeklammerte Ausdruck ausgeführt)

Es ist generell empfehlenswert, dass Sie Ihre arithmetischen Operationen in Klammern einschließen, um die Lesbarkeit und Eindeutigkeit des Transact-SQL-Codes zu verbessern.

Mathematische Funktionen

Mit den mathematischen Funktionen können Sie häufig benötigte Operationen auf mathematische Daten anwenden.

Mathematische Daten erzeugen Sie unter Verwendung der folgenden Syntax:



SELECT Funktionsname(Parameter)

Tabelle 10.2 enthält die mathematischen Funktionen sowie deren Parameter und Ergebnisse. Die Beispiele schließen Operationen wie etwa die Ermittlung des Absolutwertes, die Berechnung von trigonometrischen Funktionswerten, die Ableitung von Quadratwurzeln und die Potenzierung ein. In Tabelle 10.3 gibt dazu einige Beispiele an.

Funktion	Ergebnis
ABS(numerischer Ausdruck)	Absolutwert
ACOS ASIN(Gleitkommaausdruck)	Gibt den Winkel im Bogenmaß wieder, dessen Kosinus der angegebene float-Ausdruck ist.
ATAN(Gleitkommaausdruck)	Gibt den Winkel im Bogenmaß wieder, dessen Tangens der angegebene float-Ausdruck ist.
ATN2(Gleitkommaausdruck1, Gleitkommaausdruck2)	Gibt den Winkel im Bogenmaß zurück, dessen Tangens zwischen den beiden angegebenen float-Ausdrücken liegt.
COS SIN (Gleitkommaausdruck)	Kosinus oder Sinus (im Bogenmaß)

Tabelle 10.2: Mathematische Funktionen



Funktion	Ergebnis
COT TAN(Gleitkommaausdruck)	Kotangens oder Tangens (im Bogenmaß)
CEILING(num. Ausdruck)	Kleinster Ganzzahlwert, der größer oder gleich einem angegebenen Wert ist.
DEGREES(num. Ausdruck)	Umwandlung von Winkelangaben im Bogenmaß in Grad
EXP(Gleitkommaausdruck)	Exponentieller Wert eines angegebenen Wertes
FLOOR(num. Ausdruck)	Größter Ganzzahlwert, der kleiner oder gleich einem angegebenen Wert ist.
LOG(Gleitkommaausdruck)	Natürlicher Logarithmus
LOG10(Gleitkommaausdruck)	Logarithmus zur Basis 10
PI()	Konstante 3.141592653589793
POWER(num. Ausdruck, y)	Wert von <i>num. Ausdruck</i> in der Potenz von <i>y</i>
RADIANS(num. Ausdruck)	Umwandlung von Grad in Bogenmaß
RAND([seed])	Zufällige Gleitkommazahl zwischen 0 und 1
ROUND(num. Ausdruck, Länge)	Numerischer Ausdruck gerundet auf die angegebene Länge; Länge als Ganzzahlwert
SIGN(num. Ausdruck)	Positiv, negativ oder Null
SQUARE(Gleitkommaausdruck)	Quadratzahl des angegebenen Werts
SQRT(Gleitkommaausdruck)	Quadratwurzel des angegebenen Werts

Tabelle 10.2: Mathematische Funktionen

Anweisung	Ergebnis
SELECT SQRT(9)	3,0
SELECT ROUND(1234.56, 0)	1235
SELECT ROUND(1234.56, 1)	1234,60
SELECT ROUND(\$1234.56, 1)	1.234,60
SELECT POWER (2,8)	256,0
SELECT FLOOR(1332.39)	1332
SELECT ABS(-365)	365

Tabelle 10.3: Beispiele für den Aufruf mathematischer Funktionen



Wenn Sie mathematische Funktionen auf Währungsdatentypen anwenden, sollten Sie dem Datentyp immer ein Dollarzeichen (\$) voranstellen. Anderenfalls wird der Wert als numerischer Wert mit 4 Dezimalstellen behandelt.



Wenn Sie den Datentyp `float` verwenden, erhalten Sie unter Umständen unerwartete Ergebnisse von SQL Server Query Analyzer und anderen Programmen. Falls Sie zum Beispiel

```
SELECT ROUND(12,3456789E+5,2)
```

ausführen, erhalten Sie zurück:

```
1234567,88999999999
```

Dieses Ergebnis ist eine Funktion von ODBC (Open Database Connectivity). SQL Server führt zwar immer noch die Rundung aus, jedoch sind Gleitkommatypen ungenau und liefern nicht notwendigerweise das von Ihnen erwartete Ergebnis zurück. Im Allgemeinen ist es besser, auf Gleitkommatypen zu verzichten.

Zeichenfolgenfunktionen

Beim Umgang mit Zeichendaten können Sie sich zur Manipulation der Daten verschiedener Zeichenfolgenfunktionen bedienen (siehe Tabelle 10.4). Mit den meisten Zeichenfolgenfunktionen können nur die Datentypen `char`, `nchar`, `varchar` und `nvarchar` manipuliert werden; andere Typen müssen daher zuvor umgewandelt werden. Zeichendaten können Sie unter Verwendung der folgenden Syntax bearbeiten:



```
SELECT Funktionsname(Parameter)
```

Funktion	Ergebnis
'Ausdruck' + 'Ausdruck'	Verkettet zwei oder mehr Zeichenfolgen.
ASCII(Zeichenausdruck)	Gibt den ASCII-Code des ersten Zeichens von links zurück.
CHAR(Ganzzahlausdruck)	Zeichenäquivalent des ASCII-Wertes
CHARINDEX(Muster, Ausdruck)	Liefert die Anfangsposition des angegebenen Musters zurück.
DIFFERENCE(Zeichenausdruck1, Zeichenausdruck2)	Vergleicht zwei Zeichenfolgen und wertet deren Übereinstimmung aus; liefert einen Wert von 0 bis 4 für den besten Treffer zurück.

Tabelle 10.4: Zeichenfolgenfunktionen

Funktion	Ergebnis
LEFT(Zeichenausdruck, Ganzzahlausdruck)	Gibt <i>Ganzzahlausdruck</i> Zeichen der Zeichenfolge <i>Zeichenausdruck</i> von links beginnend zurück.
LOWER(Zeichenausdruck)	Wandelt in Kleinschreibung um.
LTRIM(Zeichenausdruck)	Liefert Daten ohne führende Leerzeichen zurück.
NCHAR(Ganzzahlausdruck)	Liefert das dem <i>Ganzzahlausdruck</i> entsprechende Unicode-Zeichen zurück.
PATINDEX('%Muster%', Ausdruck)	Liefert die Anfangsposition des ersten Auftretens in <i>Ausdruck</i> zurück.
QUOTENAME('Zeichenfolge1', 'Trennzeichen')	Liefert eine Unicode-Zeichenfolge (nvarchar(129)) mit gültigen SQL Server-Trennzeichen zurück.
REPLACE('Zeichenfolge1', 'Zeichenfolge2', 'Zeichenfolge3')	Ersetzt sämtliche Vorkommen von <i>Zeichenfolge2</i> in <i>Zeichenfolge1</i> durch <i>Zeichenfolge3</i> .
REPLICATE(Zeichenausdruck, Ganzzahlausdruck)	Wiederholt <i>Zeichenausdruck</i> so oft, wie es <i>Ganzzahlausdruck</i> angibt.
REVERSE(Zeichenausdruck)	Liefert die Umkehrung von <i>Zeichenausdruck</i> .
RIGHT(Zeichenausdruck, Ganzzahlausdruck)	Gibt <i>Ganzzahlausdruck</i> Zeichen der Zeichenfolge <i>Zeichenausdruck</i> von rechts beginnend zurück.
RTRIM(Zeichenausdruck)	Liefert Daten ohne nachgestellte Leerzeichen.
SOUNDEX(Zeichenausdruck)	Liefert einen vierstelligen (SOUNDEX-) Code zurück, der ein Maß für die Ähnlichkeit zweier Zeichenfolgen ist.
SPACE(Ganzzahlausdruck)	Liefert eine Zeichenfolge mit Leerzeichen zurück, deren Anzahl gleich <i>Ganzzahlausdruck</i> ist.
STR(Gleitkommaausdruck[,Länge[,dezimal]])	Liefert Zeichendaten zurück, die aus umgewandelten numerischen Daten bestehen; <i>Länge</i> gibt die Gesamtlänge an, während die Anzahl der Leerzeichen rechts vom Dezimaltrennzeichen <i>dezimal</i> ist.
STUFF(Zeichenausdruck1, Start, Länge, Zeichenausdruck2)	Löscht Zeichen der Länge <i>Länge</i> im <i>Zeichenausdruck1</i> bei der Position <i>Start</i> und fügt dort <i>Zeichenausdruck2</i> ein.

Tabelle 10.4: Zeichenfolgenfunktionen (Forts.)

Funktion	Ergebnis
SUBSTRING(Ausdruck, Start, Länge)	Liefert einen Teil eines Zeichen- oder Binär- ausdrucks zurück.
UNICODE('nchar_Zeichenfolge')	Liefert den Unicode-Ganzzahlwert des ersten Zeichens von 'nchar_Zeichenfolge' zurück.
UPPER(Zeichenausdruck)	Wandelt in Großschreibung um.

Tabelle 10.4: Zeichenfolgenfunktionen (Forts.)

Zum Beispiel können Sie mit der folgenden Anweisung eine Spalte namens *Name* zurück-
geben, die aus einer Zeichenfolgenverkettung von *last name*, *first initial* und *employee ID*
besteht:



```
SELECT lname + ', ' + SUBSTRING(fname,1,1) + '.'  
AS Name, emp_id as EmployeeID  
FROM employee
```



Name	EmployeeID
-----	-----
Accorti, P.	PMA42628M
Afonso, P.	PSA89086M
Ashworth, V.	VPA30890F
Bennett, H.	H-B39728F
.	.
.	.
.	.
Sommer, M.	MFS52347M
Thomas, G.	GHT50241M
Tonini, D.	DBT39435M

(43 row(s) affected)

Tabelle 10.5 zeigt weitere Beispiele für Zeichenfolgenfunktionen.

Anweisung	Ergebnis
SELECT ASCII('G')	71
SELECT LOWER('ABCDE')	abcde
SELECT PATINDEX('%BC%', 'ABCDE')	2
SELECT RIGHT('ABCDE',3)	CDE
SELECT REVERSE('ABCDE')	EDCBA

Tabelle 10.5: Weitere Beispiele für Zeichenfolgenfunktionen

Datumsfunktionen

Datetime-Werte manipulieren Sie mit den Datumsfunktionen. Datumsfunktionen können in *Spaltenliste*, der *WHERE*-Klausel und an jeder Stelle verwendet werden, an der auch ein Ausdruck stehen kann. Für Datumsfunktionen benutzen Sie die folgende Syntax:



SELECT Datumsfunktion(Parameter)

Datetime-Werte, die Sie als Parameter übergeben, müssen Sie in einfache oder doppelte Anführungszeichen einschließen. Manche Funktionen erwarten einen Parameter, der als *datepart* (Datumseinheit) bekannt ist. In Tabelle 10.6 sind die *datepart*-Werte und deren Abkürzungen aufgeführt.

datepart	Abkürzung	Werte
day	dd	1 – 31
day of year	dy	1 – 366
hour	hh	0 – 23
millisecond	ms	0 – 999
minute	mi	0 – 59
month	mm	1 – 12
quarter	qq	1 – 4
second	ss	0 – 59
week	wk	0 – 53
weekday	dw	1 – 7 (Son – Sam)
year	yy	1753 – 9999

Tabelle 10.6: *datepart*-Werte

Tabelle 10.7 führt die Datumsfunktionen, ihre Parameter und Ergebnisse auf. Tabelle 10.8 gibt einige Beispiele für Datumsfunktionen.

Funktion	Ergebnis
DATEADD(datepart, Zahl, Datum)	Fügt dem Datumsteil eines Datums eine Zahl hinzu
DATEDIFF(datepart, Datum1, Datum2)	Nummer des Datumsteils zwischen zwei Daten

Tabelle 10.7: *Datumsfunktionen*



Funktion	Ergebnis
DATENAME(datepart, Datum)	Liefert den ASCII-Wert für den Wert des angegebenen Datumsteils von Datum.
DATEPART(datepart, Datum)	Liefert den Ganzzahlwert eines angegebenen datepart für das angegebenen Datum zurück.
DAY (Datum)	Gibt einen Ganzzahlwert zurück, der den Tag repräsentiert.
GETDATE()	Aktuelles Datum und Zeit im internen Format
MONTH (Datum)	Gibt einen Ganzzahlwert zurück, der den Monat repräsentiert.
YEAR (Datum)	Gibt einen Ganzzahlwert zurück, der das Jahr repräsentiert.

Tabelle 10.7: Datumsfunktionen (Forts.)

Funktion	Ergebnis
SELECT DATEDIFF(mm, '1/1/97', '12/31/99')	35
SELECT GETDATE()	Apr 29, 1997 2:10AM
SELECT DATEADD(mm, 6, '1/1/97')	Jul 1, 1997 2:10AM
SELECT DATEADD(mm -5, '10/6/97')	May 6, 1997 2:10AM

Tabelle 10.8: Beispiele für Datumsfunktionen

Das folgende Beispiel zeigt eine komplexere Abfrage mit vielen Elementen, die Sie bis jetzt kennen gelernt haben:



```
SELECT emp_id AS EmployeeID, lname + ', ' + SUBSTRING(fname,1,1) + '.'
AS Name,
-'Has been employed for ', DATEDIFF(year, hire_date, getdate()), '
years.'
FROM employee
```



EmployeeID	Name			
PMA42628M	Accorti, P.	Has been employed for	5	years.
PSA89086M	Afonso, P.	Has been employed for	7	years.
VPA30890F	Ashworth, V.	Has been employed for	7	years.
H-B39728F	Bennett, H.	Has been employed for	8	years.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
MFS52347M	Sommer, M.	Has been employed for	7	years.



```
GHT50241M Thomas, G. Has been employed for 9 years.
DBT39435M Tonini, D. Has been employed for 7 years.

(43 row(s) affected)
```

10.3 Systemfunktionen

Zum Abrufen von Informationen aus den Systemtabellen können Sie mehrere eingebaute Systemfunktionen nutzen. Die Ergebnisdaten erhalten Sie über die folgende Syntax:



```
SELECT Funktionsname(Parameter)
```

Systemfunktionen können Sie in *Spaltenliste*, der *WHERE*-Klausel und überall dort verwenden, wo auch ein Ausdruck stehen kann.

Tabelle 10.9 enthält einige Systemfunktionen mit ihren Parametern und Ergebnissen.

Funktion	Ergebnisse
COALESCE(Ausdruck1, Ausdruck2, ... AusdruckN)	Gibt den ersten Ausdruck zurück, der nicht NULL ist
COL_NAME(Tabellen_Id, Spalten_Id)	Gibt den Spaltennamen zurück
COL_LENGTH('Tabellenname', 'Spaltenname')	Gibt die Spaltenlänge zurück
DATALENGTH('Ausdruck')	Gibt die tatsächliche Länge von <i>Ausdruck</i> eines beliebigen Datentyps zurück
DB_ID(['Datenbankname'])	Gibt eine Datenbank-ID zurück
DB_NAME([Datenbank_Id])	Gibt einen Datenbanknamen zurück
GETANSINULL(['Datenbankname'])	Gibt die standardmäßige NULL-Zulässigkeit der Datenbank zurück
HOST_ID()	Gibt die Host-Id der Arbeitsstation zurück
HOST_NAME()	Gibt den Host-Computernamen zurück
IDENT_INCR('Tabellenname')	Gibt den Schrittweitenwert zurück, der bei der Erstellung von Identitätsspalten festgelegt wurde

Tabelle 10.9: Systemfunktionen



Funktion	Ergebnisse
IDENT_SEED('Tabellen_Name')	Gibt den Ausgangswert zurück, der während der Erstellung von Identitätsspalten festgelegt wurde
INDEX_COL('Tabellenname', Index_Id, Schlüssel_Id)	Gibt den indizierten Spaltennamen zurück
ISDATE(variable Spaltenname)	Prüft die Gültigkeit eines Datumsformats. Gibt 1 für <i>gültig</i> zurück; andernfalls 0
ISNULL(Ausdruck, Wert)	Ersetzt NULL durch den angegebenen Ersatzwert
ISNUMERIC(Variable Spaltenname)	Prüft die Gültigkeit eines numerischen Formats. Gibt 1 für <i>gültig</i> zurück; andernfalls 0
NULLIF(Ausdruck1, Ausdruck2)	Gibt NULL zurück, wenn <i>Ausdruck1</i> = <i>Ausdruck2</i> ist
OBJECT_ID('Objekt_Name')	Gibt eine Objekt-Id der Datenbank zurück
OBJECT_NAME(Objekt_Id)	Gibt einen Objektnamen der Datenbank zurück
STATS_DATE(Tabellen_Id, Index_Id)	Gibt das Datum zurück, an dem die Indexstatistik zuletzt aktualisiert worden ist
SUSER_ID(['Server_Benutzername'])	Gibt die Benutzer-Id des Servers zurück
SUSER_NAME([Server_Id])	Gibt den Benutzernamen des Servers zurück
USER_ID(['Benutzername'])	Gibt die Benutzer-Id der Datenbank zurück
USER_NAME([User_Id])	Gibt den Benutzernamen der Datenbank zurück

Tabelle 10.9: Systemfunktionen (Forts.)

Die folgende Abfrage verwendet zwei Systemfunktionen, um den Namen der zweiten Spalte in der Tabelle *employee* zurückzuliefern:



```
SELECT COL_NAME(OBJECT_ID('employee'),2)
fname
(1 row(s) affected)
```



10.4 Datenumwandlungen

Viele Funktionen erwarten die Daten in einem ganz bestimmten Format oder Datentyp. Es ist daher nützlich und notwendig, einen Datentyp gegebenenfalls in einen anderen umwandeln zu können. Zum Ändern des Datentyps setzen Sie die Funktion `CONVERT` oder `CAST` ein. Die Funktion `CONVERT` können Sie überall dort verwenden, wo auch Ausdrücke möglich sind. `CONVERT` hat die folgende Syntax:



`CONVERT(Datentyp[(Länge)], Ausdruck [,Format])`

Die Tabelle 10.10 gibt die mit der Funktion verbundenen Formatparameter, einen Hinweis zum internationalen Standard und das Format der Ausgabe an.

Format ohne Jahrhundert jj	Format mit Jahrhundert jjjj	Standard	Datum Formatausgabe
-	0 oder 100	Standard	mon tt jjjj hh:mm AM (oder PM)
1	101	USA	mm/tt/jj
2	102	ANSI	jj.mm.tt
3	103	Britisch/Französisch	tt/mm/jj
4	104	Deutsch	tt.mm.jj
5	105	Italienisch	tt-mm-jj
6	106	-	tt mon jj
7	107	-	mon tt, jj
8	108	-	hh:mi:ss
-	9 oder 109	Standardwert + Millisekunden	mon tt, jjjj hh:mm:ss.ms AM_ (oder PM)
10	110	USA	mm-tt-jj
11	111	Japan	jj/mm/tt
12	112	ISO	jjmmmtt
-	13 oder 113	europ. Standard + Millisekunden	tt mon jjjj hh:mm:ss.ms(24h)
14	114	-	hh:mm:ss.ms(24h)

Tabelle 10.10: Verwenden der Funktion `CONVERT` für den Datentyp `Datetime`

Mit der folgenden Abfrage wandeln Sie das aktuelle Datum in eine Zeichenfolge mit acht Zeichen im ANSI-Format um:



```
SELECT CONVERT(CHAR(8),GETDATE(),2)
```



```
-----  
01.01.09
```

(1 row(s) affected)



```
SELECT emp_id AS EmployeeID, lname + ', ' + SUBSTRING(fname,1,1) + '.'  
AS Name, 'Has been employed for ' + CONVERT(CHAR(2),  
(DATEDIFF(year, hire_date, getdate())) + ' years.'  
FROM employee
```



```
EmployeeID Name  
-----  
PMA42628M Accorti, P. Has been employed for 9 years.  
PSA89086M Afonso, P. Has been employed for 11 years.  
VPA30890F Ashworth, V. Has been employed for 11 years.  
H-B39728F Bennett, H. Has been employed for 12 years.  
. . .  
. . .  
. . .  
MFS52347M Sommer, M. Has been employed for 11 years.  
GHT50241M Thomas, G. Has been employed for 13 years.  
DBT39435M Tonini, D. Has been employed for 11 years.
```

(43 row(s)affected)



Dieses Beispiel baut auf der Abfrage auf, die Sie im Abschnitt zu Datumsfunktionen weiter oben in der heutigen Lektion ausgeführt haben. Hier fassen Sie nun die letzten drei Spalten unter Verwendung der Funktion CONVERT und der Zeichenfolgenverkettung zu einer einzigen Spalte zusammen.

10.5 Zeilen auswählen

Bis jetzt haben Sie sich verschiedene Methoden dafür angesehen, die Spalten im Resultset einer Abfrage abzurufen, zu formatieren und zu manipulieren. Jetzt werden Sie erfahren, wie Sie auf der Grundlage bestimmter Suchkriterien die zurückzuliefernden Zeilen festlegen können. Dazu benutzen Sie die **WHERE**-Klausel der **SELECT**-Anweisung. Die Suchbedingungen schließen Vergleichsoperatoren, Bereichswerte, Listen, Zeichenfolgenmuster, unbekannte Werte, Kombinationen und Negationen dieser Bedingungen ein.

Die grundlegende Syntax für die Auswahl von Zeilen lautet wie folgt:



```
SELECT <Spaltenliste>
FROM <Tabellenliste>
WHERE <Suchkriterien>
```

Vergleichsoperatoren

Sie können die Suchbedingungen über die Vergleichsoperatoren (siehe Tabelle 10.11) festlegen, wobei die Zeilen über den Vergleich bestimmter Spaltenwerte mit einem Ausdruck oder Wert ermittelt werden. Ausdrücke können Konstanten, Spaltennamen, Funktionen oder geschachtelte Unterabfragen enthalten. Falls Sie zwei verschiedene Datentypen (wie etwa **char** und **varchar**) oder Datumsstypen (wie etwa **datetime** und **smalldatetime**) miteinander vergleichen möchten, müssen Sie diese in einfache Anführungszeichen einschließen. Doppelte Anführungszeichen wären zwar auch akzeptabel, aber einfache Anführungszeichen sind ANSI-kompatibel.

Operator	Beschreibung
=	Gleich
>	Größer als
<	Kleiner als
>=	Größer oder gleich
<=	Kleiner oder gleich
<>	Ungleich
!=	Ungleich
!>	Nicht größer als
!<	Nicht kleiner als
()	Rangfolge

Tabelle 10.11: Vergleichsoperatoren

Die Syntax der WHERE-Klausel unter Verwendung von Vergleichsoperatoren lautet wie folgt:



```
SELECT <Spaltenliste>
FROM <Tabellenliste>
WHERE <Spaltenname> Vergleichsoperator <Ausdruck>
```

Die folgende Abfrage liefert die Mitarbeiter-ID (*employee ID*), den Nachnamen (*last name*) und den Vornamen (*first name*) sämtlicher Mitarbeiter (*employees*) zurück, die vom Verleger (*publisher*) mit der *pub_id* 0877 beschäftigt werden:



```
SELECT emp_id, lname, fname
FROM employee
WHERE pub_id = '0877'
```



emp_id	lname	Fname
PMA42628M	Accorti	Paolo
VPA30890F	Ashworth	Victoria
H-B39728F	Bennett	Helen
.	.	.
.	.	.
.	.	.
M-R38834F	Rance	Martine
DBT39435M	Tonini	Daniel

(10 row(s) affected)



Falls Sie arithmetische Operatoren verwenden, die mit logischen Operatoren verknüpft sind, werden die arithmetischen Operatoren zuerst verarbeitet. Die Rangfolge können Sie selbstverständlich jederzeit mit Klammern verändern.

Bereiche

Mit dem Schlüsselwort **BETWEEN** können Sie Zeilen auf der Grundlage bestimmter Werte abrufen. Wie schon bei den Vergleichsoperatoren müssen Sie Bereichsangaben mit Zeichendatentypen (wie **char** und **varchar**) oder Datumentypen (wie **datetime** und **smalldatetime**) in einfache Anführungszeichen einschließen.

Die Syntax für die WHERE-Klausel unter Verwendung von Vergleichen lautet wie folgt:



```
SELECT Spaltenliste
FROM Tabellenliste
WHERE Spaltenname [NOT] BETWEEN Ausdruck AND Ausdruck
```

Die folgende Abfrage liefert den Nachnamen (*last name*) und die Mitarbeiter-ID (*employee ID*) für sämtliche Mitarbeiter (*employees*) zurück, die zwischen dem 01.10.92 und dem 31.12.92 eingestellt worden sind:



```
SELECT lname, emp_id
FROM employee
WHERE hire_date BETWEEN '10/1/92' AND '12/31/92'
```



lname	emp_id
Josephs	KFJ64308F
Paolino	MAP77183M

(2 row(s) affected)



Die BETWEEN-Klausel ist inklusiv; d.h., dass sowohl der 01.10.92 als auch der 31.12.92 als potentielle Einstellungsdaten einbezogen werden. Beachten Sie, dass in der BETWEEN-Klausel der kleinere Wert immer als erster genannt werden muss.

Listen

Mithilfe des Schlüsselwortes IN können Sie Zeilen mit Werten abrufen, die mit denjenigen in einer Liste übereinstimmen. Bereichsangaben mit Zeichendatentypen (wie char und varchar) oder Datumentypen (wie datetime und smalldatetime) sind in einfache Anführungszeichen einzuschließen.

Die Syntax für die WHERE-Klausel unter Verwendung von Listen lautet wie folgt:



```
SELECT <Spaltenliste>
FROM <Tabelleliste>
WHERE [NOT] Spaltenname [NOT] IN (Werteliste)
```

Falls Sie die Mitarbeiter finden möchten, die für Verleger mit einer *pub_id* 0877 oder 9999 tätig sind, führen Sie die folgende Abfrage aus:



```
SELECT emp_id, lname, fname
FROM employee
WHERE pub_id IN ('0877', '9999')
```



emp_id	lname	fname

PMA42628M	Accorti	Paolo
VPA30890F	Ashworth	Victoria
H-B39728F	Bennett	Helen
.	.	.
.	.	.
.	.	.
A-R89858F	Roulet	Annette
DBT39435M	Tonini	Daniel

(17 row(s) affected)

Bei Verwendung des Operators NOT können Sie auch Zeilen abrufen, die sich nicht in der Liste befinden. Wenn Sie sämtliche Mitarbeiter finden möchten, die nicht für den Verleger mit der *pub_id* 0877 oder 9999 arbeiten, formulieren Sie die folgende Abfrage:



```
SELECT emp_id, lname, fname
FROM employee
WHERE pub_id NOT IN ('0877', '9999')
```



emp_id	lname	fname

PSA89086M	Afonso	Pedro
F-C16315M	Chang	Francisco
PTC11962M	Cramer	Philip
A-C71970F	Cruz	Aria
AMD15433F	Devon	Ann
.....		
CGS88322F	Schmitt	Carine
MAS70474F	Smith	Margaret
HAS54740M	Snyder	Howard
MFS52347M	Sommer	Martin
GHT50241M	Thomas	Gary

(26 row(s) affected)



Versuchen Sie nach Möglichkeit, positive Suchkriterien zu verwenden. Vermeiden Sie den Einsatz von NOT, da der Abfrageoptimierer nicht in jedem Fall negative Suchkriterien erkennt. Bei einem NOT muss der SQL Server eine Menge Aufwand betreiben, um das Resultset ermitteln zu können. Die vorstehende Abfrage können Sie auch unter Verwendung der Anweisungen BETWEEN und AND formulieren.

Zeichenfolgen

Über das Schlüsselwort LIKE können Sie Zeilen auf der Grundlage von Teilen einer Zeichenfolge abrufen. LIKE wird zusammen mit Daten des Typs char, varchar, nchar, nvarchar, text, datetime und smalldatetime verwendet. Reguläre Ausdrücken gestatten Ihnen darüber hinaus den Einsatz von vier Platzhalterzeichen.

Die Syntax für die WHERE-Klausel in Verbindung mit dem Schlüsselwort LIKE lautet wie folgt:



```
SELECT <Spaltenliste>
FROM <Tabellenliste>
WHERE Spaltenname [NOT] LIKE 'string'
```

Folgende Platzhalterzeichen sind verfügbar:

- % Zeichenfolge mit Null oder mehr Zeichen
- _ Einzelnes Zeichen
- [] Einzelnes Zeichen innerhalb eines bestimmten Bereichs
- [^] Einzelnes Zeichen, das nicht innerhalb eines bestimmten Bereichs ist

Bei Verwendung der LIKE-Klausel müssen Sie die Platzhalterzeichen in einfache Anführungszeichen einschließen.

Die folgende Abfrage liefert *title_id* und *title* sämtlicher Bücher aus der Tabelle *titles* zurück, bei denen die Zeichenfolge *computer* an einer beliebigen Stelle im Titel vorkommt:



```
SELECT title_id, title
FROM titles
WHERE title LIKE '%computer%'
```




title_id title

-
-
BU1111 Cooking with Computers: Surreptitious Balance Sheets
BU2075 You Can Combat Computer Stress!
BU7832 Straight Talk About Computers
MC3026 The Psychology of Computer Cooking
PS1372 Computer Phobic AND Non-Phobic Individuals: Behavior
Variations

(5 row(s) affected)

Die folgende Abfrage liefert *au_id*, *au_lname* und *au_fname* sämtlicher Autoren (*authors*) zurück, deren Namen in der Tabelle *authors* mit B oder M beginnen. Dazu bedient sich die Abfrage einer LIKE-Klausel, bei der die beiden Buchstaben in eckigen Klammern stehen:



```
SELECT au_id, au_lname, au_fname
FROM authors
WHERE au_lname LIKE '[BM]%'
```



au_id	au_lname	au_fname
409-56-7008	Bennet	Abraham
648-92-1872	Blotchet-Halls	Reginald
724-80-9391	MacFeather	Stearns
893-72-1158	McBadden	Heather

(4 row(s) affected)

Unbekannte Werte



Ein Nullwert ist weder eine leere Zeichenfolge noch dasselbe wie eine 0 bei numerischen Daten. Eine NULL taucht immer dann auf, wenn einem Feld kein Wert zugewiesen worden ist, was nur eine andere Ausdrucksweise dafür ist, dass NULL gleich dem Wert »unbekannt« ist. In Fällen dieser Art widersteht NULL sämtlichen Vergleichen mit Leerzeichen, Nullen und anderen NULL-Werten (bei Verwendung der Vergleichsoperatoren < und >). Wie also können Sie nun Zeilen ausfindig machen, die NULL-Werte enthalten? Das lässt sich mithilfe der Schlüsselwörter IS NULL und IS NOT NULL realisieren.



Die Syntax für die WHERE-Klausel unter Verwendung der Operatoren `IS NULL` und `IS NOT NULL` lautet:



```
SELECT Spaltenliste  
FROM Tabellenliste  
WHERE column_name IS [NOT] NULL
```

Mit der folgenden Abfrage finden Sie sämtliche Bücher, deren Verkäufe (*sales*) den Wert `NULL` enthalten.



```
SELECT title_id, title  
FROM titles  
WHERE ytd_sales IS NULL
```



```
title_id title
```

```
-----  
MC3026   The Psychology of Computer Cooking  
PC9999   Net Etiquette
```

(2 row(s) affected)

Im Gegensatz zur vorigen Abfrage können Sie jetzt die Klausel `IS NOT NULL` dazu verwenden, sämtliche Bücher ausfindig zu machen, die Werte bei `ytd_sales` aufweisen:



```
SELECT title_id, title  
FROM titles  
WHERE ytd_sales IS NOT NULL
```



```
title_id title
```

```
-----  
BU1032   The Busy Executive's Database Guide  
BU1111   Cooking with Computers: Surreptitious Balance Sheets  
BU2075   You Can Combat Computer Stress!  
BU7832   Straight Talk About Computers  
.  
.  
.  
TC3218   Onions, Leeks, and Garlic: Cooking Secrets of the  
Mediterranean  
TC4203   Fifty Years in Buckingham Palace Kitchens  
TC7777   Sushi, Anyone?
```

(16 row(s) affected)

Abrufen von Zeilen über mehrere Suchkriterien

Bis jetzt haben Sie erfahren, wie Sie Zeilen auf der Grundlage spezifischer Suchkriterien (Werte, Bereiche, Listen, Zeichenfolgenvergleiche und unbekannte Werte) auswählen können. Nun werden wir uns anschauen, wie man Zeilen unter Verwendung mehrerer Suchkriterien abrufen kann.

Mehrere Suchkriterien kombinieren Sie über den Einsatz der logischen Operatoren AND, OR und NOT. Die Verwendung der Operatoren AND und OR erlaubt Ihnen die Verknüpfung von zwei oder mehr Ausdrücken. AND liefert Ergebnisse zurück, wenn sämtliche Bedingungen zutreffend (true) sind. OR liefert bereits dann Ergebnisse zurück, wenn nur eine der Bedingungen erfüllt ist.

Wenn in der WHERE-Klausel mehr als ein logischer Operator verwendet wird, kann die Rangfolge ins Spiel kommen: Dem NOT folgt dabei AND und anschließend OR.

Die Syntax für eine WHERE-Klausel unter Verwendung mehrerer Suchkriterien sieht wie folgt aus:



```
SELECT Spaltenliste
FROM Tabellenliste
WHERE [NOT] Ausdruck {AND|OR} [NOT] Ausdruck
```

Sehen Sie sich dazu die folgenden Beispiele an.

Abfrage 1

Formulieren Sie die folgende Abfrage: Es soll die Titel-ID, der Titel und der Preis sämtlicher Bücher abgerufen werden, die die *pub ID* 0877 oder das Wort *computer* im Titel haben, und deren Preis NOT NULL ist.



```
SELECT title_id, title, price, pub_id
FROM titles
WHERE title LIKE '%computer%' OR pub_id = '0877' AND price IS NOT NULL
```



title_id	title	price
BU1111	Cooking with Computers: Surreptitious Balance ...	11.95
1389		
BU2075	You Can Combat Computer Stress!	2.99
736		
BU7832	Straight Talk About Computers	19.99



Daten mit SELECT abrufen

```
1389
MC2222 Silicon Valley Gastronomic Treats 19.99
877
MC3021 The Gourmet Microwave 2.99
877
MC3026 The Psychology of Computer Cooking (null)
877
PS1372 Computer Phobic AND Non-Phobic Individuals: Beha 21.59
877
TC3218 Onions, Leeks, and Garlic: Cooking Secrets of th 20.95
877
TC4203 Fifty Years in Buckingham Palace Kitchens 11.95
877
TC7777 Sushi, Anyone? 14.99
877
```

(10 row(s) affected)

Abfrage2

Lassen Sie nun die Abfrage noch einmal laufen und versuchen Sie, den NULL-Wert in Ihrem Preis-Feld zu unterdrücken.



```
SELECT title_id, title, price, pub_id
FROM titles
WHERE (title LIKE '%computer%' OR pub_id = '0877') AND price IS NOT
NULL
```



title_id	title	price
pub_id		
-	-	-
BU1111	Cooking with Computers: Surreptitious Balance Sh	11.95
1389		
BU2075	You Can Combat Computer Stress!	2.99
736		
BU7832	Straight Talk About Computers	19.99
1389		
MC2222	Silicon Valley Gastronomic Treats	19.99
877		
MC3021	The Gourmet Microwave	2.99
877		
PS1372	Computer Phobic AND Non-Phobic Individuals: Beha	21.59
877		
TC3218	Onions, Leeks, and Garlic: Cooking Secrets of th	20.95



```
877
TC4203    Fifty Years in Buckingham Palace Kitchens      11.95
877
TC7777    Sushi, Anyone?                                   14.99
877
```

(9 row(s)affected)

In Abfrage 2 lassen sich die gewünschten Ergebnisse erzielen, indem man die Rangfolge ändert.

10.6 Duplikate entfernen

Bei der Auswahl von Daten in einer Tabelle kann es geschehen, dass Sie mehr als eine Zeile mit denselben Informationen zurückerhalten. Duplikate dieser Art können Sie unter Verwendung der Klausel `DISTINCT` im `SELECT`-Teil der `SELECT`-Anweisung ausschalten. Wenn Sie die `DISTINCT`-Klausel nicht angeben, werden alle Zeilen ausgegeben, auf die die Kriterien der `WHERE`-Klausel zutreffen.

Die Syntax der `DISTINCT`-Klausel lautet:



```
SELECT DISTINCT Spaltenliste
FROM table_name
WHERE Suchkriterien
```

Die Unterscheidung wird über die Kombination sämtlicher Spalten in *Spaltenliste* bestimmt. `NULL`-Werte werden als Duplikate von sich selbst behandelt, so dass folgerichtig nur eine einzige `NULL` zurückgeliefert wird.

Tabelle 10.12 enthält einige `SELECT`-Anweisungen mit und ohne `DISTINCT`-Klausel. Die Abfrage1 listet ohne Duplikate Bundesstaaten auf, in denen Autoren leben. Die Abfrage2 listet ohne Duplikate Städte auf, in denen Autoren leben. Die Abfrage3 schließlich listet nur unterschiedene Kombinationen von Städten und Bundesstaaten auf.

Abfrage	Abfrage2	Abfrage3
<code>SELECT DISTINCT</code> state from authors	<code>SELECT DISTINCT</code> city FROM authors	<code>SELECT DISTINCT</code> city, state FROM authors
State CA	City Ann Arbor	City und State Ann Arbor, MI

Tabelle 10.12: Jede Abfrage führt zu unterschiedlichen Ergebnissen.

Abfrage	Abfrage2	Abfrage3
IN	Berkeley	Berkeley, CA
KS	Corvallis	Corvallis, OR
MD	Covelo	Covelo, CA
MI	Gary	Gary, IN
OR	Lawrence	Lawrence, KS
TN	Menlo Park	Menlo Park, CA
UT	Nashville	Nashville, TN
	Oakland	Oakland, CA
	Palo Alto	Palo Alto, CA
	Rockville	Rockville, MD
	Salt Lake City	Salt Lake City, UT
	San Francisco	San Francisco, CA
	San Jose	San Jose, CA
	Vacaville	Vacaville, CA
	Walnut Creek	Walnut Creek, CA

Tabelle 10.12: Jede Abfrage führt zu unterschiedlichen Ergebnissen (Forts.).

Warum können Sie nun, falls Sie nur die verschiedenen Städte auflisten möchten, in denen Autoren leben, nicht einfach die Abfrage2 benutzen, die Ihnen dieselben Städte wie Abfrage3 liefert? Um diese Frage beantworten zu können, nehmen Sie nun einmal an, dass zwei Autoren Ihrer Datenbank in Portland leben. Falls Sie die Abfrage3 laufen lassen, würde diese als einen der unterschiedlichen Werte Portland zurückliefern. Einer der Autoren lebt jedoch in Portland, Oregon, während der andere in Portland, Maine, lebt, was ganz eindeutig unterschiedliche Orte sind. Daher liefert die Abfrage3 unter Verwendung der DISTINCT-Kombination von Stadt (*city*) und Bundesstaat (*state*) sowohl Portland, Oregon, als auch Portland, Maine, zurück.

Beachten Sie, dass die Ergebnisse in allen drei Abfragen sortiert sind, was aber nicht unbedingt so sein muss. In älteren Versionen von SQL Server waren immer die ersten Werte sortiert, so dass der erste Wert mit dem nächsten Wert sofort verglichen werden konnte, um die Entfernung von doppelt vorkommenden Werten einfacher zu gestalten. SQL Server 2000 verwendet dagegen zum Ermitteln der nötigen Informationen einen komplexeren Hash-Algorithmus, was die Geschwindigkeit und Effizienz vergrößert, aber den Nachteil mit sich bringen kann, dass Sie unter Umständen auf die Sortierung verzichten müssen. Um die Sortierung Ihrer Daten zu garantieren, müssen Sie in Ihren Abfragen die ORDER BY-Klausel einsetzen.

Sehen Sie sich Abfrage3 genauer an:



```
SELECT DISTINCT city, state
FROM authors
```

Das Ergebnis, in dem Portland, OR, und Portland, ME, zurückgeliefert wird, sieht folgendermaßen aus:



city	state
Portland	ME
Portland	OR

(2 row(s) affected)

10.7 Sortieren von Daten mit der Klausel ORDER BY

Ihre Abfrageergebnisse sortieren Sie mit der ORDER BY-Klausel in der SELECT-Anweisung. Die grundlegende Syntax für den Einsatz der ORDER BY-Klausel lautet wie folgt:



```
SELECT Spaltenliste
FROM Tabellenliste
[ORDER BY Spaltenname | Spaltenliste_Nummer [ASC|DESC]]
```

Die Anzahl der Spalten in der ORDER BY-Liste kann beliebig groß sein, solange die Gesamtlänge der Spalten nicht größer als 900 Byte wird. Außerdem können Sie sowohl Spaltennamen als auch die Ordinalnummern der Spalten in *Spaltenliste* angeben.

Die folgenden Abfragen liefern die gleichen sortierten Resultsets zurück:

Abfrage1

```
SELECT title_id, au_id
FROM titleauthor
ORDER BY title_id, au_id
```

Abfrage2

```
SELECT title_id, au_id
FROM titleauthor
ORDER BY 1, 2
```

Spaltennamen und Ordinalnummern können Sie in der `ORDER BY`-Klausel auch gemeinsam verwenden. Darüber hinaus können Sie noch festlegen, ob Sie die Ergebnisse in aufsteigender (ASC) oder absteigender (DESC) Reihenfolge sortiert haben möchten. Wenn Sie weder ASC noch DESC angeben, wird ASC verwendet.

Falls Sie die Ergebnisse nach einer Spalte sortieren, die NULL-Werte enthält, erscheinen die Zeilen mit den NULL-Werten zuerst.

Bei der Verwendung der `ORDER BY`-Klausel hat die Sortierreihenfolge Ihres SQL Servers einen Einfluss auf Ihre Resultsets. Die standardmäßige Sortierreihenfolge für SQL Server ist die Wörterbuchreihenfolge ohne Unterscheidung nach Groß- und Kleinschreibung. Falls Ihr SQL Server eine Sortierreihenfolge verwendet, die nach Groß- und Kleinschreibung unterscheidet, berührt dies die Sortierung Ihrer Resultsets, da in einer groß- und kleinschreibungsabhängigen Sortierreihenfolge der Großbuchstabe A nicht als gleichwertig mit dem kleinen Buchstaben a behandelt wird.

Welche die aktuell verwendete Sortierung Ihres Servers ist, finden Sie heraus, wenn Sie die gespeicherte Systemprozedur `sp_helpsort` ausführen.



Die `ORDER BY`-Klausel können Sie nicht auf Spalten anwenden, deren Datentypen `text` oder `image` sind. Dies gilt für `text`-Datentypen aus dem Grund, weil diese an einer anderen Stelle in Ihrer Datenbank gespeichert und in der Länge zwischen 0 bis 2 GB variieren können. SQL Server erlaubt keine Sortierung eines Feldes dieser Größe. `Image`-Datentypen sind ebenfalls in ihren eigenen getrennten Datenseiten mit 8 KB gespeichert und nicht sortierbar. Eine Alternative würde darin bestehen, eine andere Tabelle zu erstellen, die zusammenfassende Informationen über Ihre Grafik oder Ihren Text enthält.

10.8 Zwischenbilanz

Lassen Sie uns einmal Revue passieren, welche Themen bis jetzt in diesem Kapitel behandelt worden sind.

Sie haben nun über die Verwendung der `SELECT`-Anweisung eine solide Grundlage zum Abrufen von Daten bekommen. Außerdem haben Sie gelernt, wie Sie Ihre Spaltenüberschriften ändern und Ihrer Bildschirmausgabe Zeichenfolgenlitterale hinzufügen können. Zu diesem Zweck setzen Sie die Anweisungen

```
SELECT Spaltenname AS neuer_Spaltenname
```

zum Ändern einer Spaltenüberschrift und

```
SELECT Zeichenfolgenlitteral, Spaltenname
```

zum Ändern eines Zeichenfolgenliterals ein.



Ihre Kenntnisse der `SELECT`-Anweisung haben Sie darüber hinaus erweitert, indem Sie sich mit arithmetischen Operatoren, mathematischen Funktionen, Zeichenfolgenfunktionen sowie Datums- und Zeitfunktionen beschäftigt haben. Danach haben Sie weitere Systemfunktionen wie `GetDate` für Manipulationen Ihrer Daten kennen gelernt.

Es gibt Situationen, in denen die Daten, mit denen Sie arbeiten möchten, nicht in dem gewünschten Datentyp und Format vorliegen. Über die Verwendung des Befehls `CONVERT` zum Umwandeln von Daten haben Sie eine Möglichkeit kennen gelernt, den Datentyp zu ändern.

Anschließend haben Sie Ihre Kenntnisse der `SELECT`-Anweisung durch Beispiele und Erörterungen zur Auswahl verschiedener Datenzeilen mit Hilfe von Vergleichsoperatoren, Wertebereichen, Listen und Zeichenfolgen vertieft.

Danach haben Sie erfahren, wie Sie Zeilen mit `NULL`-Werten unterdrücken können. Außerdem wissen Sie jetzt, wie Sie Zeilen mit Duplikaten zu behandeln haben.

Den letzten Teil dieser Lektion haben Sie mit der Sortierung Ihrer Ergebnisse über die `ORDER BY`-Klausel abgeschlossen.

Nach dieser kurzen Atempause können wir nun die Lektion des heutigen Tages mit einigen erweiterten `SELECT`-Anweisungen fortsetzen. Sie werden nun mit fortgeschrittenen Techniken wie der Erzeugung von Zusammenfassungen über die Aggregatfunktionen `GROUP BY` und `HAVING` und die Anweisungen `COMPUTE` und `COMPUTE BY` arbeiten.

Außerdem lernen Sie, wie Sie Daten korrelieren und Unterabfragen ausführen. Zum Schluss werden Sie einen Blick auf weitere fortgeschrittene Abfragen werfen, die mit der Auswahl von Daten aus mehreren Tabellen arbeiten. »Weiter fortgeschritten« soll hier aber nicht bedeuten, dass diese Abfragen zu kompliziert für Sie sind, sondern es bedeutet lediglich, dass ihre Handhabung etwas mehr Übung als die einfachen `SELECT`-Anweisungen erfordert, mit denen Sie schon zu tun hatten.

10.9 Aggregatfunktionen

Aggregatfunktionen liefern zusammenfassende Werte für eine ganze Tabelle oder Zeilenbereiche einer Tabelle zurück. Sie werden normalerweise in Verbindung mit der Klausel `GROUP BY` benutzt und treten in der `HAVING`-Klausel oder in *Spaltenliste* auf. Sie werden nun nach und nach alles, was Sie zu diesem Thema wissen müssen, in der grundlegenden Form kennen lernen. Anschließend erfahren Sie, wie Sie diese Grundlagen erweitern können. In Tabelle 10.13 sind die Aggregatfunktionen mit ihren Parametern und Ergebnissen enthalten.



Funktion	Ergebnis
AVG([ALL DISTINCT] Spaltenname)	Durchschnitt der Werte im numerischen Ausdruck; entweder alle Werte oder nur die eindeutigen Instanzen
COUNT(*)	Anzahl der ausgewählten Zeilen
COUNT([ALL DISTINCT] column_name)	Anzahl der Werte im Ausdruck, entweder alle Werte oder nur die eindeutigen Instanzen
MAX(Spaltenname)	Größter Wert im Ausdruck
MIN(Spaltenname)	Kleinsten Wert im Ausdruck
STDEV(Spaltenname)	Gibt die statistische Standardabweichung sämtlicher Werte in <i>Spaltenname</i> oder im Ausdruck zurück.
STDEVP(Spaltenname)	Liefert die statistische Standardabweichung der Auffüllung für alle Werte im angegebenen Ausdruck zurück.
SUM([ALL DISTINCT] Spaltenname)	Summe der Werte im numerischen Ausdruck; entweder alle Werte oder nur die eindeutigen Instanzen
TOP n [PERCENT]	Liefert die obersten <i>n</i> Werte bzw. <i>n</i> % im Resultset zurück.
VAR(Spaltenname)	Liefert die statistische Varianz der in <i>Spaltenname</i> oder im Ausdruck aufgeführten Werte zurück.
VARP(Spaltenname)	Liefert die statistische Varianz für die Auffüllung aller Werte im angegebenen Ausdruck zurück.

Tabelle 10.13: Aggregatfunktionen

Es folgen nun einige Beispiele für den Einsatz von Aggregatfunktionen für die Datenbank *pubs*.

Die folgende Abfrage liefert die Anzahl der Zeilen in der Tabelle *employee* zurück:



```
SELECT COUNT(*)  
FROM employee
```



```
-----  
43  
  
(1 row(s) affected)
```

Die nächste Abfrage wählt den in der Spalte *ytd_sales* in der Tabelle *titles* gefundenen Maximalwert aus:



```
SELECT MAX(ytd_sales)
FROM titles
```



```
-----
22246
```

(1 row(s) affected)

Nachfolgende Abfrage wählt sämtliche Werte der Spalte *qty* in der Tabelle *sales* aus und summiert sie auf:



```
SELECT SUM(qty)
FROM sales
```



```
-----
493
```

(1 row(s) affected)

GROUP BY und HAVING

Durch die `GROUP BY`-Klausel werden zusammengefasste Daten, die den Kriterien der `WHERE`-Klausel entsprechen, als einzelne Zeilen zurückgeliefert. Die `HAVING`-Klausel benennt die Kriterien, die festlegen, welche Zeilen in der `GROUP BY`-Klausel zurückgeliefert werden sollen. So können Sie beispielsweise herausfinden, welche Bücher mehr als einen Autor haben, und dann für jedes einzelne Buch den Titel und den Autorennamen ermitteln.

Die Syntax für die `GROUP BY`- und `HAVING`-Klauseln lautet folgendermaßen:



```
SELECT Spaltenliste
FROM Tabellenliste
WHERE Suchkriterien
[GROUP BY [ALL] nicht_aggregierte(r) Ausdruck/Ausdrücke
[HAVING] Suchkriterien]
```

Die `HAVING`-Klausel hat die gleiche Wirkung auf die `GROUP BY`-Klausel, wie sie die `WHERE`-Klausel auf die `SELECT`-Anweisung hat. Zum Beispiel findet die folgende Abfrage sämtliche Bücher, die mehr als einen Autor besitzen. Sie liefert die Titel-ID und die Anzahl der Autoren für jedes Buch zurück:



Daten mit SELECT abrufen



```
SELECT title_id, count(title_id) AS Autorenanzahl
FROM titleauthor
GROUP BY title_id
HAVING count(title_id) > 1
```



```
title_id Autorenanzahl
-----
BU1032    2
BU1111    2
MC3021    2
PC8888    2
PS1372    2
PS2091    2
TC7777    3
```

Die GROUP BY ALL-Klausel liefert alle Gruppierungen einschließlich derjenigen, die nicht mit den Kriterien der WHERE-Klausel übereinstimmen.

Wenn Sie sämtliche Bücher mit aktuellen Jahresverkaufszahlen (*year-to-date sales*) von \$ 4.000 und höher finden und alle Titel-IDs auflisten möchten, können Sie die folgende Abfrage ausführen:



```
SELECT title_id, ytd_sales
FROM titles
WHERE (ytd_sales >= 4000)
GROUP BY ALL title_id, ytd_sales
```



```
title_id ytd_sales
-----
BU1032    4095
BU1111    3876
BU2075    18722
BU7832    4095
MC2222    2032
MC3021    22246
MC3026    (null)
PC1035    8780
PC8888    4095
PC9999    (null)
PS1372    375
PS2091    2045
PS2106    111
PS3333    4072
PS7777    3336
TC3218    375
TC4203    15096
TC7777    4095
```

Beachten Sie, dass auch Bücher aufgelistet werden, die nicht den Kriterien der WHERE-Klausel entsprechen.



Es gibt gewisse Anforderungen, denen die GROUP BY- und HAVING-Klauseln genügen müssen, um kompatibel zum ANSI-Standard zu sein. Eine dieser Anforderungen besteht darin, dass die GROUP BY-Klausel alle nichtaggregierten Spalten aus *Spaltenliste* enthalten muss. Eine weitere besagt, dass die Kriterien der HAVING-Klausel nur einen einzigen Wert zurückliefern dürfen.

COMPUTE und COMPUTE BY

Die Klauseln COMPUTE und COMPUTE BY werden dazu benutzt, neue Zusammenfassungs- und Detaildaten zu erzeugen. Es werden dabei die Aggregatfunktionen verwendet, die bereits weiter oben erläutert worden sind. Die COMPUTE-Klausel liefert Detailzeilen und eine abschließende zusammenfassende Zeile zurück, während die COMPUTE BY-Klausel etwa wie die Klausel GROUP BY neue Ergebniszeilen mit Zusammenfassungen liefert. Die Zeilen werden von ihr jedoch in der Form von Untergruppierungen mit Summenwerten dargestellt.

Die Syntax für die Klauseln COMPUTE und COMPUTE BY lautet:



```
SELECT Spaltenliste
FROM Tabellenliste
WHERE Suchkriterien
[COMPUTE] Aggregatausdruck/ausdrücke
[BY] Spaltenliste
```

Die folgenden Beispiele vergleichen die Klauseln COMPUTE BY und GROUP BY:



```
SELECT type, SUM(ytd_sales)
FROM titles
GROUP BY type
```



type	ytd_sales
business	30788
mod_cook	24278
popular_comp	12875
psychology	9939
trad_cook	15471
UNDECIDED	NULL



Bei Verwendung der `COMPUTE BY`-Klausel müssen Sie immer auch die Klausel `ORDER BY` benutzen. Falls Ihre `ORDER BY`-Klausel beispielsweise folgendermaßen lautet:

```
ORDER BY title_id, pub_id, au_id
```

kann Ihre `COMPUTE BY`-Klausel eine der folgenden Formen annehmen:

```
COMPUTE Aggregatfunktion (Spaltenname)  
BY title_id, pub_id, au_id
```

oder

```
COMPUTE Aggregatfunktion (Spaltenname)  
BY title_id, pub_id
```

oder

```
COMPUTE Aggregatfunktion (Spaltenname)  
BY title_id
```

Wie Sie sehen, müssen die in der Klausel `COMPUTE BY` genannten Spalten dieselben oder eine Untermenge der Spalten in der `ORDER BY`-Klausel sein. Die Reihenfolge der Spalten in der `COMPUTE BY`-Klausel muss dieselbe wie in der `ORDER BY`-Klausel sein; außerdem dürfen Sie keine Spalten überspringen. Geben Sie nun einmal Folgendes ein und untersuchen Sie dann die Ergebnisse:



```
SELECT type, ytd_sales  
FROM titles  
ORDER BY type  
COMPUTE SUM(ytd_sales) BY type
```



type	ytd_sales
business	4095
business	3876
business	18722
business	4095

```
sum  
=====
```

```
30788
```

type	ytd_sales
mod_cook	2032
mod_cook	22246

```
sum  
=====
```



24278

type	ytd_sales

popular_comp	8780
popular_comp	4095
popular_comp	(null)

sum
=====

12875

type	ytd_sales

psychology	375
psychology	2045
psychology	111
psychology	4072
psychology	3336

sum
=====

9939

type	ytd_sales

trad_cook	375
trad_cook	15096
trad_cook	4095

sum
=====

19566

type	ytd_sales

UNDECIDED	(null)

sum
=====

(null)

(24 row(s) affected)



Durch die Klauseln `COMPUTE` und `COMPUTE BY` werden neue Ergebniszeilen mit Daten in nichtrelationaler Form erzeugt, weshalb sie nicht zusammen mit der Anweisung `SELECT INTO` zum Erstellen neuer Tabellen verwendet werden dürfen. `COMPUTE` und `COMPUTE BY` können Sie nicht auf die Datentypen `text` oder `image` anwenden, weil diese sich nicht sortieren lassen.

10.10 Superaggregate (ROLLUP und CUBE)

Zum Erzeugen weiterer Zusammenfassungszeilen können Sie die Operatoren `ROLLUP` und `CUBE` einsetzen, die man auch als *Superaggregate* bezeichnet. `ROLLUP`- und `CUBE`-Operatoren benutzen Sie in Verbindung mit der Klausel `GROUP BY`.

Die Syntax für die Operatoren `ROLLUP` und `CUBE` lautet wie folgt:



```
SELECT Spaltenliste
FROM Tabellenliste
WHERE Suchkriterien
[GROUP BY [ALL] nichtaggregierte(r)_Ausdruck/Ausdrücke
[WITH {ROLLUP | CUBE}]]
```

Der Operator `ROLLUP` wird typischerweise zum Erstellen von fortlaufenden Durchschnittswerten oder Summen eingesetzt. Dazu wenden Sie die Aggregatfunktion in der `SELECT`-Spaltenliste auf jede Spalte in der `GROUP BY`-Klausel von links nach rechts an. Was diese Vorgehensweise bedeutet, lässt sich am Einfachsten mit einem Beispiel zeigen:



```
SELECT type, pub_id,
SUM(ytd_sales) AS ytd_sales
FROM titles
GROUP BY type, pub_id
WITH ROLLUP
```



type	pub_id	ytd_sales
business	736	18722
business	1389	12066
business	(null)	30788
mod_cook	877	24278
mod_cook	(null)	24278
popular_comp	1389	12875
popular_comp	(null)	12875
psychology	736	9564
psychology	877	375
psychology	(null)	9939

trad_cook	877	19566
trad_cook	(null)	19566
UNDECIDED	877	(null)
UNDECIDED	(null)	(null)
(null)	(null)	97446

(15 row(s) affected)



Der Operator `ROLLUP` liefert eine Zeile in der Ausgabe für jede Zeile in der Tabelle `titles` mit einem einzigen `type` und `pub_id`. Anschließend listet er für jeden Eintrag die `ytd_sales` auf und erstellt noch eine weitere Zeile für jeden `type` mit zusammenfassenden Informationen. Im obigen Beispiel wird in den Zeilen mit `(null)` im Feld `pub_id` für sämtliche `ytd_sales` der Wert der Funktion `SUM` für diese Gruppe von Typen angezeigt.

Anders ausgedrückt: In der Tabelle `titles` befinden sich zwei Zeilen, die sowohl einen Typ `business` als auch eine eindeutige `pub_id` besitzen (in der Originaltabelle gibt es insgesamt vier Bücher des Typs `business` mit einem Autor `[pub_id]`, der drei Bücher, sowie einem weiteren Autor, der das vierte Buch geschrieben hat). Jeder Autor, der `business`-Bücher geschrieben hat, bringt es auf `ytd_sales` von 18.722 bzw. 12.066.

Der Operator `ROLLUP` erzeugt dann ein Teilsummenfeld, das alle Bücher des Typs `business` summiert ($18.722 + 12.066 = 30.788$). Die Abfrage führt das Gleiche für jede Gruppe von Büchern und Autoren in der Tabelle durch und liefert schließlich eine Gesamtsumme (97.446), die durch die `(null)`-Werte in den Feldern `type` und `pub_id` gekennzeichnet wird.

Der `CUBE`-Operator erzeugt Zeilen mit Superaggregaten, indem er jede mögliche Kombination der Spalten in der Klausel `GROUP BY` verwendet. Wie auch der `ROLLUP`-Operator produziert der `CUBE`-Operator fortlaufende Durchschnittswerte und Summen, erzeugt aber auch gleichzeitig zusätzliche zusammenfassende Ergebniszeilen, indem er ein Crossreferenzing von Spalten durchführt. Sehen Sie sich dazu folgendes Beispiel an:



```
SELECT type, pub_id,  
SUM(ytd_sales) AS ytd_sales  
FROM titles  
GROUP BY type, pub_id  
WITH CUBE
```



type	pub_id	ytd_sales
business	736	18722
business	1389	12066
business	(null)	30788
mod_cook	877	24278

mod_cook	(null)	24278
popular_comp	1389	12875
popular_comp	(null)	12875
psychology	736	9564
psychology	877	375
psychology	(null)	9939
trad_cook	877	19566
trad_cook	(null)	19566
UNDECIDED	877	(null)
UNDECIDED	(null)	(null)
(null)	(null)	97446
(null)	736	28286
(null)	877	44219
(null)	1389	24941

(18 row(s) affected)



Der CUBE-Operator erzeugt in der Ausgabe für jede Zeile in der Tabelle *titles* eine Ergebniszeile mit einem einzigen *type* und *pub_id*. Dann zeigt er die *ytd_sales* für jeden Eintrag an und erzeugt eine zusätzliche Zeile mit zusammenfassenden Informationen sowie eine weitere Zeile für jede *pub_id*. In diesem Beispiel ist in den Zeilen mit (null) im Feld *pub_id* die Summe sämtlicher *ytd_sales* für diese Gruppe von Typen enthalten, während die Zeilen mit (null) im Feld *type* die Summe sämtlicher *ytd_sales* für diese Gruppe von *pub_ids* enthalten.

Bei Verwendung von ROLLUP und CUBE sind einige Einschränkungen bei der Klausel GROUP BY zu beachten: Es ist nur ein Maximum von 10 Spalten in der GROUP BY-Klausel erlaubt, während die Summe der Größe dieser Spalten 900 Byte nicht überschreiten darf. Außerdem dürfen Sie die Klausel GROUP BY ALL nicht verwenden.



Die Operatoren ROLLUP und CUBE erzeugen neue Zeilen von relationalen Daten. Die relationalen Ergebnisse sind den formatfreien Ergebnissen vorzuziehen, die sich aus Anweisungen mit COMPUTE und COMPUTE BY ergeben.

Außerdem ist zu beachten, dass Sie die Operatoren ROLLUP und CUBE nicht zusammen mit den Datentypen text oder image verwenden können, weil sich diese Datentypen nicht sortieren lassen.

10.11 Datenkorrelation

In diesem Abschnitt werden Sie einen Blick auf die Implementierung von Verknüpfungen zum Abrufen von Daten aus zwei oder mehreren Tabellen werfen. Die Ergebnisse werden dabei in der Form einer einzelnen Tabelle mit den Spalten aus allen Tabellen dargestellt, die in der `SELECT`-Spaltenliste angegeben sind und den Suchkriterien entsprechen. Sie werden erfahren, wie Sie Verknüpfungen unter Verwendung der ANSI- und der SQL Server-Syntax implementieren, und die verschiedenen Typen von Verknüpfungen untersuchen: innere Verknüpfungen, Kreuzverknüpfungen, äußere Verknüpfungen und Selbstverknüpfungen.

Implementieren von Verknüpfungen

Zum Verknüpfen von Tabellen müssen Sie einen Vergleich von einer oder mehreren Spalten in einer Tabelle mit einer oder mehreren Spalten in einer anderen Tabelle durchführen. Das Ergebnis dieses Vergleichs erzeugt dann neue Zeilen, indem die in der `SELECT`-Spaltenliste genannten Spalten aus den verknüpften Tabellen entsprechend den Verknüpfungsbedingungen kombiniert werden. Beim Verknüpfen von Tabellen können Sie entweder die ältere ANSI-Syntax nach `SQL-89` oder die neuere ANSI-Syntax nach `SQL-99` verwenden.



Die Ergebnismengen für diese beiden Syntaxformen sind bei den meisten Abfragen identisch. Ein Unterschied ergibt sich bei einer äußeren Verknüpfung. Nach der älteren `SQL-89`-Syntax kann der Abfragecompiler für die Abfrage entscheiden, dass die äußere Verknüpfung nicht notwendig ist, so dass eine einfache innere Verknüpfung zwischen den beiden Tabellen durchgeführt wird. Nach dem neueren `SQL-99`-Standard passiert das nicht.

ANSI SQL-99-Syntax für Verknüpfungen

Die Verknüpfungsanweisungen nach der ANSI-Syntax werden in der `FROM`-Klausel der `SELECT`-Anweisung aufgeführt:



```
SELECT Tabellenname.Spaltenname, [...]
FROM { Tabellenname [Verknüpfungstyp] JOIN Tabellenname
ON Suchkriterien}, [...]
WHERE Suchkriterien
```

Die `WHERE`-Klausel wählt aus den verknüpften Zeilen diejenigen aus, die zurückgeliefert werden sollen. Es lassen sich drei Typen von ANSI-Verknüpfungsanweisungen angeben: `INNER JOIN`, `OUTER JOIN` und `CROSS JOIN`.

ANSI SQL-89-Syntax für Verknüpfungen

Obwohl SQL Server 2000 weiterhin die ältere SQL-89-Syntax unterstützt, empfiehlt es sich, statt dessen die neuere ANSI-Standardsyntax zu verwenden:



```
SELECT Tabellenname.Spaltenname, [...]
FROM Tabellenliste
WHERE Tabellenname.Spaltenname
Verknüpfungsoperator Tabellenname.Spaltenname, [...]
```

Wenn Sie die früheren Versionen der Syntax verwenden, listet die FROM-Klausel die zu verknüpfenden Tabellen auf. Die WHERE-Klausel nennt die zu verknüpfenden Spalten und enthält darüber hinaus gegebenenfalls zusätzliche Suchkriterien, die die Zeilen betreffen, die zurückgeliefert werden sollen. Die Verknüpfungsoperatoren gemäß SQL Server-Syntax lauten wie folgt: =, >, <, >=, <=, <>, != und !<.

Innere Verknüpfungen



Mit Verknüpfungen werden zwei Tabellen auf der Grundlage einer Verknüpfungsbedingung miteinander verbunden, die als Ergebnis zu einer neuen Tabelle führt. Diese Tabelle enthält dann die Zeilen, die der Verknüpfungsbedingung entsprechen. Innere Verknüpfungen erzeugen immer dann Ergebnisse, wenn die gesuchten Informationen in beiden Tabellen gefunden worden sind. Die häufigsten Typen innerer Verknüpfungen sind *Gleichheits-* und *natürliche* Verknüpfungen.

Bei einer Gleichheitsverknüpfung werden die Spaltenwerte auf Gleichheit hin untersucht und die redundanten Spalten als Spalten im Resultset angezeigt. In der natürlichen Verknüpfung dagegen werden die redundanten Spalten nur einmal angezeigt.

Schauen Sie sich zur Verdeutlichung dieses Sachverhalts das folgende Beispiel an. Hier wählt die SELECT-Anweisung alle Spalten aus den Tabellen *publishers* und *pub_info* aus, wenn die *pub_id*-Spalten für die verknüpften Tabellen gleich sind. Beachten Sie die redundante Spalte *pub_id*.

Ältere ANSI-Syntax:



```
SELECT publishers.pub_id, pub_name, pr_info
FROM publishers, pub_info
WHERE publishers.pub_id = pub_info.pub_id
```

Die bevorzugte ANSI-Syntax:



```
SELECT publishers.pub_id, pub_name, pr_info
FROM publishers
INNER JOIN pub_info ON publishers.pub_id = pub_info.pub_id
```



pub_id	pub_name	pr_info
0736	New Moon Books	This is sample text data for New Moon Bo
0877	Binnet & Hardley	This is sample text data for Binnet & Ha
1389	Algodata Infosystems	This is sample text data for Algodata In
1622	Five Lakes Publishing	This is sample text data for Five Lakes
1756	Ramona Publishers	This is sample text data for Ramona Publ
9901	GGG&G	This is sample text data for GGG&G, publ
9952	Scootney Books	This is sample text data for Scootney Bo
9999	Lucerne Publishing	This is sample text data for Lucerne Pub

(8 row(s) affected)

Natürliche Verknüpfungen

Bei einer natürlichen Verknüpfung werden die Spaltenwerte ebenfalls auf Gleichheit untersucht, redundante Spalten jedoch aus dem Resultset ausgeschlossen. Im folgenden Beispiel wählt die `SELECT`-Anweisung alle Spalten aus der *publisher*-Tabelle und alle Spalten außer *pub_id* aus der *pub_info*-Tabelle aus.

Ältere ANSI-Syntax:



```
SELECT publishers.*, pub_info.logo, pub_info.pr_info
FROM publishers, pub_info
WHERE publishers.pub_id = pub_info.pub_id
```

Bevorzugte ANSI-Syntax:



```
SELECT publishers.*, pub_info.logo, pub_info.pr_info
FROM publishers
INNER JOIN pub_info ON publishers.pub_id = pub_info.pub_id
```



pub_id	pub_name	city	state	country
0736	New Moon Books	Boston	MA	USA
0877	Binnet & Hardley	Washington	DC	USA
1389	Algodata Infosystems	Berkeley	CA	USA

1622	Five Lakes Publishing	Chicago	IL	USA
1756	Ramona Publishers	Dallas	TX	USA
9901	GGG&G	München	NULL	Germany
9952	Scootney Books	New York	NY	USA
9999	Lucerne Publishing	Paris	NULL	France

(8 row(s) affected)

Kreuz- bzw. unbeschränkte Verknüpfungen

Kreuz- oder unbeschränkte Verknüpfungen liefern als Ergebnismenge eine Kombination sämtlicher Zeilen aller Tabellen in der Verknüpfung zurück. Eine Kreuz- oder unbeschränkte Verknüpfung wird nicht über die `WHERE`-Klausel in der SQL Server-Verknüpfung von zwei oder mehreren Tabellen erzeugt, sondern unter Verwendung des Schlüsselwortes `CROSS JOIN` für die ANSI-Verknüpfung.



Die Kombination aller Zeilen aus allen in die Verknüpfung eingebundenen Tabellen ist auch unter der Bezeichnung *Kartesisches Produkt* bekannt. In der Regel brauchen Sie diese Art der Verknüpfung nicht; es sei denn, Sie interessieren sich, wie bei manchen statistischen oder mathematischen Analysen, für sämtliche denkbaren Kombinationen. Anders ausgedrückt: Wenn Sie jede Tabelle als Matrix betrachten und diese Matrizen dann multiplizieren, erhalten Sie eine neue Matrix, wie sie in Abbildung 10.1 zu sehen ist. Jede Zeile von Tabelle 1 wurde jeder Zeile in Tabelle 2 hinzugefügt. Wenn Sie die Anzahl der Spalten beider Tabellen addieren, erhalten Sie die resultierende Anzahl der Spalten. Wenn Sie die Anzahl der Zeilen in *Tabelle 1* mit der Anzahl der Zeilen in *Tabelle 2* multiplizieren, bekommen Sie die Gesamtanzahl der Zeilen, die die Abfrage zurückliefert.



Tabellen mit `text`- oder `image`-Spalten können nicht verknüpft werden. Sie können jedoch die Länge der `text`-Spalten aus zwei Tabellen über eine `WHERE`-Klausel miteinander vergleichen, obschon kein Vergleich der eigentlichen Daten möglich ist.

A	B
C	D
E	F

Tabelle 1 hat 3 Zeilen
mit 2 Spalten

1	2	3
4	5	6
7	8	9

Tabelle 2 hat 3
Zeilen mit 3 Spalten

A	B	1	2	3
C	D	1	2	3
E	F	1	2	3
A	B	4	5	6
C	D	4	5	6
E	F	4	5	6
A	B	7	8	9
C	D	7	8	9
E	F	7	8	9

Die Kreuzverknüpfung liefert
eine Ergebnismenge von
9 Zeilen mit 5 Spalten

Abbildung 10.1:
Erzeugen des kartesischen Produkts

Es folgt nun ein Beispiel für die Erstellung eines kartesischen Produkts unter Verwendung von `CROSS JOIN` bzw. der unbeschränkten Verknüpfung. Nehmen Sie dazu einmal an, dass sämtliche Buchtitel und deren Autoren-IDs aufgelistet werden sollen:

Ältere ANSI-Syntax:



```
SELECT titles.title, titleauthor.au_id
FROM titles, titleauthor
```

Bevorzugte ANSI-Syntax:



```
SELECT titles.title, titleauthor.au_id
FROM titles CROSS JOIN titleauthor
```



title	au_id
-----	-----
The Busy Executive's Database Guide	172-32-1176
The Busy Executive's Database Guide	213-46-8915
.	.
.	.
.	.
Sushi, Anyone?	998-72-3567
Sushi, Anyone?	998-72-3567

(450 row(s) affected)



Als Ergebnis dieser Abfrage haben Sie 450 Zeilen erhalten. In der Tabelle *titles* gibt es 18 und in der Tabelle *titleauthor* 25 Zeilen. Weil eine unbeschränkte bzw. Kreuzverknüpfung alle möglichen Kombinationen zurückliefert, erhalten Sie $18 \times 25 = 450$ Zeilen – nicht ganz das, was Sie sich vorgestellt haben, oder?

Eine unbeschränkte Verknüpfung vermeiden Sie, indem Sie von der Anzahl der Tabellen, die Sie verknüpfen möchten, 1 abziehen. $N - 1$ ergibt dann die Anzahl der nötigen Verknüpfungsklauseln, wobei N die Anzahl der in die Verknüpfung involvierten Tabellen darstellt (also drei Tabellen; $3 - 1 = 2$; ergibt zwei Verknüpfungsklauseln). Die Anzahl der Verknüpfungsklauseln kann höher werden, wenn Sie Verknüpfungen auf der Grundlage eines zusammengesetzten Schlüssels durchführen.

Äußere Verknüpfungen

Durch äußere Verknüpfungen entfernen Sie Zeilen aus einer Tabelle, während alle Zeilen einer anderen Tabelle im Resultset erscheinen. Einer der häufigsten Verwendungszwecke für diesen Typ einer Verknüpfung besteht in der Suche nach verwaisten Datensätzen. Die Operatoren und Schlüsselwörter für die ANSI-Syntax lauten folgendermaßen:

LEFT OUTER JOIN	Schließt sämtliche Zeilen aus der ersten Tabelle und nur die übereinstimmenden Zeilen aus der zweiten Tabelle ein
RIGHT OUTER JOIN	Schließt sämtliche Zeilen aus der zweiten Tabelle und nur die übereinstimmenden Zeilen aus der ersten Tabelle ein
FULL OUTER JOIN	Schließt sämtliche nicht übereinstimmenden Zeilen aus beiden Tabellen ein

Nehmen wir an, Sie haben eine Tabelle mit Kunden und eine andere mit Bestellungen, die über ein Feld *CustomerID* miteinander verbunden sind. Mit einer Gleichheits- oder natürlichen Verknüpfung bekommen Sie nur dann Datensätze geliefert, wenn *CustomerID* in beiden Tabellen übereinstimmt. Äußere Verknüpfungen erweisen sich zur Anzeige einer Kundenliste als recht praktisch – falls ein Kunde eine Bestellung hat, wird obendrein auch die Bestellinformation angezeigt. Hat der Kunde dagegen keine Bestellung, wird die Information aus der Bestelltabelle als (null) angezeigt.

Wenn Sie für diese Tabellen eine linke äußere Verknüpfung ausführen und dabei die Tabelle *Kunden* als erste angeben, erhalten Sie die gewünschten Ergebnisse. Spezifizieren Sie dagegen eine rechte äußere Verknüpfung, enthalten Ihre Ergebnisse sämtliche Bestellungen. Sollte dabei die *CustomerID* einer Bestellung nicht mit der *CustomerID* in der Kundentabelle übereinstimmen, ist die Kundeninformation (null). (Nach den Regeln der referenziellen Integrität sollten Sie niemals eine Bestellung ohne eine gültige *CustomerID* haben. In diesem Fall wird Ihre rechte äußere Verknüpfung zu denselben Ergebnissen wie

eine Gleichheits- oder natürliche Verknüpfung führen: Sämtliche Bestellungen und Kunden werden angezeigt, bei denen es eine Übereinstimmung der *CustomerID* gibt.)

Die linke und die rechte äußere Verknüpfung kann je nach der Tabellenreihenfolge zu den gleichen Ergebnissen führen. Beispielsweise liefern die folgenden beiden Verknüpfungen die gleichen Informationen zurück:

```
Customers.CustomerID *= Orders.CustomerID
```

und

```
Orders.CustomerID *= Customers.CustomerID.
```



In den SQL Server-Versionen vor 7.0 konnten Sie die folgende SQL-89-Syntax verwenden:

- ▶ *= schließt alle Zeilen aus der ersten Tabelle und nur die übereinstimmenden Zeilen in der zweiten Tabelle ein (linke äußere Verknüpfung).
- ▶ *= schließt alle Zeilen aus der zweiten Tabelle und nur die übereinstimmenden Zeilen in der ersten Tabelle ein (rechte äußere Verknüpfung).

Leider ist nicht garantiert, dass diese Operatoren die korrekten Ergebnisse liefern. Sobald NULL-Werte im Spiel sind, können Probleme auftreten. Wenn Sie also mit äußeren Verknüpfungen arbeiten, sollten Sie das immer mit der neueren ANSI-Syntax nach SQL-99 formulieren.

Falls Sie sämtliche Titel finden möchten, von denen Exemplare verkauft wurden, und wissen möchten, wie viele davon verkauft worden sind, können Sie die folgende Abfrage mit der neueren SQL-99-Syntax ausführen:



```
SELECT titles.title_id, titles.title, sales.qty
FROM titles LEFT OUTER JOIN sales
ON titles.title_id = sales.title_id
```



title_id	title	qty
BU1032	The Busy Executive's D...	5
BU1032	The Busy Executive's D...	10
BU1111	Cooking with Computers...	25
BU2075	You Can Combat Compute...	35
BU7832	Straight Talk About Co...	15
...	...	...
TC4203	Fifty Years in Bucking...	20
TC7777	Sushi, Anyone?	20

(23 row(s) affected)

Selbstverknüpfungen

Wie der Name schon sagt, korreliert eine *Selbstverknüpfung* die Zeilen in einer Tabelle mit Zeilen in derselben Tabelle. Selbstverknüpfungen werden meist bei Vergleichsabfragen für ein und dieselbe Information verwendet. Wenn Sie beispielsweise alle Autoren auflisten möchten, die in derselben Stadt mit derselben Postleitzahl leben, dann vergleichen Sie die Stadt und die Postleitzahl durch Ausführen der folgenden Abfrage:

Ältere ANSI-Syntax:



```
SELECT au1.au_fname, au1.au_lname,
       au2.au_fname, au2.au_lname,
       au1.city, au1.zip
FROM authors au1, authors au2
WHERE au1.city = au2.city
AND au1.zip = au2.zip
AND au1.au_id < au2.au_id
ORDER BY au1.city, au1.zip
```

Bevorzugte ANSI-Syntax:



```
SELECT au1.au_fname, au1.au_lname,
       au2.au_fname, au2.au_lname,
       au1.city, au1.zip
FROM authors au1
INNER JOIN authors au2 ON au1.city = au2.city
AND au1.zip = au2.zip
WHERE au1.au_id < au2.au_id
ORDER BY au1.city, au1.zip
```



au_fname	au_lname	au_fname	au_lname	city	zip
Cheryl	Carson	Abraham	Bennet	Berkeley	94705
Dean	Straight	Dirk	Stringer	Oakland	94609
Dean	Straight	Livia	Karsen	Oakland	94609
Dirk	Stringer	Livia	Karsen	Oakland	94609
Ann	Dull	Sheryl	Hunter	Palo Alto	94301
Anne	Ringer	Albert	Ringer	Salt Lake City	84152

(6 row(s) affected)

Beachten Sie, dass Sie bei der Anwendung einer Selbstverknüpfung auf eine Tabelle einen Alias für den Tabellennamen erstellen, damit eine Tabelle logisch wie zwei Tabellen behandelt werden kann.



Ein Tabellenalias ist immer dann sinnvoll, wenn Sie eine Verknüpfung von mehreren Tabellen durchführen. Da Sie die Tabelle über den Alias anstelle des Tabellennamens referenzieren, können Sie außerdem eine besser lesbare und kürzere Abfrageanweisung erstellen.

10.12 Unterabfragen

Eine SELECT-Anweisung, die in eine andere SELECT-Anweisung eingebettet ist, wird als *Unterabfrage* bezeichnet. Unterabfragen können zu denselben Ergebnissen wie eine Verknüpfungsoperation führen. In dem nun folgenden Abschnitt erfahren Sie, wie man Unterabfragen einsetzt, welche Typen von Unterabfragen und welche Beschränkungen es gibt, und was korrelierte Unterabfragen sind.

Wie die Unterabfrage verwendet wird

Eine SELECT-Anweisung kann in jede andere SELECT-, INSERT-, UPDATE- oder DELETE-Anweisung eingebettet werden. Falls die Unterabfrage nur einen einzelnen Wert – beispielsweise ein Aggregat – zurückliefert, kann sie überall dort benutzt werden, wo auch der einzelne Wert benutzt werden kann. Liefert die Unterabfrage dagegen eine Liste – wie etwa eine einzelne Spalte aus mehreren Werten – zurück, lässt sich diese Unterabfrage nur in der WHERE-Klausel verwenden.

In vielen Fällen kann anstelle einer Unterabfrage eine Verknüpfung eingesetzt werden. Allerdings gibt es Instanzen, die nur in Form einer Unterabfrage verarbeitet werden. Grundsätzlich besteht zwischen einer Unterabfrage und einer Verknüpfung nur ein geringfügiger Leistungsunterschied, obschon in manchen Fällen eine Verknüpfungsoperation leistungsfähiger als eine Unterabfrage sein kann.

Die Unterabfrage ist immer in Klammern eingeschlossen und wird – außer bei einer korrelierten Unterabfrage – immer vor der Verarbeitung der äußeren Abfrage beendet. Eine Unterabfrage kann eine weitere Unterabfrage enthalten, die dann wiederum eine Unterabfrage enthalten kann usw. Die Verschachtelungstiefe von Unterabfragen ist nur durch die Systemressourcen begrenzt.

Die Syntax für eine verschachtelte SELECT-Anweisung lautet:



```
(SELECT [ALL | DISTINCT] Unterabfrage_Spaltenliste  
[FROM Tabellenliste]  
[WHERE Klausel]  
[GROUP BY Klausel]  
[HAVING Klausel])
```

Typen von Unterabfragen

Eine Unterabfrage kann eine einzelne Zeile oder einen einzelnen Wert überall da zurückliefern, wo auch ein Ausdruck für einen einzelnen Wert verwendet werden kann. Sie kann unter Verwendung der folgenden Vergleichsoperatoren eingesetzt werden: =, <, >, <=, >=, <>, != und !<. Sie kann Einzelspalten mit vielen Werten zurückliefern, die zusammen mit dem Vergleichsoperator der IN-Liste in der WHERE-Klausel verwendbar sind. Eine Unterabfrage kann auch mehrere Zeilen zurückliefern; mit dem Schlüsselwort EXISTS in der WHERE-Klausel lässt sich ein Test auf Vorhandensein durchführen.

Die folgenden Anweisungen liefern alle Autoren mit Wohnsitz in dem Bundesstaat, in dem sich auch die Buchhandlung befindet, die die Bücher ihrer jeweiligen Verleger verkauft:



```
SELECT DISTINCT au_fname, au_lname, state
FROM authors
WHERE state IN
(SELECT state FROM stores)
```

oder



```
SELECT DISTINCT au_fname, au_lname, state
FROM authors
WHERE EXISTS
(SELECT * FROM stores
WHERE state = authors.state)
```



au_fname	au_lname	state
Abraham	Bennet	CA
Akiko	Yokomoto	CA
Ann	Dull	CA
Burt	Gringlesby	CA
...		
Sheryl	Hunter	CA
Stearns	MacFeather	CA

(16 row(s) affected)

Für den Einsatz von Unterabfragen gibt es gewisse Einschränkungen. Eine Unterabfrage ...

- muss in Klammern stehen,
- muss einen Einzelwert zurückliefern, wenn man sie in einem Einzelwertausdruck verwendet,
- kann nicht zusammen mit der ORDER BY-Klausel verwendet werden,
- darf keine ORDER BY-, COMPUTE- oder SELECT INTO-Klausel enthalten,

- darf nicht mehr als eine Spalte in der *Spaltenliste* haben, wenn sie in der IN-Klausel verwendet wird,
- muss SELECT * enthalten, wenn sie in der EXISTS-Klausel verwendet wird,
- darf nicht die Typen text und image in der Auswahlliste enthalten (eine Ausnahme ist die Verwendung von *),
- darf keine GROUP BY- und HAVING-Klauseln enthalten, wenn sie zusammen mit einem unmodifizierten Vergleichsoperator (einem Operator ohne das Schlüsselwort ANY oder ALL) verwendet wird.

Korrelierte Unterabfragen

Eine *korrelierte Unterabfrage* referenziert eine Tabelle von der äußeren Abfrage aus und bearbeitet dann die Zeilen der äußeren Abfrage. Eine korrelierte Unterabfrage unterscheidet sich von einer normalen Unterabfrage, weil die korrelierte Unterabfrage von den Werten der äußeren Abfrage abhängt. Eine normale Unterabfrage wird dagegen unabhängig von der äußeren Abfrage ausgeführt.

Im nun folgenden Beispiel wurde die Verknüpfungsabfrage in Form einer korrelierten Unterabfrage neu geschrieben. Die Abfrage liefert dieselben Ergebnisse zurück und beantwortet die folgende Aufgabe: Zeige alle Autoren an, die in derselben Stadt leben und dieselbe Postleitzahl haben.

Mit einer JOIN-Klausel:



```
SELECT au1.au_fname, au1.au_lname,
       au2.au_fname, au2.au_lname,
       au1.city, au1.zip
FROM authors au1, authors au2
WHERE au1.city = au2.city
AND au1.zip = au2.zip
AND au1.au_id < au2.au_id
ORDER BY au1.city, au1.zip
```



au_fname	au_lname	au_fname	au_lname	city	zip
Cheryl	Carson	Abraham	Bennet	Berkeley	94705
Dean	Straight	Dirk	Stringer	Oakland	94609
Dean	Straight	Livia	Karsen	Oakland	94609
Dirk	Stringer	Livia	Karsen	Oakland	94609
Ann	Dull	Sheryl	Hunter	Palo Alto	94301
Anne	Ringer	Albert	Ringer	Salt Lake City	84152

(6 row(s) affected)

Mit einer korrelierten Unterabfrage:



```
SELECT au1.au_fname, au1.au_lname, au1.city, au1.zip
FROM authors au1
WHERE zip IN
(SELECT zip
FROM authors au2
WHERE au1.city = au2.city
AND au1.au_id <> au2.au_id)
ORDER BY au1.city, au1.zip
```



au_fname	au_lname	city	zip
Abraham	Bennet	Berkeley	94705
Cheryl	Carson	Berkeley	94705
Livia	Karsen	Oakland	94609
Dirk	Stringer	Oakland	94609
Dean	Straight	Oakland	94609
Sheryl	Hunter	Palo Alto	94301
Ann	Dull	Palo Alto	94301
Albert	Ringer	Salt Lake City	84152
Anne	Ringer	Salt Lake City	84152

(9 row(s) affected)

Beachten Sie, dass die gleichen Daten zurückgegeben worden sind; hier sind sie lediglich anders formatiert und besser lesbar dargestellt.

10.13 SELECT INTO

Die Anweisung `SELECT INTO` ermöglicht Ihnen das Erstellen einer neuen Tabelle auf der Grundlage von Abfrageergebnissen. Die neue Tabelle wird über die Spalten, die Sie in der Auswahlliste angegeben, die Tabellen, die Sie in der `FROM`-Klausel genannt, und die Zeilen, die Sie in der `WHERE`-Klausel ausgewählt haben, aufgebaut. Mit der Anweisung `SELECT INTO` können Sie zwei Typen von Tabellen erstellen: Permanente und temporäre. Die Syntax von `SELECT INTO` lautet wie folgt:



```
SELECT Spaltenliste
INTO neuer_Tabellenname
FROM Tabellenliste
WHERE Suchkriterien
```

Beim Erstellen einer permanenten Tabelle müssen Sie die Datenbankoption *SELECT INTO/BULK COPY* einschalten. Die Anweisung `SELECT INTO` ermöglicht es Ihnen, eine Tabelle zu definieren und mit Daten zu versorgen, ohne die üblichen Schritte in der Datendefinition absolvieren zu müssen. Der Name der neuen Tabelle muss in der Datenbank eindeutig sein und den Regeln der SQL Server-Namenskonventionen entsprechen.

Falls Spalten in der *Spaltenliste* der `SELECT`-Anweisung keine Bezeichnungen haben, wie es etwa bei den abgeleiteten Spalten in Aggregatfunktionen der Fall ist, werden die Spalten der neuen Tabelle ebenfalls keine Bezeichnung tragen. Daraus entstehen zwei Probleme:

- Spaltennamen müssen immer eindeutig sein. Falls mehr als eine Spalte keine Überschrift aufweist, wird `SELECT INTO` fehlschlagen.
- Falls es eine Spalte ohne Überschrift in der neuen Tabelle geben sollte, besteht die einzige Möglichkeit zum Abrufen dieser Spalte in der Verwendung von `SELECT *`.

Aus diesen Gründen empfiehlt es sich, immer einen Spaltenalias für abgeleitete Spalten zu erstellen. Da es sich bei `SELECT INTO` außerdem um eine nicht protokollierte Operation handelt, sollten Sie nach einer derartigen Operation immer unverzüglich Ihre Datenbank sichern. Weiterhin können Sie die `SELECT INTO`-Anweisung auch zum Erstellen von temporären Tabellen verwenden.

Es gibt zwei Typen von temporären Tabellen: lokale und globale:

- Eine *lokale* temporäre Tabelle steht nur für die Dauer der aktuellen Benutzersitzung im SQL Server zur Verfügung und wird freigegeben, wenn die Sitzung beendet wird. Eine lokale temporäre Tabelle wird erstellt, indem dem Namen einer neuen Tabelle das Symbol `#` vorangestellt wird.
- Eine *globale* temporäre Tabelle steht sämtlichen Benutzersitzungen im SQL Server zur Verfügung und wird erst dann verworfen, wenn die letzte Benutzersitzung, die auf die Tabelle zugreift, beendet wird. Eine globale temporäre Tabelle wird erstellt, indem dem neuen Tabellennamen die beiden Symbole `##` vorangestellt werden.

Diese temporären Tabellen residieren in der Datenbank *tempdb*. Das folgende Beispiel erzeugt eine temporäre Tabelle namens *#tmpTitles* mit einer Liste von Titel-IDs, dem Titel selbst und dessen Verkaufspreis:



```
SELECT title_id, title, price
INTO #tmpTitles
FROM titles
GO
SELECT * FROM #tmpTitles
GO
```



title_id	Title	price
BU1032	The Busy Executive's Database Guide	19.99
BU1111	Cooking with Computers: Surreptitious ...	11.95
BU2075	You Can Combat Computer Stress!	2.99
.	.	.
.	.	.
.	.	.
TC3218	Onions, Leeks, and Garlic: Cooking Secret ...	20.95
TC4203	Fifty Years in Buckingham Palace Kitchens	11.95
TC7777	Sushi, Anyone?	14.99

(18 row(s) affected)



Der Zweck der Anweisung `SELECT INTO` besteht darin, eine neue Tabelle zu erstellen. Falls Sie in eine schon existierende Tabelle Zeilen hinzufügen möchten, benutzen Sie dazu die Anweisungen `INSERT` oder `INSERT INTO`, die Sie beide in der morgigen Lektion kennen lernen.

10.14 Der UNION-Operator

Die Ergebnisse von zwei oder mehreren Abfragen können Sie mithilfe des `UNION`-Operators in einem einzigen Resultset zusammenfassen. Dabei werden standardmäßig Duplikatzeilen eliminiert, was Sie jedoch durch die Verwendung von `UNION` mit dem Schlüsselwort `ALL` (liefert sämtliche Zeilen einschließlich der Duplikate zurück) verhindern können.



```
SELECT Spaltenliste [INTO Klausel]
[FROM Klausel]
[WHERE Klausel]
[GROUP BY Klausel]
[HAVING Klausel]
[UNION [ALL]
SELECT Spaltenliste
[FROM Klausel]
[WHERE Klausel]
[GROUP BY Klausel]
[HAVING Klausel]...]
[ORDER BY Klausel]
[COMPUTE Klausel]
```


Für den Einsatz des UNION-Operators gelten folgende Regeln:

- Alle *Spaltenlisten* müssen dieselbe Anzahl Spalten, dieselbe Spaltenreihenfolge und gleichartige Datentypen aufweisen.
- Falls Sie in einer der Abfragen eine INTO-Klausel verwenden, muss diese in der ersten Abfrage erscheinen.
- GROUP BY- und HAVING-Klauseln können Sie nur in individuellen Abfragen einsetzen.
- ORDER BY- und COMPUTE-Klauseln sind nur am Ende der UNION-Anweisung erlaubt, um die Reihenfolge der endgültigen Ergebnisse festzulegen oder um Summenwerte zu berechnen.
- Die Spaltennamen werden über die erste SELECT-*Spaltenliste* ermittelt.

Im folgenden Beispiel können Sie den UNION-Operator verwenden, um Daten aus zwei getrennten Abfragen zusammenzuführen und die beiden Resultsets zu einem einzigen Resultset zu kombinieren:



```
SELECT title, stor_name, ord_date, qty
FROM titles, sales, stores
WHERE titles.title_id = sales.title_id
AND stores.stor_id = sales.stor_id
UNION
SELECT title, 'No Sales', NULL, NULL
FROM titles
WHERE title_id NOT IN
(SELECT title_id FROM sales)
ORDER BY qty
```



title	stor_name	ord_date	qty
-			
Net Etiquette (null)	No Sales	(null)	
The Psychology ... (null)	No Sales	(null)	
Is Anger the Enemy?	Eric the Read Books	Sep 13 1994...	3
...	...	...	...
Onions, Leeks, and Garlic: ...	News & Brews	Jun 15 19...	40
Secrets of Silicon Valley	Barnum's	May 24 1993...	50
Is Anger the Enemy?	Barnum's	Sep 13 1994...	75

(23 row(s) affected)

10.15 Zusammenfassung

In der heutigen Lektion haben Sie sich die wohl bedeutendste SQL-Anweisung überhaupt – die **SELECT**-Anweisung – erarbeitet. Mit dieser Anweisung können Sie Daten aus Datenbanken abrufen. Damit Sie mit diesen Abfragen auch Daten nach bestimmten Kriterien abrufen können, haben Sie außerdem arithmetische Operatoren, mathematische Funktionen, Zeichenfolgenfunktionen, Datums- und Zeitfunktionen sowie Systemfunktionen kennen gelernt.

Oftmals befinden sich die Daten, mit denen Sie arbeiten möchten, nicht in dem von Ihnen gewünschten Format oder Datentyp. Diese Lektion hat gezeigt, wie Sie die Funktionen der Datenumwandlung einsetzen können, um Ihre Daten von einem Typ in einen anderen zu konvertieren.

Außerdem haben Sie gelernt, wie Sie mit Vergleichsoperatoren, Bereichen, Listen und Zeichenfolgen umgehen, um Informationen gezielt abzurufen. Sie haben erfahren, wie Sie Zeilen mit **NULL**-Werten eliminieren und wie sich mit dem Schlüsselwort **DISTINCT** Duplikate aus dem Resultset entfernen lassen.

Weitere Themen in dieser umfangreichen Lektion waren das Sortieren von Daten mit der **ORDER BY**-Klausel, das Zusammenfassen mit Aggregatfunktionen und das Gruppieren mit **GROUP BY** und **HAVING**.

Darüber hinaus haben Sie noch die Klauseln **COMPUTE** und **COMPUTE BY** kennen gelernt, die zum Erstellen neuer Zeilen mit Zusammenfassungen und Detaildaten verwendet werden. Mit den Operatoren **ROLLUP** und **CUBE** lassen sich so genannte Superaggregate erzeugen. Diese Operatoren setzen Sie in Verbindung mit der **GROUP BY**-Klausel ein.

Weiterhin haben Sie gelernt, wie Sie Tabellen unter Verwendung der ANSI-Syntax und der Syntax des SQL Servers miteinander verknüpfen und wie man mit Unterabfragen arbeitet.

Die **SELECT INTO**-Anweisung ermöglicht Ihnen die Erstellung einer neuen Tabelle auf der Grundlage von Abfrageergebnissen.

Zum Schluss haben Sie noch erfahren, wie Sie die Ergebnisse von zwei oder mehr Abfragen unter Verwendung des **UNION**-Operators in einem einzigen Resultset zusammenfassen können.

10.16 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F *Es sieht so aus, als ob SELECT-Anweisungen ganz schön lang werden können. Gibt es eigentlich eine Grenze dafür, wie lang sie werden dürfen?*

A Ja, Abfragen dürfen nicht größer als 64 KB werden.

F *Wo findet man normalerweise SELECT Anweisungen?*

A SELECT-Anweisungen finden sich oft in Frontend-Anwendungen wie etwa Visual Basic und PowerBuilder. Sie finden SELECT-Anweisungen außerdem auch eingebettet in gespeicherten Prozeduren, Triggern, Ereignissen, Warnungen und an vielen anderen Stellen in Ihrem SQL Server.

F *Wann sollte man die ANSI- und wann die SQL Server-Syntax verwenden?*

A Im Allgemeinen sollten Sie vorrangig mit der ANSI-Syntax arbeiten, da diese Form der Implementierung einen Standard darstellt, was den Code portabel gestaltet. Außerdem haben Sie nur geringfügige oder gar keine Leistungsgewinne bzw. -verluste zu erwarten.

F *Was genau macht die folgende Anweisung?*

```
SELECT SUM(qty) FROM sales
```

A Sie liefert die Summe des Feldes qty in der Tabelle sales zurück.

F *Ist der Einsatz von COMPUTE und COMPUTE BY empfehlenswert?*

A Diese Anweisungen sind durchaus zulässig, allerdings liefern sie nichtrelationale Daten zurück.



Kontrollfragen

1. Welche Ergebnisse liefern diese Abfragen zurück?

a.

```
SELECT * FROM authors
WHERE au_lname LIKE 'M%'
```

b.

```
SELECT emp_id AS EmployeeID,
       lname AS LastName,
       fname AS FirstName
FROM employee
```

c.

```
SELECT ROUND ($7725.53, 1)
```

d.

```
SELECT lname + ', ' + SUBSTRING(fname,1,1) + '.' AS Name,
       emp_id AS EmployeeID
FROM employee
```

2. Können Sie eine Unterabfrage als Verknüpfung und umgekehrt ausführen?
3. Richtig oder Falsch: ROLLUP und CUBE liefern keine Summenwerte.
4. Können Sie eine SELECT INTO-Anweisung zum Aufbau einer temporären Tabelle verwenden, auf die jeder zugreifen kann?

Übungen

1. Sie möchten *title_id*, *title* und *price* für sämtliche Bücher abrufen, die eine Verleger-ID (*publisher ID*) von 0877 haben oder das Wort *computer* im Titel enthalten; außerdem soll der Preis NOT NULL sein. Wie sehen die entsprechenden Transact-SQL-Anweisungen aus? (Hinweis: Benutzen Sie die Tabelle *titles*.)
2. Schreiben Sie eine Abfrage zum Abrufen sämtlicher Bücher in der Tabelle *titles*, die Preise ungleich NOT NULL haben.
3. Schreiben Sie eine Abfrage zum Auflisten sämtlicher Buchtitel und Preise in der Tabelle *titles* in absteigender Sortierreihenfolge nach dem Preis.
4. Schreiben Sie eine Abfrage, die den Durchschnittswert der *ytd_sales*-Zahlen aus der Tabelle *titles* in der Datenbank *pubs* liefert.

5. Schreiben Sie unter Verwendung der Klauseln `GROUP BY` und `HAVING` eine Abfrage, die sämtliche Bücher mit mehr als einem Autor findet. (Hinweis: Benutzen Sie die Tabelle *titleauthor*.)
6. Schreiben Sie unter Verwendung der `COMPUTE BY`-Klausel eine Abfrage, die *stor_id* ausgibt und eine fortlaufende Summe der bestellten Menge anzeigt. Benutzen Sie dazu die Tabelle *sales*.
7. Erstellen Sie mithilfe von Verknüpfungen (entweder nach SQL-Server- oder nach ANSI-Syntax) eine Abfrage, die den Vornamen, den Nachnamen und die Buchtitel eines Autors zurückgibt. (Verwenden Sie die Felder *au_fname* und *au_lname* der Tabelle *authors* und das Feld *title* aus der Tabelle *titles*. Sie müssen zwei Verknüpfungen einsetzen – eine von *authors* mit *titleauthor* und eine von *titles* mit *titleauthor*.)
8. Erstellen Sie eine Unterabfrage zum Auffinden von Autoren (*authors*), die in denselben Bundesstaaten leben, in denen sich die Buchhandlungen (*stores*) befinden.
9. Erstellen Sie eine temporäre Tabelle, die sämtliche Informationen aus der Tabelle *employee* enthält. Testen Sie das Vorhandensein Ihrer neuen Tabelle, indem Sie Daten daraus auswählen.



Daten bearbeiten



Gestern haben Sie gelernt, wie man Daten mit der `SELECT`-Anweisung abrufen. Die heutige Lektion konzentriert sich nun darauf, wie Sie die Daten in Ihren `SQL` Server-Tabellen mit `INSERT`-, `UPDATE`- und `DELETE`-Anweisungen bearbeiten können. Bevor Sie sich jedoch mit den eigentlichen Anweisungen befassen, werfen Sie zunächst einen Blick auf den Zusammenhang zwischen Datenbearbeitungen und Transaktionsprotokollierungen.

11.1 Transaktionsprotokolle

`SQL` Server verfolgt Änderungen in der Datenbank, indem er nahezu jeden Änderungsvorgang, der in der Datenbank geschieht, im Transaktionsprotokoll aufzeichnet. Den Speicherort und die Verwaltung des Transaktionsprotokolls haben Sie bereits am Tag 4 kennen gelernt. Seien Sie sich immer dessen bewusst, dass alle Änderungen, die Sie an Ihren Daten vornehmen, auch in das Transaktionsprotokoll aufgenommen werden: Mit jeder `INSERT`-Anweisung wird eine Kopie der gesamten neuen Zeile in das Transaktionsprotokoll geschrieben; und bei einem `DELETE` wird eine Kopie des gesamten gelöschten Datensatzes ins Transaktionsprotokoll aufgenommen.

Bei der `UPDATE`-Operation ist die Sache nicht ganz so eindeutig. Bei manchen `UPDATE`-Operationen protokolliert `SQL` Server lediglich die veränderten Bytes, während er für viele andere `UPDATE`-Vorgänge gleich zwei Einträge ins Protokoll aufnehmen muss: die gesamte alte Version und die gesamte neue Version einer Zeile. Nach welchen Regeln das geschieht, können wir Rahmen dieses Buches allerdings nicht erläutern. Außerdem wird die Anzahl der Daten, die ins Transaktionsprotokoll aufgenommen werden, noch durch die Anzahl der Indizes bestimmt (wie Tag 13 zeigt).

Es gibt auch Ausnahmen bei der Protokollierung. Ein Beispiel ist das Dienstprogramm `BCP` (Bulk Copy Program, siehe dazu Tag 10), das äquivalent zu mehreren `INSERT`-Anweisungen ist. In der schnellen Version von `BCP` werden die einzelnen Zeilen beim Einfügen in die Tabelle nicht in das Transaktionsprotokoll geschrieben; bei indizierten Tabellen kommt jedoch jede neue Zeile auch ins Transaktionsprotokoll. Beachten Sie aber, dass unabhängig von der verwendeten Version von `BCP` (schnell oder langsam) immer auch eine gewisse Protokollierung des Speicherplatzes, der von der `BCP`-Operation reserviert wird, stattfindet. In der gestrigen Lektion haben Sie die Anweisung `SELECT INTO` kennen gelernt, die ebenfalls nicht protokolliert wird (mit der gerade eben erwähnten Ausnahme hinsichtlich der Speicherplatzreservierung). Die heutige Lektion stellt die Anweisung `TRUNCATE TABLE` vor, die einer `DELETE`-Operation ohne die Protokollierung jeder einzelnen gelöschten Zeile entspricht.

Nach dieser kurzen Einführung zur Transaktionsprotokollierung befassen wir uns jetzt mit den Anweisungen, die eine Protokollierung auslösen. Bevor es aber richtig losgeht, sollten Sie sich die Zeit nehmen, Ihre Daten in Sicherungstabellen zu kopieren, so dass Sie die

Tabellen, mit denen Sie im Verlauf der Übung arbeiten, später wiederherstellen können.

Starten Sie die folgenden Abfragen zum Erstellen von Kopien der Tabellen *publishers* und *sales*. Um diese Sicherungsanweisungen ausführen zu können, ist die Datenbankoption *Select Into/Bulk Copy* einzuschalten (siehe dritte Zeile).



```
USE pubs
GO
EXEC sp_dboption 'pubs', 'select into/bulkcopy', TRUE
GO
SELECT * INTO tmpPublishers FROM publishers
SELECT * INTO tmpStores FROM stores
SELECT * INTO tmpTitles FROM titles
SELECT * INTO tmpSales FROM sales
GO
EXEC sp_dboption 'pubs', 'select into/bulkcopy', FALSE
GO
```

11.2 Daten einfügen

Mit der elementaren *INSERT*-Anweisung wird immer nur eine Zeile in eine Tabelle eingefügt. Durch gewisse Änderungen der *INSERT*-Anweisung erreichen Sie, dass durch Auswahl von Daten aus einer anderen Tabelle oder über die Ausführung einer gespeicherten Prozedur mehrere Zeilen auf einmal eingefügt werden können. Bei jeder dieser Operationen müssen Sie immer auch eine Kenntnis der Struktur der Tabelle haben, für die Sie die Einfügungen vornehmen. Zu diesen Informationen gehören:

- Die Anzahl der Spalten in der Tabelle
- Der Datentyp jeder Spalte
- Der Name der Spalten (bei bestimmten *INSERT*-Anweisungen)
- Einschränkungen und Spalteneigenschaften, wie etwa *identity*, *uniqueidentifier* oder *quantity > 20*.

Mehr zu Einschränkungen und Identitätsspalten erfahren Sie in Lektion 14. Die Syntax der *INSERT*-Anweisung lautet folgendermaßen:



```
INSERT [INTO]
    {<Tabelle_oder_Sicht>}
    {[(Spaltenliste)]}
VALUES
    ({DEFAULT |
```



```
konstanter_Ausdruck) [...n]) |  
Select_Anweisung |  
Execute_Anweisung) |  
DEFAULT VALUES}
```

Wobei gilt:

```
<Tabelle_oder_Sicht> :: =  
{ Tabellename | Sichtname  
| Rowset_Funktion  
}[...n]
```

Am Einfachsten lassen sich die Spalten zusammen mit ihren Namen und Datentypen über das Dialogfeld **TABELLENEIGENSCHAFTEN** im Enterprise Manager herausfinden (siehe Abbildung 11.1). Führen Sie zum Öffnen des Dialogfelds **TABELLENEIGENSCHAFTEN** die folgenden Arbeitsschritte aus:

1. Wählen Sie eine Datenbank im Ordner *Datenbanken* aus.
2. Markieren Sie eine Tabelle im Ordner *Tabellen*.
3. Klicken Sie mit der rechten Maustaste im Detailbereich auf die Tabelle und wählen Sie **EIGENSCHAFTEN** aus dem Kontextmenü.

Falls Sie den **SQL Server Enterprise Manager** gerade nicht verfügbar haben, können Sie auch die gespeicherte Systemprozedur `sp_help` ausführen, die Ihnen die gleichen Informationen liefert:



```
USE pubs  
GO  
EXEC sp_help publishers  
GO
```

Ein Teil der Ergebnisse sieht etwa folgendermaßen aus:



Column_name	Type	Computed	Length
pub_id	char no		4
pub_name	varchar no		40
city	varchar no		20
state	char no		2
country	varchar no		30

In der vorstehenden Ausgabe sehen Sie den Spaltennamen, den Datentyp und die Position jeder einzelnen Spalte. Diese Informationen benötigen Sie dann beim Aufbau Ihrer **INSERT-Anweisungen**.



Abbildung 11.1:
Das Dialogfeld Tabelleneigenschaften für die Tabelle publishers

Informationen können Sie auch mit INFORMATION_SCHEMA-Sichten zusammentragen. Um zum Beispiel die Spaltennamen und Datentypen für das obige Beispiel zu ermitteln, führen Sie die folgende Abfrage aus:



```
USE pubs
GO
SELECT * FROM INFORMATION_SCHEMA.Columns
WHERE TABLE_NAME = 'publishers'
GO
```

Die Anweisung INSERT VALUES

Die einfachste Form der INSERT-Anweisung verlangt einen Wert für jede Spalte in der Tabelle – und zwar in der Reihenfolge, in der die Spalten definiert worden sind. Diese Reihenfolge ist auch im Beispiel oben zu sehen. Zum Einfügen einer einzelnen Spalte in die Tabelle *tmpPublishers* können Sie nun den folgenden Befehl ausführen:

```
USE pubs
GO
INSERT INTO tmpPublishers
VALUES('9956', 'A New Publisher', 'Poulsbo', 'WA', 'USA')
```

Wenn Sie INSERT-Berechtigungen für diese Tabelle besitzen und damit keine Einschränkungen verletzen, sollten Sie nun von SQL Server folgende Rückmeldung erhalten:



(1 row(s) affected)



Da sämtliche Spalten einen Zeichentyp aufweisen, sind alle Werte in Hochkommas eingeschlossen. Numerische Werte werden dagegen nicht in Hochkommas eingeschlossen.



Ungebundene Tabellen-Steuerelemente, wie sie in den grafischen Frontend-Werkzeugen eingesetzt werden, fügen Daten nicht automatisch auch in die Datenbank ein. Dazu muss der Entwickler Code schreiben, der sämtliche in der Tabelle geänderten Daten in einer Programmschleife ermittelt und dann diese Zeilen über INSERT/VALUES in die Datenbank schreibt.

Die einfachste Form der INSERT-Anweisung erfordert es, dass Sie explizit für jede Spalte einen Wert bereitstellen, der sich auch an der korrekten Position befindet. Falls Sie die Werte in einer anderen Reihenfolge oder sogar für eine Spalte überhaupt keinen expliziten Wert angeben möchten, können Sie dazu auf eine andere Form der INSERT-Anweisung zurückgreifen.

Die folgende INSERT-Anweisung enthält eine Liste von Spaltennamen vor der VALUES-Klausel. Die Liste beinhaltet lediglich eine Untermenge der Spaltennamen in der Tabelle, so dass Sie auch nur die Werte für die genannten Spalten angeben müssen.

```
INSERT INTO tmpPublishers(state, pub_id)
VALUES('AK', '9932')
```



Wenn sich Clientanwendungen wie Microsoft Visual Basic oder PowerBuilder mit einer Datenbank verbinden und Daten einfügen, verwenden sie ebenfalls die INSERT/VALUES-Anweisungen, die Sie gerade kennen gelernt haben.

Was passiert nun aber mit den Spalten, die nicht in der Spaltenliste erwähnt sind? Beim Einfügen einer neuen Zeile muss jede Spalte einen eigenen Wert besitzen. Liefern Sie keinen Wert mit, dann muss SQL Server in der Lage sein, einen solchen festzulegen. Damit SQL Server dies tun kann, muss jede in der Spaltenliste nicht aufgeführte Spalte eines der folgenden Kriterien erfüllen:

- Die Spalte besitzt einen zugeordneten Standardwert.
- Die Spalte ist eine Identitätsspalte.

- Die Spalte erlaubt NULL-Werte.
- Die Spalte ist vom Typ `rowversion` (in vorherigen SQL Server-Versionen als `timestamp` bezeichnet).

Standardwerte und Identitätsspalten behandelt Tag 14; NULL-Werte und den Datentyp `rowversion` haben Sie am 9. Tag kennen gelernt.

Die obige `INSERT`-Anweisung stellt für die Felder `pub_name`, `city` und `country` keine Werte bereit. In der Tabelle `publishers` selbst erlauben die Spalten `pub_name` und `city` beide NULL-Werte, während für `country` der Standardwert »USA« festgelegt ist. Nach der Ausführung der `INSERT`-Anweisung können Sie die folgende Abfrage laufen lassen, um die Zeile anzuzeigen, die Sie gerade eingefügt haben:



```
SELECT * FROM tmpPublishers
WHERE pub_id = '9932'
```

Beachten Sie die NULL-Werte und den Standardwert für `country`:

Pub_id	Pub_name	City	state	country
9932	(null)	(null)	AK	USA

Sie erhalten eine Fehlermeldung, wenn Sie versuchen, eine `INSERT`-Anweisung auszuführen, bei der Sie die Werte für Spalten, die nicht den genannten Kriterien entsprechen, freigelassen haben.

Die folgende `INSERT`-Anweisung stellt keinen Wert für die Spalte `pub_id` bereit:



```
INSERT INTO tmpPublishers(pub_name, city, state)
VALUES('The Best Books', 'New Orleans', 'LA')
```



```
Server: Nachr.-Nr. 515, Schweregrad 16, Status 2, Zeile 1
Der Wert NULL kann in die pub_id-Spalte, pubs.dbo.tmpPublishers-Tabelle
nicht eingefügt werden. Die Spalte lässt NULL-Werte nicht zu. Fehler
bei INSERT.
Die Anweisung wurde beendet.
```

Beachten Sie, dass die Fehlermeldung nicht ganz eindeutig ist. Das Problem besteht nicht darin, dass die Spalte `pub_id` keine NULL-Werte erlaubt, sondern dass sie auch keinen Standardwert hat und keine Identitätsspalte ist.



Standardwerte

Es gibt noch eine weitere Variante der einfachen, einzeiligen `INSERT`-Anweisung, die Sie einsetzen können, wenn Sie keine Spaltenliste angeben möchten, aber SQL Server veranlassen wollen, immer dann Standardwerte zu verwenden, wenn diese auch vorhanden sind (dies schließt auch `NULL`-Werte und Identitätswerte ein). Über das Schlüsselwort `DEFAULT` in der eigentlichen Werteliste können Sie SQL Server mitteilen, dass er den zu verwendenden Wert selbst ermitteln soll. Dazu folgendes Beispiel:

```
INSERT INTO tmpPublishers  
VALUES('9950', DEFAULT, DEFAULT, 'AK', DEFAULT)
```

Falls jede Spalte in einer Tabelle einen Standardwert besitzt, den SQL Server verwenden kann, gibt es noch eine weitere Variation, die Sie benutzen können. Dazu teilen Sie SQL Server über die Schlüsselwörter `DEFAULT VALUES` einfach mit, dass er alle Standardwerte verwenden soll, wie es folgende Anweisung zeigt:

```
INSERT INTO tmpPublishers DEFAULT VALUES
```

Einfügen von Daten mit `SELECT`

Alle bisherigen `INSERT`-Anweisungen haben nur eine einzige Zeile in eine Tabelle eingefügt. Falls Sie nun mehrere Zeilen auf einmal einfügen möchten, müssen Sie eine Datenquelle haben, in der diese Zeilen bereits vorhanden sind. Eine derartige Quelle stellt typischerweise eine andere Tabelle oder eine Verknüpfung zwischen zwei oder mehr Tabellen dar. Bei dieser Form einer `INSERT`-Anweisung ermitteln Sie die einzufügenden Datenzeilen über eine Unterabfrage, die Sie in der gestrigen Lektion kennen gelernt haben. Das Resultset der Unterabfrage ergibt in diesem Fall den Zeilenbereich, der eingefügt werden soll. Dabei muss die Anzahl der Spalten im Resultset der Unterabfrage genau mit der Anzahl der Spalten in der Tabelle übereinstimmen; außerdem müssen die Spalten kompatible Datentypen besitzen. Im folgenden ersten Beispiel erstellen Sie eine Adresstabelle:



```
CREATE TABLE address_list  
(name varchar(50) not null,  
address varchar(50) not null,  
city varchar(20) not null,  
state char(2) )
```



Befehl(e) erfolgreich abgeschlossen.

Diese Tabelle besitzt vier Zeichenfelder. Jede `SELECT`-Anweisung, über die Sie die Tabelle mit Daten füllen möchten, muss daher vier Zeichenspalten liefern. Dazu ein Beispiel:



```
INSERT INTO address_list
SELECT stor_name, stor_address, city, state
FROM tmpStores
```



(6 row(s) affected)



Anders als bei den Unterabfragen in der gestrigen Lektion wird eine Unterabfrage, die in einer `INSERT`-Anweisung steht, nicht in Klammern eingeschlossen.

Die in der Unterabfrage angegebenen Spaltennamen werden ignoriert, da die Tabelle bereits Spaltennamen für jedes Feld besitzt.

Sie können noch eine `INSERT`-Anweisung ausführen, um weitere Zeilen in die Tabelle *address_list* aufzunehmen. Nehmen Sie dazu einmal an, dass Sie die Namen und Adressen aus der Tabelle *authors* hinzufügen möchten. Anstelle eines einzigen Namens enthält die Tabelle *authors* aber eine Spalte mit dem Vornamen (*au_fname*) und eine Spalte mit dem Nachnamen (*au_lname*). Die Tabelle *address_list* erwartet jedoch nur einen einzigen Wert für ihre Spalte *name*, so dass Sie den Vor- und den Nachnamen verketteten müssen. Dabei können Sie die beiden Namensbestandteile nur ein Komma und ein Leerzeichen trennen.



```
INSERT INTO address_list
SELECT au_lname + ', ' + au_fname, address, city, state
FROM authors
```



(23 row(s) affected)

Im folgenden letzten Beispiel erstellen Sie eine Tabelle mit den Namen sämtlicher Verleger und den Buchtiteln, die jeder Verleger produziert hat.



```
CREATE TABLE publisher_list
(pub_name varchar(40) NULL,
title varchar(80) NULL)
```



Zum Füllen dieser Tabelle mit Daten benötigen Sie eine Verknüpfung zwischen den Tabellen *tmpPublishers* und *tmpTitles*. Zu diesem Zweck sollten Sie eine äußere Verknüpfung verwenden, damit auch diejenigen Verleger eingeschlossen werden, die gegenwärtig noch keine Bücher produziert haben.



```
INSERT INTO publisher_list
SELECT pub_name, title
FROM tmpPublishers LEFT OUTER JOIN tmpTitles
ON tmpPublishers.pub_id = tmpTitles.pub_id
```

Einfügen von Daten mit gespeicherten Prozeduren

SQL Server bietet noch eine weitere Möglichkeit zum Einfügen von Zeilen in eine Tabelle: Falls ein Resultset von einer gespeicherten Prozedur zurückgeliefert wird und Sie die Anzahl der Spalten in diesem Resultset und ihren Typ kennen, können Sie die beim Aufruf dieser gespeicherten Prozedur zurückgegebenen Daten auch mit einem `INSERT` in eine Tabelle einfügen.



Wenn die gespeicherte Systemprozedur mehr als ein Resultset zurückliefert, müssen Sie sich vergewissern, dass alle Ergebnisse dieselbe Anzahl Spalten und die gleichen Datentypen in den korrespondierenden Spalten zurückliefern.

Ihre eigenen gespeicherten Prozeduren schreiben Sie zwar erst am 15. Tag, aber für den Moment können Sie dafür auch eine gespeicherte Systemprozedur verwenden. Die gespeicherte Systemprozedur `sp_spaceused` liefert Informationen über den von einer einzelnen Tabelle verwendeten Speicherplatz zurück. Für die Daten, die von dieser Prozedur erzeugt werden, erstellen Sie eine eigene Tabelle, die diese Daten aufnimmt, wobei Sie zu gegebener Zeit jeweils eine `INSERT`-Anweisung zum Einfügen einer Zeile verwenden. Auf diese Weise können Sie das Wachstum der Tabelle im Laufe der Zeit verfolgen.

Sehen Sie sich nun die Ausgabe der gespeicherten Systemprozedur `sp_spaceused` an (beachten Sie, dass die Ergebnisse auf Ihrem System etwas anders aussehen können):



```
EXEC sp_spaceused tmpPublishers
```



name	rows	reserved	data	index_size	unused
tmpPublishers	11	32 KB	2 KB	4 KB	26 KB

Die Prozedur liefert sechs Spalten zurück, die sämtlich Zeichendaten darstellen. Es sieht zwar so aus, als ob die zweite Spalte numerisch ist, aber in Wirklichkeit ist sie es nicht. Den Datentyp können Sie feststellen, indem Sie sich entweder den Code für die gespeicherte Systemprozedur `sp_spaceused` ansehen (wie, erfahren Sie am Tag 15) oder versuchen, eine Tabelle mit einer Ganzzahlspalte zu erstellen. Beachten Sie die Fehlermeldung, die Sie erhalten, wenn Sie eine Zeile einfügen wollen.

Die Ergebnisse können Sie in der folgenden Tabelle speichern:



```
CREATE TABLE space_usage
(table_name varchar(30) not null,
 rows varchar(9),
 reserved varchar(10),
 data varchar(10),
 index_size varchar(10),
 unused varchar(10) )
```

Zum Einfügen in diese Tabelle führen Sie die gespeicherte Systemprozedur `sp_spaceused` aus:



```
INSERT INTO space_usage
EXEC sp_spaceused tmpPublishers
```

Eine interessante Erweiterung dieser Möglichkeit, Zeilen über eine gespeicherte Systemprozedur einfügen zu können, besteht darin, Zeilen über eine remote gespeicherte Prozedur einzufügen. Wenn Sie zum Beispiel eine Prozedur auf einem Remoteserver liegen haben, die sämtliche Zeilen in einer Tabelle auswählt, können Sie diese Prozedur dazu benutzen, alle Zeilen vom remoten zum lokalen Server zu kopieren. Zum Ausführen einer remote gespeicherten Prozedur müssen Sie folgende Angaben im remoten Prozeduraufruf machen:

- Servername
- Datenbankname
- Name des Besitzers der Prozedur
- Prozedurname

Wenn Sie beispielsweise einen SQL Server namens *Wildlife* mit einer Datenbank namens *Water* und eine Prozedur namens *Fish* mit dem Besitzer *dbo* haben, können Sie die folgende Abfrage ausführen:

```
INSERT INTO local_table
EXEC Wildlife.Water.dbo.Fish
```



Beim Einfügen von Daten in eine Tabelle mit einer Identitätsspalte gibt es noch einige weitere Dinge in Betracht zu ziehen, auf die Tag 14 eingeht. Dort erfahren Sie auch, welche Restriktionen auftauchen können, wenn Sie Werte in Tabellen mit Prüfeinschränkungen oder Einschränkungen der referenziellen Integrität einfügen wollen.

11.3 Daten löschen

Mit einer weiteren Anweisung zur Datenbearbeitung haben Sie die Möglichkeit, eine oder mehrere Zeilen in einer Tabelle zu löschen:

```
DELETE [FROM] {Tabellenname | Sichtname}
[WHERE Klausel]
```



Beachten Sie, dass das Schlüsselwort `FROM` ebenso wie die `WHERE`-Klausel optional ist.

Die folgende `DELETE`-Anweisung entfernt alle Zeilen aus der Tabelle *tmpSales*:



```
DELETE tmpSales
```



(21 row(s) affected)

Zum Entfernen einer Untermenge von Zeilen aus einer Tabelle spezifizieren Sie über die `WHERE`-Klausel diejenigen Zeilen, die entfernt werden sollen. Zu den Bedingungen in einer `WHERE`-Klausel können sämtliche Bedingungen gehören, die Sie am 10. Tag kennen gelernt haben, einschließlich relationaler Operatoren (`<`, `>`, `=`) und der Schlüsselwörter `IN`, `LIKE` und `BETWEEN`.

Die folgende `DELETE`-Anweisung entfernt sämtliche Bücher mit dem *pub_name* »New Moon Books« aus der Tabelle *publisher_list*:



```
DELETE publisher_list
WHERE pub_name = 'New Moon Books'
```

DELETE über eine Suchtabelle

Eine einzelne DELETE-Anweisung entfernt nur die Zeilen einer einzigen Tabelle. SQL Server gibt Ihnen jedoch auch die Möglichkeit, eine weitere Tabelle in Ihre DELETE-Anweisung einzubinden, die als *Suchtabelle* (auch als *Lookup-Tabelle* bezeichnet) verwendet werden kann. Eine Suchtabelle erscheint normalerweise in einer Unterabfrage. Im nächsten Beispiel sollen Sie sämtliche von New Moon Books publizierten Titel entfernen, wobei die Tabelle *tmpTitles* eine der zu bearbeitenden Tabellen ist, die jedoch nur die Verleger_ID (*pub_id*), nicht jedoch den Verlegernamen (*pub_name*) enthält. Die Verleger_ID für New Moon Books müssen Sie über die Tabelle *tmpPublishers* ausfindig machen, woraus sich dann die Zeilen ergeben, die in der Tabelle *tmpTitles* zu entfernen sind.



```
DELETE publisher_list
WHERE pub_name =
      (SELECT pub_name FROM tmpPublishers
       WHERE pub_id = '9956')
```



(1 row(s) affected)

Die Unterabfrage greift auf *tmpPublishers* zu und liefert einen einzigen Wert für *pub_name* zurück. Dieser Wert wird dann dazu verwendet, die Zeilen in *publisher_list* zu ermitteln, die Sie löschen müssen; d.h., sämtliche Zeilen, in denen *pub_name* gleich dem Rückgabewert ist. Denken Sie daran, dass nur eine Zeile zurückgegeben wird, weil *pub_id* der Primärschlüssel der Tabelle *publishers* ist. Falls Ihre Unterabfrage mehr als eine Zeile zurückgeben würde, müssten Sie anstelle des Gleichheitszeichens (=) das Schlüsselwort IN benutzen.

Die obige Abfrage lässt sich effizienter mit einer JOIN-Klausel formulieren. Wenn Sie einen JOIN mit einem DELETE-Befehl verwenden, geben Sie zwei FROM-Bedingungen an: eine spezifiziert die Tabelle, aus der die Datensätze zu löschen sind, und eine gibt die Verknüpfungstabelle an. Das folgende Beispiel führt den gleichen Löschvorgang aus, wie weiter oben gezeigt:

```
DELETE publisher_list
FROM publisher_list p1 JOIN tmpPublishers p
ON p1.pub_name = p.pub_name
AND p.pub_id = '9956'
```

Im folgenden Beispiel wird eine Suchtabelle verwendet, die mehr als einen Wert zurückliefert. Diesmal sollen Sie alle Zeilen aus der Tabelle *tmpSales* entfernen, die auf den Verkauf von Wirtschaftsliteratur (*business books*) hinweisen. Dabei enthält die Tabelle

tmpSales die *title_id* des verkauften Buches, jedoch nicht dessen Kategorie (*type*). Um die *title_id* herauszufinden, die mit Wirtschaftsliteratur korrespondiert, müssen Sie auf die Tabelle *tmpTitles* zugreifen. Da Sie nun die Daten in der Tabelle *tmpSales* weiter oben in dieser Lektion schon gelöscht haben, müssen Sie sie nun erneut mit Daten versorgen. Vielleicht erinnern Sie sich noch daran, dass Sie zu Beginn dieses Kapitels sämtliche *sales*-Daten in eine andere Tabelle kopiert haben? Sie werden nun die Tabelle *tmpSales* löschen, neu erstellen und dann unter Verwendung einer `SELECT INTO`-Anweisung mit Daten füllen. Führen Sie dazu den folgenden Code aus:



```
USE pubs
GO
DROP TABLE tmpSales
GO
SELECT * INTO tmpSales
FROM Sales
GO
```

Entfernen Sie nun sämtliche Zeilen aus der Tabelle *tmpSales*, deren Kategorie (*type*) auf »business« lautet.



```
DELETE tmpSales
WHERE title_id IN
    (SELECT title_id FROM tmpTitles
     WHERE type = 'business')
(5 row(s) affected)
```

Die Unterabfrage greift auf die Tabelle *tmpTitles* zu und liefert eine Liste der Werte von *title_id* zurück. Diese Werte werden dann anschließend dazu benutzt, die Zeilen in *tmpSales* zu bestimmen, die gelöscht werden sollen; also sämtliche Zeilen, bei denen die *title_id* gleich einem der zurückgelieferten Werte ist.

Transact-SQL weist eine Erweiterung auf, die Ihnen die Möglichkeit gibt, `DELETE`-Anweisungen unter Verwendung einer `FROM`-Klausel zu schreiben, die mehrere Tabellen enthält. Durch diese Verfahrensweise bekommt die `DELETE`-Anweisung das Aussehen einer Verknüpfungsoperation, obschon nur die Zeilen einer einzigen Tabelle gelöscht werden. Die damit mögliche Funktionalität entspricht derjenigen einer Unterabfrage. Die zweite Tabelle wird nur als Suchtabelle eingesetzt.

Das folgende Beispiel zeigt, wie die `DELETE`-Anweisungen mehrere Tabellen in einer `FROM`-Klausel verwenden:

```
DELETE tmpSales
FROM tmpSales s JOIN tmpTitles t
ON s.title_id = t.title_id
WHERE t.type = 'business'
```



Ob man die Methode mit der Unterabfrage oder mit der Verknüpfung wählt, ist eine rein subjektive Entscheidung. Der Vorteil der Unterabfrage liegt darin, dass keinerlei Verwirrungen bezüglich der zu ändernden Tabelle und der Tabelle, die als Suchtabelle fungiert, auftreten können. Außerdem sollten Sie nicht vergessen, dass die Verknüpfungsmethode nicht dem ANSI-Standard entspricht. Dagegen hat die JOIN-Methode den Vorteil, dass SQL Server Verknüpfungen effizienter als Unterabfragen ausführt.

TRUNCATE TABLE

Am Anfang dieses Abschnitts haben Sie ein Beispiel für eine DELETE-Anweisung ohne WHERE-Klausel kennen gelernt, die alle Zeilen in der Tabelle löschen würde. Falls Sie nun tatsächlich sämtliche Daten in einer Tabelle entfernen, aber die Tabellenstruktur intakt lassen möchten, gibt es noch eine weitere Alternative: Die Anweisung TRUNCATE TABLE.

```
TRUNCATE TABLE tmpSales
```

Anders als die DELETE-Anweisung liefert diese Anweisung keine Meldung über die Anzahl der verarbeiteten Zeilen (*rows affected*) zurück. Außerdem gibt es noch weitere Unterschiede zwischen DELETE ohne WHERE-Klausel und TRUNCATE TABLE:

- DELETE protokolliert jede einzelne Zeile beim Löschen mit, während TRUNCATE TABLE lediglich die aufgehobenen Seiten- und Blockreservierungen in das Transaktionsprotokoll schreibt.
- DELETE verwaltet Indizes, indem es die Zeiger nacheinander löscht und jede Indexanpassung protokolliert. TRUNCATE TABLE dagegen entfernt den Index in einem einzigen Arbeitsschritt und schreibt auch dann nur die aufgehobenen Seitenreservierungen in das Transaktionsprotokoll.
- TRUNCATE TABLE kann nur vom Tabellenbesitzer, einem Mitglied der Datenbankrolle *db_owner* oder der Serverrolle *sysadmin* ausgeführt werden, während DELETE von jedem Benutzer ausgeführt werden kann, der die entsprechende Berechtigung für die Tabelle bekommen hat.
- Ein DELETE-Trigger wird nicht beim Abschneiden einer Tabelle, aber beim Löschen von Zeilen ausgelöst (siehe Tag 15 für weitere Informationen über Trigger).
- TRUNCATE TABLE setzt alle Identitätswerte auf den Ausgangswert zurück, während DELETE keinen Einfluss auf den nächsten zu verwendenden Identitätswert hat.

11.4 Aktualisieren mit UPDATE

Die dritte Anweisung zum Bearbeiten von Daten ist die `UPDATE`-Anweisung, die Ihnen die Änderung der Spaltenwerte in einer Tabellenzeile ermöglicht. Bevor Sie die weiteren Beispiele ausprobieren, sollten Sie sich zunächst einmal um einen »frischen« Datenbestand kümmern. Starten Sie dazu die folgenden Anweisungen, um die Tabellen *tmpSales*, *tmpPublishers*, *tmpStores* und *tmpTitles* neu aufzubauen:



```
USE pubs
GO
DROP TABLE tmpSales
DROP TABLE tmpPublishers
DROP TABLE tmpStores
DROP TABLE tmpTitles
GO
SELECT * INTO tmpPublishers FROM Publishers
SELECT * INTO tmpSales FROM Sales
SELECT * INTO tmpStores FROM sales
SELECT * INTO tmpTitles FROM titles
GO
```

UPDATE

Die Syntax für die `UPDATE`-Anweisung lautet:



```
UPDATE
    {Tabellenname | Sichtname}
SET
    Spaltenname1 = {Ausdruck1 | NULL | (Select_Anweisung)}
    [, Spaltenname2 = ...]
[WHERE Suchkriterien]
```

Die `SET`-Klausel spezifiziert die Spalten, die zu aktualisieren sind. Wie schon bei der `DELETE`-Anweisung ist auch hier die `WHERE`-Klausel optional.

Die folgende `UPDATE`-Anweisung ändert die Spalte *ytd_sales* (aktuelle Jahresverkäufe) der Tabelle *tmpTitles* in allen Zeilen auf den Wert 0; ein Beispiel für eine Aktion, die Sie vermutlich nach dem Jahresabschluss durchführen werden.



```
update tmpTitles
set ytd_sales = 0
```



(18 row(s) affected)

Ohne WHERE-Klausel ändert diese Anweisung den Wert der Spalte *ytd_sales* in allen Zeilen der Tabelle auf 0.

Im folgenden Beispiel wird die Spalte *city* für den Verleger Algodata Infosystems aktualisiert:



```
UPDATE tmpPublishers  
SET city = 'El Cerrito'  
WHERE pub_name = 'Algodata Infosystems'
```



(1 row(s) affected)

Eine UPDATE-Anweisung ist auch in der Lage, den neuen Wert einer Spalte abhängig vom ursprünglichen Wert zu ändern. Das folgende Beispiel verringert den Preis aller Psychologiebücher um 10% des ursprünglichen Preises:



```
UPDATE tmpTitles  
SET price = price * 0.90  
WHERE type = 'psychology'
```



(5 row(s) affected)

Mit einer UPDATE-Anweisung können Sie auch mehrere Spalten gleichzeitig aktualisieren, indem Sie nach der einmaligen Nennung des Schlüsselwortes SET einfach nur durch Komma getrennt die Spalten aufzählen, die aktualisiert werden sollen. Im folgenden Beispiel wird der Preis aller populären Computerbücher um 20% erhöht und dem Feld *notes* in jeder Zeile die Zeichenfolge (price increase) angehängt.



```
UPDATE tmpTitles  
SET price = price * 1.2, notes = notes + ' (price increase)'  
WHERE type = 'popular_comp'
```



(3 row(s) affected)

UPDATE über eine Suchtabelle

Mit einer einzelnen UPDATE-Anweisung kann immer nur eine einzige Tabelle aktualisiert werden. SQL Server erlaubt es jedoch, in die UPDATE-Anweisung eine weitere Tabelle einzubinden, die als Suchtabelle fungiert. Eine Suchtabelle erscheint normalerweise nur in einer Unterabfrage. Die Unterabfrage wiederum kann entweder in der WHERE- oder der SET-Klausel der UPDATE-Anweisung erscheinen. Im folgenden Beispiel werden Sie den Verlegernamen sämtlicher Wirtschaftsbücher auf New Moon Books abändern.



```
UPDATE tmpTitles
SET pub_id =
    (SELECT pub_id FROM tmpPublishers
     WHERE pub_name = 'New Moon Books')
WHERE type = 'business'
```



(4 row(s) affected)

Der Verlegername erscheint nur in der Tabelle *tmpPublishers*, es ist aber die Tabelle *tmpTitles* zu aktualisieren. Die Unterabfrage greift dazu auf die Tabelle *tmpPublishers* zu und liefert die Verleger-ID (*pub_id*) für New Moon Books zurück. Dieser Wert wird dann als der neue Wert in der Spalte *pub_id* der Tabelle *tmpTitles* verwendet.

Wie auch bei der DELETE-Anweisung verfügt Transact-SQL über eine Erweiterung, die es erlaubt, in UPDATE-Anweisungen eine FROM-Klausel einzusetzen, die mehrere Tabellen enthalten kann. Durch diese Vorgehensweise scheint das UPDATE wieder zu einer Verknüpfungsoperation zu werden, obschon nur die Zeilen in einer Tabelle geändert werden. Die zur Verfügung gestellte Funktionalität ist dieselbe wie in Unterabfragen, wobei die zweite Tabelle lediglich als Suchtabelle verwendet wird.

Das folgende Beispiel zeigt, wie die obenstehende UPDATE-Anweisung so umgeschrieben werden kann, dass nun über eine FROM-Klausel mehrere Tabellen angesprochen werden:

```
UPDATE tmpTitles
SET tmpTitles.pub_id = p.pub_id
FROM tmpTitles t JOIN tmpPublishers p
ON t.pub_id = p.pub_id
WHERE p.pub_name = 'New Moon Books'
AND type = 'business'
```

Die Wahl ob Unterabfrage oder Verknüpfung hängt wieder ganz von den persönlichen Präferenzen ab. Wie auch schon bei der DELETE-Anweisung führt die Methode mit der Unterabfrage generell zu einem übersichtlicheren Eindruck hinsichtlich der zu ändernden Tabellen, des neuen Wertes von *pub_id* und der Zeilen, die tatsächlich aktualisiert werden.

Vergessen Sie auch hier nicht, dass die Verknüpfungsmethode nicht dem ANSI-Standard entspricht.

Es gibt darüber hinaus auch noch einige UPDATE-Anweisungen, die bei Verwendung der Verknüpfungsmethode weitaus anspruchsvoller in der Formulierung sind. Zu diesen gehören UPDATE-Anweisungen, in denen Unterabfragen sowohl in der SET- als auch in der WHERE-Klausel eingesetzt werden.

Das nächste Beispiel ändert den Verleger sämtlicher Psychologiebücher, die von New Moon Books publiziert worden sind, auf den Namen Binnet & Hardley.



```
UPDATE tmpTitles
SET pub_id =
    (SELECT pub_id FROM tmpPublishers
     WHERE pub_name = 'Binnet & Hardley')
WHERE type = 'psychology' AND pub_id =
    (SELECT pub_id FROM tmpPublishers
     WHERE pub_name = 'New Moon Books')
```



(4 row(s) affected)



Noch einmal sei darauf hingewiesen, dass der Verlegername nur in der Tabelle *tmpPublishers* erscheint, während es die Tabelle *tmpTitles* ist, die eigentlich aktualisiert werden soll. Die erste Unterabfrage greift auf die Tabelle *tmpPublishers* zu und liefert die Verleger-ID (*pub_id*) für Binnet & Hardley zurück. Der Wert dieser *pub_id* wird sodann als neuer Wert für die Spalte *pub_id* in der Tabelle *tmpTitles* verwendet. Die zweite Unterabfrage greift noch einmal auf die Tabelle *tmpPublishers* zu, um die *pub_id* für New Moon Books zu ermitteln, deren Wert *pub_id* dann dazu verwendet wird, die Zeilen festzulegen, die in der Tabelle *tmpTitles* aktualisiert werden müssen.

Diese UPDATE-Anweisung wäre übrigens bei Verwendung der Verknüpfungsmethode weitaus schwieriger zu formulieren, weil die Tabelle *tmpPublishers* zweimal erscheinen müsste – einmal zum Festlegen des neuen Wertes von *pub_id* und ein zweites Mal zum Festlegen der *pub_id* derjenigen Zeilen, die aktualisiert werden müssten. Dies ist ein weiterer Grund dafür, warum die Methode über eine Unterabfrage gegenüber der Verknüpfungsmethode vorzuziehen ist.



11.5 Zusammenfassung

In der heutigen Lektion haben Sie SQL-Anweisungen kennen gelernt, mit denen Sie die Daten in Ihren SQL Server-Tabellen ändern können. Sie sind nun in der Lage, über die INSERT-Anweisung neue Zeilen in eine Tabelle einzufügen – entweder Zeile für Zeile oder als Zeilenbereich, der aus einer anderen Tabelle stammt. Außerdem können Sie mit der DELETE-Anweisung Zeilen aus einer Tabelle löschen und Werte mit der UPDATE-Anweisung in vorhandenen Zeilen ändern.

Neben den einfachen Anweisungen zum Einfügen, Aktualisieren und Löschen – INSERT, UPDATE und DELETE – hat diese Lektion auch die Anweisungen INSERT VALUES, INSERT INTO und TRUNCATE TABLE sowie den Einsatz von Unterabfragen, Verknüpfungen und Suchtabellen behandelt. Es wurde auch erläutert, wie sich die einzelnen Operationen im Transaktionsprotokoll niederschlagen.

11.6 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F *Wie kann man eine DELETE- oder UPDATE-Operation nach der Ausführung wieder zurücknehmen?*

A Vollzogene Änderungen gelten standardmäßig als übertragen und damit auch als permanent – ein UNDO gibt es in SQL Server leider nicht. Allerdings haben Sie die Möglichkeit, eine DELETE- oder UPDATE-Anweisung innerhalb einer Transaktion auszuführen, wobei die Transaktion dann später als Ganzes wieder zurückgesetzt werden kann. Transaktionen sind Thema von Tag 12.

F *Was bewirkt die folgende Anweisung?*

```
INSERT INTO Publishers DEFAULT VALUES?
```

A Mit dieser Anweisung wird eine Zeile in die Tabelle publishers eingefügt, wobei für die Spalten die Standardwerte verwendet werden, sofern solche vorhanden sind. Sind sie nicht vorhanden, fügt SQL Server NULL-Werte ein.

Kontrollfragen

1. Was geschieht, wenn Sie eine DELETE-Anweisung ohne WHERE-Klausel verwenden?
2. Richtig oder falsch: Beim Einfügen einer neuen Zeile müssen Sie für jede Spalte einen Wert bereitstellen.
3. Welche Möglichkeiten bieten Verknüpfungen in einer DELETE- oder UPDATE-Anweisung?

Übungen

Die Übungen setzen voraus, dass sich Ihre Tabelle *tmpTitles* der Datenbank *pubs* im Originalzustand befindet. Falls Sie die Tabelle zum Üben der Anweisungen in dieser Lektion verändert haben, müssen Sie die Datenbank *pubs* neu aufbauen. Führen Sie dazu das Skript *instpubs.sql* aus, das Sie im Verzeichnis *...\\MSSQL\\Install* finden. Stellen Sie außerdem die Tabelle *tmpTitles* mit folgenden Anweisungen wieder her:

```
USE pubs
GO
SELECT * INTO tmpTitles
FROM titles
GO
```

1. Erstellen Sie eine temporäre Tabelle, die die Titel-ID (*title ID*), den Titel (*title*), die Verleger-ID (*publisher ID*) und den Preis (*price*) sämtlicher moderner Kochbücher (*type* = 'mod_cook') enthält. Legen Sie diese Tabelle den folgenden Übungen zugrunde.
2. Fügen Sie die Titel-ID (*title ID*), den Titel (*title*), die Verleger-ID (*publisher ID*) und den Preis (*price*) sämtlicher traditioneller Kochbücher (*type* = 'trad_cook') in die temporäre Tabelle ein.
3. Setzen Sie die Preise sämtlicher Bücher um 20 Prozent herauf.
4. Verringern Sie den Preis sämtlicher von Binnet & Hardley herausgegebener Bücher um 10 Prozent.
5. Löschen Sie alle Bücher mit einem Preis kleiner als \$10.
6. Löschen Sie alle Bücher mit aktuellen Jahresverkäufen (*year-to-date sales*) größer als 10.000.



Transact-SQL- Erweiterungen

In der gestrigen Lektion haben Sie gelernt, wie Sie mit den Anweisungen INSERT, UPDATE und DELETE Daten modifizieren können. Diese Anweisungen sollte man nicht leichtfertig einsetzen, was insbesondere für die Anweisung zum Löschen – DELETE – gilt. Mit gespeicherten Prozeduren kann man sicherstellen, dass die richtigen Dinge gelöscht werden und dabei auch die Datenintegrität gewahrt bleibt. Dem Thema gespeicherte Prozeduren ist Tag 15 gewidmet.

Die heutige Lektion zeigt zunächst, wie Sie mit der Sprache Transact-SQL programmieren können. Als Erstes machen Sie sich mit der Stapelverarbeitung und mit Skripts vertraut. Dann untersuchen Sie Transaktionen in SQL Server. Ein weiterer Schwerpunkt sind die Anweisungen, mit denen sich der Programmfluss steuern lässt, beispielsweise IF...ELSE und WHILE. Zum Abschluss untersuchen Sie die verschiedenen Arten von Sperren, mit denen SQL Server die gleichzeitige Verwendung von Datenbanken steuert.

12.1 Stapel



Eine *Stapel* oder *Batch* ist ein Satz von Transact-SQL-Anweisungen, die SQL Server als Einheit interpretiert. Die Anweisungen werden zusammen vorgelegt und das Schlüsselwort GO markiert das Ende der Stapelverarbeitung. Hier ein Beispiel für eine Stapelverarbeitung im SQL Server Query Analyzer:

```
USE PUBS
SELECT au_id, au_lname FROM authors
SELECT pub_id, pub_name FROM publishers
INSERT publishers VALUES
    ('9998', 'SAMS Publishing', 'Seattle', 'WA', 'USA')
GO
```

Stapel folgen mehreren Regeln:

- Alle SQL-Anweisungen werden als Gruppe kompiliert.
- Wenn an einer beliebigen Stelle des Stapels ein Syntaxfehler auftritt, wird der gesamte Stapel abgebrochen.
- Manche Anweisungen lassen sich in einem Stapel kombinieren, während andere eingeschränkt sind. Die folgenden CREATE-Anweisungen kann man innerhalb ein und desselben Stapels binden:

```
CREATE DATABASE
CREATE TABLE
CREATE INDEX
```

- Die folgenden Anweisungen lassen sich nicht mit anderen im selben Stapel kombinieren:

```
CREATE RULE
CREATE TRIGGER
CREATE PROCEDURE
CREATE DEFAULT
CREATE VIEW
```

- Wenn Sie versuchen, diese Anweisungen zu kombinieren, erhalten Sie eine Fehlermeldung wie die folgende:

```
Server: Nachr.-Nr. 111, Schweregrad 15, Status 1, Zeile 2
'CREATE VIEW' muss die erste Anweisung in einem Abfragebatch sein.
```

- Wenn Sie die Tabelle ändern, können Sie die neuen Spalten nicht im selben Stapel verwenden.
- SET-Anweisungen wirken sich sofort aus – ausgenommen die Optionen QUOTED_IDENTIFIER und ANSI_NULLS.

12.2 Skripts



Ein *Skript* ist ein als Gruppe gespeicherter Satz von einem oder mehreren Stapeln. Normalerweise werden Skripts als eine zu realisierende Arbeitseinheit – wie zum Beispiel das Laden von Daten oder die Datenbankwartung – ausgeführt. Listing 12.1 zeigt ein Beispiel für ein Skript.

Listing 12.1: Beispiel für ein Skript

```
USE PUBS
SELECT au_id, au_lname FROM authors
SELECT pub_id, pub_name FROM publishers
INSERT publishers VALUES
    ('9997','SAMS Publishing', 'Seattle', 'WA','USA')
GO

SELECT * FROM stores
GO

DELETE publishers WHERE pub_id = '9997'
GO
```

Stapel und Skripts haben nicht unbedingt etwas mit Transaktionen zu tun, worauf diese Lektion in Kürze eingeht. Zum Lieferumfang von SQL Server gehören zahlreiche Skripts, die Sie im Ordner `...\\MSSQL\\Install` finden; die betreffenden Dateien weisen die Erweiterung `.sql` auf. Diese Skripts sind hervorragende Beispiele, die Ihnen als Orientierung beim Erstellen eigener Transact-SQL-Skripts dienen können.

12.3 Transaktionen



Eine *Transaktion* ist eine Arbeitseinheit, in der entweder alle Anweisungen erfolgreich verarbeitet werden oder überhaupt keine Anweisung verarbeitet wird. Mit Transaktionen haben Sie ständig zu tun, auch wenn Sie es gar nicht bemerken. Was wäre beispielsweise, wenn ein Bankkassierer 50 DM von einem Girokonto auf ein Sparkonto transferieren sollte, aber vergisst, das Geld auf dem Sparkonto gutzuschreiben? Der Bankkunde wäre darüber sicherlich verärgert. Man erwartet einfach, dass das vom Konto abgebuchte Geld auf dem Sparkonto erscheint. Das ist eine Transaktion. Die Arbeitseinheit beendet alle Kommandos erfolgreich oder sie schlägt fehl und macht alle Aktionen rückgängig. Als SQL Server-Entwickler müssen Sie Transaktionen so programmieren, dass sie korrekt arbeiten. Aber auch als Administrator müssen Sie Transaktionen verstehen, damit sie richtig verwendet werden und sich nicht negativ auf das Transaktionsprotokoll auswirken.

Transaktionen müssen die folgenden vier Eigenschaften aufweisen, die man mit dem Akronym ACID für Atomicity (Unteilbarkeit), Consistency (Konsistenz), Isolation und Durability (Beständigkeit) beschreibt:

- **Unteilbarkeit:** Eine Transaktion läuft entweder in ihrer Gesamtheit vollständig ab oder wird insgesamt abgebrochen. Wenn irgendeine Anweisung fehlschlägt, scheitern alle Anweisungen, die Teil der Transaktion sind.
- **Konsistenz:** Eine Transaktion hinterlässt die Datenbank in einem konsistenten Zustand, nachdem sie erfolgreich abgeschlossen oder fehlgeschlagen ist. Die durch die Transaktion bewirkten Änderungen sind von einem Zustand zum anderen konsistent.
- **Isolation:** Eine Transaktion gilt als isoliert, wenn sie nicht mit irgendeiner anderen Transaktion in der Datenbank in Wechselwirkung oder in Konflikt steht.
- **Beständigkeit:** Eine Transaktion gilt dann als beständig, wenn die Arbeitseinheit dauerhaft in das System übernommen wird und dort vollständig bleibt, ungeachtet dessen, was mit der Datenbank nach Abschluss der Transaktion passiert. Wenn der Strom ausfällt und der Datenbankserver abstürzt, wird garantiert, dass die Transaktion nach wie vor vollständig ist, sobald der Server wieder hochfährt.



Transaktionen garantieren, dass die auszuführende Arbeit vollständig erfolgreich oder vollständig gescheitert sein kann, wie in der vorhergehenden Liste beschrieben. *Sperren* bieten einen Teil dieser Garantie. Während einer Transaktion kann keine andere Transaktion Daten bearbeiten, die Ihre Transaktion geändert hat, bis Sie darüber entschieden haben, ob die Änderung permanent ist. Während Sie die Daten bearbeiten, belegen Sie diese mit einer *exklusiven Sperre*. Umgekehrt können Sie die Daten einer anderen Transaktion nicht lesen, wenn diese gerade bearbeitet werden. Für diese anderen Daten fordern Sie eine *gemeinsame Sperre* an, aber die andere Transaktion verwendet eine exklusive Sperre für ihre Daten, die Sie am Lesen hindert. Auf Sperren geht diese Lektion später ein.

Transaktionstypen

Es gibt drei Typen von Transaktionen: explizit, implizit und automatisch (in der Online-Dokumentation von Microsoft als »autocommit« bezeichnet).

Explizite Transaktionen



Explizite Transaktionen sind Transaktionen, die Sie manuell konfigurieren. Den Anfang und das Ende einer expliziten Transaktion kennzeichnen Sie explizit mit reservierten Wörtern. Dazu gehören: `BEGIN TRANSACTION`, `COMMIT TRANSACTION`, `COMMIT WORK`, `ROLLBACK TRANSACTION`, `ROLLBACK WORK` und `SAVE TRANSACTION`.

Um mit einer expliziten Transaktion zu beginnen, tippen Sie die Schlüsselwörter `BEGIN TRAN` (oder `BEGIN TRANSACTION`, falls Sie in Tipplaune sind) ein. Um SQL Server zu zeigen, dass Ihre Transaktion vollständig ist und die gesamte Arbeit gespeichert werden soll, geben Sie die Anweisung `COMMIT TRAN` (oder `COMMIT WORK`) ein. Eine typische Transaktion sieht etwa so aus:

```
BEGIN TRAN
```

```
UPDATE authors
SET city = 'San Jose' WHERE au_lname = 'Smith'
INSERT titles
VALUES ('BU1122','SQL Server 2000 in 21 Tagen',
      'business','9998',$35.00, $1000.00,10,4501, 'Ein tolles Buch!',
      '8/1/1998')
SELECT * from titleauthor
```

```
COMMIT TRAN
```

Transaktionen muss man auch abbrechen können. Dazu verwenden Sie die Anweisung `ROLLBACK TRAN` (oder `ROLLBACK WORK`). Hier ist ein Beispiel für die Anweisung `ROLLBACK TRAN`:

```
BEGIN TRAN
    Delete sales where title_id = 'BU1032'
    IF @@ERROR > 0
        ROLLBACK TRAN
    ELSE
        COMMIT TRAN
```



Die Anweisung `ROLLBACK TRAN` bricht die Transaktion vollständig ab. Jegliche Arbeit, die in der Transaktion bis zu diesem Punkt erfolgt ist, wird wieder rückgängig gemacht (zurückgerollt) oder abgebrochen. Sie haben auch die Möglichkeit, Sicherungspunkte innerhalb einer Transaktion zu erstellen und dann selektiv bis zu diesen Punkten zurückzurollen. Wieder illustriert dies am besten ein Codebeispiel:

```
BEGIN TRAN
    UPDATE table1 SET col1 = 5 WHERE col2 = 14
    SAVE TRAN savepoint1
    INSERT table2 values (3,16)
    IF @@error > 0
        ROLLBACK TRAN savepoint1
        DELETE table3 WHERE col1 > 2
    IF @@error > 0
        ROLLBACK TRAN
    ELSE
        COMMIT TRAN
```

Nach der Anweisung `SAVE TRAN` steht der so genannte *Sicherungspunktname*. Indem Sie den Sicherungspunktnamen in den ersten Rollback einschließen, bringen Sie zum Ausdruck, dass Sie die Transaktion bei einem Scheitern nicht bis zum Anfang rückgängig machen möchten, sondern nur bis zu einem benannten Punkt – im Beispiel `savepoint1`. Das `INSERT` in `table2` würde abgebrochen, falls der erste Rollback ausgelöst wird, die Transaktion selbst würde jedoch fortgesetzt. Praktisch würde das `INSERT` aus der Transaktion »entfernt«. Weil die später folgende `ROLLBACK TRAN`-Anweisung nicht mit einem Sicherungspunkt benannt ist, wird dafür ein Rollback bis zurück zur Anweisung `BEGIN TRAN` durchgeführt.

Sehen Sie sich nun die einzelnen `TRANSACTION`-Anweisungen näher an, beginnend mit `BEGIN` und `COMMIT`.



```
BEGIN TRANsaction [transaction_name]
COMMIT TRANsaction [transaction_name]
COMMIT [WORK]
```

Der Transaktion können Sie den optionalen Namen *transaction_name* zuweisen. Sie können eine Variable einsetzen, wo immer sie *transaction_name* sehen, solange sie sich zu einem gültigen Transaktionsnamen auflösen lässt. Es ist nicht notwendig, eine Transaktion zu benennen, eigentlich ist es besser, dass Sie eine Transaktion nicht benennen, außer der ersten `BEGIN`-Anweisung in Ihrer Transaktion. Der *transaction_name* muss ein gültiger SQL Server-Bezeichner sein. Achten Sie darauf, dass Sie bei der `COMMIT WORK`-Anweisung keinen Transaktionsnamen angeben dürfen.



Transaktionen lassen sich zwar verschachteln, doch handelt es sich dabei um eine rein syntaktische Angelegenheit. Man kann Transaktionen nicht wirklich verschachteln. In einem Skript können zwar mehrere Transaktionen auftreten, wovon tatsächlich aber nur eine wirkliche Transaktion verwendet wird.

Die Syntax für die Anweisung `SAVE TRANSACTION` lautet:



```
SAVE TRANsaction [savepoint_name]
```

savepoint_name kennzeichnet einen Sicherungspunkt, zu dem eine Transaktion zurückkehren kann, wenn ein Teil der Transaktion bedingt abgebrochen wird. Der *savepoint_name* muss ein gültiger SQL Server-Bezeichner sein.

Die Syntax für die Anweisung `ROLLBACK TRANSACTION` lautet:



```
ROLLBACK TRANsaction [transaction_name | savepoint_name]  
ROLLBACK [WORK]
```

Sehen Sie sich die obigen Erläuterungen zu den Anweisungen `BEGIN/COMMIT TRAN` und `SAVE TRAN` hinsichtlich der optionalen Namen an. Beachten Sie wieder, dass es die Anweisung `ROLLBACK WORK` nicht gestattet, einen Transaktionsnamen oder einen Sicherungspunktnamen anzugeben.

Die globale Variable @@trancount

Eine globale Variable, `@@trancount`, betrifft direkt diese Transaktionen. Wenn Sie eine `BEGIN TRAN`-Anweisung ausführen, wird `@@trancount` um eins erhöht. Ein `SAVE TRAN` hat keine Wirkung auf `@@trancount`. Ein `ROLLBACK TRAN` kann mehrere Auswirkungen haben, je nachdem, ob ein Transaktionsname angegeben ist. Enthält die Anweisung keinen Transaktionsnamen, wird `@@trancount` auf 0 zurückgesetzt (die gesamte Arbeit wird rückgängig gemacht). Ist der letzte Transaktionsname angegeben, wird `@@trancount` um eins vermindert. Eine `ROLLBACK WORK`-Anweisung setzt `@@trancount` immer auf 0 zurück und bricht auch alle Arbeiten ab, die vom ersten `BEGIN TRAN` an ausgeführt wurden. Studieren Sie dazu die Arbeitsweise des folgenden Codes:

```

SELECT @@TRANCOUNT -- sollte 0 zurückgeben.
BEGIN TRAN t1
SELECT @@TRANCOUNT -- sollte 1 zurückgeben.
SAVE TRAN savepoint1
SELECT @@TRANCOUNT -- Ist immer noch auf 1 gesetzt.
ROLLBACK TRAN savepoint1
SELECT @@TRANCOUNT -- Ist immer noch auf 1 gesetzt.
BEGIN TRAN t2
SELECT @@TRANCOUNT -- Sollte 2 zurückgeben.
ROLLBACK TRAN
SELECT @@TRANCOUNT -- Ist wieder zurück auf 0.

```

Zulässige und nicht zulässige Anweisungen

Einige Anweisungen sind als Teil expliziter Transaktionen nicht zulässig. Dazu gehören folgende:

```

ALTER DATABASE
DROP DATABASE
RECONFIGURE
BACKUP LOG
RESTORE DATABASE
CREATE DATABASE
RESTORE LOG
UPDATE STATISTICS

```

Andere Anweisungen können zusammen innerhalb einer Transaktion erscheinen, was aber keine der Regeln berührt, die Sie zuvor gelernt haben.

Sicherungspunkte

Es gibt keinen veröffentlichten Grenzwert der Zahl von Sicherungspunkten innerhalb einer Transaktion. Sobald eine Transaktion bestätigt ist, lässt sie sich nicht mehr rückgängig machen.

Wenn Sie doppelte Sicherungspunktnamen in derselben Transaktion angeben, wird nur die letzte Instanz des Sicherungspunktnamens tatsächlich verwendet, wenn Sie einen Rollback bis zu diesem Sicherungspunkt durchführen. Im folgenden Beispiel kehrt die Anweisung `ROLLBACK TRAN transave1` nur bis zum zweiten Sicherungspunkt `transave1` zurück. Der erste Sicherungspunkt wird ignoriert, nachdem der Name erneut verwendet wird.

```

BEGIN TRAN
    INSERT
    UPDATE
    SAVE TRAN transave1
    DELETE
    INSERT
    SELECT

```

```

SAVE TRAN transave1
INSERT
DELETE
IF @@ERROR <> 0
    ROLLBACK TRAN transave1
ELSE
    COMMIT TRAN

```



Wenn Sie eine `ROLLBACK TRAN`-Anweisung innerhalb eines Triggers auslösen und kein Sicherungspunkt angegeben ist, wird die gesamte Transaktion rückgängig gemacht und der Rest des Stapels nicht ausgeführt. Die Verarbeitung kann jedoch mit dem nächsten Stapel fortfahren. In Ihrem Code müssen Sie diesen Sachverhalt prüfen und den Rollback in angemessener Form behandeln. Möglicherweise ist es am besten, der aufrufenden Routine aus dem Trigger heraus einen Fehlercode zu übergeben und es der aufrufenden Routine zu überlassen, den Zustand der Transaktion passend zu behandeln. Diese Lösung ist wesentlich sicherer.

Remote Prozeduraufrufe

Aufrufe an remote gespeicherte Prozeduren werden normalerweise nicht als Teil einer Transaktion angesehen:

```

BEGIN TRANSACTION
UPDATE table1 SET col1 = 5 WHERE col1 = 1
DELETE table1 WHERE col1 = 5
EXEC server2.pubs..usp_insertpublisher parm1 parm2 parm3
COMMIT TRAN

```

Wenn die auf *Server2* gespeicherte Prozedur `pubs..usp_insertpublisher` eine `ROLLBACK TRAN`-Anweisung auslöst, hat das keinen Einfluss auf die lokale Transaktion.

Es gibt jedoch eine Serverkonfigurationsoption namens `REMOTE_PROC_TRANSACTIONS`. Dieser Konfigurationsparameter, der normalerweise auf 0 (aus) gesetzt ist, steuert, ob remote gespeicherte Prozeduren automatisch verteilt werden. Um die automatische Verteilung von remote gespeicherten Prozeduren in Transaktionen zu aktivieren, führen Sie die folgenden Anweisungen aus:



```

EXEC sp_configure "REMOTE_PROC_TRANSACTIONS", 1
RECONFIGURE WITH OVERRIDE

```

Dies beeinflusst alle Aufrufe von remote gespeicherten Prozeduren – setzen Sie diese Option also mit Bedacht ein. Auf verteilte Transaktionen geht diese Lektion später näher ein.

Automatische Transaktionen

Auch wenn scheinbar keine Transaktion läuft – hinter den Kulissen sind Transaktionen immer am Werk. Jede Ausführung einer Datenänderungsanweisung in SQL Server ist eine *implizite Transaktion*.

Im folgenden Stapel ist jede SQL-Anweisung eine separate Transaktion. Somit umfasst dieser Stapel eigentlich drei separate Transaktionen. Wenn eine der Anweisungen fehlschlägt, beeinträchtigt sie die anderen nicht. Jede Anweisung ist eigenständig – sie kann unabhängig von den anderen Anweisungen im Stapel erfolgreich verlaufen oder scheitern.

```
INSERT table1 VALUES (1,'abcde')
UPDATE table1 SET col1 = 5 WHERE col1 = 1
DELETE FROM table1 WHERE col1 = 5
GO
```

Transaktionen können auch Leistungsvorteile bringen. Da die Transaktionsprotokolle erst auf die Platte geschrieben werden, wenn die Transaktion vollendet ist, kann SQL Server für eine effektivere Ausnutzung der Platte sorgen. Es ist daher günstig, Anweisungen zu gruppieren. Wenn Sie die vorhergehenden Anweisungen zur folgenden Transaktion gruppieren, werden nur fünf statt neun Protokolleinträge geschrieben:

```
BEGIN TRAN
INSERT table1 VALUES (1,'abcde')
UPDATE table1 SET col1 = 5 WHERE col1 = 1
DELETE FROM table1 WHERE col1 = 5
COMMIT TRAN
```

Wenn Sie die Gruppierung von Anweisungen steigern, können Transaktionen die Effizienz Ihrer Datenbearbeitungsanweisungen dramatisch erhöhen.

Implizite Transaktionen

Implizite Transaktionen werden bereitgestellt, um mit dem American National Standards Institute (ANSI) konform zu sein. Wenn implizite Transaktionen aktiviert sind, lösen ausgewählte Transact-SQL-Anweisungen automatisch ein `BEGIN TRANSACTION` aus. Diese Anweisungen müssen Sie mit `COMMIT` oder `ROLLBACK TRAN` explizit bestätigen oder rückgängig machen.

Implizite Transaktionen aktivieren Sie auf Sitzungsebene mit der folgenden Anweisung:

```
SET IMPLICIT_TRANSACTIONS ON
```

Während der restlichen Sitzung müssen die folgenden Anweisungen explizit bestätigt (oder rückgängig gemacht) werden:

ALTER TABLE	GRANT	REVOKE
CREATE	DROP	DELETE
SELECT	INSERT	UPDATE
TRUNCATE TABLE	FETCH	OPEN

Implizite Transaktionen deaktivieren Sie mit:

```
SET IMPLICIT_TRANSACTIONS OFF
```

Beispielsweise sind die folgenden beiden Codefragmente (hinsichtlich der Transaktionen) identisch. Beachten Sie, dass Sie dies mit der Variablen @@trancount verifizieren können.

```
CREATE TABLE table1 (col1 int NOT null)
BEGIN TRAN
INSERT table1 VALUES (1)
SELECT @@trancount
COMMIT TRAN
```

und

```
SET IMPLICIT_TRANSACTIONS ON
INSERT table1 values (2)
SELECT @@trancount
COMMIT TRAN
```



Wenn Sie die Option IMPLICIT_TRANSACTIONS aktivieren, müssen Sie daran denken, jede Transaktion zu bestätigen oder rückgängig zu machen. Man vergisst das leicht – länger als gewünscht bleiben dann Transaktionen geöffnet und Sperren werden aufrecht erhalten. Diese Option sollten Sie also nur dann aktivieren, wenn Sie Ihre Transaktionen tatsächlich immer bestätigen.

Wie arbeiten Transaktionen?

Nachdem Sie die verschiedenen Transaktionsarten – explizite, implizite und automatische – kennen, soll dieser Abschnitt anhand der Beispielanweisungen von Listing 12.2 klären, wie Transaktionen in SQL Server arbeiten.

Listing 12.2: Schrittweise durch einen Satz von Transaktionen gehen

```
BEGIN TRAN
  INSERT table1 values (1,'abcde')
  UPDATE table1 SET col1 = 5 WHERE col1 = 1
  DELETE FROM table1 WHERE col1 = 5
COMMIT TRAN
```



Die Anweisungen in diesem Listing bewirken Folgendes:

1. Wird die Anweisung `BEGIN TRAN` an die Datenbank geschickt, erkennt der SQL Server-Parser die Anforderung, mit einer expliziten Transaktion zu beginnen. SQL Server 2000 reserviert jedoch erst einen Protokolldatensatz im Speicher, wenn tatsächlich eine Arbeitseinheit ausgeführt wurde. Aus technischer Sicht hat die Transaktion also noch nicht begonnen.
2. Die `INSERT`-Anweisung wird ausgeführt. SQL Server erstellt einen Protokolldatensatz im Speicher und reserviert eine Transaktionskennung für diese neue Transaktion. Die neue Zeile wird im Transaktionsprotokoll aufgezeichnet und dann wird die Datenseite für `table1` im Speicher bearbeitet (siehe Abbildung 12.1). Wenn sich die benötigte Seite nicht im Speicher befindet, wird sie von der Festplatte gelesen.
3. Die nächste Anweisung läuft in ähnlicher Weise ab. Zuerst wird die Anweisung `UPDATE` im Transaktionsprotokoll aufgezeichnet und dann die Datenseite im Speicher bearbeitet (siehe Abbildung 12.2).

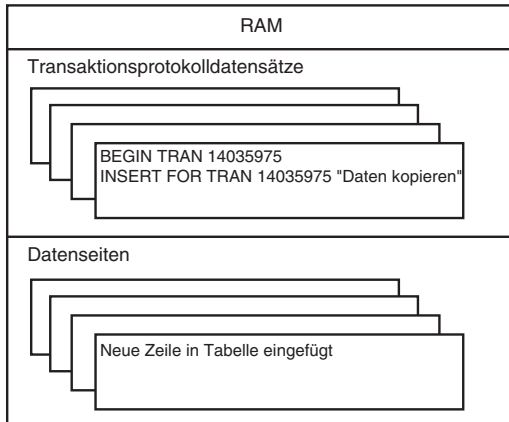


Abbildung 12.1:
Schritt 2 des Transaktionsprozesses



In diesem Beispiel wird für die Aktualisierung lediglich eine einzige Zeile im Transaktionsprotokoll aufgezeichnet. In der Regel finden Sie aber dafür im Transaktionsprotokoll eine Anweisung zum Löschen, gefolgt von einer Anweisung zum Einfügen. Es wäre eine Aktualisierung »an Ort und Stelle« erforderlich, damit nur ein einziger Bearbeitungsdatensatz in das Protokoll geschrieben würde. Zu diesem Thema sei auf die einschlägige Literatur verwiesen.

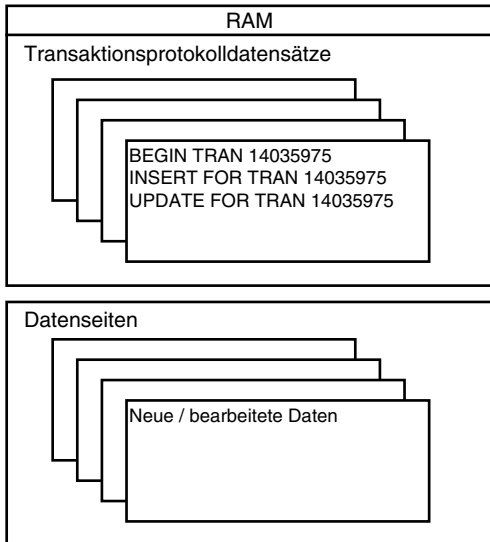


Abbildung 12.2:
Schritt 3 des Transaktionsprozesses

4. Wenn SQL Server die Anweisung `COMMIT TRAN` erhält, wird der Protokolldatensatz in die Transaktionsprotokolldatei für die Datenbank geschrieben (siehe Abbildung 12.3). Dies ist Ihre Garantie, dass die Transaktion wiederhergestellt werden kann. Weil die Protokolländerungen auf die Platte geschrieben werden, garantieren sie, dass die Transaktion wiederherstellbar ist – selbst wenn der Strom ausfällt oder die Datenbank abstürzt, bevor die Datenseite auf die Platte geschrieben wird.

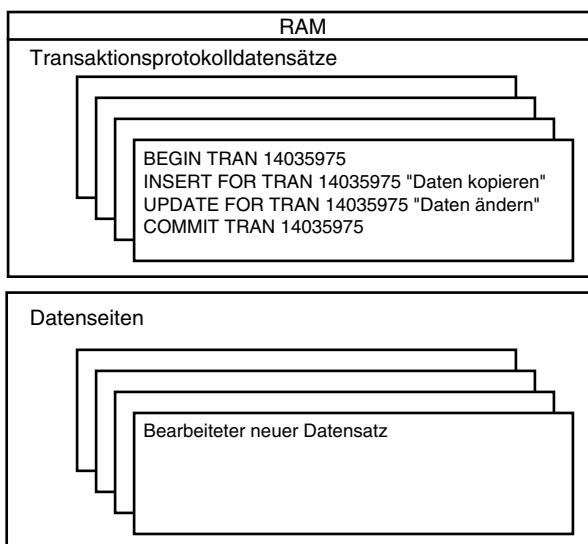


Abbildung 12.3:
Schritt 4 des Transaktionsprozesses

In SQL Server 2000 werden Protokolldatensätze in eine separate Datei (oder einen Satz von Dateien) geschrieben und sind per Transact-SQL nicht zugänglich. Microsoft liefert kein Hilfsprogramm mit, um auf das Transaktionsprotokoll zuzugreifen (abgesehen vom Befehl DBCC LOG, dessen Behandlung in diesem Buch zu weit führen würde). Nur interne SQL Server-Prozesse wie Datensicherung und Wiederherstellung brauchen den Zugriff auf das Transaktionsprotokoll.

Der CHECKPOINT-Prozess



Nach dieser Einführung in das Thema Transaktionen fragen Sie sich möglicherweise, wann die Datenseiten auf die Platte geschrieben werden. Die Protokolldatensätze werden geschrieben, wenn SQL Server die Anweisung COMMIT TRAN ausführt. Und wann kommen die Daten auf die Platte? Die Antwort liegt im CHECKPOINT-Prozess. Dabei handelt es sich um einen internen Vorgang, bei dem SQL Server den Speicher leert und die Datenseiten auf die Festplatte schreibt (kopiert).

Durch Prüfpunkte (Checkpoints) lässt sich vermeiden, dass die Wiederherstellung von bestätigten Transaktionen übermäßig lange dauert. Nachdem ein Prüfpunkt aufgetreten ist, wird ein Protokolleintrag geschrieben, der anzeigt, dass alle bearbeiteten Seiten im Speicher auf die Platte geschrieben worden sind. Damit erhält der SQL Server-Wiederherstellungsprozess einen Punkt im Transaktionsprotokoll, bis zu dem keine Suche nach früheren bestätigten Transaktionen erfolgen muss, um eine vollständige Wiederherstellung zu garantieren.

Es gibt zwei Arten von Prüfpunkten in SQL Server: automatische und manuelle. Der automatische Prüfpunktprozess findet basierend auf internen SQL Server-Berechnungen statt. Das Intervall zwischen automatischen Prüfpunkten legen Sie mit der Serverkonfigurationsoption *RECOVERY INTERVAL* fest. Diese Option gibt die maximale Zeitdauer (in Minuten) an, die für die Wiederherstellung aller Datenbanken im System erforderlich ist. Wenn SQL Server feststellt, dass es so lange oder länger dauern würde, um eine Datenbank wieder herzustellen, löst er den automatischen Prüfpunkt aus. Dabei werden alle bearbeiteten Datenseiten und alle Protokolldatensätze für diese Datenbank auf die Festplatte geschrieben. Der automatische Prüfpunktprozess durchläuft alle 60 Sekunden jede Datenbank und bestimmt, ob für die Datenbank ein Prüfpunkt auszulösen ist. Die Standardeinstellung für die Option *RECOVERY INTERVAL* ist 0 – was bedeutet, dass SQL Server entscheidet, wann ein Prüfpunkt nötig ist.

Ein automatischer Prüfpunkt tritt auch unter zwei anderen Bedingungen auf:

- Wenn Sie die gespeicherte Systemprozedur *sp_dboption* ausführen, um eine Datenbankoption zu ändern.
- Wenn Sie SQL Server herunterfahren. Dazu können Sie den Transact-SQL-Befehl SHUTDOWN ausführen oder den MSSQLServer-Dienst beenden.

Sie können jederzeit einen manuellen Prüfpunkt erzwingen, indem Sie die Transact-SQL-Anweisung `CHECKPOINT` ausführen. Beachten Sie, dass Sie dazu ein Mitglied der Rolle `db_owner` einer Datenbank sein müssen. Wenn Sie einen manuellen Prüfpunkt auslösen, werden alle modifizierten Seiten aus dem Speicher auf die Festplatte geschrieben – genau wie beim automatischen Prüfpunktprozess.



Bei Datenbanken mit aktivierter Option *Truncate log on Checkpoint* schneidet der Prüfpunktprozess die bestätigten Transaktionen regelmäßig (oder wenn das Transaktionsprotokoll gefüllt ist) aus dem Transaktionsprotokoll ab. Ein manueller Prüfpunkt hat keine Wirkung auf das Transaktionsprotokoll (abgesehen vom Kopieren der Transaktionsdatensätze auf die Festplatte) – unabhängig davon, ob die Option *Truncate log on Checkpoint* gesetzt ist oder nicht.

Verteilte Transaktionen

Alle bisher behandelten Transaktionen hatten nur mit einem einzigen Server zu tun. SQL Server 2000 unterstützt auch Transaktionen, die sich über mehrere Server erstrecken. Diese Fähigkeit wird von dem Dienst Microsoft Distributed Transaction Coordinator (MSDTC) unterstützt.

Verteilte Transaktionen lassen sich nach drei verschiedenen Verfahren einsetzen:

- Beim Programmieren mit den DB-Lib-, ODBC- oder OLE DB-APIs. ActiveX-Datenobjekte (ADO) erlauben ebenfalls den Einsatz von verteilten Transaktionen. Diese Optionen gehen aber über den Rahmen dieses Buches hinaus.
- Mit der Transact-SQL-Syntax `BEGIN DISTRIBUTED TRANSACTION`.
- Durch Setzen der Option `SET REMOTE_PROC_TRANSACTIONS`, um verteilte Transaktionen für eine einzelne Sitzung zu aktivieren.

Wir gehen hier auf die beiden letzten Methoden ein. Um den MSDTC-Dienst in Ihrer Transaktion anzumelden und seine Koordinierungsaktivität über mehrere Server zur Verfügung zu haben, müssen Sie lediglich die Anweisung `BEGIN DISTRIBUTED TRANSACTION` genauso wie bisher die Anweisung `BEGIN TRAN` aufrufen. Die vorhergehende Diskussion über Transaktionen hat erwähnt, dass remote gespeicherte Prozeduren außerhalb der aktuellen Transaktion operieren, aus der heraus sie aufgerufen wurden. Bei verteilten Transaktionen sind die remote gespeicherten Prozeduren nun in die Transaktion eingeschlossen. Wenn der Remoteserver scheitert, beeinträchtigt dies auch den oder die aufrufenden Server. Scheitert im folgenden Codebeispiel die von Server2 aufgerufene Transaktion, ist davon auch die Transaktion auf Ihrem Server betroffen:

```
BEGIN DISTRIBUTED TRANSACTION
UPDATE table1 SET col1 = 5 WHERE col1 = 1
DELETE table1 WHERE col1 = 5
EXEC server2.pubs..usp_insertpublisher parm1 parm2 parm3
COMMIT TRAN
```

SQL Server 2000 erlaubt es, Remoteserver als Teil einer beliebigen Transact-SQL-Anweisung abzufragen. Somit ist auch das folgende Codesegment eine gültige verteilte Transaktion in SQL Server 2000:

```
BEGIN DISTRIBUTED TRANSACTION
INSERT remoteserver.pubs..remotetable (col1, col2) VALUES (1,2)
UPDATE localtab SET col2 = 5 WHERE col1 = 1
COMMIT TRAN
```

Die oben aufgeführte dritte Methode ist die Anweisung `SET REMOTE_PROC_TRANSACTIONS`. Wenn Sie diese Option in einer Sitzung festlegen, werden alle während der Anweisung aufgerufenen gespeicherten Prozeduren als Teil der Transaktion angesehen. Sie können diese Option auch mit der gespeicherten Systemprozedur `sp_configure` einstellen – die Option ist dann von diesem Zeitpunkt an für alle Sitzungen wirksam. Bevor Sie diese Option serverweit festlegen, sollten Sie die Auswirkungen auf alle vorhandenen remote gespeicherten Prozeduren untersuchen.

Bei Verbindungen zu Remotedatenbanken sind in SQL Server zwei Typen zu unterscheiden: Remoteserver und Verbindungsserver. Die Option Remoteserver dient lediglich der Abwärtskompatibilität; die Option Verbindungsserver ist in SQL Server 2000 verbessert worden.

Remoteserver

Remoteserver erlauben die Ausführung von gespeicherten Prozeduren in remoten SQL Server-Datenbanken. Je nachdem, ob Sie die Transact-SQL-Anweisung `BEGIN DISTRIBUTED TRANSACTION` oder `SET REMOTE_PROC_TRANSACTION ON` verwendet haben, können remote gespeicherte Prozeduren Teil einer Transaktion sein oder nicht. Damit sich remote gespeicherte Prozeduren ausführen lassen, müssen Sie den Remoteserver wie folgt einrichten:

1. Fügen Sie den Namen des Remoteservers in die Systemtabelle `sys.servers` auf Ihrem SQL Server hinzu. Führen Sie dazu die Anweisung `exec sp_addserver <Remoteserver> aus`.
2. Fügen Sie Ihren Server in die Tabelle `sys.servers` in der remoten SQL Server-Datenbank hinzu. Führen Sie auch hier die Anweisung `exec sp_addserver <IhrServer> aus`.
3. Führen Sie auf Ihrem Server die folgenden Transact-SQL-Anweisungen aus:

```
EXEC sp_addremotelogin <Remoteserver>, sa, sa
EXEC sp_remoteoption <Remoteserver>, sa, sa, trusted, true
```

4. An dem entfernten Server führen Sie den Code aus Schritt 3 aus, jedoch mit dem Verweis auf Ihren Server.

Jetzt können Sie remote gespeicherte Prozeduren ausführen, während sie als *sa* angemeldet sind. Um andere Benutzernamen einzurichten, müssen Sie die obigen gespeicherten Prozeduren erneut ausführen. Zur Referenzierung der remote gespeicherten Prozedur geben Sie den vierteiligen Namen – *servername.dbname.owner.procname* – in der Anweisung *execute* an. Wenn Sie beispielsweise die gespeicherte Prozedur *byroyalty* in der Datenbank *pubs* auf dem Server *gizmo* referenzieren, führen Sie folgenden Code aus:

```
exec gizmo.pubs.dbo.byroyalty 10
```

Diese Lösung ist zwar sehr nützlich, aber nicht so flexibel wie der Einsatz von Verbindungsservern.

Verbindungsserver

Microsoft hat Verbindungsserver in SQL Server 2000 verbessert. Verbindungsserver ermöglichen es, auf Rowsets aus den OLE DB-Datenquellen als Tabellen zu verweisen. Im Unterschied zu den oben erwähnten Remoteservern können Sie mit Verbindungsservern auch Daten von Nicht-SQL Server-Datenquellen abrufen. Auf diese Weise ist der Zugriff auf jede OLE DB-Datenquelle möglich. Folglich können Sie mit Daten von Oracle, Microsoft Access, Microsoft Excel oder jeder anderen OLE DB-Datenquelle arbeiten.

Die Möglichkeiten von Verbindungsservern gehen über das, was bei Remoteservern erlaubt ist, weit hinaus. Sie können immer noch remote gespeicherte Prozeduren ausführen, aber Sie können auch remote Tabellen verknüpfen. Um einen Verbindungsserver einzurichten, verwenden Sie die gespeicherten Systemprozeduren *sp_addlinkedserver* und *sp_addlinkedsrvlogin*:

```
Exec sp_addlinkedserver 'Remoteserver','SQL Server'  
Exec sp_addlinkedsrvlogin 'Remoteserver','TRUE'
```



Sie müssen das Skript *instcat.sql* aus dem SQL Server-2000-Installationsverzeichnis für alle Server mit SQL Server 6.5 ausführen, wenn Sie verteilte Abfragen oder remote gespeicherte Prozeduren mit den verketteten Serveroptionen ausführen wollen.

Der vorhergehende Code fügt den Remoteserver auf Ihrem SQL Server hinzu und gibt an, dass Sie sich bei einem Verbindungsversuch zum Remoteserver mit allen Sicherheitsinformationen anmelden, die Sie auf dem lokalen Server verwenden.

Nun können Sie Abfragen wie die folgende ausführen, die die *titles* von Ihrem lokalen Server mit der Tabelle *publishers* von einem Remoteserver verknüpft:

```
SELECT t1.title, t2.pub_name
FROM pubs..titles t1                -- Lokale Tabelle
INNER JOIN remoteserver.pubs..publishers t2 -- Remote Tabelle
ON t1.pub_id = t2.pub_id
```

Aus den bisherigen Darstellungen dürfte klar sein, dass Sie eine Menge zusätzlicher Informationen brauchen, um Verbindungsserver zu implementieren – vor allem, wenn Sie SQL Server-Tabellen mit Nicht-SQL Server-Daten mithilfe der OLE DB-Unterstützung von heterogenen Daten verknüpfen wollen. Da solche Themen über den Rahmen dieses Buches hinausgehen, sei hier wieder einmal auf die einschlägige Literatur verwiesen.

12.4 Transact-SQL-Steuerungsstrukturen

Schickt man einen Anweisungstapel an SQL Server, werden die Anweisungen normalerweise in der angegebenen Reihenfolge abgearbeitet. Die Sprache Transact-SQL bietet verschiedene Steuerungsstrukturen, mit denen sich die Reihenfolge der Programmausführung ändern lässt. Diese Anweisungen sind vor allem in Transaktionen, Triggern, Funktionen und gespeicherten Prozeduren nützlich.

Blöcke (BEGIN...END)

Bei vielen der hier vorgestellten Steuerungsanweisungen ist als Teil ihrer Syntax eine einzelne Anweisung angegeben. Wenn Sie an diesen Stellen tatsächlich nur eine einzelne Anweisung auszuführen haben, brauchen Sie keine weiteren Vorkehrungen zu treffen. Sind aber mehrere Anweisungen zusammen auszuführen, müssen Sie sie in die Schlüsselwörter BEGIN und END einschließen. Diese Konstruktion finden Sie in verschiedenen Beispielen der nächsten Abschnitte wieder.

Benutzerdefinierte Meldungen (PRINT)

Bislang haben Sie nur eine Möglichkeit kennen gelernt, um Informationen von SQL Server an ein Clientprogramm zurückzugeben: die SELECT-Anweisung. Zusätzlich bietet SQL Server die PRINT-Anweisung, die allerdings recht eingeschränkt ist:



```
PRINT { 'ASCII-Text' | @lokale_Variable | @@globale_Variable }
```

Mit der PRINT-Anweisung können Sie lediglich eine ASCII-Zeichenfolge (Zeichenfolgenkonstante) oder eine Variable eines Zeichentyps (fester oder variabler Länge) wie im folgenden Beispiel ausgeben:

```
PRINT "Hello"
PRINT @@version
```

Wenn Sie etwas Komplexeres ausgeben möchten, müssen Sie die Zeichenfolge in einer Zeichenvariablen zusammensetzen und dann diese Variable mit der PRINT-Anweisung ausgeben.

Das folgende Beispiel verwendet die Variable @msg, um eine längere Zeichenfolge zu erstellen. In einer PRINT-Anweisung sind die Verkettungsoperationen und numerischen Variablen nicht zulässig. Allerdings können Sie diese Elemente in einer Zuweisung mit SET oder SELECT verwenden und dann das Ergebnis ausgeben.



```
USE pubs
DECLARE @msg varchar(50),
        @numWA tinyint
SELECT @numWA = COUNT(*) FROM stores
WHERE state = 'WA'
SELECT @msg = 'Es gibt ' + CONVERT(varchar(3), @numWA) +
           ' Geschäfte in Washington'

PRINT @msg
```



```
(1 row(s) affected)
(1 row(s) affected)
Es gibt 2 Geschäfte in Washington.
```

Bedingte Ausführung (IF...ELSE)

Mit einem IF...ELSE-Block ist es möglich, eine Anweisung bedingt auszuführen. Auf das Schlüsselwort IF folgt ein Ausdruck, der entweder wahr (true) oder falsch (false) ist. Ist der Ausdruck wahr, wird die nächste Anweisung ausgeführt. Nach dem optionalen Schlüsselwort ELSE lässt sich eine alternative Anweisung angeben, die ausgeführt wird, wenn der Ausdruck nach IF das Ergebnis falsch geliefert hat.



```
IF BoolescherAusdruck
{ SQL_Anweisung | Anweisungsblock }
[ELSE [BoolescherAusdruck]
{ SQL_anweisung | Anweisungsblock }]
```



Zum Beispiel gibt die folgende Anweisung einen Wochenbericht mit der Prozedur `uspWeeklyReport` aus, wenn `today` gleich Freitag ist, andernfalls wird ein täglicher Bericht mit der Prozedur `uspDailyReport` ausgegeben:

```
IF (datename(dw, getdate()) = 'Freitag')
    BEGIN
        PRINT 'Wochenbericht'
        EXEC uspWeeklyReport
    END
ELSE
    BEGIN
        PRINT 'Tagesbericht'
        EXEC uspDailyReport
    END
```

Beachten Sie, dass das Ende des `IF...ELSE`-Blocks nicht mit einem speziellen Schlüsselwort zu kennzeichnen ist. Der auf `IF` folgende boolesche Ausdruck kann eine `SELECT`-Anweisung enthalten. Wenn diese `SELECT`-Anweisung einen einzelnen Wert liefert, lässt sich dieser mit einem anderen Wert vergleichen, so dass sich ein boolescher Ausdruck ergibt.

Ist im folgenden Beispiel der durchschnittliche Buchpreis größer als \$15, gibt die Anweisung die erste Meldung aus, andernfalls die zweite Meldung:

```
IF (SELECT avg(price) FROM titles) > 15
    PRINT 'Umsatz halten'
ELSE
    PRINT 'Zeit für eine Preisanhebung'
```

Liefert die nach `IF` angegebene `SELECT`-Anweisung mehrere Werte zurück, können Sie eine spezielle Form von `IF` verwenden: `IF EXISTS`:



```
IF EXISTS (SELECT Anweisung)
    {SQL_Anweisung | Anweisungsblock}
[ELSE
    {SQL_Anweisung | Anweisungsblock}]
```

Die Anweisung `IF EXISTS` gibt *wahr* zurück, wenn die Ergebnismenge der zugehörigen `SELECT`-Anweisung aus mindestens einer Zeile besteht, und *falsch*, wenn die `SELECT`-Anweisung überhaupt keine Zeile zurückgibt. Das folgende Beispiel liefert alle Informationen über alle Bücher, die der Verleger mit der ID 9933 veröffentlicht hat:

```
IF EXISTS
    (SELECT * FROM titles WHERE pub_id = '9933')
BEGIN
    PRINT 'Folgende Bücher wurden gefunden: '
    SELECT * FROM titles WHERE pub_id = '9933'
END
```



```
ELSE
    PRINT 'Keine Bücher von diesem Verleger'
```

Die folgende Abfrage liefert die gleichen Ergebnisse, stellt die Frage aber entgegengesetzt:

```
IF NOT EXISTS
(SELECT * FROM titles WHERE pub_id = '9933')
BEGIN
    PRINT 'Keine Bücher von diesem Verleger: '
    RETURN
END
ELSE
    BEGIN
        PRINT 'Folgende Bücher wurden gefunden: '
        SELECT * FROM titles WHERE pub_id = '9933'
    END
END
```

In manchen Fällen bietet IF EXISTS eine bessere Leistung als alternative Verfahren, weil SQL Server die Verarbeitung stoppen kann, sobald die erste Zeile gefunden ist.



Verwenden Sie IF EXISTS nicht mit Aggregaten, da diese immer Daten zurückgeben, auch wenn der Wert dieser Daten gleich 0 ist. Wenn Sie also sehen möchten, ob der Verleger 9933 irgendwelche Bücher veröffentlicht hat, lässt sich das mit der folgenden Anweisung *nicht* erledigen:

```
IF EXISTS
(SELECT COUNT(*) FROM titles WHERE pub_id = '9933')
```

Diese SELECT-Anweisung gibt immer eine Zeile zurück. Wenn die Datenbank keine Bücher von diesem Verleger enthält, hat diese eine Zeile den Wert 0, aber IF EXISTS liefert true zurück.

Bedingungsliste (CASE)

Programmierer müssen häufig einen Bedingungsausdruck innerhalb einer anderen Anweisung anwenden. Mit dem CASE-Ausdruck lassen sich Transact-SQL-Ausdrücke für Bedingungs-werte einfacher formulieren. Mit dieser Konstruktion kann man unterschiedliche Werte zurückgeben, abhängig vom Wert einer Steuervariablen oder Bedingung.

Ein einfacher CASE-Ausdruck hat folgende Syntax:



```
CASE Eingabeausdruck
    WHEN WhenAusdruck1 THEN Ergebnisausdruck1
    [[WHEN WhenAusdruck2 THEN Ergebnisausdruck2][...]]
    [ELSE Alternativergebnisausdruck]
END
```

Die Syntax der komplexen CASE-Anweisung lautet:



```
CASE
  WHEN BoolescherAusdruck1 THEN ErgebnisAusdruck1
  [[WHEN BoolescherAusdruck2 THEN ErgebnisAusdruck2[[...]]]
  [ELSE AlternativErgebnisausdruck]
END
```

Die einfache CASE-Konstruktion vergleicht einen *Eingabeausdruck* mit jedem aufgeführten *WhenAusdruck* und gibt bei Übereinstimmung den zugehörigen *Ergebnisausdruck* zurück. Wenn keiner dieser Ausdrücke eine Übereinstimmung liefert, wird der *Alternativergebnisausdruck* nach dem Schlüsselwort ELSE zurückgegeben.

Sehen wir uns einige CASE-Konstruktionen in der Praxis an. Der Code im folgenden Beispiel vergleicht für jede Zeile in der Tabelle *titles* den Wert in der Spalte *type* mit jedem Wert in der CASE-Liste. Beim ersten übereinstimmenden Wert gibt die CASE-Anweisung den zugehörigen Ergebnisausdruck zurück. Liefert keiner der Ausdrücke das Ergebnis true, wird der Ergebnisausdruck nach ELSE zurückgegeben.



```
SELECT title_id, type = CASE type
  WHEN 'business' THEN 'Business Book'
  WHEN 'psychology' THEN 'Psychology Book'
  WHEN 'mod_cook' THEN 'Modern Cooking Book'
  WHEN 'trad_cook' THEN 'Traditional Cooking Book'
  WHEN 'popular_comp' THEN 'Popular Computing Book'
  WHEN 'undecided' THEN 'No type determined yet'
END
FROM titles
```

Die Ergebnismenge sieht folgendermaßen aus:



title_id	type
BU1032	Business Book
BU1111	Business Book
BU2075	Business Book
BU7832	Business Book
MC2222	Modern Cooking Book
MC3021	Modern Cooking Book
MC3026	No type determined yet
PC1035	Popular Computing Book
PC8888	Popular Computing Book
PC9999	Popular Computing Book
PS1372	Psychology Book
PS2091	Psychology Book
PS2106	Psychology Book

```

PS3333  Psychology Book
PS7777  Psychology Book
TC3218  Traditional Cooking Book
TC4203  Traditional Cooking Book
TC7777  Traditional Cooking Book

```

(18 row(s) affected)

Das nächste Beispiel wertet mehrere Ausdrücke für jede Zeile in der Tabelle *titles* aus. In diesem Fall wird der Wert von *price* mit verschiedenen Werten verglichen. Der erste Ausdruck, der *wahr* ist, bestimmt den zurückgegebenen Ausdruck.



```

SELECT title_id, cost = CASE
    WHEN price < 10 THEN 'Billig'
    WHEN price BETWEEN 10 AND 20 THEN 'Mittel'
    ELSE 'Teuer'
END
FROM titles

```

Diese Anweisung liefert folgende Ausgabe:



```

title_id cost
-----
BU1032  Mittel
BU1111  Mittel
BU2075  Billig
BU7832  Mittel
MC2222  Mittel
MC3021  Billig
MC3026  Teuer
PC1035  Teuer
PC8888  Mittel
PC9999  Teuer
PS1372  Teuer
PS2091  Mittel
PS2106  Billig
PS3333  Mittel
PS7777  Billig
TC3218  Teuer
TC4203  Mittel
TC7777  Mittel

```

(18 row(s) affected)

Das erste CASE-Beispiel ist eigentlich nur eine Spezialversion der komplexen CASE-Konstruktion. Man kann das erste Beispiel wie folgt formulieren und erhält die gleichen Ergebnisse:

```
SELECT title_id, type = CASE
    WHEN type = 'business' THEN 'Business Book'
    WHEN type = 'psychology' THEN 'Psychology Book'
    WHEN type = 'mod_cook' THEN 'Modern Cooking Book'
    WHEN type = 'trad_cook' THEN 'Traditional Cooking Book'
    WHEN type = 'popular_comp' THEN 'Popular Computing Book'
    ELSE 'No type determined yet'
END
FROM titles
```

Die einfache Form der CASE-Anweisung setzt man ein, wenn alle Bedingungstests auf Gleichheit erfolgen. Anstelle von CASE kann man auch zwei andere Funktionen verwenden: COALESCE und NULLIF.

Die Funktion COALESCE

Manchmal hat ein CASE-Ausdruck die folgende Form:

```
CASE
    WHEN WhenAusdruck1 IS NOT NULL THEN Ergebnisausdruck1
    [[WHEN WhenAusdruck2 IS NOT NULL THEN Ergebnisausdruck2]
    [...]]
    [ELSE Alternativergebnisausdruck]
END
```

Sobald hier der erste Wert in der Liste der *WhenAusdruck*-Werte ungleich NULL ist, wird der zugehörige *Ergebnisausdruck* zurückgegeben. Man kann diese Konstruktion auch mithilfe der Funktion COALESCE formulieren.

Im folgenden Beispiel enthält eine Lohntabelle drei Spalten mit Informationen über den Stundenlohn, das Gehalt und die Provision eines Mitarbeiters. Aus diesen Angaben lässt sich der Jahreslohn berechnen. Normalerweise erhält aber ein Mitarbeiter nur eine Art dieser Lohnzahlungen, so dass nur in einer der drei Spalten ein Wert erscheint. Die anderen beiden Spalten sind NULL. Erstellen Sie zunächst die Beispieltabelle mit einigen Daten:

```
USE pubs
GO
CREATE TABLE Lohn (
    PersNr int NOT NULL,
    Stunde money NULL,
    Gehalt money NULL,
    Provision money NULL,
    Absatz bigint NULL)
GO
INSERT Lohn VALUES (111, NULL, 52000, NULL, NULL)
INSERT Lohn VALUES (112, 14, NULL, NULL, NULL)
```

```

INSERT Lohn VALUES (113, NULL, NULL, 0.15, 50000)
INSERT Lohn VALUES (114, NULL, 73000, NULL, NULL)
INSERT Lohn VALUES (115, 4.90, NULL, NULL, NULL)
INSERT Lohn VALUES (116, NULL, 28500, NULL, NULL)
GO

```

Nachdem Sie diese Tabelle erstellt und mit Daten gefüllt haben, können Sie den Jahreslohn für alle Mitarbeiter berechnen. Die folgende Anweisung bedient sich dazu der COALESCE-Funktion, um nur die Nicht-NULL-Werte aus den Spalten *Stunde*, *Gehalt* und *Provision* zu übernehmen:



```

SELECT PersNr, "Jahreslohn" = CONVERT(money,
    (COALESCE(Stunde * 40 * 52, Gehalt, Provision * Absatz)))
FROM Lohn

```



PersNr	Jahreslohn
111	52000.0000
112	29120.0000
113	7500.0000
114	73000.0000
115	10192.0000
116	28500.0000

(6 row(s) affected)

Die Funktion NULLIF

Eine CASE-Anweisung hat manchmal auch die folgende Form:

```

CASE
    WHEN Ausdruck1 = Ausdruck2 THEN NULL
    [ELSE Ausdruck1]
END

```

Hier gibt die CASE-Anweisung den Wert NULL zurück, wenn die beiden Ausdrücke gleich sind; andernfalls wird der Wert des ersten Ausdrucks zurückgegeben.

Nehmen wir nun eine Tabelle wie im vorherigen Beispiel an – die Tabelle *Lohn* mit den gleichen Daten, wobei jetzt aber die NULL-Werte als 0 gespeichert sind. Mit dem folgenden Code können Sie für dieses Beispiel die NULL-Werte in 0 ändern:



```
USE pubs
GO
UPDATE Lohn
SET Stunde = ISNULL(Stunde, 0),
    Gehalt = ISNULL(Gehalt, 0),
    Provision = ISNULL(Provision, 0),
    Absatz = ISNULL(Absatz, 0)
```

Den Einsatz von `NULLIF` soll folgendes Szenario deutlich machen. Wenn Sie in einer `SELECT`-Anweisung die Anzahl der Mitarbeiter mit Stundenlohn, die Anzahl der Gehaltsempfänger und die Empfänger von Provision zurückgeben wollen, funktioniert der folgende Code *nicht*:

```
USE pubs
GO
SELECT Stunde = COUNT(Stunde),
    Gehalt = COUNT(Gehalt),
    Provision = COUNT(Provision)
FROM Lohn
```

Die Funktion `COUNT` zählt, wie viele Werte in einer Spalte enthalten sind – und 0 zählt als Wert. Nur die `NULL`-Werte erscheinen nicht im Ergebnis von `COUNT`. Mit der Funktion `NULLIF` können Sie nun die 0-Werte in `NULL`-Werte umwandeln und die Nicht-`NULL`-Werte unverändert lassen. Mit dem folgenden Code erhalten Sie dann die gewünschten Antworten:



```
SELECT Stunde = COUNT(NULLIF(Stunde, 0)),
    Gehalt = COUNT(NULLIF(Gehalt, 0)),
    Provision = COUNT(NULLIF(Provision, 0))
FROM Lohn
```



Stunde	Gehalt	Provision
2	3	1

(1 row(s) affected)

Schleifen (WHILE)

Transact-SQL verfügt über eine `WHILE`-Konstruktion, mit der sich Anweisungen wiederholt ausführen lassen, bis eine bestimmte Bedingung erfüllt ist. In vielen Fällen kann man auf eine derartige Schleifenkonstruktion verzichten, weil `SQL` von Natur aus mit Ergebnismengen arbeitet. Beispielsweise brauchen Sie keine `WHILE`-Anweisung, um jede Zeile einer Tabelle zu durchlaufen – das erledigt die `SELECT`-Anweisung selbst und prüft dabei, ob die `WHERE`-Kriterien erfüllt sind.

In bestimmten Situationen sind aber Aktionen wiederholt auszuführen. Man kann sie in einer Transaktion wiederholen (nicht empfohlen, weil Transaktionen möglichst klein und schnell sein sollen), in einer gespeicherten Prozedur oder in einer SQL-Funktion. Auch bei der Navigation durch einen Cursor können WHILE-Anweisungen sinnvoll sein.

Eine WHILE-Konstruktion wiederholt die Ausführung, solange eine spezifizierte Bedingung `true` bleibt. Enthält die Bedingung eine `SELECT`-Anweisung, gilt analog zur `IF...ELSE`-Anweisung, dass die gesamte `SELECT`-Anweisung in Klammern stehen muss.

Das folgende Beispiel prüft wiederholt die Durchschnittspreise für alle Bücher. Solange dieser Durchschnitt unter \$20 liegt, wird der Preis jedes Buches aktualisiert. Die Schleife wird verlassen, wenn der Durchschnittspreis größer oder gleich \$20 ist:

```
WHILE (SELECT AVG(price) FROM titles) < 20
BEGIN
    UPDATE titles SET price = price * 1.1
    PRINT 'Alle Preise wurden um 10% angehoben.'
END
```



Wie in jeder Programmiersprache kann es vorkommen, dass die Schleife endlos läuft. Es liegt in der Verantwortlichkeit des Programmierers, die Abbruchbedingung geeignet zu formulieren.

Im Beispiel wiederholt sich die Ausführung, solange der Preis kleiner als ein bestimmter Betrag ist. Deshalb muss innerhalb der Schleife eine Operation stattfinden, die den Preis erhöht. Dadurch erreicht der Durchschnittspreis nach einigen Schleifendurchläufen schließlich einen Wert, der über der Bedingung liegt. (Nur wenn alle Preise gleich \$0.00 sind, funktioniert das nicht.) Falls die `UPDATE`-Anweisung den Buchpreis verringert (`price = price * 0.9`), haben Sie es mit einer Endlosschleife zu tun.

12.5 Sperren

Der Begriff Sperren ruft häufig negative Assoziationen hervor; dabei tragen aber Sperren dazu bei, dass in einer Datenbank parallele Vorgänge ablaufen können. Oft hört man von Sperrproblemen, nur selten redet man vom Nutzen der Sperren. Fest steht: Ohne Sperren hätte SQL Server keinen Mechanismus, um mehrere Benutzer daran zu hindern, Daten gleichzeitig zu ändern.

Im Allgemeinen finden Sie vier Typen von Sperren in SQL Server:

- **Gemeinsame Sperren** – auch als *Lesesperren* bezeichnet – ermöglichen, dass gleichzeitige Transaktionen eine Ressource lesen können. Eine gemeinsame Sperre hindert andere Benutzer daran, die Daten zu ändern, während Sie sie anzeigen. Gemeinsame Sperren sind kompatibel mit anderen gemeinsamen Sperren.

- *Exklusive Sperren* verhindern, dass mehrere Transaktionen gleichzeitig auf eine Ressource zugreifen. Eine exklusive Sperre verwenden Sie, wenn Sie Daten verändern wollen. Andere Benutzer können die Daten erst wieder anzeigen oder bearbeiten, wenn Sie die Sperre aufheben. Exklusive Sperren sind nicht kompatibel mit anderen Sperren.
- *Aktualisierungssperren* verwenden Sie fast wie exklusive Sperren. Aktualisierungssperren hindern andere Benutzer an der Bearbeitung von Daten, während Sie diese Daten ändern.
- Eine *beabsichtigte Sperre* zeigt an, dass SQL Server plant, eine Sperre (der oben beschriebenen Arten) auf ein Objekt weiter unten in der Hierarchie anzuwenden. Über beabsichtigte Sperren werden Sie in Kürze mehr lernen.



Aktualisierungssperren sind notwendig, wenn eine Abfrage zwei Phasen durchläuft, um Daten zu bearbeiten: eine Suchphase und eine Bearbeitungsphase. Wenn SQL Server eine gemeinsame Sperre während der Suchphase verwendet, könnte ein anderer Benutzer ebenfalls eine gemeinsame Sperre auf dasselbe Objekt anwenden. Wenn die Suchtransaktion versucht, die Daten zu bearbeiten, braucht sie eine exklusive Sperre. Die andere Transaktion könnte bereits versucht haben, eine exklusive Sperre zu erhalten und SQL Server wird Ihnen keine exklusive Sperre geben. Daher könnte eine *Blockierungssituation* oder ein *Deadlock* auftreten. Um dies zu verhindern, wird eine Aktualisierungssperre verwendet, die eine andere Transaktion daran hindert, exklusives Sperren auf das Objekt anzuwenden, das zur Aktualisierung gesperrt wird.

Sperrentypen

Es lassen sich unterschiedliche Ebenen oder Typen von Objekten sperren:

- *RID* – Zeilen-ID – ist ein anderer Begriff für eine Sperre auf Zeilenebene. Bei diesem Typ ist jeweils nur eine Zeile zu einem bestimmten Zeitpunkt gesperrt.
- *Schlüssel* ist eine Zeilensperre in einem Index. Die Schlüsselsperre sperrt entweder einen einzigen Schlüsselwert oder mehrere Schlüsselwerte (als Schlüsselbereichssperre bezeichnet). Dieser Sperrentyp wird verwendet, um Schlüsselbereiche in serialisierbaren Transaktionen zu schützen.
- *Seite* ist die Standardeinheit von 8 KB in SQL Server. Eine Seitensperre sperrt die gesamten Inhalte auf einer einzigen Seite – was eine oder viele Zeilen betreffen kann.
- Ein *Block* (aufeinanderfolgende Gruppe von acht Datenseiten oder Indexseiten) wird angefordert, wenn für ein bestimmtes Objekt keine Seiten mehr verfügbar sind und weitere Daten hinzugefügt werden müssen.

- Eine *Tabellensperre* kann man entweder automatisch über die Sperrenausweitung oder explizit anfordern. Alle Seiten in der Tabelle werden als Einheit gesperrt.
- Eine *beabsichtigte* Sperre ist ein Weg, um auf Tabellenebene anzuzeigen, dass es Seiten- oder Zeilensperren gibt oder dass bei einer Seite eine Zeilensperre vorhanden ist. Wenn beispielsweise eine einzige gemeinsame Seitensperre innerhalb einer Tabelle besteht, wird eine beabsichtigte gemeinsame Sperre auf Tabellenebene eingerichtet.

In SQL Server 2000 ist das Sperren vollständig dynamisch; vor der Version 7.0 war die Sperrung auf Seitenebene der Standard. SQL Server 2000 entscheidet über den Sperrtyp, wenn eine Abfrage optimiert wird. Bei Auswahloperationen für eine sehr kleine Datenmenge oder beim Einfügen, Aktualisieren oder Löschen mit geringem Umfang kommen wahrscheinlich Sperren auf Zeilenebene (und/oder Sperren im Schlüsselbereich) infrage. Bei sehr großen Auswahloperationen (wie `SELECT * FROM Riesentabelle`) ist es möglicherweise effektiver, Sperren auf Seiten- oder sogar Tabellenebene einzusetzen.

Bei einer Zeilensperre ist jeweils nur eine einzige Zeile statt einer ganzen Seite oder Tabelle von der Sperre betroffen. Eine Zeilensperre ist nützlich, weil eine einzige Seite viele Zeilen erhalten kann. Allgemein gesagt ist die parallele Programmausführung (die Fähigkeit mehrerer Benutzer, gleichzeitig auf Daten zuzugreifen) um so besser, je kleiner die Sperreinheit ist. Nachteilig dabei ist aber, dass zum Beispiel 5000 Zeilensperren wesentlich mehr Ressourcen und somit Zeit verbrauchen als eine einzige Tabellensperre. Weil Sperren in der Regel mit Kompromissen verbunden sind und der optimale Sperrtyp nicht immer offensichtlich ist, sollte man SQL Server die Sperrentscheidungen überlassen.

Steuern des Sperrverhaltens

Normalerweise müssen Sie sich um das Sperrverhalten keine Gedanken machen. Bei den Operationen `INSERT`, `UPDATE` und `DELETE` verwendet SQL Server eine exklusive Sperre. SQL Server kann jedoch mit so genannten Sperrhinweisen in `SELECT`-, `INSERT`-, `UPDATE`- und `DELETE`-Anweisungen die Sperren auf der Basis einzelner Abfragen konfigurieren. Sie geben sie nach dem Namen der Tabelle in Ihrer Abfrage an. Gelegentlich müssen Sie das Standardsperrverhalten ändern, weil Probleme mit Transaktionen auftreten, die im Konflikt miteinander stehen oder sich gegenseitig blockieren.

Um beispielsweise ein exklusives Tabellensperren für die Tabelle *authors* zu erzwingen, so dass niemand die Tabelle bearbeiten kann, während Sie sie bearbeiten, können Sie die folgende Abfrage ausführen:

```
SELECT *  
FROM authors (TABLOCKX)
```

Es gibt verschiedene Parameter, die Sie anstelle von TABLOCKX verwenden können:

- NOLOCK fordert, dass keine Sperren verwendet werden. (Derartige Lesevorgänge bezeichnet man als *Dirty Read* – »unsauberes Lesen«.) Die Verwendung dieses Optimierungshinweises erlaubt einer Abfrage das Lesen von Daten, die zum exklusiven Gebrauch gesperrt sind. Dadurch ist es möglich, dass geänderte aber noch nicht bestätigte Daten als Teil der Abfrage gelesen werden. Auch wenn diese Option unter bestimmten Umständen nützlich ist, sollten Sie sie nur einsetzen, wenn Sie sich genau über die Konsequenzen im Klaren sind.
- READUNCOMMITTED entspricht der Option NOLOCK.
- READPAST überspringt gesperrte Zeilen, die es normalerweise verhindern, dass die Transaktion mit einer Leseoperation fortfahren kann (die Transaktion wird »blockiert«, bis die Sperren aufgehoben werden). Diese Option ist mit Vorsicht zu genießen, weil im Resultset Daten fehlen können.
- REPEATABLE READ gibt an, dass das Sperren wie eine Transaktion auf der Isolationsstufe REPEATABLE READ durchzuführen ist. Im Allgemeinen heißt das, dass Lesesperren erst dann aufgehoben werden, wenn die Transaktion beendet ist. Somit lassen sich die von einer Transaktion gelesenen Daten erst dann durch eine andere Transaktion modifizieren, wenn die erste Transaktion abgeschlossen ist. Daher werden Aktualisierungen und Löschvorgänge verhindert, aber Einfügungen sind erlaubt.
- HOLDLOCK erhält eine gemeinsame Sperre für die Dauer einer Transaktion aufrecht. Normalerweise werden während einer Auswahlenweisung gemeinsame Sperren angefordert und sofort wieder aufgehoben, wenn die nächste Zeile oder Seite abgerufen wird. Mit dieser Option werden solche gemeinsamen Sperren erst aufgehoben, wenn die aktuelle Transaktion entweder bestätigt oder rückgängig gemacht worden ist. Zudem werden Sperren so ausgegeben, dass auch keine Einfügeoperationen erlaubt sind. Dieses Verhalten ist im Allgemeinen bei Schlüsselbereichssperren implementiert.
- SERIALIZABLE entspricht der Option HOLDLOCK.
- UPDLOCK fordert eine Aktualisierungssperre statt einer gemeinsamen Sperre an. Diese Option verwendet man normalerweise nicht.
- ROWLOCK erzwingt die Verwendung von Sperren auf Zeilenebene.
- PAGLOCK fordert eine gemeinsame Seitensperre an.
- TABLOCK fordert eine gemeinsame Sperre auf Tabellenebene, statt einzelne Seiten oder Zeilen zu sperren.
- TABLOCKX belegt eine Tabelle mit einer exklusiven Sperre.

Sperrenausweitung

Sperrenausweitung ist der Vorgang, bei dem eine Sperre auf niedriger Ebene (beispielsweise Zeilen- oder Seitensperre) zu einer Sperre auf höherer Ebene (zum Beispiel Tabellensperre) konvertiert wird. SQL Server weitet Sperren automatisch aus, wenn eine Transaktion die entsprechende Eskalationsschwelle überschreitet. Sie haben keine Kontrolle darüber, wann dies geschieht.

Die Option LOCKS

Für Sperren steht eine weitere Konfigurationsmöglichkeit zur Verfügung: die Option LOCKS. Diese gehört zu den erweiterten Konfigurationsoptionen. Der Standardwert 0 bedeutet, dass SQL Server die Anzahl der Sperren dynamisch je nach Systemanforderungen einstellt. Dabei reserviert SQL Server 2 % des verwendeten Speichers für Sperren. Sie können zwar die Anzahl der Sperren manuell einstellen, doch wenn Ihnen die Sperren ausgehen, schreibt SQL Server Fehlermeldungen sowohl in das Anwendungsereignisprotokoll von Windows NT/2000 als auch in das Fehlerprotokoll von SQL Server. Alle Aktivitäten, die Sperren auf dem System verwenden, halten an, bis mehr Sperren verfügbar werden. Da jede Sperre etwas Speicher benötigt, sollten Sie nur dann Sperren manuell zuweisen oder die Parameter höher setzen, wenn Sie sich vollkommen über die Auswirkungen dieser Veränderung im Klaren sind.

Sperren überwachen

Um Sperren zu überwachen, können Sie die gespeicherten Prozeduren `sp_lock` oder `sp_processinfo` ausführen, wie es das folgende Beispiel zeigt:



```
EXEC sp_lock
GO
```



spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	1	0	0	DB		S	GRANT
6	1	0	0	DB		S	GRANT
7	1	0	0	DB		S	GRANT
7	2	0	0	DB		S	GRANT
7	5	0	0	DB		S	GRANT
7	2	0	0	EXT	1:80	X	GRANT
7	1	117575457	0	TAB		IS	GRANT
8	1	0	0	DB		S	GRANT

(8 row(s) affected)

Wie die Ausgabe der Prozedur `sp_lock` zeigt, kann es schwierig sein, die Ergebnisse zu interpretieren. Sie müssen die *dbid* (Datenbank-ID) aus der Tabelle *sysdatabases* in der *master*-Datenbank und die *ObjID* (Objekt-ID) aus der Tabelle *sysobjects* in der entsprechenden Datenbank nachschlagen. Gegebenenfalls müssen Sie auch die *IndId* (Index-ID) aus der Tabelle *sysindexes* in dieser Datenbank herausuchen. Die Spalte *Type* gibt die Art des gesperrten Objekts an (Datenbank, Tabelle, Block, Schlüssel, Seite oder Zeile). Die Spalte *Resource* zeigt je nach Typ der Sperre an, was gesperrt ist:

- EXT (Block) hat das Format *Dateinummer:Erweiterung*. Beispielsweise bedeutet 1:80, dass die Datei 1 der Datenbank im Block 80 mit einer Blocksperre belegt ist.
- PAG (Seite) hat das Format *Dateinummer:Seitennummer*. Beispielsweise ist 1:2401 eine Seitensperre auf Seitennummer 2401 in Dateinummer 1 in der Datenbank.
- RID (Zeile) hat das Format *Dateinummer:Seitennummer:Zeilen-ID*. Somit gibt 1:32:16 an, dass Dateinummer 1, Seite 32, Zeile 16 auf dieser Seite gesperrt ist.
- KEY (Schlüsselbereich) zeigt interne Informationen von SQL Server, die der Benutzer nicht interpretieren kann.

Der Spermodus (Spalte *Mode*) gibt den Typ der Sperre an (S für gemeinsam, X für Exklusiv, I für beabsichtigt usw.).

Wahrscheinlich sind Sie vor allem an der Spalte *Status* interessiert. Der Status GRANT (Erteilt) bedeutet, dass die Sperre derzeit in Benutzung ist, der Status WAIT (Warten) zeigt an, dass die Sperre wegen eines anderen Vorgangs blockiert ist, und CNVT (Konvertieren) heißt, dass eine Sperre zu einer anderen Sperre umgewandelt – höchstwahrscheinlich ausgeweitet – wird. Im Hinblick auf Probleme beim Blockieren lassen sich CNVT und WAIT als vergleichbar einstufen.

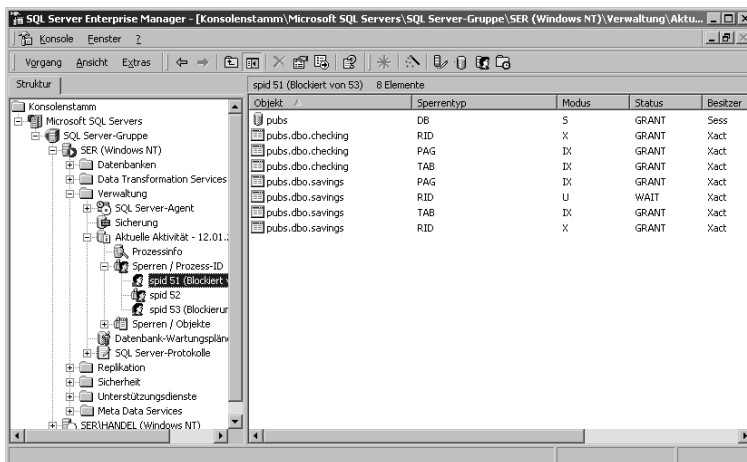


Abbildung 12.4:
Das Fenster *Aktuelle Aktivität* für Sperren / Prozess-ID

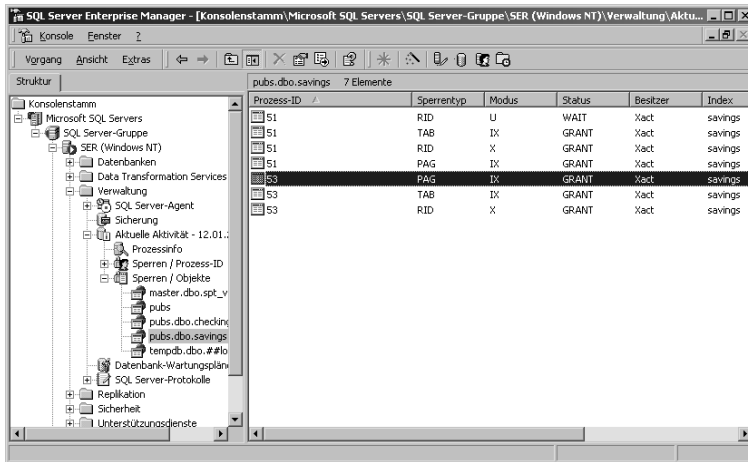


Abbildung 12.5:
Das Fenster *Aktuelle Aktivität* für *Sperren / Objekte*

Es ist vielleicht einfacher, die Sperren mit dem SQL Server Enterprise Manager im Fenster **AKTUELLE AKTIVITÄT** zu beobachten. Um dieses Fenster zu öffnen, erweitern Sie den Knoten für Ihren Server, dann den Ordner *Verwaltung* und schließlich den Knoten *Aktuelle Aktivität*. Hier können Sie sehen, welche Prozesse (Benutzer) aktiv sind, welche Sperren ein Prozess hält und welche Sperren ein Objekt hält. Abbildung 12.4 zeigt ein Beispiel für das Fenster **AKTUELLE AKTIVITÄT**.

Wenn Sie den Knoten *Sperren / Objekte* erweitern, können Sie die Sperren beobachten, die für jedes Objekt wirksam sind (siehe Abbildung 12.5).

Kompatibilität von Sperren

Sperren können mit anderen Sperren kompatibel sein oder nicht. Tabelle 12.1 zeigt eine Kompatibilitätsmatrix für Sperren.

	IS	S	U	IX	SIX	X
Beabsichtigt gemeinsam (IS)	Ja	Ja	Ja	Ja	Ja	Nein
Gemeinsam (S)	Ja	Ja	Ja	Nein	Nein	Nein
Aktualisierung (U)	Ja	Ja	Nein	Nein	Nein	Nein
Beabsichtigt exklusiv (IX)	Ja	Nein	Nein	Ja	Nein	Nein
Gemeinsam mit beabsichtigt exklusiv (SIX)	Ja	Nein	Nein	Nein	Nein	Nein
Exklusiv (X)	Nein	Nein	Nein	Nein	Nein	Nein

Beachten Sie, dass exklusive Sperren mit jeder anderen Sperre unverträglich sind. Das erscheint auch sinnvoll, denn wenn Sie Daten ändern, möchten Sie sicherlich nicht, dass ein anderer Benutzer genau dieselben Daten ändert. Andererseits sind gemeinsame Sperren ausreichend flexibel. Wenn Sie Daten lesen, ist es in Ordnung, wenn andere Benutzer dieselben Daten lesen.

Deadlocks



Ein *Deadlock* ist eine Situation, in der zwei Transaktionen miteinander im Konflikt stehen und die einzige Lösung darin besteht, eine der Transaktionen abubrechen. Ein Deadlock lässt sich am Einfachsten mit einem Beispiel veranschaulichen.

1. Erstellen Sie mit dem Query Analyzer die folgenden beiden Tabellen und füllen Sie sie mit Daten:



```
USE pubs
GO

CREATE TABLE checking
(acct_num INT NOT NULL,
 last_name CHAR(30) NOT NULL,
 balance MONEY NOT NULL
)
GO
CREATE TABLE savings
(acct_num INT NOT NULL,
 last_name CHAR(30) NOT NULL,
 balance MONEY NOT NULL
)
GO
INSERT checking VALUES (1,'smith', $500.00)
INSERT checking VALUES (2,'Jones', $300.00)
INSERT savings VALUES (1,'smith', $100.00)
INSERT savings VALUES (2,'Jones', $200.00)
GO
```

2. Öffnen Sie jetzt ein zweites Query Analyzer-Fenster (über DATEI / VERBINDEN). Im ersten Fenster führen Sie die folgende Anweisung aus:



```
USE pubs
GO
BEGIN TRAN
```

```
UPDATE checking
SET balance = balance + $100.00
WHERE acct_num = 1
```

Sie sollten zurückbekommen:



(1 row(s) affected)

3. Im zweiten Fenster führen Sie diese Anweisungen aus:



```
USE pubs
GO
BEGIN TRAN
    UPDATE savings
    SET balance = balance - $100.00
    WHERE acct_num = 2
```

Dabei sollten Sie das gleiche Ergebnis erhalten. So weit, so gut.

4. Jetzt gehen Sie wieder in das erste Fenster und führen die folgenden Anweisungen aus:



```
UPDATE savings
SET balance = balance - $100.00
WHERE acct_num = 1
```

Beachten Sie, dass die Transaktion noch läuft, während sie das andere Abfragefenster blockiert. Führen Sie in einem dritten Fenster die gespeicherte Systemprozedur `sp_lock` aus. Dabei erhalten Sie etwa folgende Ergebnisse:

spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
9	5	0	0	DB		S	GRANT
9	5	1429580131	0	RID	1:170:0	X	GRANT
9	5	1429580131	0	RID	1:170:1	U	WAIT
9	5	1413580074	0	RID	1:167:0	X	GRANT
9	5	1429580131	0	PAG	1:170	IX	GRANT
9	5	1413580074	0	PAG	1:167	IX	GRANT
9	5	1413580074	0	TAB		IX	GRANT
9	5	1429580131	0	TAB		IX	GRANT
10	5	0	0	DB		S	GRANT
10	5	1429580131	0	RID	1:170:1	X	GRANT
10	5	1429580131	0	PAG	1:170	IX	GRANT
10	5	1429580131	0	TAB		IX	GRANT

(12 row(s) affected)

Beachten Sie, dass für eine Serverprozess-ID (spid) der Status `WAIT` angezeigt wird. Das weist darauf hin, dass der Prozess wartet, um eine Sperre zu erhalten, und er erst dann fortfahren kann, wenn eine andere Serverprozess-ID ihre Sperre auf der angeforderten Ressource aufhebt.

5. Führen Sie nun im zweiten Fenster die folgenden Anweisungen aus:



```
UPDATE checking
SET balance = balance + $100.00
WHERE acct_num = 2
```

Daraufhin erhalten Sie die Fehlermeldung:

Server: Nachr.-Nr. 1205, Schweregrad 13, Status 50, Zeile 1

Ihre Transaktion (Prozess-ID #12) befindet sich wegen eines anderen Prozesses in einer Deadlocksituation und wurde als Deadlock-Opfer ausgewählt. Führen Sie die Transaktion erneut aus.

6. Die ursprüngliche Abfrage ist jetzt beendet. Führen Sie den folgenden Code zum Aufräumen im ersten Fenster aus:

```
COMMIT TRAN
GO
DROP TABLE checking
DROP TABLE savings
GO
```

Das Vermeiden von Deadlocks ist wichtig, weil Zeit und Ressourcen verschwendet werden, wenn ein Deadlock auftritt. Ein Weg zum Vermeiden von Deadlocks besteht darin, auf Tabellen immer in der gleichen Reihenfolge zuzugreifen. Im vorhergehenden Beispiel wäre es nicht zum Deadlock gekommen, wenn beide Transaktionen mit der gleichen Tabelle begonnen hätten. Eine der Transaktionen hätte auf die Fertigstellung der anderen gewartet, bevor sie begonnen hätte. Wenn Sie in eine Deadlock-Situation geraten, empfiehlt es sich im Allgemeinen, eine oder zwei Sekunden zu warten und die Transaktion dann erneut zu starten.



Ein verbreiteter Irrglaube ist, dass keine Deadlocks auftreten, wenn Sperren auf Zeilenebene wirksam sind. Wie Sie aus dem vorhergehenden Beispiel sehen können, haben Deadlocks weniger damit zu tun, dass es Sperren auf Zeilenebene gibt, sondern damit, dass die falschen Sperren zur falschen Zeit gebraucht werden. Umfangreiche Tests sind wichtig, um möglichst alle Deadlocks in Ihren Anwendungen zu vermeiden.

Isolationsstufen von Transaktionen

Isolationsstufen von Transaktionen beeinflussen die Standardarten und die Dauer von Sperren während einer `SELECT`-Anweisung. Wie weiter vorn in dieser Lektion gezeigt, lassen sich die Sperrarten für jede Abfrage einzeln überschreiben. Die Isolationsstufe verhält sich ähnlich, kann jedoch auf Basis der Sitzungsebene überschrieben werden (das heißt, dass diese Einstellung für alle Abfragen während dieser Sitzung wirksam ist). Die Isolationsstufe einer Transaktion legen Sie mit der Anweisung `SET TRANSACTION ISOLATION LEVEL` fest:



```
SET TRANSACTION ISOLATION LEVEL
    {READ COMMITTED | READ UNCOMMITTED | REPEATABLE READ |
    SERIALIZABLE}
```

- Der Standardwert ist `READ COMMITTED`, was bedeutet, dass eine Auswahlabfrage nur Daten sieht, für die die Abfrage eine gemeinsame Sperre erhalten kann (sie führt keine Dirty Reads durch).
- `READ UNCOMMITTED` entspricht dem Sperrhinweis `NOLOCK` und erlaubt Dirty Reads in allen Abfragen während einer bestimmten Sitzung. Seien Sie vorsichtig mit dieser Option, weil Sie möglicherweise ungültige – d. h. noch nicht bestätigte – Daten sehen.
- `REPEATABLE READ` hebt gemeinsame Sperren erst nach Fertigstellung der Transaktion auf und ist gleichwertig mit dem Sperrhinweis `HOLDLOCK`. Diese Option verhindert Aktualisierungen und Löschvorgänge von Daten, die Sie gelesen haben.
- `SERIALIZABLE` verhindert nicht nur Aktualisierungen und Löschvorgänge von gelesenen Daten, sondern auch Einfügevorgänge im Schlüsselbereich gelesener Daten.

Sowohl `REPEATABLE READ` als auch `SERIALIZABLE` haben dramatische Auswirkungen auf die parallele Programmausführung und Sie sollten sehr darauf achten, dass sie die Folgen der jeweiligen Verwendung verstehen, bevor Sie diese Isolationsstufen implementieren.

Mit der Anweisung `DBCC USEROPTIONS` können Sie die wirksame Isolationsstufe anzeigen:



```
DBCC USEROPTIONS
GO
```

Wenn Sie vorher noch nicht die Anweisung `SET TRANSACTION ISOLATION LEVEL` ausgeführt haben, erhalten Sie folgende Ausgabe:



Set Option	Value

textsize	64512
language	Deutsch
dateformat	dmy
datefirst	1
ansi_null_dflt_on	SET

(5 row(s) affected)

DBCC-Ausführung abgeschlossen. Falls DBCC Fehlermeldungen ausgegeben hat, wenden Sie sich an den Systemadministrator.

Achten Sie auf den Unterschied, nachdem Sie mit den folgenden Anweisungen die Option `READ UNCOMMITTED` für Dirty Reads eingeschaltet haben:



```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
GO
DBCC USEROPTIONS
GO
```



Set Option	Value

textsize	64512
language	Deutsch
dateformat	dmy
datefirst	1
ansi_null_dflt_on	SET
isolation level	read uncommitted

(6 row(s) affected)

DBCC-Ausführung abgeschlossen. Falls DBCC Fehlermeldungen ausgegeben hat, wenden Sie sich an den Systemadministrator.

12.6 Zusammenfassung

Die heutige Lektion hat sich mit Transaktionen, verteilten Transaktionen, Sperren, Stapeln und Skripts beschäftigt.

Sie haben gesehen, wie Transaktionen (und verteilte Transaktionen) verwendet werden, um »Arbeitseinheiten« mit SQL Server zu realisieren. Sie haben die ACID-Eigenschaften einer Transaktion untersucht, um zu verstehen, warum sie für zuverlässige und dauerhafte Datenkonsistenz sorgen.

Prüfpunkte dienen dazu, periodisch Daten aus dem Arbeitsspeicher auf die Festplatte zu kopieren. Das Wiederherstellungsintervall können Sie SQL Server festlegen lassen oder selbst anpassen.

Ein weiterer Schwerpunkt dieser Lektion waren die Sperren. Dabei haben Sie gelernt, welche Sperren in SQL Server 2000 zur Verfügung stehen, wie die Syntax für Sperren aussieht, wie Sie die Entscheidungen von SQL Server hinsichtlich der zu verwendenden Sperren überschreiben können und wann dieses Vorgehen sinnvoll ist.

Außerdem wurde gezeigt, wie man Sperren mit Transact-SQL und im SQL Server Enterprise Manager beobachtet. Schließlich haben Sie eine Deadlock-Situation provoziert und gelernt, wie man derartige Blockierungen vermeiden kann.

12.7 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F Zeigt das folgende Beispiel einen gültigen Stapel?

```
BEGIN TRAN
  SELECT * FROM titles
  UPDATE titles set PRICE = $12.99 WHERE TITLE_ID = 'BU1032'
  ROLLBACK TRAN
  SELECT * FROM titles
COMMIT TRAN
```

A Nein. Wenn Sie den vorhergehenden Satz von SQL-Anweisungen ausgeführt hätten, würden Sie am Ende die folgende Fehlermeldung erhalten:

Server: Nachr.-Nr. 3902, Schweregrad 16, Status 1, Zeile 6
Die COMMIT TRANSACTION-Anforderung hat keine entsprechende BEGIN TRANSACTION.

Dies liegt daran, weil die Transaktion bereits rückgängig gemacht wurde, wenn die Programmausführung die Anweisung COMMIT TRAN erreicht.

- F** *Wahr oder Falsch? Die Anweisung COMMIT TRAN schreibt alle bearbeiteten Seiten im Speicher auf die Platte.*
- A** Falsch. Der CHECKPOINT-Prozess schreibt bearbeitete Seiten vom Speicher in Datenbankdateien im SQL Server (sowohl Datenseiten als auch Protokolldatensätze). Die Anweisung COMMIT TRAN schreibt die bearbeiteten Protokolldatensätze für jede Transaktion in die Transaktionsprotokolldateien für jede Datenbank.
- F** *Welche Art von Sperren sind kompatibel mit einer gemeinsamen Sperre?*
- A** Gemeinsame Sperren sind kompatibel mit gemeinsamen Sperren und Aktualisierungssperren. Es kommt nicht darauf an, ob sie Zeilen-, Seiten- oder Tabellensperren sind. Blöcke und Datenbanken erhalten praktisch keine gemeinsamen Sperren.
- F** *Wie nennt man es, wenn zwei Transaktionen eine exklusive Sperre auf einer Ressource haben und die jeweils andere Transaktion die Ressource sperren muss, die die andere Transaktion gesperrt hat?*
- A** Diese Situation heißt Deadlock. Man kann Deadlocks vermeiden, indem man bei getrennten Prozessen – vorzugsweise in gespeicherten Prozeduren – in der gleichen Reihenfolge auf die Tabellen zugreift.

Kontrollfragen

1. Welche CREATE-Anweisungen sind in ein und demselben Stapel erlaubt?
2. Welche Sperren sind bei Ausführung der COMMIT TRAN-Anweisung im folgenden Stapel wirksam?

```
BEGIN TRAN
    UPDATE authors SET au_lname = 'Johnson' WHERE au_lname = 'Smith'
    INSERT publishers VALUES('9991', 'SAMS', 'Indianapolis', 'IN', 'USA')
    SELECT * FROM publishers (HOLDLOCK)
COMMIT TRAN
```
3. Wie lässt sich festlegen, dass remote gespeicherte Prozeduren jederzeit automatisch Teil einer verteilten Transaktion sind?
4. Wie brechen Sie eine Transaktion ab, die Sie mit impliziten Transaktionen begonnen haben?
5. Wenn Sie eine Transaktion aus einem Trigger für *table1* heraus abbrechen, wird dann die Anweisung SELECT * FROM authors im folgenden Code ausgeführt?

```
BEGIN TRAN
INSERT table1 VALUES (3)
GO
SELECT * FROM authors
COMMIT TRAN
```

Übung

1. Sie haben Ihrem Bestand ein neues Buch hinzugefügt (*Surfen im Netz in drei leichten Schritten*). Nicht nur der Titel ist neu, sondern auch der Herausgeber (*Waycool Publishers*) und die beiden Autoren (*Ann Jackson* und *Bob Greene*). Schreiben Sie ein Skript, um alle diese Informationen hinzuzufügen. Dabei sollen entweder alle Anweisungen des Skripts erfolgreich ausgeführt werden oder insgesamt scheitern.



Indizes

In der gestrigen Lektion haben Sie die Programmerweiterungen zu SQL Server und Transact-SQL sowie Transaktionen kennen gelernt. Mit Transaktionen können Sie Datenbankänderungen logisch gruppieren und diese Änderungen als Arbeitseinheit entweder vollständig übernehmen oder vollständig verwerfen lassen. Außerdem haben Sie gelernt, wie man Sperren einsetzt, um ein relationales Datenbanksystem zu schützen und es wie ein Einzelbenutzersystem aussehen zu lassen.

Die heutige Lektion befasst sich mit Indizes. Indizes stellen einen Satz logischer Zeiger auf Ihre Daten zur Verfügung, die dem Index eines Buches ähnlich sind, mit dessen Hilfe Sie nach Einträgen in Ihrem Buch suchen können. Alle Abfragen, die Sie bisher in dieser Woche untersucht haben (SELECT, INSERT, UPDATE, DELETE), kommen zwar ohne Indizes aus, laufen aber normalerweise mit Indizes schneller.

Als Erstes untersucht diese Lektion die Frage, warum man Indizes verwendet, und geht dann auf einige Grundlagen zu B+-Bäumen ein – eine Datenstruktur, mit der Indizes in SQL Server implementiert sind. Dann geht es um verschiedene Optionen und Leistungsprobleme, die mit Indizes verbunden sind. Außerdem untersuchen Sie Indizes für Sichten und es wird erläutert, warum gerade derartige Indizes so hilfreich sind. Das Thema Indizes runden verschiedene DBCC-Anweisungen und Themen zu Wartungsproblemen ab. Zum Abschluss werfen Sie noch einen kurzen Blick auf die Volltextindizierung von SQL Server 2000.

13.1 Warum eigentlich Indizes?



Indizes verwendet man aus vielen Gründen, an erster Stelle steht aber die Geschwindigkeit. Ohne Indizes greift SQL Server auf die Daten zu, indem er jede Datenseite für jede in der SQL-Anweisung spezifizierte Tabelle liest. In manchen Fällen ist ein derartiger *Tabellenscan* (das Einlesen jeder einzelnen Datenseite einer Tabelle) eine hervorragende Methode für die Datenwiedergabe; beispielsweise wenn die Tabelle klein ist oder wenn Sie auf einen großen Teil einer Tabelle zugreifen müssen. In der Regel ist der Datenzugriff jedoch schneller, wenn man einen Index benutzt. Unter Umständen lassen sich damit auch Verknüpfungen zwischen Tabellen beschleunigen.

Ein weiterer Grund zum Erstellen eines Index ist das Erzwingen der Eindeutigkeit. Es stellt zwar keine Fehlerbedingung dar, wenn in einer Tabelle zwei identische Zeilen auftauchen, jedoch gehört das nicht zu den Prinzipien der effizienten Speicherung und widerspricht der Normalisierung. Stellen Sie sich dazu ein System vor, das Kunden speichert. Wenn Sie Ihre Kunden nicht auseinander halten können und ihnen falsche Rechnungen schicken, sind Sie Ihre Kunden höchstwahrscheinlich über kurz oder lang los. Um die Kunden eindeutig zu identifizieren, stehen Ihnen mehrere Methoden zur Verfügung.

Sie können ihnen Nummern zuweisen, ihre Namen und Geburtstage kombinieren, ihre Kreditkartennummern benutzen oder andere derartige Werte bzw. Kombinationen von Werten einsetzen. Unabhängig von der konkreten Wahl teilen Sie SQL Server Ihre Entscheidung durch einen *eindeutigen* Index mit. Auch wenn die in Lektion 14 behandelten *eindeutigen Einschränkungen* ein anderes Mittel sind, um die Eindeutigkeit zu erzwingen, erzeugt SQL Server selbst in diesem Fall einen eindeutigen Index.

Indexstrukturen



Ein Index besteht typischerweise aus einer Anzahl von Seiten, die als *B-Baum* konfiguriert sind (siehe dazu Abbildung 13.1).

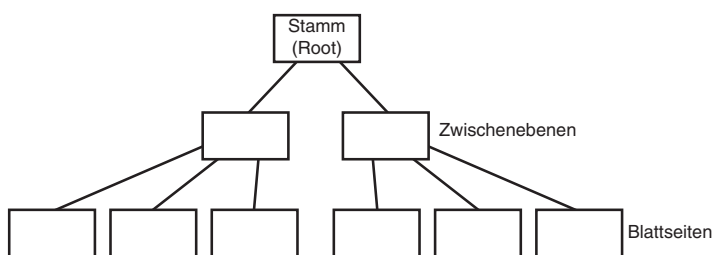


Abbildung 13.1:
Ein B-Baum

Wie schon oben erwähnt, hilft ein Index bei der schnellen Suche nach Daten. Um eine bestimmte Datenzeile zu suchen, durchlaufen Sie den B-Baum, um den Verweis auf die Datenzeile zu finden, und gehen dann zur betreffenden Datenzeile. Dabei beginnen Sie mit der Stammseite, für die ein Zeiger in der Tabelle *sysindexes* existiert (eigenartigerweise in der *root*-Spalte für *nicht gruppierte* Indizes). Die Stammseite enthält die Indexeinträge (die Daten für die Spalte bzw. Spalten, die Sie indiziert haben) und Zeiger auf jede Seite unterhalb der Stammseite. Jeder Index kann eine oder mehr dazwischenliegende Ebenen haben. Jeder Eintrag hat auch hier einen Indexwert und einen Zeiger auf die nächste weiter unten folgende Seite.



Die Einträge auf den *Blattseiten* (den niedrigsten Ebenen im Baum) hängen davon ab, ob die Tabelle einen *gruppierten Index* besitzt oder nicht. Logisch gesehen, würden Sie einen Eintrag sowohl für jede indizierte Zeile in der Tabelle als auch einen Zeiger auf die Datenseite und die Zeilennummer finden, die zur aktuellen Datenzeile gehört. Wenn die Tabelle auch einen *gruppierten Index* hat, würden alle nicht gruppierten Indizes anstelle der Datenseite und der Zeilennummer den Schlüsselwert des gruppierten Index enthalten. Gruppierte und nicht gruppierte Indizes werden weiter unten erörtert. Fürs Erste können Sie sich einen gruppierten Index etwa wie eine Methode zum

Vorsortieren der eigentlichen Daten vorstellen. Tabellen ohne einen gruppierten Index werden *Heaps* genannt, wobei nicht gruppierte Indizes getrennte Indexstrukturen darstellen, die keine direkte Sortierung der eigentlichen Daten beinhalten.

Die Daten selbst werden in so genannten Datenseiten gespeichert. Jede Seite fasst 8.192 Byte einschließlich eines Headers mit 96 Byte, so dass also jede Datenseite 8.096 verfügbare Speicherbyte für die eigentlichen Daten enthält. Alle Seiten weisen denselben grundlegenden Aufbau auf.

Abbildung 13.2 zeigt ein Beispiel für das Aussehen eines Index, der für eine Vornamenspalte eingerichtet ist.

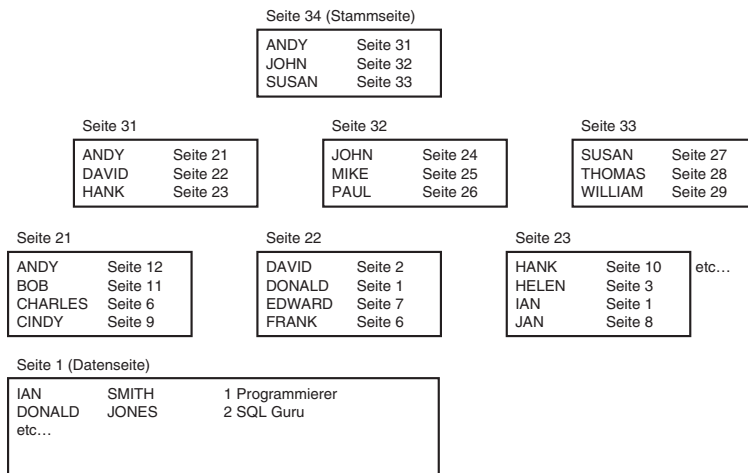


Abbildung 13.2:
Beispiel für einen B-Baum

Jede Ebene des Index stellt eine doppelt verkettete Liste dar. Jede Seite kennt ihre Vorgängerseite und diejenige Seite, die ihr logisch folgt. Auf der Ebene des Stammseite und der Zwischenebenen des Index stellt jeder Indexwert den ersten Wert auf der Seite der darunter liegenden Ebene dar. Auf der Blattebene des Index finden Sie pro Zeile der Tabelle einen Eintrag. Beachten Sie, dass der Index auf der Grundlage der Spalte (bzw. Spalten) sortiert ist, die Sie als Ihren Indexschlüssel gewählt haben. Die physikalische Anordnung der Daten wird dadurch nicht berührt.

Wenn Sie Daten in der Tabelle ändern, werden auch alle Indizes für die Tabelle geändert. SQL Server garantiert Ihnen dabei die Konsistenz zwischen den Daten in Ihrer Tabelle und deren Indizes, was insofern gut ist, als Sie dadurch eine hervorragende Datenintegrität erhalten. Im Gegenzug bedeutet es aber auch, dass die INSERT-, UPDATE- und DELETE-Operationen, die vorher vielleicht recht schnell gewesen sind, nun etwas mehr zu tun bekommen und langsamer laufen. Wenn Sie eine neue Zeile mit einer INSERT-Anweisung in die

Tabelle einfügen, benötigen Sie dazu normalerweise zwei Ein-/Ausgabe- (E/A-) Operationen, nämlich eine für die Daten und eine weitere für das Protokoll. Mit zwei Indizes in der Tabelle sind nun mindestens zwei E/A-Operationen mehr erforderlich – vielleicht sogar noch mehr als nur diese. Deshalb müssen Sie zwischen Ihren Anforderungen zur Datenänderung und schnelleren Abfragen abwägen.

Wie verwendet man einen Index? Suchen Sie dazu eine Datenzeile in Abbildung 13.2. Wir nehmen dabei an, dass der Index für Vornamen eingerichtet ist. Zum Aufsuchen der Zeile mit dem Vornamen Donald sehen Sie auf der Stammseite nach. Weil Donald »kleiner« als John ist, folgen Sie nun dem Eintrag für Andy auf die Seite 31. Auf Seite 31 stellen Sie fest, dass Donald wiederum »größer« als David, aber »kleiner« als Hank ist, so dass Sie zur Seite 22 gehen können. Auf der Seite 22 finden Sie den Eintrag für Donald, der auf die Seite 1 zeigt (die eine Datenseite ist, weil sie die Blattebene des Index darstellt). In SQL Server finden Sie außerdem noch eine Zeilennummer für die Zeile, die Donald als Vornamen enthält, die jedoch aus Gründen der Vereinfachung aus der Abbildung ferngehalten worden ist. Jetzt können Sie die Zeile auf der Seite 1 lesen und auf die Informationen über Donald zugreifen, die auf der Seite gespeichert sind. Genau auf diese Weise benutzt auch SQL Server den Index zum Aufsuchen von Daten.

Indexoptionen

Für Indizes stehen Ihnen verschiedene Optionen zur Verfügung. Bevor Sie Ihre eigenen Indizes erstellen können, müssen Sie diese Optionen festlegen können und die dahinterstehenden Konzepte verstanden haben.

Gruppierte Indizes



Es gibt zwei Optionen für die physikalische Speicherung Ihrer Indizes, von denen die erste unter der Bezeichnung »gruppiert« bekannt ist. Ein *gruppiert*er Index sortiert die Daten physikalisch neu, so dass Sie keine komplett separate Indexstruktur mehr haben (wie die oben beschriebene), sondern eine Blattebene des Index, die die Daten selbst enthält. Der Zugriff auf die Daten eines gruppierten Index ist fast immer schneller als die Verwendung eines nicht gruppierten Index, da die zusätzlichen Suchvorgänge für Datenseite/Zeile auf der Blattebene des Index wegfallen.

Da die Daten physikalisch in der Reihenfolge des Indexschlüssels sortiert sind, können Sie auch nur einen gruppierten Index für eine Tabelle haben (da Sie ja kaum mehrere Kopien der Daten speichern wollen). Weil nur ein gruppiert Index verfügbar ist, müssen Sie die-

sen auch besonders sorgfältig bestimmen. Die Bestimmung eines gruppierten Index ist zwar ein ziemlich komplexes Thema, jedoch gibt es dazu einige grundlegende Richtlinien, die hier besprochen werden sollen.

Ein Hauptproblem, das mit gruppierten Indizes in Verbindung steht, ist freier Speicherplatz. Die Erstellung eines gruppierten Index verlangt, dass mindestens 120 % der Tabellengröße als temporärer Platz zur Verfügung stehen. Dieser freie Platz muss in der Datenbank zum Zeitpunkt der Erstellung des Index zur Verfügung stehen. Zum Erstellen des Index kopiert SQL Server die Tabelle, sortiert den Inhalt in der Reihenfolge der Indexwerte (aufsteigend), baut die Indexstrukturen auf (die Stammseite und alle benötigten Zwischenseiten) und löscht anschließend die Ursprungstabelle. Nach Abschluss dieser Operation nimmt der gruppierte Index nur noch etwa 5% mehr Platz als die Tabelle selbst ein. Der Verwaltungsaufwand (die Seiten ohne Daten) dieses Indextyps ist relativ klein, hängt aber von der Größe der Werte ab, die indiziert werden.

Eigentlich klingt dies nicht schlecht, da Sie den freien Platz nur während der Erstellung des Index benötigen. Allerdings kann gerade dies schwierig einzuschätzen sein: Falls Sie eine Datenbank mit 500 MB haben, aber eine Tabelle in der Datenbank, für die Sie einen gruppierten Index anlegen wollen, allein schon 100 MB groß ist, benötigen Sie mindestens 120 MB freien Speicherplatz in der Datenbank.

Eine andere kritische Thematik in SQL Server 2000 besteht darin, dass die Schlüsselwerte (indizierte Spalten), die Sie für einen gruppierten Index festlegen, mit den nicht gruppierten Indizes »zusammengelegt« werden. Falls Sie einen breit angelegten gruppierten Index gewählt haben (beispielsweise `char(30)`), dauert es aus diesem Grund nicht nur länger, um den gruppierten Index selbst zu durchsuchen, sondern Ihre sämtlichen nicht gruppierten Indizes müssen den Wert von `char(30)` aus Ihrem gruppierten Index ebenfalls in jeder Zeile des nicht gruppierten Index mitführen. Das zieht einen beträchtlichen Verwaltungsaufwand nach sich. Deshalb sollte man gruppierte Indexschlüssel möglichst klein halten und zudem sicherstellen, dass der gruppierte Schlüssel nur selten aktualisiert wird, da andernfalls auch alle Ihre nicht gruppierten Indizes zu aktualisieren sind.

Außerdem ist die Reihenfolge zu beachten, in der Sie Ihre Indizes erstellen. Da der gruppierte Indexschlüssel ein Bestandteil der Schlüsselwerte jedes nicht gruppierten Index ist, müsste jeder nicht gruppierte Index neu aufgebaut werden. Aus diesem Grund sollten Sie Ihren gruppierten Index immer als ersten erstellen.

Abbildung 13.3 zeigt ein Beispiel eines gruppierten Index. Beachten Sie, dass die Daten in der Reihenfolge des Indexschlüssels (wieder unser Vorname) sortiert und die Daten der Blattebene des Index (die Datenseite) nach dem Vornamen sortiert sind.

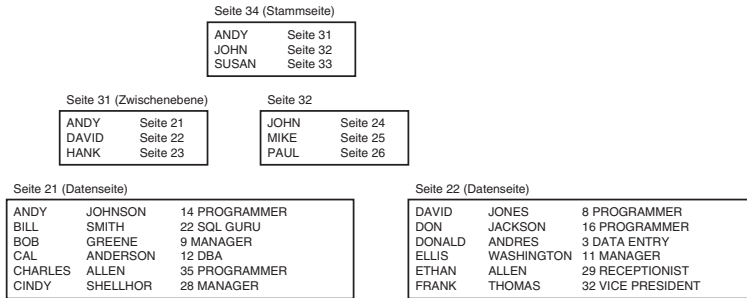


Abbildung 13.3:
Beispiel für einen
gruppierten Index

Nicht gruppierte Indizes



Ein *nicht gruppierter Index* ist grundsätzlich das Gleiche wie ein normaler B-Baumindex. Jeder Index besitzt eine Stammseite, eine oder mehrere Ebenen mit zwischenliegenden Seiten und eine Blattebene, die eine Zeile für jede Tabellenzeile enthält. Nicht gruppierte Indizes erfordern mehr Gesamtspeicherplatz als gruppierte Indizes, verlangen dafür aber weniger Platz während des Prozesses der Erstellung.

Für eine einzige Tabelle können Sie bis zu 249 nicht gruppierte Indizes anlegen. Die Reihenfolge, in der Sie sie erstellen, ist nicht von Bedeutung. Wenn Sie einen nicht gruppierten Index anlegen, wird dadurch – anders als bei einem gruppierten Index – die Reihenfolge der Daten nicht geändert. Die Zeilen in der Blattebene des Index werden nach der Reihenfolge der Spalten sortiert, die als Bestandteil des Index ausgewählt worden sind. Jede Zeile enthält einen Zeiger auf Seitennummer und Zeile der Daten in der Tabelle, sofern kein gruppierter Index vorhanden ist, oder aber den Wert des gruppierten Indexschlüssels für den Fall, dass die Tabelle noch einen gruppierten Index besitzt. Ein Beispiel eines nicht gruppierten Index finden Sie in Abbildung 13.2.

Eindeutige und nicht eindeutige Indizes

Die Eindeutigkeit legt fest, ob in Ihrem Index Duplikatwerte erlaubt sind. Im Index für den Vornamen, den Sie weiter oben gesehen haben, wäre es bei einem eindeutigen Index nicht erlaubt, dass zwei Personen denselben Vornamen tragen. Standardmäßig sind die Indizes in SQL Server nicht eindeutig, was bedeutet, dass Duplikatwerte erlaubt sind.

Falls Ihre Daten dies entsprechend unterstützen, kann die Leistung durch Erstellung und Verwendung eines Index bedeutend erhöht werden. Wenn der Wert, nach dem Sie suchen, gefunden wurde, ist kein weiteres Durchsuchen des Index mehr erforderlich (da Sie wissen, dass es nur einen Eintrag für den Wert gibt, können Sie nach dem Auffinden dieses Wertes die weitere Suche einstellen).

Gruppierte Indizes sind besonders gute Kandidaten für eindeutige Indizes, da SQL Server intern ohnehin immer die Eindeutigkeit gruppierter Indizes erzwingt. Falls Sie keinen eindeutigen gruppierten Index erzeugt haben, sorgt SQL Server für einen zusätzlichen versteckten Schlüssel, der die Eindeutigkeit des Index erzwingt. Wozu also SQL Server dies machen lassen, wenn Sie bereits einen geeigneten Schlüsselwert haben, der eindeutig ist?

Einspaltige und mehrspaltige Indizes



Viele Indizes bestehen aus nur einer Spalte. Die Erstellung mehrspaltiger Indizes ist jedoch auch kein Problem. Mehrspaltige Indizes können sehr nützlich sein, da Sie damit die Anzahl der von SQL Server benutzten Indizes reduzieren und eine bessere Leistung erzielen können. Falls Sie in Abfragen zwei Spalten häufig zusammen verwenden, sind diese hervorragende Kandidaten für einen so genannten *zusammengesetzten Index* (was nur ein anderer Name für einen Index mit mehreren Spalten ist). Zusammengesetzte Indizes können gruppiert oder nicht gruppiert sein. Sie können aus 2 bis 16 Spalten zusammengesetzt und bis zu 900 Byte breit sein.

Ein zu breiter Index ist allerdings uneffektiv, da es dann schneller sein kann, anstelle des Index gleich die ganze Tabelle zu durchsuchen. Indizes schließen ganz allgemein eine Reihe von Kompromissen ein und führen recht selten zu einer eindeutigen Entscheidung, die für alle Anwendungszwecke geeignet ist.

Aufsteigende / absteigende Indizes

In allen vorherigen Versionen von SQL Server ist man immer von aufsteigenden Indizes ausgegangen (wie man sich Indizes im Allgemeinen auch vorstellt). Sortiert man also das Alphabet, kommt A zuerst und Z zuletzt. Manchmal muss man aber eine Liste in absteigender Reihenfolge sortieren (von Z nach A). SQL Server unterstützt nun auch absteigende Indizes.

Abbildung 13.4 zeigt einen nicht gruppierten Index für den Vornamen (den gleichen Index wie in Abbildung 13.2), wobei aber der Index jetzt in absteigender Reihenfolge gespeichert ist. Die Zeilen finden Sie nach der gleichen Methode wie bisher, wobei Sie bei der Stammseite beginnen und durch den Index navigieren. Hier haben Sie aber den Vorteil, dass keine zusätzliche Sortierung erforderlich ist, wenn die Abfrage die Daten in absteigender Reihenfolge zurückgeben soll.

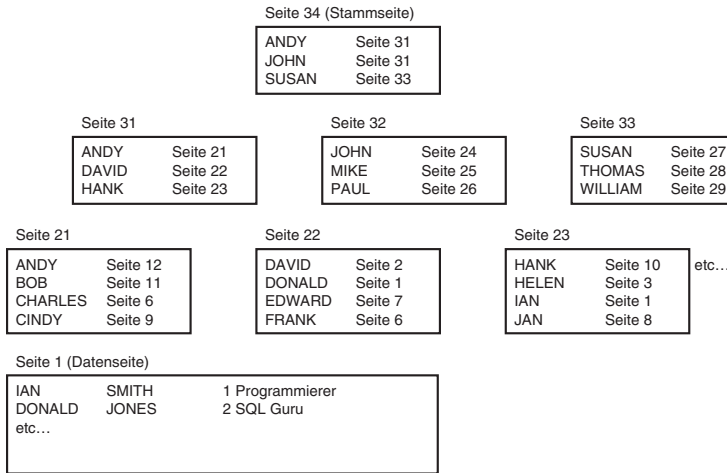


Abbildung 13.4:
Ein absteigender
nicht gruppierter
Index

13.2 Indizes in SQL Server

SQL Server implementiert B-Bäume, um seine Indizes zu erstellen. Zum Erstellen Ihrer Indizes verwenden Sie die Anweisung `CREATE INDEX`:



```
CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX Indexname
ON [Besitzer.][Tabellenname | Sichtname]
(Spaltenname [ASC|DESC][,...n])
[WITH
    [PAD_INDEX][[,] FILLFACTOR = x]
    [[,] IGNORE_DUP_KEY]
    [[,] DROP_EXISTING]
    [[,] STATISTICS_NORECOMPUTE]]
    [[,] SORT_IN_TEMPDB]]
ON Dateigruppe]
```

Die Elemente in dieser Syntax haben folgende Bedeutung:

- **UNIQUE** gibt an, dass für diesen Index keine Duplikate erlaubt sind. Der Standardwert lautet auf nicht eindeutige und Duplikatwerte.
- **CLUSTERED** legt fest, dass die Daten selbst physikalisch sortiert und zur Blattebene des Index werden. Gruppierte Indizes müssen immer eindeutig sein. Falls Sie einen gruppierten **UNIQUE**-Index erstellen, gibt es kein Problem. Falls Sie jedoch einen nicht eindeutigen gruppierten Index erstellen, werden jedem gruppierten Indexschlüssel vier Byte hinzugefügt, um die Eindeutigkeit zu gewährleisten.

- *NONCLUSTERED* legt fest, dass ein normaler B-Index als ein vollständig separates Objekt erstellt wird. Dies ist der standardmäßig vorgegebene Indextyp.
- *Indexname* ist der eindeutige SQL Server-Name für dieses Objekt.
- *Tabellenname* ist der Name der Tabelle, die die zu indizierenden Spalten enthält.
- *Sichtname* ist der Name der Sicht, die die zu indizierenden Spalten enthält.
- *Spaltenname* ist der Name der Spalte (bzw. Spalten), die indiziert werden soll(en). Sie können einen Index mit bis zu 16 Spalten mit einer Gesamtlänge von 900 Byte erzeugen. Die Spalten dürfen nicht vom Typ *text*, *image*, *bit* oder *ntext* sein.
- *ASC | DESC* gibt an, ob die Spalte in aufsteigender (*ASC*) Reihenfolge (zum Beispiel von A bis Z) oder in absteigender (*DESC*) Reihenfolge (zum Beispiel von Z bis A) zu sortieren ist. Standard ist aufsteigend, was mit vorherigen Versionen von SQL Server konsistent ist, die keine absteigende Option hatten.
- *Dateigruppe* ist der Name der Dateigruppe, in der der Index erstellt werden soll. Wenn dieses Argument nicht angegeben ist, wird der Index in der standardmäßigen Dateigruppe erstellt.

Zum Beispiel erstellen die folgenden Codezeilen eine Tabelle namens *myauthors* in der Datenbank *pubs* und kopieren anschließend sämtliche Daten aus der Tabelle *authors*. Danach erstellt der Code einen Index auf der Spalte *au_id* in der Tabelle *myauthors*. Der Index ist gruppiert und erzwingt die Eindeutigkeit:

```
USE PUBS
-- Die Tabelle erstellen:
CREATE TABLE dbo.myauthors (
    au_id id NOT NULL ,
    au_lname varchar (40) NOT NULL ,
    au_fname varchar (20) NOT NULL ,
    phone char (12) NOT NULL ,
    address varchar (40) NULL ,
    city varchar (20) NULL ,
    state char (2) NULL ,
    zip char (5) NULL ,
    contract bit NOT NULL
)
-- Die Daten von authors nach myauthors kopieren:
INSERT myauthors select * from authors
-- Den eindeutigen gruppierten Index für myauthors erstellen:
Create unique clustered index myauind on myauthors (au_id)
```


Der folgende Code erstellt einen nicht eindeutigen, nicht gruppierten und absteigenden Index für die Spalte *au_fname* derselben Tabelle:

```
Use pubs
Create index mynamindex on myauthors (au_fname DESC)
```

Der folgende Transact-SQL-Befehl ist funktionell identisch und eventuell verständlicher:

```
Use pubs
Create nonclustered index mynameindex on myauthors (au_fname)
```

Die Optionen fillfactor und pad_index

Die Option *fillfactor* gibt an, wie weit jede Seite in der Blattebene eines Index gefüllt werden soll. Der standardmäßige Füllfaktor ist 0. Da *fillfactor* ein Konfigurationsparameter ist, sollten Sie sich vergewissern, dass er sich nicht geändert hat. Denken Sie daran, dass Sie diese Überprüfung entweder über den SQL Server Enterprise Manager oder mit der gespeicherten Prozedur *sp_configure* ohne Parameter vornehmen können. Die Ausgabe der Prozedur sieht etwa wie folgt aus:



```
Exec sp_configure
go
```



name	minimum	maximum	config_value	run_value
allow updates	0	1	0	0
default language	0	9999	0	0
fill factor (%)	0	100	0	0
language in cache	3	100	3	3
max async IO	1	255	32	32
max text repl size (B)	0	2147483647	65536	65536
max worker threads	10	1024	255	255
nested triggers	0	1	1	1
network packet size (B)	4096	65535	4096	4096
recovery interval (min)	0	32767	0	0
remote access	0	1	1	1
remote proc trans	0	1	0	0
show advanced options	0	1	0	0
user options	0	4095	0	0

(14 row(s) affected) ...

Suchen Sie in der Ausgabe nach der Bezeichnung »fill factor« und überprüfen Sie, dass die Werte der Spalten *config_value* und *run_value* beide den Wert 0 enthalten.

Wenn Sie zusammen mit Ihrer Anweisung `CREATE INDEX` keinen Füllfaktor angeben, wird der Standardwert (typischerweise 0) verwendet. Ein Wert 0 bedeutet, dass die Blattseiten Ihres Index nahezu vollständig gefüllt werden, während die Seiten, die keine Blätter sind (Zwischenebenenseiten und die Stammseite), immer noch Platz für mindestens zwei weitere Zeilen haben. Falls der Füllfaktor gleich 100 ist, werden sämtliche Blattseiten zu 100% gefüllt, wobei kein Platz für zusätzliche Zeilen gelassen wird. Die Stamm- und die Zwischenebenenseiten werden aber nach wie vor noch Platz für zusätzliche Zeilen haben. Alle anderen Werte geben den Prozentsatz an, mit dem jede Blattseite mit Zeilen gefüllt wird. SQL Server rundet dabei den Prozentsatz auf die nächstmögliche Zeilengröße ab, so dass Sie selten den exakten gewünschten Prozentsatz erhalten werden, obschon SQL Server dafür sorgt, dass er so nahe wie möglich an diesen Wert heranreicht.

Falls Sie einen gruppierten Index mit einem Füllfaktor von 50 angegeben haben, wird jede Seite zu etwa 50% gefüllt. In Abbildung 13.5 können Sie erkennen, dass die Blattseite des gruppierten Index nur zur Hälfte gefüllt worden ist. Der dazugehörige Code kann etwa so lauten:

```
CREATE INDEX aurnameindex on authors (au_fname)
WITH FILLFACTOR = 50
GO
```

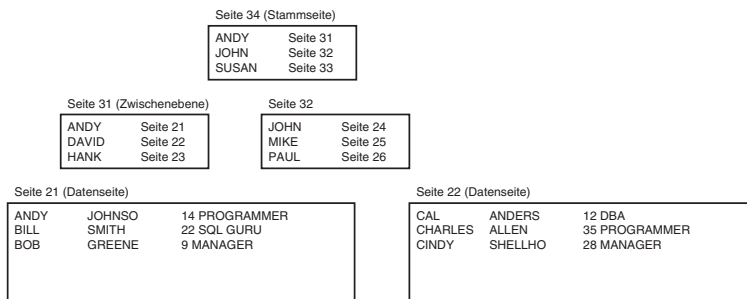


Abbildung 13.5:
Ein gruppiertes
Index mit einem
Füllfaktor von 50



Beachten Sie, dass `fillfactor` nicht streng auf den Index angewendet wird. Falls Sie einen Index mit einem Füllfaktor von 50 spezifizieren (d.h., jede Seite halb voll ist), wird es sich wahrscheinlich im Laufe der Zeit so ergeben, dass manche Seiten gefüllt und andere nahezu leer sein werden. Wenn Sie Ihren Füllfaktor wieder in Kraft treten lassen möchten, können Sie dies nach einer der folgenden drei Methoden erreichen:

- ▶ Sie löschen den Index und erstellen ihn zusammen mit dem erneut angegebenen Füllfaktor neu.
- ▶ Sie führen den Befehl `DBCC DBREINDEX` aus.
- ▶ Sie führen den Befehl `DBCC INDEXDEFRAG` aus.

Jede dieser Möglichkeiten wird Ihren Füllfaktor wieder in Kraft setzen. Das Werkzeug DBCC SHOWCONTIG gibt Ihnen eine Entscheidungshilfe dafür an die Hand, wann Sie neu indizieren müssen (wird weiter unten in dieser Lektion erörtert). SQL Server nimmt aber auch automatisch eine gewisse Aufräumarbeit vor.

Wenn Sie die Option `pad_index` zusammen mit einem Füllfaktor angeben, dann legt diese Option fest, dass der Füllfaktor nicht nur auf die Blattseiten, sondern auch auf die Seiten des Index, die keine Blattseiten sind, angewendet wird. Die Option `pad_index` lässt sich am Einfachsten mit einem Beispiel verdeutlichen.

Falls Sie denselben Index wie in Abbildung 13.5 erstellen, jedoch die Option `pad_index` hinzufügen, wird nun auch auf die Seiten, die keine Blattseiten sind, der Füllfaktor angewendet. Wenn Ihre Zeilen nicht genau passen, versucht SQL Server, so nahe wie möglich an den von Ihnen festgelegten Füllfaktor heranzukommen. Der folgende Code könnte beispielsweise einen Index wie den in Abbildung 13.6 erzeugen:

```
CREATE INDEX aurnameindex on authors (au_fname)
WITH FILLFACTOR = 50, PAD_INDEX
GO
```

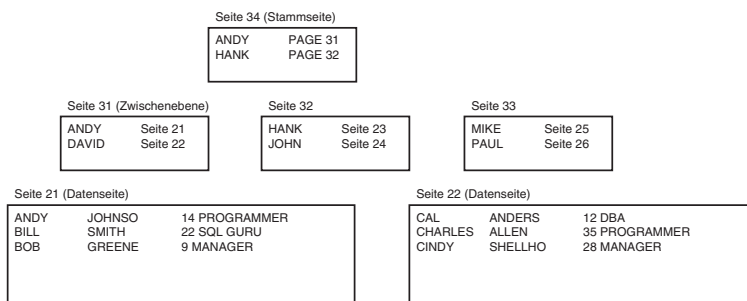


Abbildung 13.6:
Ein gruppierter
Index mit `pad_index`



Jetzt wissen Sie zwar, wie `fillfactor` und `pad_index` funktionieren, aber noch nicht, warum Sie diese Optionen überhaupt verwenden sollen. Die Festlegung eines Füllfaktors kann an vielen Stellen sinnvoll sein. Um dies zu verstehen, sollten Sie sich zunächst mit einer Anzahl von Begriffen vertraut machen. Der erste ist *Seitenteilung*. Wenn eine Seite von SQL Server gefüllt ist und eine weitere Zeile auf dieser Seite untergebracht werden soll, entsteht eine Seitenteilung. Dabei wird dem Index oder der Tabelle eine neue Seite zugewiesen und 50% der Zeilen werden auf die neue Seite verschoben. Die neue Zeile wird dann anschließend an der passenden Stelle untergebracht. Wie Sie sich vorstellen können, kann dies eine aufwendige Operation werden, sofern die Seitenteilung häufiger auftritt. Falls Sie beim Erstellen Ihres Index einen Füllfaktor verwendet haben, können Sie damit bewirken, dass neue Zeilen hinzugefügt

werden, ohne gleichzeitig Seitenteilungen auszulösen. Die Wahl eines geeigneten Füllfaktors hat daher möglicherweise Auswirkungen auf die Systemleistung, obschon der Wert für den besten Füllfaktor nur aus der Erfahrung gewonnen werden kann.

Für gruppierte Indizes kann sich dies als besonders nützlich erweisen, da die Blattebene des Index aus den eigentlichen Datenseiten besteht. Seien Sie sich jedoch dessen bewusst, dass die Anwendung eines Füllfaktors von beispielsweise 50% auf einen gruppierten Index schätzungsweise zu einer Verdoppelung der Tabellengröße führen wird, was eine dramatische Auswirkung auf den Speicherplatz haben kann und daher nur mit Bedacht vorgenommen werden sollte.



Aufgrund der Speichermechanismen nicht gruppiert Indizes können Seitenteilungen bei *Heaps* (Tabellen ohne einen gruppierten Index) sehr kostenintensiv sein. Jede Zeile in einem nicht gruppierten Index hat einen Zeiger auf Seitennummer und Zeilennummer der Datenzeile. Findet nun eine Seitenteilung im Heap statt, muss nahezu die Hälfte der Zeilen verschoben werden, was zur Folge hat, dass bei der Hälfte der Zeilen jeder Eintrag für jeden nicht gruppierten Index geändert werden muss. Während all diese Operationen vorgenommen werden, müssen Sie auf das Ergebnis Ihrer Einfüge- oder Aktualisierungsoperationen warten – die Reaktionszeit verlängert sich also. Aus diesem Grund wäre wahrscheinlich bei den meisten Tabellen ein gruppierter Index empfehlenswerter. Wie Sie schon weiter oben erfahren haben, verweist der nicht gruppierte Index nicht auf die physikalische Position der Zeilen, wenn gleichzeitig ein gruppierter Index vorhanden ist – Seitenteilungen in einem gruppierten Index betreffen die nicht gruppierten Indizes nicht.

Auch durch die Option `pad_index` können Sie Teilungen von Seiten, die keine Blattseiten sind, vermeiden. Der Speicherplatzbedarf wird hierdurch zwar vergrößert, aber nicht so stark wie bei der Option `fillfactor`.

Die Option `DROP_EXISTING`

Die Option `DROP_EXISTING` bewirkt, dass ein Index gelöscht und neu erstellt wird (im Wesentlichen ist das wie eine »Reorganisation« des Index). Vielleicht denken Sie beim Löschen und Neuerstellen eines gruppierten Index daran, dass auch die nicht gruppierten Indizes in die Aktion eingeschlossen sind. Allerdings ist SQL Server 2000 intelligent genug, um wirklich nur den gruppierten Index zu löschen und neu aufzubauen. Da die nicht gruppierten Indizes als Zeilenidentifikation den Schlüsselwert des gruppierten Index benutzen und dieser Wert durch eine `CREATE INDEX`-Anweisung mit der Klausel `DROP_EXISTING` nicht verändert wird, gibt es keine Veranlassung zur Änderung von nicht gruppierten Indizes.

Diese Option ist zwar in erster Linie für gruppierte Indizes sinnvoll, kann aber auch auf nicht gruppierte Indizes angewendet werden. Die Option `DBCC DBREINDEX`, die diese Lektion später erläutert, besitzt zwar eine ähnliche Funktion, jedoch stellt die Option `DROP_EXISTING` langfristig wohl die bessere Wahl dar. Allerdings hat Microsoft mit SQL Server 2000 ein Dienstprogramm für die Onlinereorganisation des Index eingeführt: `DBCC INDEXDEFRAG`. Darauf geht diese Lektion später noch ein.

Die Option `STATISTICS_NORECOMPUTE`

SQL Server 2000 berechnet die Statistiken über Indizes automatisch nach Bedarf neu; diese Funktion können Sie über die Option `STATISTICS_NORECOMPUTE` abschalten. Um diese Funktion erneut zu aktivieren, führen Sie `UPDATE STATISTICS` ohne `NORECOMPUTE` aus. Weitere Informationen über `UPDATE STATISTICS` erhalten Sie später in der heutigen Lektion. Das Abschalten der automatischen Statistiken für Ihre Indizes ist im Allgemeinen nicht sinnvoll, da der SQL Server-Abfrageoptimierer von genauen Statistiken zur Optimierung Ihrer Abfragen abhängig ist.

Die Option `IGNORE_DUP_KEY`

Die Option `IGNORE_DUP_KEY` legt fest, dass doppelt vorkommende Zeilen bei einer mehrzeiligen Aktualisierung in einer Tabelle mit einem eindeutigen gruppierten Index stillschweigend und ohne einen Fehler bei der Ausführung übergangen werden. SQL Server gibt aber trotzdem noch eine Warnung aus. Das folgende Codebeispiel veranschaulicht, was bei dieser Aktion passiert:

```
create table t1 (col1 int not null, col2 char(5) not null)
go
create unique clustered index myind on t1 (col1) with ignore_dup_key
go
create table t2 (col1 int not null, col2 char(5) not null)
go
insert t2 values (1, 'abcde')
insert t2 values (2, 'abcde')
insert t2 values (2, 'abcde')
insert t2 values (3, 'abcde')

insert t1 select * from t2
```

Nachdem Sie diesen Code ausgeführt haben, erhalten Sie die folgende Meldung:



Server: Nachr.-Nr. 3604, Schweregrad 16, Status 1, Zeile 6
Doppelter Schlüssel wurde ignoriert.

Wenn Sie eine `SELECT`-Anweisung für die Tabelle `tl` ausführen, erhalten Sie erwartungsgemäß drei Zeilen.

13.3 Die Indexnutzung

Bis jetzt haben Sie die Grundlagen von Indizes kennen gelernt und erfahren, wie Sie SQL Server-Indizes erstellen. Allerdings haben Sie noch nichts darüber gehört, welche Indizes Sie erzeugen bzw. wann Sie sie am besten einsetzen können.

Wann wird mein Index benutzt?

Starten Sie an dieser Stelle einmal mit der nahe liegenden Frage: »Woher weiß ich, wann ein Index benutzt wird?« Sobald Sie eine Abfrage an SQL Server schicken, wird die Abfrage zerlegt und analysiert. Dabei stellt hinsichtlich der Indexoptimierung die `WHERE`-Klausel den wichtigsten Teil der Abfrage dar. Denn an einer Anweisung in der `WHERE`-Klausel kann SQL Server am besten erkennen, wie die Abfrage für die Verwendung eines Index zu optimieren ist. Nur weil Ihre Abfrage eine `WHERE`-Klausel enthält, heißt dies aber noch nicht, dass der Index auch benutzt wird. Sehen Sie sich als Beispiel einmal folgende Abfrage an:

```
SELECT au_id, au_fname, au_lname
FROM pubs..authors
WHERE state = 'CA'
GO
```

Wenn es einen Index für die Spalte Staat (*state*) gibt, die meisten Autoren (*authors*) jedoch aus Kalifornien stammen, gibt es keine Notwendigkeit, dafür einen Index zu verwenden. Ein Tabellenscan (d. h., ein Einlesen jeder einzelnen Seite der Tabelle) dürfte wahrscheinlich der effizienteste Weg dafür sein, die Daten zu bekommen. »Effizient« bedeutet an dieser Stelle Minimierung der Anzahl der von SQL Server einzulesenden Seiten. Eine andere Ausdrucksweise hierfür wäre die *Minimierung der E/A-Operationen* und der *logischen Seitenlesevorgänge*.

Eine Abfrage, die nur eine einzige Zeile zurückliefert, würde die Verwendung eines Index sinnvoll machen. Falls es beispielsweise einen gruppierten Index für die Spalte `au_id` gibt, dann würde die folgende Abfrage mit an Sicherheit grenzender Wahrscheinlichkeit einen Index benutzen:

```
SELECT au_id, au_fname, au_lname
FROM pubs..authors
WHERE au_id = "341-22-1782"
```

Wie kann ich die Indexnutzung überprüfen?

Die nächste Frage besteht darin, wie Sie überprüfen können, welcher Index in Ihrer Abfrage tatsächlich benutzt worden ist. SQL Server 2000 enthält zu diesem Zweck eine Reihe von Optionen, mit denen die Indizes, die zur Unterstützung Ihrer Abfragen verwendet werden, untersucht werden. Sie können dazu beispielsweise die Transact-SQL-Anweisungen `SET SHOWPLAN_ALL ON` oder `SET SHOWPLAN_TEXT ON` benutzen, die Sie auch bei Verwendung des SQL Servers Query Analyzers für die Ausführungspläne einsetzen können. Bei Verwendung eines Befehlszeilenprogramms wie `isql.exe` oder `osql.exe` jedoch nehmen Sie besser die Anweisungsoption `SHOWPLAN_TEXT`. Damit Sie diesen Unterschied erkennen, führen Sie den Code von Listing 13.1 aus, um eine Tabelle zu erstellen und einige Daten einzufügen. Das Skript muss deshalb ein paar Daten eingeben, weil die Indizes sonst nicht benutzt werden.



Beachten Sie, dass die SET-Optionen für `showplan` in einem eigenen Stapel stehen müssen.

Listing 13.1: Überwachen von Indizes

```
USE PUBS
CREATE TABLE PUBS..INDEXTAB
(col1 int not null,
 col2 varchar(250) not null,
 col3 varchar(250) not null,
 col4 varchar(250) not null,
 col5 varchar(250) not null)

insert indextab values (1,'adam','col3','col4','col5')
insert indextab values (2,'bob','col3','col4','col5')
insert indextab values (3,'charles','col3','col4','col5')
insert indextab values (4,'david','col3','col4','col5')
insert indextab values (5,'edward','col3','col4','col5')
insert indextab values (6,'frank','col3','col4','col5')
insert indextab values (7,'george','col3','col4','col5')
insert indextab values (8,'hank','col3','col4','col5')
insert indextab values (9,'ida','col3','col4','col5')
insert indextab values (10,'john','col3','col4','col5')
insert indextab values (11,'kim','col3','col4','col5')
insert indextab values (12,'loni','col3','col4','col5')
insert indextab values (13,'mike','col3','col4','col5')
insert indextab values (14,'nikki','col3','col4','col5')
insert indextab values (15,'oprah','col3','col4','col5')
insert indextab values (16,'paul','col3','col4','col5')
```

```
insert indextab values (17,'quan','col3','col4','col5')
insert indextab values (18,'richard','col3','col4','col5')
insert indextab values (19,'sam','col3','col4','col5')
insert indextab values (20,'tom','col3','col4','col5')
insert indextab values (21,'uma','col3','col4','col5')
insert indextab values (22,'vera','col3','col4','col5')
insert indextab values (23,'walter','col3','col4','col5')
insert indextab values (24,'xray','col3','col4','col5')
insert indextab values (25,'yuma','col3','col4','col5')
insert indextab values (26,'zane','col3','col4','col5')
insert indextab values (27,'ann','col3','col4','col5')
insert indextab values (28,'bill','col3','col4','col5')
insert indextab values (29,'cathy','col3','col4','col5')
insert indextab values (30,'dawn','col3','col4','col5')
insert indextab values (31,'ellen','col3','col4','col5')
insert indextab values (32,'fran','col3','col4','col5')
insert indextab values (33,'grant','col3','col4','col5')
insert indextab values (34,'helen','col3','col4','col5')
insert indextab values (35,'irwin','col3','col4','col5')
insert indextab values (36,'jack','col3','col4','col5')
insert indextab values (37,'kathy','col3','col4','col5')
insert indextab values (38,'lance','col3','col4','col5')
insert indextab values (39,'molly','col3','col4','col5')
insert indextab values (40,'nancy','col3','col4','col5')
CREATE CLUSTERED INDEX CL_MYINDEX on indextab (col1)
```



Es empfiehlt sich, den Anzeigemodus von SQL Query Analyzer auf *Ergebnisse in Text* einzustellen, um die nächsten Anweisungen leichter interpretieren zu können. Im Standardmodus *Ergebnisse in Raster* ist es schwierig, die Ausgaben zu lesen.

Starten Sie nun als Erstes mit der Option SHOWPLAN_TEXT:



```
USE PUBS
GO
SET SHOWPLAN_TEXT ON
GO
select col1, col2 from indextab
where col2 = 'ann'
```


Die von diesen Anweisungen generierte Ausgabe sieht etwa folgendermaßen aus:



StmtText

```
-----
select col1, col2 from indextab
where col2 = 'ann'
```

(1 row(s) affected)

StmtText

```
-----
|--Clustered Index
Scan(OBJECT:([pubs].[dbo].[INDEXTAB].[CL_MYINDEX]),
WHERE:([INDEXTAB].[col2]=[@1]))
```

(1 row(s) affected)

Wie Sie an der Ausgabe des obigen Ausführungsplanes erkennen können, hat SQL Server zur Ausführung dieser Abfrage eine Suche über den gruppierten Index gewählt. Untersuchen Sie diese Abfrage als Nächstes mit der Option `SHOWPLAN_ALL`:



```
USE PUBS
GO
SET SHOWPLAN_ALL ON
GO
select col1, col2 from indextab
where col2 = 'ann'
```

Als Ergebnis erhalten Sie eine Ausgabe wie in Abbildung 13.7 dargestellt.

Beachten Sie, dass es sich hier um die gleichen Informationen wie bei der Option `SHOWPLAN_TEXT` handelt, allerdings mit zusätzlichen Details. Am besten bleiben Sie zunächst bei der Option `SHOWPLAN_TEXT`, solange Sie noch kein Experte im Optimieren von Abfragen geworden sind und die Bedeutung von Optionen wie `TotalSubtreeCost` noch nicht ergründet haben.

Wenn Sie sehen möchten, welche Indizes für die Tabelle vorhanden sind, können Sie die gespeicherte Systemprozedur `sp_helpindex` ausführen:



```
-- Zuerst alle SHOWPLAN-Optionen deaktivieren:
SET SHOWPLAN_ALL OFF
GO
EXEC sp_helpindex indextab
GO
```

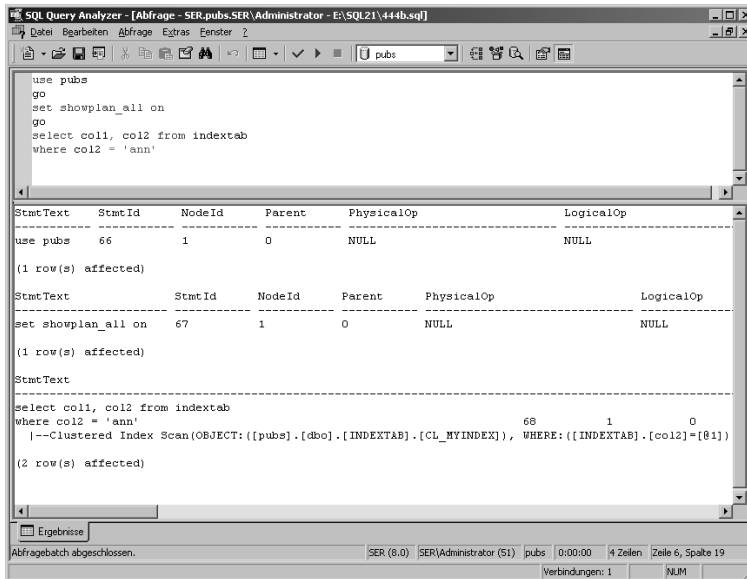


Abbildung 13.7:
Die Option
SHOWPLAN_ALL
im SQL Query
Analyzer



index_name	index_description	index_keys
CL_MYINDEX	clustered located on PRIMARY	col1

(1 row(s) affected)

Erstellen Sie nun einen zweiten Index für die Tabelle, damit Sie einen Ausführungsplan für einen Index untersuchen können, der nicht gruppiert ist.



```
CREATE INDEX NONCL_MYINDEX on indextab (col2)
GO
SET SHOWPLAN_TEXT ON
GO
select col1, col2 from indextab
where col2 = 'ann'
```



```
StmtText
-----
select col1, col2 from indextab
where col2 = 'ann'
```

(1 row(s) affected)

```
StmtText
-----
```

```
--Index Seek(OBJECT:([pubs].[dbo].[INDEXTAB].[NONCL_MYINDEX]),
SEEK:([INDEXTAB].[col2]=[@1]) ORDERED FORWARD)
```

(1 row(s) affected)

Beachten Sie, dass SQL Server jetzt eine »index seek« (Indexsuche) für den Index noncl_myindex durchgeführt hat. Das bedeutet einfach, dass nun der Index zum Auffinden sämtlicher Zeilen für ann verwendet wird.

Sie können die Pläne auch über die grafischen Ausführungsplan-Optionen des SQL Server Query Analyzers verfolgen. Es stehen zwei Pläne zur Verfügung: Der geschätzte Ausführungsplan und der tatsächliche Ausführungsplan bei Ablauf der Abfrage. Der Unterschied besteht darin, dass der tatsächliche Ausführungsplan exakte Werte enthält, während der geschätzte Plan nur zeigt, was der SQL Server-Abfrageoptimierer als Endergebnis vorausagt. Führen Sie nun die gleiche Abfrage wie vorher aus, wobei Sie jetzt jedoch die Option AUSFÜHRUNGSPLAN ANZEIGEN einschalten (entweder im Menü ABFRAGE oder durch Aktivieren der Option über die Schaltfläche AUSFÜHRUNGSMODUS). Klicken Sie im Ergebnisfenster auf das Register Ausführungsplan. Wenn Sie den Mauszeiger auf ein Element der Grafik setzen, erscheinen Detailinformationen zum betreffenden Abfrageschritt (siehe Abbildung 13.8). Die hier gezeigte Ausgabe ist relativ einfach; bei sehr großen Ausführungsplänen ist die grafische Darstellung jedoch wesentlich übersichtlicher als die textorientierte Ausgabe.

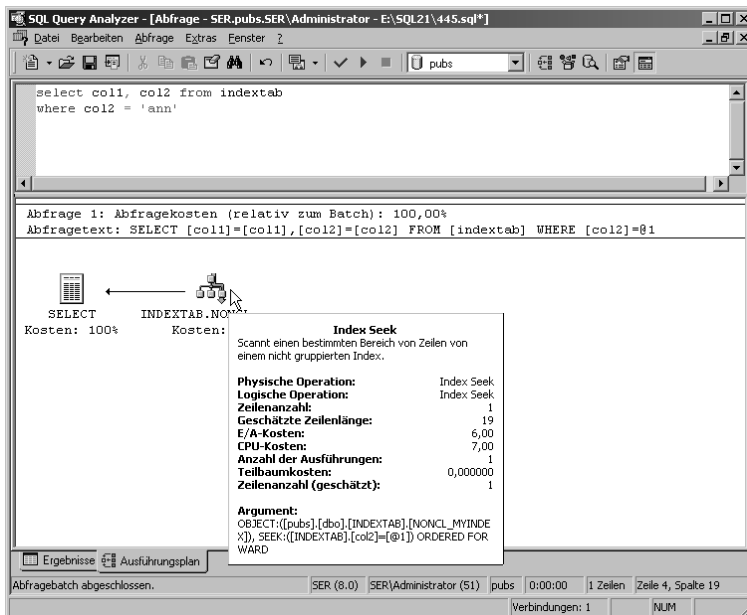


Abbildung 13.8:
Die grafische Darstellung des Ausführungsplans in SQL Query Analyzer

Was Sie bis jetzt erkennen konnten, war die Wahl, die SQL Server getroffen hat. Ein weiterer Teil des Puzzles, das Sie zusammensetzen müssen, besteht darin, wie SQL Server seine Entscheidung trifft. Die Optimierung erfolgt meistens auf der Basis, wie viel Arbeit für die Beantwortung der Abfrage erforderlich ist. Da SQL Server Ihre Abfrage so schnell wie möglich beantworten will, versucht er natürlich, mit so wenig Aufwand wie möglich auszukommen. Anders ausgedrückt heißt das in der Regel, dass er die Anzahl der logischen E/A-Operationen minimiert.



Logische E/A-Operationen sind die Datenzugriffe durch SQL Server. Jede logische E/A-Operation bedeutet, dass SQL Server eine einzelne 8 KB große Seite lesen muss. Die Gesamtanzahl der Seiten, auf die er zugreift, ist immer gleich der Anzahl der logischen E/A-Operationen. SQL Server unterscheidet dabei zwischen logischen und physischen E/A-Operationen. Falls eine Abfrage 50 logische und nur 20 physische E/A-Operationen benötigt, bedeutet dies, dass sich 30 von den 50 Seiten bereits im Hauptspeicher befinden und sofort gelesen werden können (daher der Begriff *physische E/A-Operation*). SQL Server versucht, den Umfang der physischen E/A-Operationen zu minimieren. Wie viele E/A-Operationen eine Abfrage benötigt, können Sie mithilfe der Anweisung `SET STATISTICS IO ON` ermitteln:



```
SET STATISTICS IO ON
GO
```

```
SELECT col1,col2
FROM indextab
where col1 < 3
GO
```



col1	col2
1	adam
2	bob

(2 row(s) affected)

INDEXTAB-Tabelle. Scananzahl 1, logische Lesevorgänge 2, physische Lesevorgänge 0, Read-Ahead-Lesevorgänge 0.



Vergewissern Sie sich, dass vor Verwendung der statistischen E/A-Anweisung die Option `NOEXEC` abgeschaltet ist (sofern Sie sie eingeschaltet hatten), da Sie nur dann statistische E/A-Informationen erhalten, wenn Sie auch tatsächlich eine Abfrage laufen lassen. Weil die `SHOWPLAN`-Optionen eine Abfrage nicht wirklich ausführen, sind sie auch nicht mit den `STATISTICS IO`-Informationen kompatibel.

Beachten Sie, dass zur Beantwortung dieser Abfrage zwei logische Lesevorgänge erforderlich waren: Einer zum Einlesen der Stammseite des gruppierten Index und ein weiterer zum Einlesen der Blattseite des Index. Denken Sie daran, dass die Blattseite die eigentliche Datenseite ist, da es sich um einen gruppierten Index handelt. SQL Server bevorzugt normalerweise immer einen gruppierten Index.

In Abbildung 13.9 können Sie diese beiden logischen Lesevorgänge erkennen. SQL Server muss zunächst die Stammseite des Index und anschließend die Blattebene des Index einlesen, um die Daten zurückliefern zu können. Beachten Sie, dass dieser Index keine Zwischenebenenseiten hat (weil er zu klein ist).

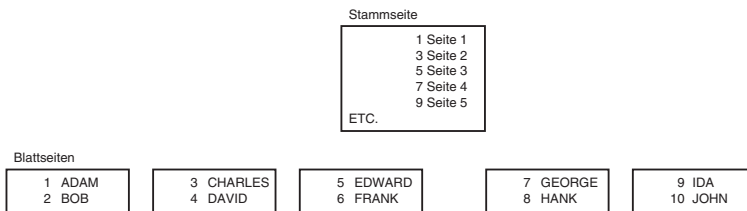


Abbildung 13.9:
Der gruppierte
Index

Der SQL Server greift außerdem noch auf Informationen über den verfügbaren Arbeitsspeicher, die zur Verarbeitung jeder Abfrageoption nötige CPU-Zeit und die Auslastung des Systems zurück. Aus diesem Grund wird SQL Server vielleicht einen Plan wählen, der sogar mehr E/A-Operationen als ein anderer erforderlich macht, weil er letztendlich bei Abwägung aller Einflussfaktoren weniger Gesamtkosten verursacht. Der SQL Server-Abfrageoptimierer ist gegenüber den Versionen vor SQL Server 7.0 weitaus intelligenter geworden und wertet mehr Informationen aus, um die Entscheidung über die zu verwendenden Indizes zu treffen. Wenn Sie die Ausgabe des `SHOWPLAN_ALL`-Berichts genauer untersuchen, finden Sie viele dieser Werte wieder. Dieses Buch kann allerdings nicht darauf eingehen, wie man diese Werte verwendet.

Da Sie nun wissen, wie SQL Server auf Ihre Tabellen zugreift, stellt sich die Frage, was Sie mit diesen Informationen anfangen können. Außerdem ist die Frage interessant, wie Sie SQL Server dazu bringen können, die von Ihnen erstellten Indizes auch tatsächlich zu verwenden. Die Antwort lautet: Sie müssen den Abfrageoptimierer umgehen!

Den Abfrageoptimierer umgehen

Um den Abfrageoptimierer zu umgehen, können Sie Abfragehinweise spezifizieren. Als Erstes müssen Sie sich dessen bewusst sein, dass der SQL Server-Abfrageoptimierer in der Regel den richtigen Ausführungsplan wählt. Wenn Sie aber von Ihrer Lösung überzeugt sind und glauben, dass sich SQL Server irrt, und wenn Sie Ihre Statistiken aktualisiert und

die SHOWPLAN-Ausgaben genau untersucht haben, können Sie Abfragehinweise in Betracht ziehen. Dieser Abschnitt untersucht diese Optionen, um dem Abfrageoptimierer die richtigen Informationen zu vermitteln.

Zur Auswahl eines Index geben Sie als Teil Ihrer Auswahlanweisung entweder den Namen oder die Nummer des Index an. Die Nummer eines Index finden Sie durch Abfrage der Systemtabelle *sysindexes* in Ihrer Datenbank heraus:



```
SELECT col1, col2
FROM indextab (index = noncl_myindex)
```

Diese Abfrage verwendet normalerweise eine gruppierte Indexsuche. In diesem Fall haben Sie aber angewiesen, den nicht gruppierten Index, den Sie weiter oben angelegt haben, zu benutzen.



SQL Server verwendet ungeachtet dessen, was der Abfrageoptimierer für die bessere Wahl hält, immer den von Ihnen gewählten Index und nimmt niemals eine Neubewertung Ihrer Auswahl vor. Vergessen Sie aus diesem Grund nicht, Ihre Wahl von Zeit zu Zeit zu überprüfen, was insbesondere dann gilt, wenn Sie neue Service Packs installieren oder SQL Server aktualisieren. Aus den genannten Gründen ist es eigentlich nur selten erforderlich (und manchmal sogar ausgesprochen gefährlich), Abfragehinweise einzusetzen.

Wie können Sie nun SQL Server dazu zwingen, einen Tabellenscan zu benutzen? Die Beantwortung dieser Frage erfordert etwas Verständnis darüber, wie die Indizes in der Tabelle *sysindexes* nummeriert sind. Nicht gruppierte Indizes sind von 2 bis 250 nummeriert. Existiert ein gruppierter Index, dann bekommt dieser die Indexnummer 1. Gibt es keinen, befindet sich in der Tabelle *sysindexes* eine Zeile mit der Indexnummer 0. Es ist möglich, im Abfragehinweis nicht den Indexnamen, sondern statt dessen die Indexnummer anzugeben, so dass die SQL-Anweisung oben auch folgendermaßen aussehen kann (unter der Annahme, dass zu *noncl_myindex* die Indexnummer 2 gehört):



```
SELECT col1, col2
FROM indextab (INDEX = 2)
```

Zum Erzwingen einer gruppierten Indexsuche können Sie nun die folgende Abfrage schreiben, wobei Sie sehen werden, dass SQL Server wirklich Ihren Wunsch erfüllt und eine gruppierte Indexsuche einsetzt:



```
SET SHOWPLAN_TEXT ON
GO
SELECT col1, col2
FROM indextab (INDEX = 0)
WHERE col1 < 3
```



```
StmtText
-----
SELECT col1, col2
FROM indextab (INDEX = 0)
WHERE col1 < 3
```

(1 row(s) affected)

```
StmtText
-----
```

```
--Clustered Index
Scan(OBJECT:([pubs].[dbo].[INDEXTAB].[CL_MYINDEX]),
WHERE:([INDEXTAB].[col1]<3))
```

(1 row(s) affected)

Um einen anderen Verwendungszweck dieser Option sehen zu können, löschen Sie nun den gruppierten Index für diese Tabelle, was aus der Tabelle einen »Heap« macht. Führen Sie dann die gleiche Abfrage mit dem Abfragehinweis INDEX=0 aus:



```
set showplan_text off
go
drop index indextab.CL_MYINDEX
GO
SET SHOWPLAN_TEXT ON
GO
SELECT col1, col2
FROM indextab (INDEX = 0)
WHERE col1 < 3
```



```
StmtText
-----
SELECT col1, col2
FROM indextab (INDEX = 0)
WHERE col1 < 3
```

(1 row(s) affected)

```
StmtText
```

```
-----
|--Table Scan(OBJECT:([pubs].[dbo].[INDEXTAB]),
  WHERE:([INDEXTAB].[col1]<3))
```

```
(1 row(s) affected)
```

Beachten Sie, dass nun ein Tabellenscan verwendet worden ist, und zwar aus dem Grund, weil es keinen gruppierten Index für die Tabelle gegeben hat. Vergessen Sie nicht, dass ein Tabellenscan und ein gruppierter Index im Wesentlichen auf die gleiche Operation hinauslaufen, denn da in der Blattebene eines gruppierten Index die Daten selbst enthalten sind, kommt eine Suche auf dieser Ebene dem Einlesen jeder einzelnen Seite (einem Tabellenscan) gleich.

13.4 Indexwartung

Indizes müssen gewartet werden, damit sie auch über längere Zeiträume hinweg effizient und nützlich bleiben. Die Indexwartung muss manuell vorgenommen werden (oder aber Sie entwickeln zu diesem Zweck automatisierte Abläufe). In jedem Fall ist es erforderlich, dass Sie sich Grundlagen der Indexwartung beherrschen.

Statistiken

SQL Server erhält die statistischen Informationen über Ihre Daten von einem speziellen Objekt in der Datenbank (das im Folgenden einmal *statistischer Blob* – Binary Large Object, d.h. großes Binärobjekt – genannt werden soll). Der statistische Blob enthält Informationen darüber, wie die Daten in den Tabellen verteilt sind, und berechnet die Durchschnittszahl von Duplikaten für indizierte Spalten. Sobald Sie einen Index für eine Tabelle erstellen, in der sich bereits Daten befinden, wird ein statistischer Blob erzeugt und mit den Informationen über Ihre Spalten versorgt, die Sie indiziert haben. Falls Sie zuerst den Index erstellen und dann erst die Daten in die Tabelle hinzufügen und die automatische Generierung von Statistiken nicht abgeschaltet haben, wird ebenfalls ein statistischer Blob erzeugt. Schalten Sie dagegen die Einrichtung in SQL Server ab, die automatisch die statistischen Daten über einen Index verwaltet, dann wird dieser statistische Blob nicht erzeugt. Wenn nun diese Informationen nicht vorhanden sind, muss SQL Server schätzen, wie Ihre Daten verteilt sind. Enthält Ihre Abfrage eine Klausel wie etwa `col1 < 3`, dann zieht SQL Server den statistischen Blob zur Schätzung der Frage heran, wie viele Zeilen diese Abfrage zurückliefern wird. Je aktueller Ihr statistischer Blob ist, um so besser ist der Job, den SQL Server bei der Auswahl des richtigen Index (bzw. bei der Entscheidung zu einem Tabellenscan) erledigen kann. Aus diesem Grund sollten Sie das automatische Generieren von Statistiken niemals abschalten.

Übervorsichtige Zeitgenossen sorgen außerdem noch dafür, dass in periodischen Abständen auf manuellem Wege Statistiken ermittelt werden. Zum Aktualisieren des statistischen Blobs führen Sie den Befehl `UPDATE STATISTICS` aus:



```
UPDATE STATISTICS {Tabelle | Sicht}
[Index
|([Index_oder_Spalte[, ...n])]
[WITH [ [FULLSCAN]
| SAMPLE Nummer {PERCENT | ROWS} ] ]
[[,] [ALL | COLUMNS | INDEX]
[[,] NORECOMPUTE] ]
```

Zum Beispiel aktualisieren Sie die Statistiken für die Tabelle *authors* mit der folgenden Anweisung:

```
UPDATE STATISTICS authors
```

Solange die automatischen Statistiken eingeschaltet sind, gibt es keinen Grund dafür, diesen Befehl häufiger auszuführen. Auf vielen Servern wird `UPDATE STATISTICS` vielleicht einmal die Woche (meistens aus Gewohnheit) für jede Tabelle in der Datenbank gestartet. Microsoft liefert für diesen Zweck eine praktische gespeicherte Systemprozedur namens `sp_statistics` mit. Zum Starten dieser Prozedur geben Sie in einem Abfragefenster `exec sp_statistics` ein – mehr nicht, keine Parameter und keine anderen Optionen. Das genügt, damit SQL Server automatisch Statistiken über die Indizes aller Tabellen in der Datenbank sammelt.

Abschalten der automatischen Statistiken

Die automatischen Statistiken können Sie über folgende gespeicherte Systemprozedur abschalten (und natürlich auch wieder einschalten, weil sie nützlich sind):

```
sp_autostats 'Tabellenname' [, 'Statsflag'] [, 'Indexname']
```

Die Argumente haben folgende Bedeutung:

- *Tabellenname* ist der Name der Tabelle, für die Sie die Option `autostats` ändern möchten.
- *Statsflag* ist entweder `on` (aktiviert) oder `off` (deaktiviert). Damit schalten Sie die automatischen Statistiken ein bzw. aus (der Standardwert ist `off`).
- *Indexname* ist der Name des Index, den Sie ändern möchten. Falls Sie diesen Parameter weglassen, sind sämtliche Indizes einer gegebenen Tabelle von der Änderung betroffen. Falls Sie sowohl *Statsflag* als auch *Indexname* weglassen, liefert `sp_autostats` Informationen über den aktuellen Status der automatischen Statistiken für die Indizes der Tabelle zurück.

Zum Abschalten der automatischen Statistiken für die Tabelle *authors* in der Datenbank *pubs* führen Sie daher die folgende Anweisung aus:



```
Use pubs
Exec sp_autostats 'authors','Off'
```



```
(4 row(s) affected)
Automatic statistics maintenance turned OFF for 4 indices.
```

Spaltenstatistiken

SQL Server 2000 kann auch Statistiken zu nicht indizierten Spalten – so genannte *Spaltenstatistiken* – verfolgen. Dazu benutzen Sie wie schon zuvor den Befehl `UPDATE STATISTICS` und geben die Option `WITH COLUMNS` für die Spalten an, über die Sie Statistiken erhalten möchten. Zum Beispiel sorgt die folgende Anweisung für die Aktualisierung der Statistiken über die Spalte *city* der Tabelle *authors*:

```
UPDATE STATISTICS authors (city) WITH COLUMNS
```

Vielleicht fragen Sie sich, wozu solch eine Einrichtung gut sein soll. Die Antwort besteht darin, dass der SQL Server-Abfrageoptimierer (der Teil des Systems, der die zu verwenden- den Indizes festlegt) diese statistischen Daten ähnlich wie indizierte Spalten mit Statistiken für die Entscheidung verwenden kann, wie eine Tabelle am besten zu lesen ist. Wenn Sie daher in der `WHERE`-Klausel Ihrer Auswahlanweisungen eine Spalte nennen, aber diese Spalte nicht indiziert haben, kann sie über die Spaltenstatistiken trotzdem zu Optimierungszwecken verwendet werden. Diese Spaltenstatistiken tauchen später noch einmal bei der Besprechung des Indexoptimierungs-Assistenten auf.

Wann wurden die Statistiken denn das letzte Mal erstellt?

Wenn Sie als Administrator mit einem neuen SQL Server betraut werden, ist natürlich für Sie der aktuelle Status Ihrer SQL Server-Datenbanken von Interesse. Dazu gehört auch die Feststellung, wie und wann die Statistiken zuletzt aktualisiert worden sind. Mit der Funktion `STATS_DATE` lässt sich das leicht herausfinden:



```
SELECT STATS_DATE(Tabellen_id, Index_id)
```

Diese Funktion ist zwar ganz einfach, verlangt aber von Ihnen die Angabe der *Tabellen_id* und der *Index_id*, die Sie wiederum über die Funktion OBJECT_ID bekommen:



```
Use pubs
Go
Declare @tabid int
Declare @indid int
Select @tabid = object_id ('indextab')
Select @indid = 2 /* Der nicht gruppierte Index */
SELECT STATS_DATE (@tabid, @indid)
```

Die Ausgabe sieht folgendermaßen aus:



```
-----
2001-01-13 17:11:05.860
```

```
(1 row(s) affected)
```



Wenn Sie nicht das oben gezeigte Ergebnis erhalten, ist vielleicht noch eine SHOWPLAN-Option aktiviert. Öffnen Sie ein neues Abfragefenster und kopieren Sie den Code in das neue Fenster oder führen Sie die folgenden Anweisungen aus:

```
SET SHOWPLAN_TEXT OFF
GO
SET SHOWPLAN_ALL OFF
GO
```

Der Befehl DBCC UPDATEUSAGE

Die Anweisung DBCC UPDATEUSAGE meldet und korrigiert Ungenauigkeiten in der Systemtabelle *sysindexes* von SQL Server. Die Informationen in dieser Tabelle teilen SQL Server zum Beispiel mit, wie viele Seiten in der Tabelle sind und wie viele Seiten von jedem Index eingenommen werden. Diese Informationen werden automatisch verwaltet, wobei die Möglichkeit besteht, dass die entsprechende Systemtabelle unter Umständen nicht mehr aktuell ist. Zur Aktualisierung dieser Informationen führen Sie DBCC UPDATEUSAGE aus:



```
DBCC UPDATEUSAGE ({0 | Datenbankname}
[, {'Tabellenname' | 'Sichtname'}
[, Index_id] | 'Indexname'])
[ WITH [COUNT_ROWS] [, NO_INFOMSGS ]]
```

Die Argumente haben folgende Bedeutung:

- *Datenbankname* ist der Name der Datenbank, in der Sie arbeiten. Falls Sie nur den Datenbanknamen angeben, werden die Informationen in *sysindexes* für sämtliche Heaps und Indizes in der Datenbank aktualisiert.
- *Tabellenname* oder *Sichtname* ist der Name einer Tabelle bzw. Sicht, für die Sie die Informationen in der Systemtabelle *sysindexes* aktualisieren möchten.
- *Index_id* ist die Indexnummer eines Index für eine bestimmte Tabelle, für die Sie die Informationen in der Systemtabelle *sysindexes* aktualisieren möchten. Die *Index_id* eines Index lässt sich mit der gespeicherten Systemprozedur *sp_helpindex* ermitteln; Sie können aber auch den Namen des Index angeben.

Der Befehl DBCC SHOWCONTIG

Ihre Indizes müssen in periodischen Abständen reorganisiert werden. Der Hintergrund liegt darin, dass die Indizes im Laufe der Zeit nicht mehr den Füllfaktoren entsprechen oder mehr und mehr fragmentiert und damit weniger brauchbar werden. Ob Ihre Indizes fragmentiert sind und reorganisiert werden müssen, erfahren Sie über den Befehl DBCC SHOWCONTIG:



```
DBCC SHOWCONTIG [( {Tabellenname | Tabellen_Id | Sichtname | Sicht_Id}
[, Indexname | Index_Id] ) ]
[WITH {ALL_INDEXES | FAST [, ALL_INDEXES]
| TABLERESULTS [, {ALL_INDEXES } ]
[, { FAST | ALL_LEVELS } ] } ]
```

Die Argumente haben folgende Bedeutung:

- *Tabellen_Id* oder *Sicht_Id* ist die ID der Tabelle bzw. Sicht, für die Sie die Fragmentierung untersuchen möchten. Die ID eines Objekts erhalten Sie über die Funktion *object_id('Objektnamen')*.
- *Tabellenname* oder *Sichtname* ist der Name der Tabelle oder Sicht, die Sie untersuchen möchten. Anstelle der ID können Sie den Namen angeben. Diese Option ist neu in SQL Server 2000.
- *Index_Id* ist die interne Nummer des Index, den Sie untersuchen möchten. Ohne diesen Wert werden die Informationen für den gruppierten Index (falls vorhanden) oder den Heap (eine Tabelle ohne einen gruppierten Index) ermittelt.
- *Indexname* ist der Name des zu untersuchenden Index. Auch diese Alternative für die *Index_Id* ist eine neue Option in SQL Server 2000.

Mit dem folgenden Skript können Sie sich die Ergebnisse von DBCC SHOWCONTIG ansehen. Führen Sie das Skript in einem neuen Fenster aus oder schalten Sie alle SHOWPLAN- und anderen Optionen ab, die Sie in den letzten Beispielen gegebenenfalls aktiviert haben.



```
USE PUBS
GO
DECLARE @tableid int
SELECT @tableid = object_id ('INDEXTAB')
DBCC SHOWCONTIG (@tableid)
GO
```



DBCC SHOWCONTIG scannt die INDEXTAB-Tabelle...
 Tabelle: 'INDEXTAB' (1445580188), Index-ID: 0, Datenbank-ID:5
 Scan auf Ebene TABLE wurde ausgeführt.

- Gescannte Seiten.....: 1
- Gescannte Blöcke.....: 1
- Blockwechsel.....: 0
- Seiten pro Block (Durchschnitt).....: 1.0
- Scandichte [Bester Wert:Tatsächlicher Wert].....: 100.00% [1:1]
- Blockscanfragmentierung: 0.00%
- Byte frei pro Seite (Durchschnitt): 6438.0
- Mittlere Seitendichte (voll).....: 20.46%

DBCC-Ausführung abgeschlossen. Falls DBCC Fehlermeldungen ausgegeben hat, wenden Sie sich an den Systemadministrator.



Eine komfortable Neuerung der SQL Server 2000-DBCC-Befehle ist die Unterstützung für die Option `tableresults`. Dadurch können die DBCC-Befehle einen gültigen SQL Server-Rowset zurückgeben. In der Tat unterstützen fast alle DBCC-Befehle diese Option, selbst wenn sie in den Syntaxbeschreibungen der Onlinedokumentation nicht angegeben ist. Aufgrund dieser Unterstützung lassen sich die Ergebnisse eines DBCC-Befehls in eine Tabelle einfügen. Man kann damit bequem umfangreiche DBCC-Ergebnisse nach dem relationalen Prinzip untersuchen. Wenn Sie zum Beispiel die Anweisungen

```
USE pubs
DBCC SHOWCONTIG ('indextab') WITH TABLERESULTS
GO
```

ausführen, erhalten Sie die Antwort als einzelne Zeile in einer virtuellen Tabelle (Rowset) zurück.



Es gibt hier eine beträchtliche Menge an Informationen, von denen jedoch nur zwei oder drei Schlüsselwerte für Sie interessant sind. Als Erstes können Sie aus der Zeile mit Index-ID: 0 ableiten, dass diese Tabelle keinen gruppierten Index hat. Als nächster Wert ist die Scandichte interessant. Je höher der Prozentwert,

desto geringer ist die Fragmentierung der Tabelle. Auch der Wert Mittlere Seitendichte (*voll*) ist wichtig. Er sollte möglichst in der Nähe Ihres festgelegten Füllfaktors liegen.

Wenn diese Werte abfallen sollten (einen festen Grenzwert kann man dazu leider nicht angeben), dann ist wahrscheinlich der Zeitpunkt gekommen, die Indizes zu reorganisieren. Dazu löschen Sie den Index und bauen ihn neu auf. Handelt es sich um einen gruppierten Index, dann können Sie die Aktion durch die Option `DROP_EXISTING` beschleunigen, die den Index einschließlich der Neuanwendung Ihrer Füllfaktoren zurück in den Ursprungszustand versetzt. Außerdem werden hiervon Ihre nicht gruppierten Indizes ausgenommen, da die Reihenfolge und die Gesamtindexstruktur Ihres gruppierten Index nicht von der Option `DROP_EXISTING` betroffen sind. Falls Sie den gruppierten Index ohne diese Option löschen und neu erstellen, reorganisieren Sie damit sowohl die Daten- als auch die Indexstruktur.

`DBCC SHOWCONTIG` kann entweder von den Mitgliedern der festen Serverrolle *sysadmin* oder der festen Datenbankrollen *db_owner* oder *db_ddladmin* ausgeführt werden.

Der Befehl DBCC DBREINDEX

Es gibt noch eine weitere Option, die sich vielleicht als vorteilhafter herausstellen könnte: Der Befehl `DBCC DBREINDEX`:



```
DBCC DBREINDEX ([ 'Datenbank.Besitzer.Tabellenname' [, Indexname
[, Füllfaktor ]])
[WITH NOINFOMSGS]
```

Viele der Optionen sind die gleichen wie in der Anweisung `CREATE INDEX`, so dass Sie sich diese Anweisung noch einmal ansehen sollten, wenn Sie sich für die Sortierung von Daten und die Optionen des Füllfaktors interessieren. Andernfalls wird der Füllfaktor eingestellt, den Sie angegeben haben. Falls Sie jedoch einen Füllfaktor von 0 angeben, wird der ursprüngliche Faktor verwendet, der bei der Erstellung des Index angegeben worden ist. Wenn Sie jedoch noch irgendeinen anderen optionalen Parameter angeben möchten, müssen Sie auch sämtliche davor stehenden Optionen spezifizieren. Geben Sie beispielsweise einen Füllfaktor an, dann müssen Sie auch wieder den Indexnamen angeben; sogar dann, wenn Sie nur einen leeren Namen mitliefern. Zum Beispiel baut die folgende Anweisung alle Indizes der Tabelle *authors* mit dem ursprünglichen Füllfaktor neu auf:



```
DBCC DBREINDEX ('authors','', 0)
```

Falls Sie eine Tabelle spezifizieren, wie im folgenden Beispiel, dann wird jeder Index gelöscht und anschließend neu erstellt, wie nachstehende Ausgabe zeigt:



```
DBCC DBREINDEX ('indextab')  
GO
```



DBCC-Ausführung abgeschlossen. Falls DBCC Fehlermeldungen ausgegeben hat, wenden Sie sich an den Systemadministrator.

Sie können auch einen individuellen Index angeben, mit dem Sie arbeiten wollen. Im folgenden Beispiel erstellen Sie als Erstes einen zusätzlichen nicht gruppierten Index für die Tabelle *authors*, führen dann den Befehl DBREINDEX aus und geben dabei den neuen Index an, den Sie gerade erstellt haben:



```
CREATE INDEX noncl_col2_indextab on indextab (col4)  
GO  
DBCC DBREINDEX ('indextab','noncl_col2_indextab')
```



Seien Sie vorsichtig, wenn Sie zusammen mit DBCC DBREINDEX einen Füllfaktor angeben. Falls Sie keinen bestimmten Index angeben, aber die Füllfaktoreinstellung verwenden wie etwa:

```
DBCC DBREINDEX ('indextab', '', 50)
```

wird dadurch der Füllfaktor für sämtliche Indizes der Tabelle zurückgesetzt (in diesem Fall auf 50), was nicht unbedingt Ihrer Absicht entsprechen muss.

DBCC DBREINDEX ist gegenüber dem Löschen und Neuerstellen von Indizes im Vorteil. Wie Tag 14 noch zeigt, erstellen manchmal Einschränkungen der referenziellen Integrität Indizes hinter den Kulissen. Mit DBCC DBREINDEX können Indizes, die mit Einschränkungen erstellt worden sind, sicher gelöscht und neu aufgebaut werden, wohingegen der einzige alternative Weg im Löschen und Neuerstellen der Einschränkungen besteht.

DBCC DBREINDEX kann von Mitgliedern der Serverrolle *sysadmin*, der Datenbankrollen *db_owner* oder *db_ddladmin* oder vom Besitzer einer Tabelle ausgeführt werden.

Der Befehl DBCC INDEXDEFRAG

Der Befehl `DBCC INDEXDEFRAG` ist neu in SQL Server 2000. Auf den ersten Blick scheint diese Option verglichen mit `DBCC DBREINDEX` begrenzt zu sein, sicherlich werden Sie aber mit diesem Befehl bevorzugt arbeiten, um Indizes zu reorganisieren. Wenn Sie `DBCC DBREINDEX` verwenden, ist Ihre Tabelle nicht für andere Abfragen zugänglich, bis die Reorganisation fertig gestellt ist. Bei `DBCC INDEXDEFRAG` dagegen können andere Benutzer weiterhin auf die Tabelle zugreifen. Diese Fähigkeit bezeichnet man als *Onlinereorganisation*. Sehen Sie sich zuerst die Syntax an:



```
DBCC INDEXDEFRAG
( { Datenbankname | Datenbank_Id | 0 }
  , { Tabellename | Tabellen_Id | 'Sichtname' | Sicht_Id }
  , { Indexname | Index_Id }
) [ WITH NO_INFOMSGS ]
```

Auch hier sind die Optionen gleich den bereits in dieser Lektion behandelten Optionen, so dass wir die Bedeutung nicht noch einmal wiederholen. Worin liegt aber nun der Unterschied zwischen diesem Befehl und `DBCC DBREINDEX`?

`DBCC INDEXDEFRAG` reorganisiert nur die Blattebenen der Indizes. Wenn Ihr Ziel also in einem perfekt von oben nach unten organisierten Index besteht, ist diese Option nichts für Sie. Wollen Sie allerdings eine Hintergrundaufgabe ausführen und den größten Teil der Indexfragmentierung beseitigen oder können Sie es sich nicht leisten, dass eine Tabelle während der Ausführung von `DBREINDEX` nicht verfügbar ist, dann ist `DBCC INDEXDEFRAG` genau richtig.

`DBCC INDEXDEFRAG` kann von Mitgliedern der Serverrolle *sysadmin*, der Datenbankrollen *db_owner* oder *db_ddladmin* oder vom Besitzer einer Tabelle ausgeführt werden.

Indizierte Sichten

SQL Server 2000 führt auch die Möglichkeit ein, Indizes für Sichten zu erstellen. In allen vorherigen Versionen von Microsoft SQL Server lassen sich Indizes nur für Tabellen anlegen. Obwohl indizierte Sichten konzeptionell einfach sind, müssen Sie sich mit einigen komplexeren Konfigurationsproblemen auseinandersetzen. Um jedoch die Grundlagen zu verstehen, brauchen Sie sich noch nicht mit diesen Problemen herumzuschlagen.

Wie Sie in den Abschnitten der heutigen Lektion zum Thema Index bereits gesehen haben, lässt sich an allen Stellen, wo man eine Tabelle spezifizieren kann, auch ein Sichtname angeben. Da Tabellen tatsächlich Daten enthalten, lässt sich natürlich ein Index auf ihnen erstellen (und ist ohne weiteres verständlich). Wie verhält es sich aber mit einer Sicht, die lediglich eine gespeicherte Abfrage darstellt, wie es Tag 15 noch näher erläutert?

Wenn Sie einen Index auf einer Sicht erstellen, werden die Abfrageergebnisse berechnet und dann gespeichert. Sobald sich in den Zeilen, die den Ergebnissen der Sicht zugrunde liegen, Änderungen ergeben, werden auch die Indizes aktualisiert.

Dieses Szenario lässt sich am besten an einem Beispiel erläutern. Stellen Sie sich eine Sicht vor, die die Tabellen *titles*, *titleauthor* und *authors* in der Datenbank *pubs* kombiniert, um eine einzelne Sicht der Tabellen zu präsentieren. Die Sicht könnte folgendermaßen aussehen:

```
create view myauthorview
with schemabinding
as
select a.au_lname, a.au_fname, t.title, ta.royaltyp
from dbo.authors a
    inner join dbo.titleauthor ta on a.au_id = ta.au_id
    inner join dbo.titles t on t.title_id = ta.title_id
```

Führen Sie nun die folgende Abfrage aus:

```
select * from myauthorview
```

Abbildung 13.10 zeigt den Ausführungsplan für diese Anweisung.

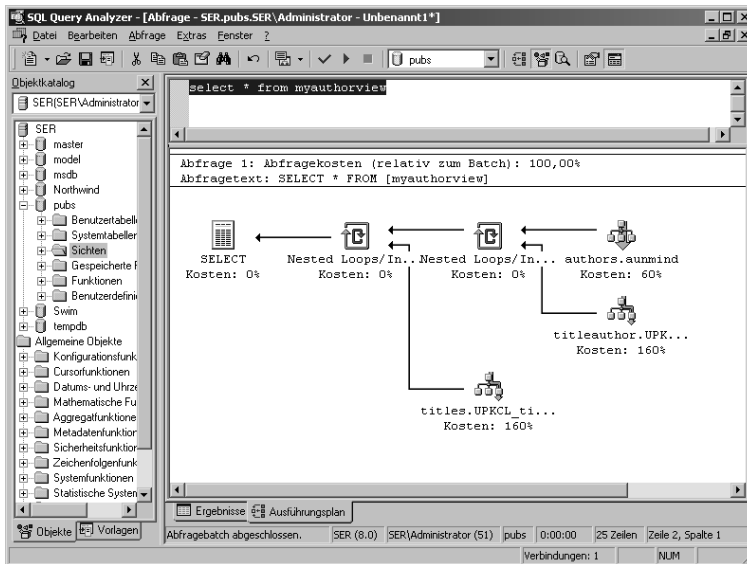


Abbildung 13.10:
Der Ausführungs-
plan ohne Verwen-
dung des gruppier-
ten Index

Weil diese Tabellen so klein sind, kann man den Abfrageoptimierer nur schwer dazu bewegen, den gruppierten Index zu verwenden. Man kann ihn aber mit einem Abfragehinweis dazu zwingen. Führen Sie jetzt folgende Anweisung aus:

```
select * from myauthorview (noexpand)
```

Der Ausführungsplan (siehe Abbildung 13.11) zeigt jetzt, dass der gruppierte Index verwendet wurde.

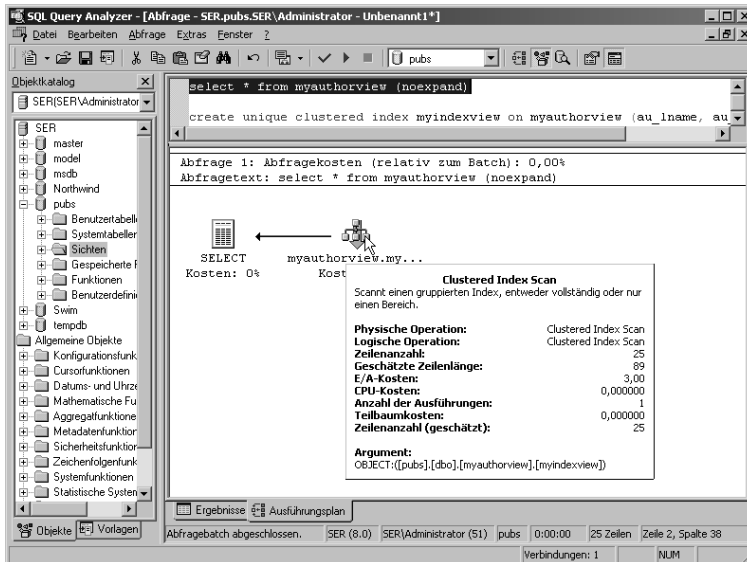


Abbildung 13.11:
Der Ausführungsplan zeigt die Verwendung des gruppierten Index.

Bei großen Tabellen mit vielen Verknüpfungen kann man sich leicht vorstellen, dass sich die Leistung bedeutend verbessert, wenn man die indizierte Sicht verwenden kann, weil die Antworten bereits vorberechnet sind. Diese Sicht ist insbesondere nützlich in einer Berichtsumgebung, wo man derartige Abfragen von vornherein kennt.

Wie wird eine indizierte Sicht verwaltet? Jedes Mal, wenn Sie eine Zeile in einer der der Sicht zugrunde liegenden Tabellen (*authors*, *titles* oder *titleauthor*) ändern, muss auch die indizierte Sicht aktualisiert werden. Wenn also bereits die Verwaltung eines normalen Index teuer erscheint, sollten Sie sich den Overhead für die Aktualisierung dieses Index vorstellen. Verbessert sich allerdings die Abfrageleistung drastisch, ist dieser Nachteil möglicherweise zu verschmerzen.

Einschränkungen bei indizierten Sichten

Wie bereits in der Einführung zu indizierten Sichten angerissen, sind bestimmte Einschränkungen zu beachten:

- Sie müssen zuerst einen eindeutigen gruppierten Index für die Sicht erzeugen, bevor Sie einen nicht gruppierten Index erstellen können.
- Sie müssen die Sicht mit der Option `SCHEMABINDING` erstellen. Auf diese Option geht Tag 15 näher ein.

- Die Eigenschaften `ANSI_NULLS` und `QUOTED_IDENTIFIER` müssen für die Datenbankverbindung, in der Sie die Sicht erstellen, aktiviert sein (im SQL Query Analyzer sind diese Optionen standardmäßig eingeschaltet).
- Mit der Sicht lassen sich nur Tabellen abfragen und keine anderen Sichten.
- Alle in der Sicht verwendeten Objekte müssen denselben Besitzer haben und sich in derselben Datenbank befinden wie die Sicht.
- In allen Abfrageverweisen müssen Sie den Namen des Besitzers angeben, wie es die Beispielabfrage weiter oben in diesem Abschnitt gezeigt hat.

In der Onlinedokumentation sind noch weitere Einschränkungen aufgeführt, die aber nicht ganz so wichtig sind.



Indizierte Sichten werden vom Abfrageoptimierer nur in der Enterprise Edition von SQL Server 2000 verwendet. Indizes für Sichten können Sie mit jeder Edition von SQL Server 2000 erstellen, aber nur die Enterprise Edition verwendet sie tatsächlich. Diese Fähigkeit ist verfügbar, damit die Anweisungen der Datendefinitionssprache (DDL) nicht bei den verschiedenen Versionen von SQL Server scheitern, selbst wenn sie nicht umfassend in den anderen Editionen des Produkts zum Einsatz kommen.

13.5 Indizes mit dem Enterprise Manager verwalten

Im SQL Server Enterprise Manager können Sie Indizes im Dialogfeld für die Indexeigenschaften verwalten. Als Erstes müssen Sie wissen, dass es zwei verschiedene Dialogfelder für die Indexkontrolle gibt. Sie können die Datenbank erweitern, mit der Sie arbeiten möchten (in den folgenden Beispielen ist dies die Datenbank *pubs*), den Ordner *Tabellen* markieren und im Detailbereich mit der rechten Maustaste auf die gewünschte Tabelle klicken. Dann wählen Sie aus dem Kontextmenü `TABELLE BEARBEITEN`, um das Dialogfeld zum Bearbeiten der Tabelle zu öffnen. Abbildung 13.12 zeigt dieses Dialogfeld für die Tabelle *authors*.

Wenn Sie auf die zweite Schaltfläche (`TABELLEN- UND INDEXEIGENSCHAFTEN`, siehe Abbildung 13.12) klicken, erscheint ein Eigenschaftsdialog für die Tabelle. Beachten Sie, dass sich dieses Dialogfeld von demjenigen für die allgemeinen Eigenschaften einer Tabelle unterscheidet. Klicken Sie auf die Registerkarte `INDIZES/SCHLÜSSEL`, die Abbildung 13.13 zeigt.

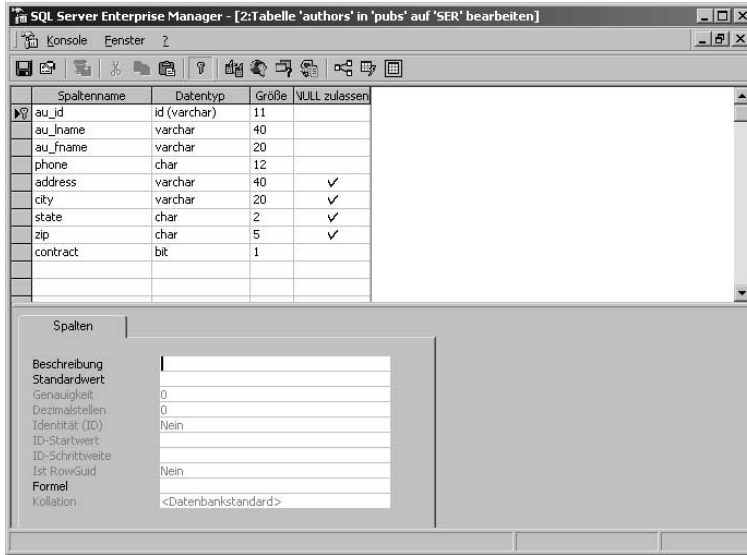


Abbildung 13.12:
Das Fenster *Tabelle*
bearbeiten

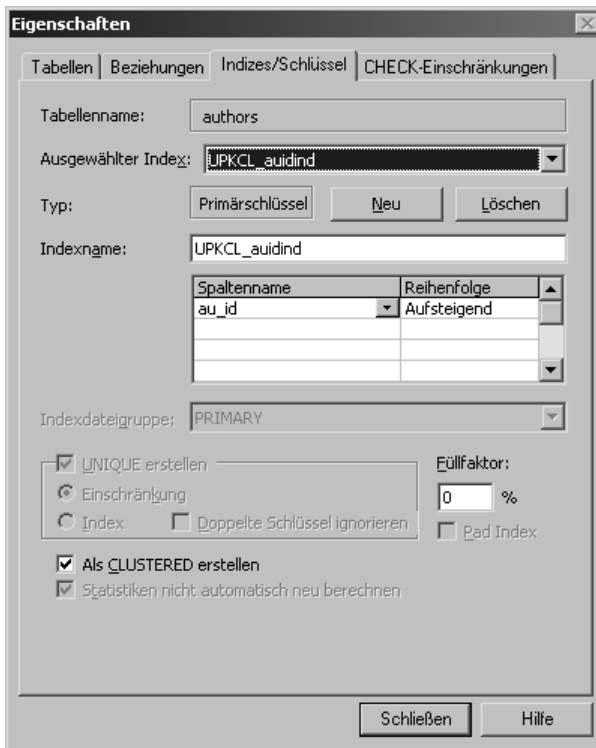


Abbildung 13.13:
Das Eigenschaftsdialogfeld für das
Bearbeiten einer Tabelle

Von diesem Dialogfeld aus können Sie nun die meisten Indizierungsoperationen wie Erstellen, Entfernen oder Umbenennen steuern. Sie können festlegen, ob Sie einen eindeutigen Index wünschen, und wenn ja, ob es eine eindeutige Einschränkung oder ein eindeutiger Index sein soll. In der Praxis werden Sie feststellen, dass die eindeutige Einschränkung gegenüber einem eindeutigen Index normalerweise vorgezogen wird. Wenn Sie es vorziehen, es bei einem eindeutigen Index zu belassen, können Sie die Option **DOPPELTE SCHLÜSSEL IGNORIEREN** einstellen, die Sie schon weiter oben im Abschnitt »Die Option IGNORE_DUP_KEY« kennen gelernt haben.

Darüber hinaus lässt sich festlegen, dass der Index gruppiert sein und die automatische Generierung von Statistiken abgeschaltet werden soll. Beachten Sie, dass automatische Einstellungen immer eingeschaltet werden, solange Sie sie nicht explizit abschalten. Außerdem stehen an dieser Stelle noch der Füllfaktor und der *Pad_Index* zur Verfügung. Eine weitere Option stellt die Dateigruppe dar. Falls Sie Dateigruppen verwenden (siehe dazu Tag 4), können Sie hier die Indexdateigruppe ändern, die mit dem Index verknüpft ist.

Es gibt jedoch noch ein besseres Dialogfeld (siehe Abbildung 13.14), das Ihnen in viel übersichtlicherer Art und Weise zeigt, welche Indizes vorhanden und welche Spalten indiziert sind. Zu diesem Dialogfeld gelangen Sie, indem Sie mit der rechten Maustaste auf eine Tabelle klicken und **ALLE TASKS / INDIZES VERWALTEN** wählen.



Abbildung 13.14:
Das Dialogfeld Indizes ver-
walten mit ausgewählter
Datenbank authors

Die Leistungsfähigkeit dieser Option wird offensichtlich, wenn Sie einen neuen Index erstellen möchten. Klicken Sie dazu auf die Schaltfläche NEU, um zum Dialogfeld NEUEN INDEX ERSTELLEN zu gelangen (siehe Abbildung 13.15). Beachten Sie, dass Sie nun jede verfügbare Option von SQL Server 2000 in grafischer Form zur Auswahl angeboten bekommen.



Abbildung 13.15:
Das Dialogfeld Neuen Index erstellen



Warum gibt es eigentlich zwei komplett verschiedene Dialogfelder zur Verwaltung von Indizes? Dasjenige aus den Tabelleneigenschaften ist eine Codekomponente, die es in vielen Microsoft-Produkten gibt (wie etwa Microsoft Visual Studio), und die zur Unterstützung der unterschiedlichsten Datenbanken, nicht nur von SQL Server 2000, entwickelt worden ist. Das Dialogfeld INDIZES VERWALTEN ist im Gegenteil dazu nur für SQL Server 2000 konzipiert und stellt sämtliche spezifischen Optionen von SQL Server zur Verfügung.

13.6 Indexauswahl

Sie kennen jetzt die verfügbaren Optionen und wissen, wie Sie Indizes aufbauen können. Die nächste Frage müsste nun lauten: »Welche Spalten sollte man eigentlich indizieren, und wie sollte man sie indizieren?«

Geeignete Kandidaten für Indizes sind:

- Primärschlüsselspalten (werden am 14. Tag erörtert)
- Fremdschlüsselspalten (werden ebenfalls am 14. Tag erörtert)
- Spalten, die Sie in der Klausel `ORDER BY` verwenden
- Spalten, die Sie in der Klausel `GROUP BY` verwenden
- Spalten, die Sie explizit in Ihrer `WHERE`-Klausel angegeben haben

Generell sollten Sie Spalten indizieren, die in der `WHERE`-Klausel als Bedingungen, die keine Funktionen oder Berechnungen enthalten, eingebunden sind. Das folgende Beispiel deutet auf einen wirklich guten Kandidaten für einen Index in Form der Spalte *emp_id* hin:

```
WHERE emp_id = 5
```

Im folgenden Beispiel wird wahrscheinlich kein Index verwendet werden können, so dass es auch keinen Sinn macht, einen zu erstellen (jedenfalls nicht für diese Abfrage):

```
WHERE SUBSTRING(emp_id,1,5) = 'Annet'
```



Anweisungen wie die erste `WHERE`-Klausel im letzten Absatz heißen auch `SEARCHABLE`-Argumente. Derartige Argumente kann `SQL Server` verwenden. Zusätzlich zu den exakten Treffern können Sie bei Zeichenspalten manchmal auch angenäherte Treffer einsetzen. Zum Beispiel ist

```
WHERE emp_lname LIKE 'w%'
```

ein `SEARCHABLE`-Argument, dagegen

```
WHERE emp_lname LIKE '%w%'
```

nicht. Beachten Sie, dass das Symbol `%` ein Platzhalterzeichen darstellt. Im ersten Beispiel wird die Spalte *emp_lname* nach Werten durchsucht, die mit einem `w` beginnen, während im zweiten Beispiel *emp_lname* nach Werten durchsucht wird, in denen ein `w` an beliebiger Stelle vorkommt. Stellen Sie sich nun einmal vor, dass Sie eine derartige Suche in einem Telefonbuch durchführen würden. Wie leicht fällt es Ihnen, die zweite Suchbedingung zu erfüllen? `SQL Server` hat das gleiche Problem – wenn Sie das erste Zeichen spezifizieren, ist wenigstens klar, wo die Suche beginnt.

Ein anderer Gesichtspunkt, der berücksichtigt werden muss, besteht darin, dass nicht jede Spalte auch ein guter Kandidat zum Indizieren ist. Die folgenden Spalten sollten Sie nicht indizieren:

- Spalten, die vom Datentyp `text`, `image` oder `bit` sind
- Spalten, die nur wenige eindeutige Werte liefern können (wie etwa männlich oder weiblich)
- Spalten, die zu breit sind, um sinnvoll als Indizes dienen zu können

Die letzte Option hängt zwar von Ihrer Anwendung und Ihrer Tabelle ab, aber sicherlich lässt sich sagen, dass eine Spalte mit der Definition `char(200)` nur einen zweitrangigen Kandidaten für einen Index darstellt.

Jeder Index zieht weitere Fragen hinsichtlich der Pflege, des Speicherplatzbedarfs und der Systemleistung nach sich, so dass Sie immer mit Vorsicht handeln sollten. Generell ist zu empfehlen, dass Sie nicht mehr als drei bis vier Indizes für eine Tabelle mitführen. Jede Regel hat zwar auch ihre Ausnahmen, aber diese stellt grundsätzlich eine recht nützliche Richtlinie dar. Nachdem Sie die zu indizierenden Spalten ermittelt haben, müssen Sie im nächsten Schritt den Typ des Index festlegen: Gruppirt oder nicht gruppiert.

Grupperte Indizes

Da nur ein einziger gruppierter Index möglich ist, sollten diesen als ersten festlegen. Zu den guten Kandidaten für gruppierte Indizes gehören:

- Sehr spezifische Abfragen (beispielsweise `coll = 5`).
- Abfragen mit Datenbereichen (beispielsweise `WHERE coll BETWEEN 5 AND 30` oder `WHERE coll > 20`).
- Abfragen für Spalten, für die Sie häufig ein `ORDER BY` oder `GROUP BY` angeben.

Eine kritische Frage hinsichtlich der Auswahl Ihres gruppierten Index in SQL Server 2000 besteht darin, dass die Schlüsselwerte (indizierte Spalten), die Sie für einen gruppierten Index bestimmen, auch in den nicht gruppierten Indizes mitgeführt werden. Falls Sie daher einen breiten gruppierten Index wie etwa `char(30)` wählen, dauert es nicht nur länger, den gruppierten Index zu durchsuchen, sondern auch alle Ihre nicht gruppierten Indizes müssen den Wert von `char(30)` aus Ihrem gruppierten Indexschlüssel für jede Zeile im nicht gruppierten Index mitführen. Weil dies einen bedeutenden Verwaltungsaufwand zur Folge hat, sollten Sie Ihre gruppierten Indexschlüssel so klein wie möglich halten. Außerdem sollten Sie sicherstellen, dass der gruppierte Schlüssel, den Sie ausgewählt haben, nicht so häufig aktualisiert wird, weil andernfalls auch alle Ihre nicht gruppierten Indizes aktualisiert werden müssten.

Eine andere Möglichkeit besteht darin, Fremdschlüsselspalten zu indizieren. Dazu mehr in Lektion 14.

Nicht gruppierte Indizes

Nach der Auswahl Ihres gruppierten Index verbleiben nur noch die nicht gruppierten Indizes. Gute Kandidaten dafür sind:

- Spezifische Abfragen (sind tatsächlich für beide Indextypen gut geeignet).
- Abfragen, die gänzlich über einen Index beantwortet werden können (so genannte *covering index queries*). Falls Sie beispielsweise einen Index auf den Spalten *au_lname* und *au_fname* in der Tabelle *authors* der Datenbank *pubs* haben, würde die folgende Abfrage vollständig über den Index und ohne Zugriff auf die Daten befriedigt werden können:

```
SELECT au_fname from pubs..authors where au_lname = "White"
```

- Spalten, die sortiert oder gruppiert werden.
- Spalten, die Teil von Funktionen sind (wie etwa MIN, MAX oder COUNT).

Der springende Punkt hier besteht darin, dass Sie nur Spalten indizieren, die in der WHERE-Klausel Ihrer Abfrage verwendet werden, und dass Sie weiterhin überprüfen, ob die von Ihnen gewählten Indizes auch von SQL Server eingesetzt werden (dies können Sie über die Option SHOWPLAN oder den grafischen Ausführungsplan im SQL Query Analyzer ermitteln).

Der Indexoptimierungs-Assistent

Den Indexoptimierungs-Assistenten untersuchen Sie detailliert am 20. Tag. Er wird an dieser Stelle einfach nur deshalb erwähnt, damit Sie später wissen, dass er nicht vergessen worden ist. Nichtsdestotrotz müssen Sie, bevor Sie diesen Assistenten sinnvoll einsetzen können, vorher noch etwas über den SQL Server Profiler wissen.

13.7 Volltextindizes

Mit Volltextindizes sind Sie in der Lage, einen Index für eine text- wie auch jede andere Zeichenspalte zu erstellen. Beginnend mit SQL Server 2000 wird diese Funktion standardmäßig installiert (sofern Sie nicht SQL Server unter Windows 9x ausführen). Die Volltextindizierung wurde mit der Einbindung der Microsoft Index Server-Technologie in SQL

Server 2000 möglich. Die Funktion setzt voraus, dass der Microsoft Search-Dienst läuft (was standardmäßig dann der Fall ist, wenn Sie während der Installation auch die Volltextindizierung installiert haben). Außerdem wird ein eindeutiger Index für jede Tabelle verlangt, die Sie für die Volltextindizierung verwenden möchten. Einen Volltextindex kann nur der Besitzer einer Tabelle erstellen.

Pro Tabelle können Sie nur einen einzigen Volltextindex anlegen. Außerdem wird der Index physisch außerhalb von SQL Server gespeichert (nämlich im Verzeichnis `\mssql\FTData`). Anders als andere Indizes von SQL Server werden Volltextindizes nicht automatisch verwaltet – dazu benötigen Sie einen periodisch ausgeführten Auftrag, der den Index aktualisiert. Aufträge werden detailliert am Tag 18 behandelt. Die Einrichtung und Verwendung eines Volltextindex geschieht anders als bei den normalen Indizes; aber auch bei dieser Aufgabe unterstützt Sie ein Assistent.

Markieren Sie im SQL Server Enterprise Manager die Tabelle, für die Sie einen Volltextindex erstellen möchten, und wählen Sie dann EXTRAS / VOLLTEXTINDIZIERUNG im Menü. Nun startet der Volltextindizierungs-Assistent, der Sie durch die einzelnen Arbeitsschritte zum Erstellen eines Index führt. Im vorliegenden Beispiel wird die Tabelle `pub_info` in der Datenbank `pubs` verwendet, die sich deshalb anbietet, weil man mit ihr einen Index auf eine Spalte mit dem Datentyp `text` erstellen kann, was mit den normalen Indizes von SQL Server nicht möglich ist. Nach dem Start des Assistenten erhalten Sie einen Einführungsbildschirm. Klicken Sie auf WEITER, um zum nächsten Dialogfeld zu gelangen (siehe Abbildung 13.16).



Abbildung 13.16:
Auswahl eines eindeutigen Index für die Tabelle

Wie schon erwähnt, wird für jeden Volltextindex, den Sie erstellen möchten, ein eindeutiger Index benötigt (als Zeilenlokator). Wählen Sie nun den eindeutigen Index aus (es gibt nur einen für *pub_info*) und klicken Sie auf WEITER . Damit kommen Sie zum Dialogfeld gemäß Abbildung 13.17.

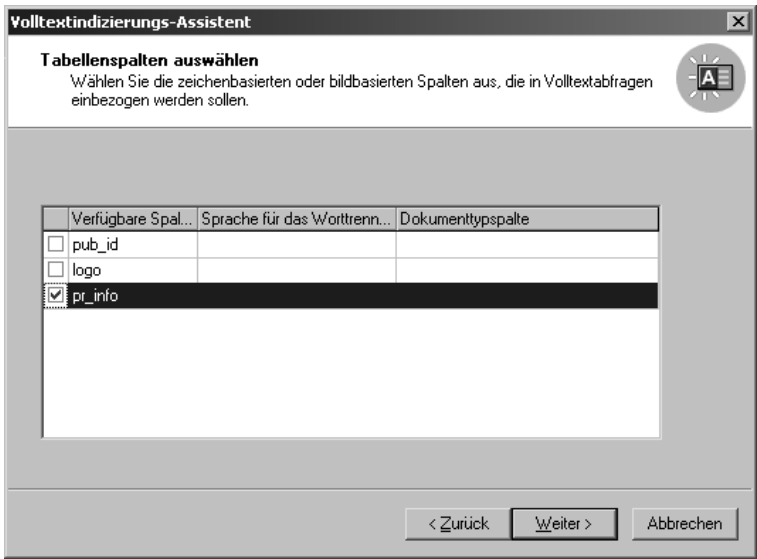


Abbildung 13.17:
Tabellenspalten
auswählen

Wie Sie sehen, wurde im Beispiel die Spalte *pr_info* gewählt, die den Datentyp *text* hat. Die anderen Optionen in diesem Dialogfeld sind neue Merkmale von SQL Server 2000: Sie können Dokumente indizieren, für die Dokumentfilter verfügbar sind. Dazu gehören zum Beispiel Microsoft Word-Dokumente. Man kann sogar die Sprache des Dokuments festlegen, so dass die Indexfilterung nach der Sprache der jeweiligen Ländereinstellung erfolgt. Klicken Sie nun auf WEITER, um zum nächsten Dialogfeld zu gelangen, in dem Sie einen Katalog auswählen können (siehe Abbildung 13.18). Legen Sie am besten einen neuen Katalog fest, um das Beispiel übersichtlich zu halten (wahrscheinlich ist ohnehin noch kein Katalog auf Ihrem System eingerichtet). Nennen Sie den neuen Katalog *pub_info* (wie es Abbildung 13.18 zeigt).

Klicken Sie auf WEITER. Sie haben nun die Möglichkeit, einen Zeitplan zum Füllen und Aktualisieren Ihres Volltextindex festzulegen. Klicken Sie auf WEITER, um das Dialogfeld für dieses Beispiel zu überspringen. Den Volltextindex können Sie später auch manuell aktualisieren. Klicken Sie nun auf FERTIG STELLEN. Nach kurzer Zeit erscheint das in Abbildung 13.19 dargestellte Dialogfeld mit der Meldung, dass Sie die Erstellung des Volltextindex beendet haben. Beachten Sie den Hinweis, dass der Volltextindex noch keine Daten enthält.

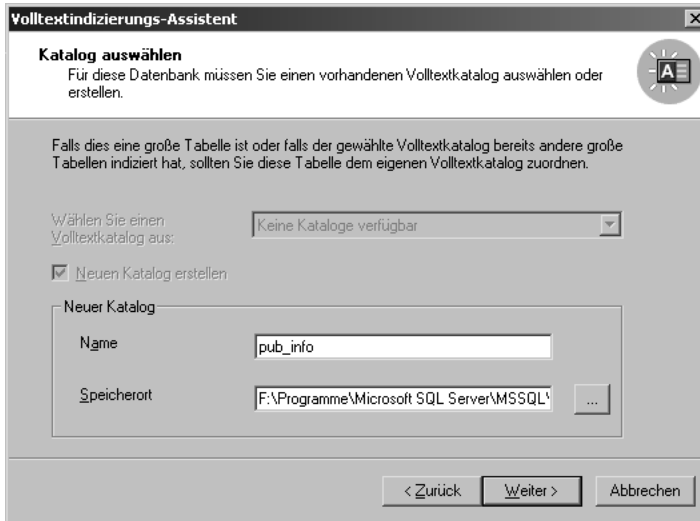


Abbildung 13.18:
Einen Katalog auswählen, um den Volltextindex zu speichern.



Abbildung 13.19:
Das abschließende Dialogfeld nach Fertigstellung des Assistenten

Klicken Sie nun auf OK und erweitern Sie wieder die Datenbank *pubs*, sofern dies noch nicht im SQL Server Enterprise Manager geschehen sein sollte. Markieren Sie den Eintrag *Volltextkatalog*, klicken Sie mit der rechten Maustaste im Detailbereich auf den eben erzeugten Katalog (namens *pub_info*) und wählen Sie VOLLSTÄNDIGES AUFFÜLLEN STARTEN aus dem Kontextmenü (siehe Abbildung 13.20).

Das war's auch schon! Der Index ist gefüllt und bereit zur Abfrage. An diesem Punkt werden die Dinge schwieriger. Die Abfrage von Volltextindizes ist in SQL Server 2000 nämlich keine einfache Sache verglichen mit normalen Abfragen, weil Sie sich erst noch mit einer speziellen Syntax für Volltextindizes bekannt machen müssen. Wenn Sie zum Beispiel die folgende Abfrage im Query Analyzer ausführen, erhalten Sie alle Zeilen zurück, die das Wort *sample* enthalten:

```
select pub_id, substring(pr_info,1,100) from pub_info
where CONTAINS (pr_info, 'sample')
```

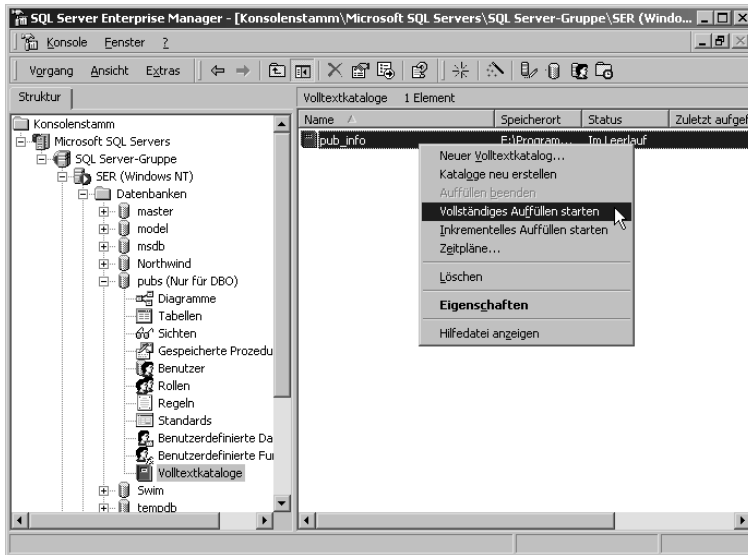


Abbildung 13.20:
Das Auffüllen des
Volltextkataloges
initialisieren

Volltextabfragen können kompliziert werden. Am besten ziehen Sie zunächst die Online-dokumentation von SQL Server zu Rate, bevor Sie anspruchsvollere Abfragen für Volltextindizes ausführen.

13.8 Zusammenfassung

Indizes sind nicht zwingend erforderlich, aber extrem nützlich für die Beschleunigung Ihrer Abfragen – sie stellen im Wesentlichen eine Art »Gaspedal« in SQL Server dar. Zum Erstellen von Indizes steht eine Reihe von Optionen zur Verfügung; außerdem müssen noch verschiedene Fragen zu Pflege und Verwaltung berücksichtigt werden. Ihre Indizes helfen Ihnen beim schnellen Auflösen von Abfragen und sorgen insgesamt für eine bessere Systemleistung, solange Sie sie richtig pflegen.

Vielleicht werden Sie nie in die Lage kommen, den Abfrageoptimierer eines Besseren belehren zu müssen – die Option dafür steht Ihnen jedenfalls in SQL Server in der Form von Abfragehinweisen zur Verfügung. Diese erfordern ein Verständnis der jeweiligen Abfrage, der abzufragenden Daten sowie der verfügbaren Indizes, die für die Optimierung in Frage kommen. Falls Sie sich hinsichtlich Ihres Wissensstandes bezüglich all dieser Dinge nicht sicher sind, sollten Sie am besten den SQL Server für die Auswahl der richtigen Indizes Sorge tragen lassen. Sofern Sie Ihre Indizes sorgfältig gepflegt haben, wird seine Entscheidung fast immer richtig sein.

Indizes können Sie fast gänzlich in Transact-SQL oder über den SQL Server Enterprise Manager verwalten. Obwohl Sie wahrscheinlich meistens die Dialogfelder des Enterprise Managers nutzen werden, sollten Sie sich auf lange Sicht die erforderlichen Kenntnisse über beide Verfahrensweisen aneignen. Natürlich ist der Enterprise Manager einfacher im Umgang, als es die exakte Handhabung der Syntax von Transact-SQL ist.

13.9 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Welche Unterschiede gibt es zwischen gruppierten und nicht gruppierten Indizes?*
- A** Die Anzahl der verfügbaren Indizes (1 für gruppierte und 249 für nicht gruppierte) und der Inhalt der Blattebene des Index (Daten beim gruppierten Index und Zeilen, die auf Daten zeigen, beim nicht gruppierten Index).
- F** *Wie entscheidet SQL Server, wann welcher Index zu verwenden ist?*
- A** Er analysiert sowohl die WHERE-Klausel in Ihrer Abfrage als auch die Statistiken des statistischen Blob sowie die in der Systemtabelle sysindexes enthaltenen Informationen.
- F** *Wann macht die Angabe eines Füllfaktors für einen Index Sinn?*
- A** Wenn Sie in Ihren Indizes freien Platz für das Wachstum der Daten und zum Reduzieren von Seitenteilungen lassen möchten.
- F** *Wie überschreibt man die Indexwahl des SQL Servers?*
- A** Mit Abfragehinweisen wie etwa `SELECT ... FROM indextab (index = 1)`.

Kontrollfragen

1. Wie erzwingen Sie, dass SQL Server einen Tabellenscan wählt?
2. Wie viel freier Platz ist zum Erstellen eines gruppierten Index erforderlich?
3. Welche Option müssen Sie einschalten, damit die Indexwahl von SQL Server überprüft werden kann?

Übungen

1. Schreiben Sie eine Abfrage zum Auswählen von Daten aus der Tabelle *sales* in der Datenbank *pubs* mit der Bedingung `title_id = 'BU1032'`. Zeigen Sie dann die Indexwahl von SQL Server und die erforderlichen E/A-Operationen auf.
2. Überschreiben Sie die Indexwahl von Übung 1 und verwenden Sie einen gruppierten Index.
3. Erstellen Sie eine neue Tabelle. Richten Sie dafür einen eindeutigen gruppierten Index und zwei nicht gruppierte Indizes ein – jeden mit einem anderen Füllfaktor. Reorganisieren Sie anschließend die Indizes mit `DBCC DBREINDEX` und setzen Sie den Füllfaktor auf 50 zurück.
4. Wiederholen Sie Übung 3, nehmen Sie aber die `DBCC DBREINDEX`- und Füllfaktoränderungen über den SQL Enterprise Manager vor.



Datenintegrität

Gestern haben Sie Indizes und alle diesbezüglichen Optionen in SQL Server kennen gelernt. Indizes stellen zwar einen großen Gewinn hinsichtlich der Systemleistung dar, können aber, wenn es zu viele sind, ein System auch in die Knie zwingen. Die Auswahl der richtigen Indexstruktur ist entscheidend für die Leistung einer Datenbank.

Die heutige Lektion befasst sich mit der Datenintegrität. Dabei lernen Sie zwei Mechanismen kennen, um die Datenintegrität zu erzwingen: prozedurale und deklarative. Die deklarative Integrität von SQL Server kombiniert mit der `IDENTITY`-Eigenschaft wird in der Microsoft-Dokumentation auch als *deklarative referenzielle Integrität* (DRI) bezeichnet, die jedoch weitaus mehr als das umfasst, was der Name aussagt.

14.1 Wie die Datenintegrität gewährleistet wird

Bei der Gewährleistung der Datenintegrität versuchen Sie sicherzustellen, dass die Daten in Ihrer Datenbank sowohl im wörtlichen Sinne als auch von den geschäftlichen Arbeitsabläufen her einwandfrei sind. Wie schon oben angedeutet, können Sie die Integrität entweder prozedural oder deklarativ gewährleisten.



Prozedurale Integrität bedeutet den Einsatz von Programmierstrukturen und separaten Objekten, um die Datenintegrität zu erzwingen; eine Maßnahme, die in der Regel auf der Anwendungsprogrammzebene vorgenommen wird. Allerdings bedeutet dies auch, dass alle Datenmodifikationen, die außerhalb dieses Programms stattfinden, nicht Ihren Integritätsregeln unterliegen. Aus diesem Grund erscheint es sinnvoll, möglichst viele dieser Datenintegritätsregeln auf die Datenbankebene zu verlagern. Die Integrität erzwingen Sie dann über Objekte wie Trigger und gespeicherte Prozeduren, auf die Tag 15 näher eingeht. Im Rahmen der Datenintegrität behandelt die heutige Lektion Standards und Regeln sowie benutzerdefinierte Datentypen.



Ein anderer Lösungsansatz ist die *deklarative Integrität*. Hierbei ist der Mechanismus zur Erzwingung der Integrität als Teil der Objekte (Tabellen) in der Datenbank deklariert und damit direkt in diese Objekte integriert. Mit diesen Objekten haben Sie vor allem zu tun, weil sie die entscheidende Funktionalität liefern, die anderweitig nur durch großen Programmieraufwand zu erreichen ist.

14.2 Typen der Integrität

Zur Gewährleistung der Integrität im SQL Server 2000 werden zahlreiche Komponenten eingesetzt, die sich folgenden Kategorien zuordnen lassen: Domänenintegrität, referenzielle Integrität und Entitätsintegrität.

Domänenintegrität



Mit *Domänenintegrität* bezeichnet man die Tatsache, dass die Datentypen allein schon die Festlegung unterstützen, welche Werte für eine gegebene Spalte gültig sind. Die *Domäne* ist einfach nur der Satz der gültigen Werte für eine bestimmte Spalte. Eine weitere Option zur Festlegung der gültigen Werte in der Domäne ist die NULL-Zulässigkeit – d.h. in diesem Fall, ob der unbekannte Wert (NULL) gültig ist. Bei Verwendung der »traditionellen« Integritätsobjekte von SQL Server erlegen Sie der Domäne einer Spalte durch benutzerdefinierte Datentypen, Regeln und Standards weitere Einschränkungen auf. Andererseits können Sie die Domänenintegrität auch mit ANSI-Einschränkungen (Standard- und CHECK-Einschränkungen) erzwingen. Beachten Sie, dass die Datentypen und die NULL-Zulässigkeit immer eine Rolle spielen, während die anderen Komponenten optional sind.

Referenzielle Integrität



Die *referenzielle Integrität* bezieht sich auf die Verwaltung der Beziehungen zwischen den Daten in mehreren Tabellen. Die referenzielle Integrität wird, wie schon weiter oben erwähnt, über DRI gewährleistet. Zwar können Sie die Integrität auch durch den Einsatz von Triggern und Programmlogik herbeiführen, jedoch würde dies eine Menge an Mehraufwand bedeuten.

SQL Server 2000 bietet zwei Arten von kaskadierenden Aktionen für Fremdschlüssel: kaskadierendes Löschen (neu in SQL Server 2000) und das Erzwingen von »keiner Aktion«. Letztere Einschränkung bedeutet Folgendes: Wenn Sie einen Kunden löschen wollen und referenzielle Integrität »ohne Aktion« wirksam ist, lässt sich der Kunde nicht löschen, wenn noch Rechnungen für ihn ausstehen. Ist dagegen kaskadierendes referenzielles Löschen aktiv und Sie löschen den Kunden, werden auch alle Rechnungen aus der Datenbank gelöscht. Um die kaskadierende Löschintegrität in allen vorherigen Versionen von SQL Server zu realisieren, musste man das Ganze in eigener Regie programmieren (mithilfe von Triggern, gespeicherten Prozeduren oder Standard-Transact-SQL-Anweisungen).

Diese bedeutende Fähigkeit von SQL Server 2000 soll einen hohen Programmieraufwand auf Seiten des Datenbankentwicklers vermeiden helfen.

Entitätsintegrität



Der letzte Integritätstyp, den Sie kennen sollten, ist die *Entitätsintegrität*. Damit ist gemeint, dass Sie jede Zeile in einer Tabelle eindeutig identifizieren können. Sie erreichen dies mit einem eindeutigen Index (siehe die gestrige Lektion) oder über die deklarative Integrität (Primärschlüssel oder eindeutige Einschränkungen), um die es heute geht.

14.3 Traditionelle Methoden der Integrität

Traditionelle (d.h. abwärtskompatible) Methoden zur Sicherstellung der Integrität beinhalten benutzerdefinierte Datentypen, Standards und Regeln. Die folgenden Abschnitte zeigen, wie Sie die Datenbankintegrität mit diesen Objekten durchsetzen können.



Obwohl Sie zur Erzwingung der Integrität auch benutzerdefinierte Datentypen, Standards und Regeln verwenden können, ist es ratsam, von diesen Mechanismen abzusehen und statt dessen nach Möglichkeit die ANSI-Einschränkungen einzusetzen, die generell flexibler sind. Ein weiterer Vorteil der ANSI-Einschränkungen besteht darin, dass der SQL Server-Abfrageoptimierer (der Ihnen bei der Auswahl des besten physischen Ausführungsplanes für Abfragen hilft) ANSI-Einschränkungen versteht und benutzen kann, um seine Entscheidungen über die Zugriffspläne zu unterstützen.

Benutzerdefinierte Datentypen

Benutzerdefinierte Datentypen ermöglichen Ihnen die Sicherstellung der Domänenintegrität. Bei der Erstellung einer Tabelle definieren Sie normalerweise jede Spalte mit einem vom System bereitgestellten Datentyp zusammen mit der Festlegung, ob in der Spalte NULL-Werte erlaubt sein sollen. Zwar können Sie auch eigene Datentypen in SQL Server definieren, jedoch müssen diese benutzerdefinierten Datentypen unter Verwendung der systemseitig bereitgestellten Typen definiert werden. SQL Server erlaubt Ihnen nicht, beispielsweise Definitionen von Strukturen als neue Datentypen vorzunehmen.

Die benutzerdefinierten Datentypen können Sie dazu einsetzen, wenn Sie die logischen Datentypen Ihres Datenmodells in die physischen Datentypen von SQL Server übertragen

möchten. Haben Sie beispielsweise in mehreren Tabellen die Postleitzahlen als Bestandteil Ihres Datenmodells gewählt, dann kann ein benutzerdefinierter Datentyp sinnvoll sein. Wenn Sie im Modell dafür `char(10)` vorgesehen haben, dann können Sie nun einen Datentyp wie etwa `postleitzahl_datentyp` erstellen, der ebenfalls als `char(10)` definiert ist.

Zum Erstellen dieses Datentyps rufen Sie die gespeicherte Systemprozedur `sp_addtype` wie folgt auf:



```
Exec sp_addtype postleitzahl_datentyp, 'char(10)'
```

Die gespeicherte Systemprozedur `sp_addtype` verwendet folgende Syntax:



```
sp_addtype Typname, Phystyp [, Nulltyp][, Besitzer]
```

Die Argumente haben folgende Bedeutung:

- *Typname* ist der Name Ihres benutzerdefinierten Datentyps, den Sie dann anstelle des Systemdatentyps in Ihrer `CREATE TABLE`-Anweisung verwenden.
- *Phystyp* ist der Systemdatentyp, der zu verwenden ist, wenn Sie auf diesen benutzerdefinierten Datentyp zugreifen. Die Anführungszeichen sind zwar in manchen Fällen optional, aber es ist sicherer, sie immer zu benutzen.
- *Nulltyp* ist optional. Ist dieses Argument (in Anführungszeichen) angegeben, entscheidet es über die NULL-Zulässigkeit einer Spalte, wenn Sie sie nicht in Ihrer `CREATE TABLE`-Anweisung festlegen. Falls Sie diesen Parameter nicht spezifizieren, wird für den Datentyp die systemseitig festgelegte NULL-Zulässigkeit (die in SQL Server 2000 standardmäßig NULL ist, sofern Sie dies nicht ändern) verwendet.



Aus Sicherheitsgründen sollten Sie in Ihrer `CREATE TABLE`-Anweisung die NULL-Zulässigkeit Ihrer Spalten immer explizit angeben und sich nicht auf diese Funktionalität verlassen.

- *Besitzer* ist ebenfalls optional. Ist dieses Argument (in Anführungszeichen) angegeben, legt es den Besitzer des benutzerdefinierten Datentyps fest. Der Standard ist der Ersteller des Datentyps.

Typen werden in der Systemtabelle `systypes`, die sich in jeder Datenbank befindet, gespeichert.

Zum Erstellen des Datentyps `laendercode_typ` des Typs `char(2)`, der standardmäßig keine NULL-Werte erlaubt, führen Sie den folgenden Aufruf aus:



```
EXEC sp_addtype laendercode_typ, 'char(2)', 'not null'
```

Diesen Typ können Sie beispielsweise wie folgt benutzen:



```
CREATE TABLE meinetabelle2  
(coll laendercode_type)
```

`coll` erlaubt keine NULL-Werte, weil Sie im Datentyp festgelegt haben, dass diese nicht erlaubt sein sollen, was ungeachtet dessen gilt, ob die Datenbank standardmäßig auf NULL-Zulässigkeit eingestellt ist oder nicht.

Wenn Sie den benutzerdefinierten Datentyp nicht mehr benötigen (weil Sie ihn entweder nie gebraucht oder alle Tabellen gelöscht haben, in denen er Verwendung fand), können Sie ihn mit der gespeicherten Systemprozedur `sp_droptype` löschen:



```
sp_droptype Typname
```

Das Argument *Typname* ist der Name des benutzerdefinierten Datentyps, den Sie löschen möchten.

Sollte sich ein benutzerdefinierter Datentyp noch in Gebrauch befinden, dann werden Sie beim Versuch, ihn zu löschen, einen Fehler erhalten. Bevor Sie ihn endgültig entfernen können, müssen Sie zunächst alle Tabellen löschen, die diesen Datentyp verwenden.

Wie Sie wahrscheinlich erwartet haben, stellt der SQL Server Enterprise Manager eine Schnittstelle zur Unterstützung von benutzerdefinierten Datentypen zur Verfügung. Erweitern Sie dazu die Datenbank, mit der Sie arbeiten möchten (denken Sie an dieser Stelle daran, dass Datentypen datenbankspezifisch sind) und lassen Sie sich die Datentypen anzeigen, die Sie in der Datenbank erstellt haben. Sie erscheinen im Detailbereich des SQL Server Enterprise Managers. Zum Erstellen eines neuen benutzerdefinierten Datentyps klicken Sie mit der rechten Maustaste auf das Symbol *Benutzerdefinierte Datentypen* in der Konsolenstruktur und wählen aus dem Kontextmenü **NEUER BENUTZERDEFINIERTER DATENTYP**. Alternativ dazu können Sie auch, wenn Sie den Ordner *Benutzerdefinierte Datentypen* markiert haben, **VORGANG / NEUER BENUTZERDEFINIERTER DATENTYP** aus dem Menü wählen oder auf die Schaltfläche **NEU** (sieht wie ein gelber Stern aus) in der Symbolleiste des Enterprise Managers klicken. In jedem Fall erhalten Sie das Dialogfeld **BENUTZERDEFINIERTER DATENTYP – EIGENSCHAFTEN** (siehe Abbildung 14.1).



Abbildung 14.1:
Das Dialogfeld Benutzerdefinierter
Datentyp – Eigenschaften

Zum Hinzufügen eines neuen benutzerdefinierten Datentyps geben Sie einfach den Namen des Typs ein, legen einen Systemdatentyp fest, machen eine Angabe zur Länge (wird unter Umständen abgeblendet dargestellt) und markieren gegebenenfalls das Kontrollkästchen NULL-WERTE ZULASSEN, falls Sie für den Datentyp standardmäßig NULL-Werte zulassen möchten. Weitere Informationen über Standards und Regeln erhalten Sie in der morgigen Lektion. Klicken Sie zum Hinzufügen des benutzerdefinierten Datentyps auf OK.

Zum Löschen des benutzerdefinierten Datentyps klicken Sie mit der rechten Maustaste auf den Datentyp (bzw. die Datentypen) in der rechten Fensterhälfte des SQL Server Enterprise Managers und wählen LÖSCHEN aus dem Kontextmenü, woraufhin das Dialogfeld OBJEKTE LÖSCHEN erscheint. Klicken Sie zum Löschen der benutzerdefinierten Datentypen auf ALLE LÖSCHEN. Beachten Sie an dieser Stelle noch einmal, dass Sie keinen benutzerdefinierten Datentyp löschen können, der sich noch in Gebrauch befindet. Ob Ihr Datentyp in Gebrauch ist, finden Sie heraus, indem Sie ihn markieren und dann die Schaltfläche ABHÄNGIGKEITEN ANZEIGEN anklicken. Sie erhalten daraufhin eine Liste der Objekte angezeigt, die Ihren benutzerdefinierten Datentyp verwenden. Diese Liste besteht aus den Tabellen und Spalten, in denen dieser Typ verwendet wird.



Falls Sie diese Liste der Abhängigkeiten (siehe Abbildung 14.2) sehen möchten, ohne einen benutzerdefinierten Datentyp zu löschen, klicken Sie doppelt auf den benutzerdefinierten Datentyp (oder klicken mit der rechten Maustaste und wählen EIGENSCHAFTEN aus dem Kontextmenü) und klicken dann auf WIRD VERWENDET VON. Alternativ dazu können Sie den benutzerdefinierten Datentyp auch anklicken und ALLE TASKS / ABHÄNGIGKEITEN ANZEIGEN aus dem Kontextmenü wählen.



Abbildung 14.2:
Die Liste der Abhängigkei-
ten für einen benutzerdefi-
nierten Datentyp

Standards

Standardwerte werden dazu verwendet, einen Wert zu definieren, der dann einer Spalte hinzugefügt wird, wenn Sie selbst in diese Spalte keinen Wert einfügen. Es gibt zwei Arten von Standardwerten in SQL Server: ANSI-Einschränkungen und selbstständige Objekte namens Standards. In diesem Abschnitt werden Sie die Standards kennen lernen, während die ANSI-Einschränkungen später in dieser Lektion untersucht werden. Für die Übungen benötigen Sie die Berechtigung für die Anweisung CREATE TABLE oder aber Sie müssen ein Mitglied der festen Datenbankrollen *db_owner* oder *db_ddladmin* sein.



Es empfiehlt sich, dass Sie anstelle der mit CREATE DEFAULT erzeugten Standards die ANSI-Einschränkungen benutzen. Die ANSI-Einschränkungen bieten dieselbe Funktionalität bei Sicherstellung der Konsistenz gegenüber anderen ANSI-Einschränkungen.

Wie Sie vielleicht schon vermutet haben, können Sie Standards mit der folgenden CREATE DEFAULT-Anweisung erstellen:



```
CREATE DEFAULT Standardname
AS konstanter_Ausdruck
```


Die Argumente haben folgende Bedeutung:

- *Standardname* muss in der Datenbank eindeutig sein und den Regeln für SQL Server-Bezeichner entsprechen.
- *konstanter_Ausdruck* ist eine Konstante, die für den Datentyp jeder Spalte, die Sie für diesen Datentyp verwenden möchten, geeignet ist. Falls Sie beispielsweise einen Datentyp `char(2)` referenzieren, dann kann Ihre Konstante entweder `x` oder `xx` sein, wobei `x` jedes gültige Zeichen repräsentiert. Bei einer Ganzzahl sollte Ihre Konstante ebenfalls eine Ganzzahl sein. Falls Sie binäre Daten referenzieren, müssen diese mit dem Zeichen `0x` beginnen (wie etwa `0x13` für den binären Wert 13). Währungsdatentypen müssen mit einem Dollarzeichen beginnen (`$`). Unicode-Datentypen fangen mit dem Zeichen `N` an. Sie können auch Systemfunktionen wie `getdate()` einsetzen, sofern diese die für den Datentyp geeigneten Daten zurückliefern.

Standards werden in der Systemtabelle *sysobjects* jeder Datenbank aufgezeichnet. Der Text der Anweisung `CREATE DEFAULT` wird in der Tabelle *syscomments* gespeichert. Wie bei anderen Objekten in SQL Server 2000 auch wird die Definition des Objekts im Unicode-Format festgehalten.

Wie Sie schon am 11. Tag gelernt haben, können Sie Standards während einer Einfügeoperation (`INSERT`) entweder mithilfe des Schlüsselwortes `DEFAULT` oder einfach durch Weglassen der Spalte in der Spaltenliste festlegen.

Es gibt eine Reihe von Punkten, die Sie beim Einsatz von Standards beachten müssen:

- Der Standardwert muss den Regeln oder ANSI-CHECK-Einschränkungen entsprechen, die Sie derselben Spalte oder den benutzerdefinierten Datentypen zugewiesen haben. Falls Sie auf der Spalte eine Regel liegen haben (wie etwa bei einer Spalte *Telefonnummer*), die diese Spalte in der Form `(xxxxxx)xxxxxxx` formatiert, dann kann Ihr Standardwert nicht `Unknown` lauten, sondern muss etwas in der Art wie `(00000)0000000` sein. Sie werden in Kürze Näheres über Regeln erfahren.
- Der Standardwert muss mit dem Datentyp der Spalte oder dem benutzerdefinierten Datentyp, dem er zugewiesen ist, kompatibel sein. Sie können keinen Standardwert `Unknown` für eine Spalte mit `char(2)` festlegen. Dementsprechend funktioniert ein Standardwert `Unknown` nicht für eine numerische Spalte. Beachten Sie an dieser Stelle, dass `Unknown` hier die literale Zeichenfolge und nicht der `NULL`-Wert ist.

Bis jetzt dürfte klargeworden sein, dass Standards auf Spalten oder benutzerdefinierte Datentypen Anwendung finden können. Wenn Sie einen Standardwert erstellen, ist dieser ein separates Objekt in der Datenbank und nicht an ein bestimmtes anderes Objekt gebunden. Um Standards auch tatsächlich nutzen zu können, müssen Sie sie an eine Spalte oder einen benutzerdefinierten Datentyp binden. Zum Binden eines Standards setzen Sie die gespeicherte Systemprozedur `sp_bindefault` ein:



`sp_bindefault Defname, Objname [, futureonly]`

Die Argumente haben folgende Bedeutung:

- *Defname* ist der Name des Standards, den Sie vorher erstellt haben.
- *Objname* ist der Name des Objekts, an den Sie Ihren Standardwert binden möchten. Er muss in Anführungszeichen eingeschlossen werden. Sie können entweder eine Spalte oder einen benutzerdefinierten Datentyp binden. SQL Server ist darüber informiert, dass Sie eine Spalte binden möchten, wenn sich Ihr Text in Anführungszeichen in dem Format *Tabellenname.Spaltenname* befindet. Erscheint dagegen kein Punkt im Text in Anführungszeichen, dann wird angenommen, dass es sich um den Namen eines benutzerdefinierten Datentyps handelt. Einen Standardwert können Sie nicht an einen systemseitigen Datentyp binden.
- *futureonly* gilt nur für benutzerdefinierte Datentypen. Falls angegeben, wird der Standardwert an keiner Stelle angewendet, an der der benutzerdefinierte Datentyp bereits Verwendung gefunden hat. Der Standardwert wird aber bei jeder zukünftigen Verwendung des benutzerdefinierten Datentyps gebunden. Es ist nicht ratsam, diese Option zu verwenden, weil es manchmal etwas verwirrend sein kann, dass der Standardwert in Abhängigkeit davon, wann die gespeicherte Systemprozedur `sp_bindefault` ausgeführt wurde, verwendet wird oder nicht.

Nach Abschluss der Bindung erhalten alle benutzerdefinierten Datentypen bzw. Spalten, an die der Standardwert gebunden ist, bei einer Einfügung diesen Wert, falls er anwendbar ist.

Um die Bindung eines Standards an eine Spalte oder einen benutzerdefinierten Datentypen aufzuheben, führen Sie die gespeicherte Systemprozedur `sp_unbindefault` aus:



`sp_unbindefault Objname [, futureonly]`

Die Argumente haben folgende Bedeutung:

- *Objname* entspricht dem gleichnamigen Objekt in der gespeicherten Systemprozedur `sp_bindefault`.
- *futureonly* gilt (wie bei `sp_bindefault`) nur für benutzerdefinierte Datentypen und hebt die Bindung des Standards an den benutzerdefinierten Datentyp auf. Der Standardwert ist allerdings noch überall dort in Kraft, wo er ursprünglich auch verwendet worden ist. Dagegen tritt bei einer zukünftigen Verwendung des benutzerdefinierten Datentyps der Standardwert nicht mehr in Kraft. Aus denselben schon zuvor

geschilderten Gründen ist der Gebrauch dieser Option nicht empfehlenswert. Um den Standardwert löschen zu können, müssen Sie explizit die Bindung zu allen infrage kommenden Spalten aufheben, für die der Standardwert wirksam gewesen ist, bevor Sie Prozedur `sp_unbindefault` mit der Option `futureonly` ausgeführt haben.

Nachdem Sie die Bindung eines Standards an einen Datentyp oder eine Spalte aufgehoben haben, wird er nicht mehr während einer Einfügeoperation verwendet. Einen Standardwert können Sie erst dann löschen, wenn Sie ihn von sämtlichen Datentypen und Spalten entbunden haben.

Einen Standardwert löschen Sie mit der Anweisung `DROP DEFAULT`:



```
DROP DEFAULT Default_Name [, Default_Name...]
```

Dies dürfte ohne weiteres zu verstehen sein. Sie können beliebig viele Standards über eine einzige Anweisung löschen. Falls Sie selbst der Besitzer des Standards sind, haben Sie auch das inhärente Recht zum Löschen. Andernfalls müssen Sie ein Mitglied der festen Datenbankrollen `db_owner` oder `db_ddladmin` oder der festen Datenbankrolle `sysadmin` sein.

Mit dem folgenden Beispielcode können Sie eine Tabelle und einen benutzerdefinierten Datentyp erstellen und dann einige Standardwerte zuweisen:



```
Use pubs
Go
Exec sp_addtype my_uddt_type, money
Go
CREATE DEFAULT intdefault as 0
Go
CREATE DEFAULT char5default as 'Hello'
Go
CREATE DEFAULT moneydefault as $10.00
Go
CREATE TABLE mytab
(intcol int not null,
 char5col char(5) not null,
 uddtcol my_uddt_type not null)
Go
```

Jetzt binden Sie die Standards an die Datentypen und die Spalten:



```
Exec sp_bindefault moneydefault, 'my_uddt_type'
Exec sp_bindefault intdefault, 'mytab.intcol'
Exec sp_bindefault char5default, 'mytab.char5col'
Go
```

Fügen Sie nun eine Zeile mit Standards ein und rufen Sie sie ab:



```
INSERT mytab DEFAULT VALUES
Go
SELECT * FROM mytab
Go
```

Die Ergebnismenge zeigt, dass die Standards tatsächlich verwendet wurden:



```
(1 row(s) affected)

intcol      char5col  uddtcol
-----
0           Hello    10.0000

(1 row(s) affected)
```

Sehen Sie sich dieses Ergebnis über den SQL Server Enterprise Manager an. Erweitern Sie den Ordner *Datenbanken* und dann die Datenbank *pubs*. Markieren Sie den Eintrag *Standards* in der Konsolenstruktur, um die Standards im Detailbereich anzuzeigen (falls Sie die eben erzeugten Standards nicht sehen sollten, klicken Sie in der Symbolleiste auf die Schaltfläche AKTUALISIEREN). Klicken Sie mit der rechten Maustaste auf das Symbol *Standards* und wählen Sie NEUER STANDARD. Sie erhalten nun das Dialogfeld STANDARDEIGENSCHAFTEN (siehe Abbildung 14.3). Dieses Dialogfeld erhalten Sie auch durch Wahl von VORGANG / NEUER STANDARD im Menü.



Abbildung 14.3:
Das Dialogfeld Standardeigen-
schaften

Geben Sie nun den Namen des Standards ein, den Sie erstellen möchten. Geben Sie im Feld WERT dann die Zeichenfolge, den numerischen, den Währungs- oder den binären Wert ein, den Sie als Standardwert verwenden möchten. Nach Beginn der Eingabe in die-

sem Fenster wird die Schaltfläche OK verfügbar. Klicken Sie auf OK, um den Standardwert in die Datenbank zu übernehmen.

Nach Erstellung Ihres Standards können Sie ihn sich im Detailbereich des SQL Server Enterprise Managers ansehen. Um einen Standard zu ändern, anzuzeigen oder die Bindung zu ändern, klicken Sie mit der rechten Maustaste auf den Standard und wählen EIGENSCHAFTEN (bzw. doppelklicken auf den Standard). Falls Sie den Standard an eine oder mehrere Spalten in Ihrer Datenbank binden möchten, klicken Sie im Dialogfeld Standardeigenschaften auf die Schaltfläche SPALTEN BINDEN. Damit gelangen Sie zum Dialogfeld gemäß Abbildung 14.4.



Abbildung 14.4:
Einen Standard mit dem
Enterprise Manager an
eine Spalte binden

Wählen Sie dazu im Listenfeld TABELLE die Tabelle aus, deren Spalten Sie für die Bindung benutzen möchten und markieren Sie dann die Spalte bzw. die Spalten, die Sie binden möchten. Durch Klicken auf die Schaltfläche HINZUFÜGEN verschieben Sie sie in das Listenfeld GEBUNDENE SPALTEN. Wenn Sie OK oder ÜBERNEHMEN anklicken, wird die Bindung vorgenommen. Ist die Bindung aus irgendeinem Grund fehlgeschlagen, erhalten Sie eine entsprechende Fehlermeldung. Um die Bindung zu einer Spalte aufzuheben, suchen Sie einfach nur die entsprechende Tabelle auf und markieren die Spalte im Listenfeld GEBUNDENE SPALTEN. Anschließend klicken Sie auf die Schaltfläche ENTFERNEN.

Zum Ändern von benutzerdefinierten Datentypen klicken Sie auf die Schaltfläche UDTs BINDEN. Daraufhin öffnet sich das Dialogfeld STANDARD AN BENUTZERDEFINIERTER DATENTYPEN BINDEN (siehe Abbildung 14.5). Sie erhalten darin eine Liste sämtlicher existierender benutzerdefinierter Datentypen. Sie markieren dann einfach nur das jeweilige Kontrollkästchen in der Spalte BINDEN, um den Standard an den benutzerdefinierten Datentyp zu binden.



Abbildung 14.5:
Einen benutzerdefinierten
Datentyp mit dem Enterprise
Manager binden

Zum Löschen des Standards klicken Sie ihn mit der rechten Maustaste an und wählen LÖSCHEN. Wie auch bei den benutzerdefinierten Datentypen können Sie die Abhängigkeiten überprüfen, die der Standard hat. Außerdem müssen Sie vor dem Löschen des Standards die Bindung zu sämtlichen benutzerdefinierten Datentypen und Spalten aufheben.

Regeln

Regeln ermöglichen durch eine erweiterte Überprüfung von Werten auf ihre Gültigkeit eine erhöhte Sicherheit bei der Domänenintegrität. Mit Regeln stellt man sicher, dass Werte

- einem Muster entsprechen (ähnlich einer LIKE-Klausel)

- einer Liste von Werten entsprechen (ähnlich einer IN-Klausel)
- in einen Bereich von Werten fallen (ähnlich einer BETWEEN-Klausel)

Regeln sind wie Standards separate Objekte, die eine bestimmte Berechtigung zum Erstellen voraussetzen. Sie müssen zum Erstellen einer Regel ein Mitglied der Rollen *db_owner* oder *sysadmin* sein. Regeln werden wie Ihre Standards in denselben Systemtabellen *sysobjects* und *syscomments* gespeichert. Bei der Ausführung von Einfügungen und Aktualisierungen werden Verletzungen der Regeln überwacht (und zwar immer dann, wenn die Spalte, die von der Regel betroffen ist, referenziert wird).



Auf die Gefahr hin, dass es Ihnen schon wie eine gesprungene Schallplatte vorkommt: In den weitaus meisten Fällen sollten Sie für diesen Integritätstyp ANSI-Einschränkungen statt Regeln implementieren. Allerdings kann es spezielle Fälle geben, in denen Sie beispielsweise ein und dieselbe Integritätsprüfung auf 100 Tabellen anwenden möchten, wobei es dann bequem wäre, wenn man diese einmal erstellt und dann einfach nur noch die Regel 100 mal aufruft. Jedoch ist auch hier die Verwendung von ANSI-Einschränkungen ratsamer, weil der SQL Server-Abfrageoptimierer Einschränkungen für die Abfrageoptimierung auswertet, was zur Folge haben kann, dass die Verwendung von Einschränkungen anstelle von Regeln unter Umständen Ihre Abfrage schneller macht.

Regeln erstellen Sie über die Anweisung `CREATE RULE`:



```
CREATE RULE Regelname
AS bedingter_Ausdruck
```

Die Argumente haben folgende Bedeutung:

- *Regelname* ist ein gültiger und eindeutiger Name in der Datenbank, in der die Regel erstellt wird.
- *bedingter_Ausdruck* hat das Format `@Variablenname <WHERE-Klausel>`. Die `<WHERE-Klausel>` kann jede gültige WHERE-Klausel einschließlich arithmetischer Operatoren, BETWEEN, IN, LIKE, AND, OR, NOT und anderer Operatoren sein. Eine Regel kann jedoch keine variablen Werte oder andere Spalten in der Datenbank referenzieren. Hierzu müssen Sie eine Prüfeinschränkung oder einen Trigger verwenden.

Sollen zum Beispiel die Werte in einer Spalte *Teilenummer* mit einem p oder t beginnen, formulieren Sie zunächst die folgende Regel:

```
CREATE RULE myrule AS @myvar like 'p%' OR @myvar like 't%'
```



Der Name @myvar ist rein willkürlich gewählt- Sie können auch @hanswilhelm sagen, wenn Sie wollen. Häufig wählt man einen Variablennamen in Anlehnung an den Namen der Spalte oder des Datentyps, auf den er sich bezieht.

Regeln werden wie Standards ebenfalls an einen Datentyp oder eine Spalte gebunden, was man über die gespeicherte Systemprozedur `sp_bindrule` erledigt:



`sp_bindrule Regelname, Objname [, futureonly]`

Die Argumente haben folgende Bedeutung:

- *Regelname* ist die Regel, für die Sie die Bindung vornehmen möchten.
- *Objname* ist (wie bereits weiter oben erläutert) entweder eine Kombination von *Tabellenname.Spaltenname* oder ein benutzerdefinierter Datentyp.
- *futureonly* hat die gleiche Bedeutung wie bei `sp_bindefault`.

Wenn die Regeln gebunden sind, arbeiten sie wie Standards: Bei der Ausführung einer INSERT-Operation werden die Daten, die Sie einfügen wollen, anhand der Regel überprüft, um ihre Gültigkeit sicherzustellen. Sind sie nicht gültig, erhalten Sie eine Fehlermeldung der folgenden Art:

Server: Nachr.-Nr. 513, Schweregrad 16, Status 1, Zeile 1
 Beim Einfügen oder Aktualisieren einer Spalte trat ein Konflikt mit einer Regel auf, die in einer vorherigen CREATE RULE-Anweisung festgelegt wurde. Die Anweisung wurde beendet. Der Konflikt trat in der pubs-Datenbank, mytab-Tabelle, intcol-Spalte auf.
 Die Anweisung wurde beendet.

Um die Bindung einer Regel aufzuheben, rufen Sie die gespeicherte Systemprozedur `sp_unbindrule` auf:



`sp_unbindrule Objname [, futureonly]`

In dieser Syntax ist *Objname* der Name des benutzerdefinierten Datentyps oder die Kombination *Tabellenname.Spaltenname*, das Argument *futureonly* hat die gleiche Bedeutung wie bei `sp_unbindefault`.

Zum Löschen einer Regel führen Sie die Anweisung `DROP RULE` aus:

`DROP RULE Regelname [, Regelname...]`

Bei Verwendung der Anweisung `DROP RULE` geben Sie einfach nur die Regelnamen an, die Sie löschen möchten. Wie auch bei der Anweisung `DROP DEFAULT` können Sie mit einem einzigen Befehl so viele Regeln auf einmal löschen, wie Sie möchten. Die Regeln dürfen dabei nicht an Spalten oder benutzerdefinierte Datentypen gebunden sein.

Für eine Untersuchung der Arbeitsweise der Regeln wie auch ihrer Zusammenarbeit mit Standards können Sie dieselbe Tabelle benutzen, die auch schon weiter oben eingesetzt worden ist. Beachten Sie, dass die folgenden Anweisungen voraussetzen, dass Sie das Skript für die vorher erwähnten Standards ausgeführt haben.



```
CREATE RULE char5rule AS @col LIKE 'h%'
Go
CREATE RULE intrule AS @intval < 100
Go
CREATE RULE moneyrule AS @moneyval BETWEEN $5.00 AND $10.00
Go
Exec sp_bindrule 'char5rule', 'mytab.char5col'
Exec sp_bindrule 'intrule', 'mytab.intcol'
Exec sp_bindrule 'moneyrule', 'my_uddt_type'
Go
```

Fügen Sie nun eine gültige Spalte ein (auf der Grundlage dieser Regeln):



```
INSERT mytab VALUES (90,'Howdy',$6.00)
Go
```

Testen Sie, ob Ihre Standards mit den Regeln konform gehen.



```
INSERT mytab DEFAULT VALUES
Go
```

In beiden Fällen erhalten Sie die Meldung: (1 row(s) affected).

Fügen Sie nun einen Satz ungültiger Werte ein (auf der Grundlage dieser Regeln):



```
INSERT mytab VALUES (101,'Ralph',$20.00)
Go
```

Der SQL Server erlaubt nicht, dass die ungültigen Werte eingegeben werden. Jetzt haben Sie eine Vorstellung davon erhalten, wie die beiden Mechanismen zusammenarbeiten und Ihnen die Kontrolle der Domänenintegrität hinsichtlich gültiger Spaltenwerte erleichtern können.

Natürlich können Sie für alle diese Dinge auch den SQL Server Enterprise Manager verwenden. Um das Dialogfeld Regeleigenschaften (siehe Abbildung 14.6) anzuzeigen, klicken Sie in der Konsolenstruktur mit der rechten Maustaste auf das Symbol *Regeln* Ihrer Datenbank und wählen dann aus dem Kontextmenü NEUE REGEL. Wie schon weiter oben erwähnt, können Sie dieses Dialogfeld auch über VORGANG / NEUE REGEL erreichen; es entspricht dem Dialogfeld STANDARDEIGENSCHAFTEN, das Sie weiter oben gesehen haben.



Abbildung 14.6:
Das Dialogfeld Regeleigenschaften

14.4 ANSI-Einschränkungen und Mechanismen der deklarativen Integrität

Die ANSI-Einschränkungen und die deklarative Integrität setzen Sie zur Sicherstellung der Genauigkeit Ihrer Daten ein. Hierzu gehören die Verwendung der Eigenschaft `IDENTITY` sowie die folgenden Einschränkungen: Standard, Prüfung, Primärschlüssel, Fremdschlüssel und Eindeutigkeit.

Die Eigenschaft `IDENTITY`

Die Eigenschaft `IDENTITY` ermöglicht Ihnen den Einsatz von systemgenerierten Werten in Ihren Tabellen, was ähnlich dem Autowert in Datenbanken von Microsoft Access und Microsoft FoxPro ist. Jede Tabelle darf eine Spalte mit der Eigenschaft `IDENTITY` enthalten.

Identitätsspalten werden typischerweise dazu verwendet, so genannte systemverwaltete Schlüssel zuzuweisen. Zur Sicherstellung der Entitätsintegrität müssen Sie jede Zeile in einer Tabelle eindeutig kennzeichnen. Kommt dafür keine vorhandene Spalte oder Spaltenkombination infrage, können Sie eine Identitätsspalte erstellen.

Die Eigenschaft `IDENTITY` können Sie nur dann verwenden, wenn die Spalte, der diese Eigenschaft zugewiesen werden soll, vom Typ Ganzzahl oder kompatibel mit diesem Typ ist. Daher sind die folgenden Typen möglich:

- `tinyint`
- `smallint`
- `integer`
- `numeric`
- `decimal`

Die Typen `numeric` und `decimal` können Sie nur dann verwenden, wenn diese einen Dezimalstellenbereich von 0 haben (wie etwa `numeric (12,0)`). Außerdem müssen `NULL`-Werte erlaubt sein. Da die `IDENTITY`-Eigenschaft Werte nicht wiederverwendet und auch nicht nach Überschreiten des verfügbaren Wertebereichs von vorn beginnt, verwenden Sie gegebenenfalls besser die Datentypen, deren Dezimalstellenbereich größer werden kann. Ist der Wertebereich des Datentyps nämlich einmal »erschöpft«, sind keine weiteren Einfügungen mehr möglich.

Bei jedem Einfügen in eine Tabelle, bei der die `IDENTITY`-Eigenschaft für eine Spalte aktiviert ist, wird in diese Spalte automatisch der nächste mögliche Wert eingetragen:



`IDENTITY [(Ausgangswert, Schrittweite)]`

Die Argumente haben folgende Bedeutung:

- *Ausgangswert* ist der Startwert der Identitätsspalte. Er wird der ersten in die Tabelle eingefügten Zeile zugewiesen.
- *Schrittweite* ist der (ganzzahlige) Wert, der die Änderung zwischen jedem einzelnen Wert angibt. Der Wert kann positiv oder negativ sein.

Sofern Sie keinen Ausgangswert oder Schrittweitenwert angeben, werden diese standardmäßig auf 1 gesetzt. Somit erhält die erste Zeile den Wert 1, die zweite den Wert 2 usw.

Die Eigenschaft `IDENTITY` wird bei den Anweisungen `CREATE TABLE` oder `ALTER TABLE` verwendet, die Tag 9 behandelt hat. Das folgende Beispiel ergibt beim Einfügen der ersten Zeile den Wert 1 für `coll`, und anschließend 101, 201 usw.:

```
CREATE TABLE mytable5
(col1 int not null IDENTITY(1,100),
 col2 char(5) not null)
```

Weil der Wertebereich von Identitätsspalten nicht wieder von vorn beginnt, ist es durchaus möglich, dass er sich erschöpft. Da Sie in SQL Server den Datentyp nach dem Erstellen der Tabelle nicht mehr ändern können, sollten Sie für die Identitätsspalte einen Datentyp wählen, der für alle in der Praxis möglicherweise vorkommenden Werte in dieser Spalte ausreichend ist.

Wenn Sie `IDENTITY` ohne die optionalen Parameter verwendet haben, werden die Werte auf (1,1) gesetzt.

Anstelle des eigentlichen Spaltennamens können Sie für den Verweis auf eine solche Spalte auch das Schlüsselwort `IDENTITYCOL` benutzen. Da es in jeder Tabelle immer nur eine Spalte mit der aktiven Eigenschaft `IDENTITY` geben kann, weiß SQL Server auch immer genau, um welche Spalte es sich dabei handelt. Im Beispiel oben hätten Sie daher die Anweisung `SELECT IDENTITYCOL FROM mytable5` ausführen können, woraufhin Ihnen SQL Server die Daten nur für `col1` zurückgeliefert hätte.

Informationen über die Identitätsspalte einer Tabelle erhalten Sie über die gespeicherte Systemprozedur `sp_help`, wobei Sie als Parameter den Namen der Tabelle angeben. Alternativ dazu können Sie auch die Systemfunktionen `IDENT_INCR` oder `IDENT_SEED` verwenden.



```
IDENT_SEED('Tabellenname')
IDENT_INCR('Tabellenname')
```

Diese Funktionen erfüllen folgende Aufgaben:

- `IDENT_SEED` liefert den Ausgangswert zurück, der bei der Erstellung oder Änderung der Tabelle verwendet wurde.
- `IDENT_INCR` liefert den Schrittweitenwert zurück, der bei der Erstellung oder Änderung der Tabelle verwendet wurde.

Aus programmiertechnischer Sicht taucht nun eine interessante Frage auf: Woher wissen Sie eigentlich, welcher Wert als letzter Identitätswert eingefügt worden ist? Hier kommt der globale Wert `@@IDENTITY` ins Spiel – immer dann, wenn Sie einen Wert in eine Tabelle mit einer Identitätsspalte einfügen, wird auch der Wert von `@@IDENTITY` aktualisiert.

Wie dies funktioniert, können Sie mit dem folgenden Code herausfinden. In diesem Beispiel arbeiten Sie mit *myidenttab*, die Sie bereits weiter oben in dieser Lektion angelegt haben.



```
CREATE TABLE myidenttab  
(col1 int not null IDENTITY(1,100),  
 col2 char(5) not null)  
Go  
INSERT myidenttab (col2) VALUES ('howdy')  
SELECT @@identity
```

An diesem Punkt sollte der Wert von @@identity gleich 1 sein.



```
INSERT myidenttab(col2) VALUES ('Movie')  
SELECT @@identity
```

Jetzt müsste @@identity den Wert 101 haben.

Wenn Sie die Anweisung `TRUNCATE TABLE` ausführen, wird der Identitätswert wieder auf den ursprünglichen Ausgangswert zurückgesetzt. Gewöhnliche Löschvorgänge haben diesen Effekt jedoch nicht – auch nicht, wenn Sie alle Zeilen in der Tabelle löschen. Probieren Sie es aus:



```
DELETE myidenttab  
Go  
INSERT myidenttab (col2) VALUES ('Zebra')  
SELECT @@identity
```

Der Wert von @@identity sollte jetzt 201 betragen, obwohl die Tabelle nur noch eine einzige Zeile enthält.



```
TRUNCATE TABLE myidenttab  
Go  
INSERT myidenttab (col2) VALUES ('howdy')  
SELECT @@identity
```

Jetzt sollte der Wert von @@identity auf 1 zurückgesetzt sein.

Neu in **SQL Server 2000** ist die Funktion `SCOPE_IDENTITY`, die den Wert von @@IDENTITY zurückgibt. Allerdings ist eine Besonderheit zu beachten. @@IDENTITY liefert den zuletzt aktualisierten Identitätswert zurück. `SCOPE_IDENTITY` gibt den letzten Identitätswert aus dem aktuellen Gültigkeitsbereich zurück. Den Unterschied zwischen `SCOPE_IDENTITY` und @@IDENTITY soll folgendes Beispiel klären: Nehmen wir an, Sie haben einen Trigger eingerichtet (darauf geht die morgige Lektion ein), und der Trigger führt Code aus, der eine andere Tabelle mit einer Identitätsspalte modifiziert. Wenn Sie den Wert von @@IDENTITY auswählen, erhalten Sie den Identitätswert der durch den Trigger modifizierten Tabelle zurück und nicht den Wert aus der Originaltabelle, weil das der zuletzt modifizierte Identi-

tätswert ist. Dagegen gibt `SCOPE_IDENTITY` den Wert der letzten Identitätszeile zurück, die Sie direkt in die Originaltabelle eingefügt haben. Führen Sie am besten den folgenden Code aus, um diesen Sachverhalt zu verdeutlichen:



```
create table triggertab (col1 int not null identity(1,1),
    col2 char(5) null)
create table othertab (cola int not null identity(100,1))
go
create trigger mytrigger on triggertab for insert as
    insert othertab DEFAULT VALUES
go

insert triggertab (col2) values ('howdy')
select @@identity
select scope_identity()
```



(1 row(s) affected)

100

(1 row(s) affected)

1

(1 row(s) affected)

Beachten Sie, dass das Ergebnis von `@@IDENTITY` gleich 100 ist, weil der Trigger für die Tabelle *triggertab* den `IDENTITY`-Wert der Tabelle *othertab* modifiziert hat, so dass es sich dabei in der Tat um die zuletzt aktualisierte Identitätsspalte handelt. Das ist wahrscheinlich nicht das erwartete Ergebnis. Deshalb gibt es jetzt in SQL Server 2000 die Funktion `SCOPE_IDENTITY`, mit der Sie den letzten Identitätswert prüfen können, den Sie in *triggertab* eingefügt haben. In allen vorherigen Versionen von SQL Server ist der in *triggertab* eingefügte Datenwert nicht mehr zugänglich gewesen.

Identitätswerte werden als Pool im Hauptspeicher verwaltet. Eine garantierte Verwendung jedes einzelnen Wertes gibt es nicht, weil unter Umständen Transaktionen wieder abgebrochen werden oder der eine oder andere Server-Crash auftreten kann. Falls Sie garantiert eindeutige Identitätsspalten haben möchten, müssen Sie einen eindeutigen Index auf diese Spalte legen. Manchmal möchte man Identitätswerte wiederverwenden. Standardmäßig

gibt es jedoch keine Möglichkeit für manuelle Einfügungen in Identitätsspalten; gegebenenfalls können Sie dies innerhalb einer einzelnen Sitzung mit der Option `identity_insert` überschreiben.



```
SET identity_insert [datenbank.[Besitzer.]]Tabellenname ON|OFF
```

Diese Anweisung aktiviert die Fähigkeit, direkte Einfügungen in die Identitätsspalte einer Tabelle vorzunehmen. Um diese Option aktivieren zu können, müssen Sie ein Mitglied der Rollen `sysadmin`, `db_owner` oder `db_ddladmin` oder der Tabellenbesitzer sein.

Um diese Fähigkeit auf `myidenttab` anzuwenden, führen Sie den folgenden Transact-SQL-Code aus. Er muss gänzlich innerhalb derselben Kopie des SQL Server Query Analyzers ablaufen, weil die SET-Anweisung sitzungsspezifisch ist.



```
SET identity_insert myidenttab ON
Go
INSERT myidenttab (col1, col2) VALUES (2, 'jolly')
Go
SET identity_insert myidenttab OFF
Go
SELECT * FROM myidenttab
```

Dieser Code fügt die angeforderte Zeile in die Tabelle ein. Sie müssen dabei immer die Spaltenliste angeben, auch dann, wenn Sie ausdrücklich für jede einzelne Spalte in der Tabelle Werte spezifizieren.



Anstelle der Identität (insbesondere bei Primärschlüsseln) können Sie auch in Erwägung ziehen, den Datentyp `UNIQUEIDENTIFIER` zu benutzen, obschon es sich dabei nicht im strengen Sinne um eine Einschränkung oder DRI-Option handelt. Wenn Sie eine Spalte mit dem Datentyp `UNIQUEIDENTIFIER` definieren und automatisch über die Funktion `NEWID` bei jedem Einfügen einer neuen Zeile einen neuen und global eindeutigen Standardwert erzeugen, steht Ihnen damit eine leistungsfähige Möglichkeit zur Sicherstellung eindeutiger Schlüsselwerte zur Verfügung. Anders als bei Verwendung der `IDENTITY`-Eigenschaft sind die Werte von `UNIQUEIDENTIFIER` in jeder Tabelle auf jedem System eindeutig. Da sie aber viel größer als die typischen Identitätsspalten mit Ganzzahldatentypen sind, sollten Sie sie nicht zu häufig einsetzen. Außerdem ist es viel schwieriger, auf dem Bildschirm statt einer einfachen Ganzzahl eine binäre Zahl mit 16 Byte anzusprechen.

ANSI-Einschränkungen

ANSI-Einschränkungen ähneln funktionell den herkömmlichen Objekten, die Sie bereits weiter oben kennen gelernt haben. Nichtsdestotrotz handelt es sich bei ihnen nicht um separate Objekte, sondern um einen Teil der Definition der Tabellen in Ihrer Datenbank. ANSI-Einschränkungen können wie Standards und Regeln die Domänenintegrität mit DEFAULT- und CHECK-Einschränkungen oder die referenzielle Integrität mit Primär- und Fremdschlüsseln erzwingen. Die Entitätsintegrität können Sie außerdem mit eindeutigen Einschränkungen oder Primärschlüsseln durchsetzen.

Gegenüber Standards und Regeln kann dies eine große Verbesserung bedeuten, da es keinerlei Notwendigkeit dafür gibt, separate Objekte für die Verwendung mit Ihren Tabellen und Bindungen zu verwalten. Einschränkungen werden in jeder Datenbank in den Systemtabellen *sysreferences*, *syscomments* und *sysobjects* und gegebenenfalls *sysforeignkeys* gespeichert.

Schauen Sie sich nun die Syntax von Einschränkungen als Erweiterung der Anweisungen CREATE TABLE und ALTER TABLE an:



```
CREATE TABLE [Datenbank.[Besitzer].]Tabellenname
    ({Spaltenname Spalteneigenschaften [Einschränkung]
    [,] {nächster_Spaltenname | nächste_Einschränkung}...})
    [ON Dateigruppe]
    [TEXTIMAGE_ON Dateigruppe]
```

Dies ist der gleiche Code, den Sie schon am 9. Tag kennen gelernt haben – allerdings befassen Sie sich jetzt mit den Teilen, die mit *Einschränkung* gekennzeichnet sind. Einschränkungen haben die folgende syntaktische Form:

```
[CONSTRAINT <Name>] <Typ der Einschränkung> [<Einschränkungsoptionen>]
```

Es folgt nun die vollständige Syntax der Einschränkungsoption (keine Sorge – sie wird im Laufe dieser Lektion in allen Einzelheiten besprochen):

```
<Spalteneinschränkung> ::= [CONSTRAINT Einschränkungsnamen]
[ [ NULL | NOT NULL ] | [ { PRIMARY KEY | UNIQUE }
[CLUSTERED | NONCLUSTERED]
[WITH [FILLFACTOR = Füllfaktor] ]
[ON {Dateigruppe | DEFAULT} ]]
]
| [ [FOREIGN KEY] REFERENCES Ref_Tabelle [(Ref_Spalte) ]
    [ON UPDATE { CASCADE | NO ACTION }]
    [ON DELETE { CASCADE | NO ACTION }]
[NOT FOR REPLICATION]
]
```



```

| CHECK [NOT FOR REPLICATION] (logischer_Ausdruck)
}

<Tabelleneinschränkung > ::= [CONSTRAINT Einschränkungsnamen]
{ [ { PRIMARY KEY | UNIQUE } [ CLUSTERED | NONCLUSTERED]
  { ( Spalte[,...n] ) }
  [ WITH [FILLFACTOR = Füllfaktor] ]
  [ON {Dateigruppe | DEFAULT} ]]
| FOREIGN KEY [(Spalte[,...n])] REFERENCES Ref_Tabelle [(Ref_Spalte[,...n])]
  [ON UPDATE { CASCADE | NO ACTION }]
  [ON DELETE { CASCADE | NO ACTION }]
  [NOT FOR REPLICATION] | CHECK [NOT FOR REPLICATION] (Suchkriterien)
}

ALTER TABLE Tabelle
{ [ALTER COLUMN Spaltenname
  {[ neuer_Datentyp [ (Genauigkeit[, Dezimalstellen] ) ]
  [ COLLATE Sortierungsname] [ NULL | NOT NULL ] ]
  | [ {ADD | DROP} ROWGUIDCOL ] }}
| ADD { [ <Spaltendefinition> ]
  | Spaltenname AS berechneter_Spaltenausdruck [,...n]
  | [WITH CHECK | WITH NOCHECK] ADD [ <Tabelleneinschränkung> ] [,...n]
  | DROP { [CONSTRAINT] Einschränkungsnamen | COLUMN Spalte } [,...n]
  | {CHECK | NOCHECK} CONSTRAINT {ALL | Einschränkungsnamen[,...n]}
  | {ENABLE | DISABLE} TRIGGER {ALL | Triggernamen[,...n]} } }

```

Beachten Sie, dass dies eine vereinfacht dargestellte Syntax ist.

Einschränkungen lassen sich auf Spaltenebene und auf Tabellenebene festlegen. Einschränkungen auf Spaltenebene werden beim Erstellen der Tabelle auf der Spaltenebene angewendet. Einschränkungen auf Tabellenebene werden wie zusätzliche Spalten hinzugefügt. Einige Beispiele helfen bei der Unterscheidung dieser beiden Formen.

Einschränkungen auf Spaltenebene

```

CREATE TABLE mytablea
(col1 int not null CONSTRAINT DF_a_col1 DEFAULT (0))

```

Einschränkungen auf Tabellenebene

```

CREATE TABLE mytableb
(col1 int not null)

ALTER TABLE mytableb ADD
CONSTRAINT DF_b_col1 DEFAULT (0) FOR col1

```

Beachten Sie hier die Option `FOR coll`, die angibt, auf welche Spalte die `DEFAULT`-Einschränkung angewendet werden soll. Bei einer Einschränkung auf Spaltenebene ist die Spalte implizit enthalten.

DEFAULT-Einschränkungen

`DEFAULT`-Einschränkungen ähneln den Standards von SQL Server. Allerdings werden sie immer nur auf Spalten und nie auf benutzerdefinierte Datentypen angewendet. Außerdem können Sie keine `DEFAULT`-Einschränkungen auf Spalten anwenden, die gleichzeitig Identitätsspalten darstellen oder die mit dem Datentyp `timestamp` definiert worden sind. Der Unterschied besteht hier darin, dass der Standardwert ein »Teil« der Spalte ist und nicht an die Spalte gebunden werden muss. Diese Einschränkungen werden nur bei Einfügevorgängen erzwungen – genau wie bei SQL Server-Standardobjekten.



Spaltenebene:

```
[CONSTRAINT Einschränkungsnamen] DEFAULT {konstanter_Ausdruck}
```

Tabellenebene:

```
[CONSTRAINT Einschränkungsnamen] DEFAULT {konstanter_Ausdruck}
FOR Spaltenname
```

Der Teil `CONSTRAINT Einschränkungsnamen` der Syntax ist optional. Er stellt klar, dass Sie explizit eine Einschränkung hinzufügen (sollte man aus Dokumentationsgründen immer tun) und ihr gleichzeitig einen Namen geben. Wenn Sie einer Einschränkung keinen Namen geben, kommt dabei etwa Folgendes heraus:

```
DF__mytab__coll__117F9D94
```

Die Benennung von Einschränkungen ist daher immer sinnvoll – einige Operationen verlangen sogar, dass Sie die jeweilige Einschränkung namentlich angeben, mehr dazu in Kürze.

Als Nächstes folgt das Schlüsselwort `DEFAULT` und dann entweder eine Konstante des passenden Datentyps, `NULL` oder eine *NILADIC-Funktion*. Zu den *NILADIC-Funktionen* (d.h. Funktionen, die keine Parameter haben) gehören:

- `CURRENT_TIMESTAMP`: Holt das aktuelle Datum und die Zeit; gleichwertig mit `SELECT getdate()`.
- `SYSTEM_USER`: Ermittelt den aktuellen Benutzernamen; gleichwertig mit `SELECT suser_name()`.
- `CURRENT_USER`, `USER` und `SESSION_USER`: Ermitteln den aktuellen Datenbankbenutzernamen; gleichwertig mit `SELECT user_name()`.

Funktionell bestehen bei DEFAULT-Einschränkungen keine Unterschiede zwischen Tabellen- und Spaltenebene. DEFAULT-Einschränkungen müssen als Spaltenebene während einer CREATE TABLE-Anweisung und Tabellen- oder Spaltenebeneinschränkungen während einer ALTER TABLE-Anweisung spezifiziert werden.

Es folgen nun einige Beispiele zur Veranschaulichung von DEFAULT-Einschränkungen



```
CREATE TABLE defaulttbl
( intcol int NOT NULL CONSTRAINT df_intcol DEFAULT 0,
  char5col char(5) NOT NULL DEFAULT 'Hello',
  anumber numeric(10,0) NOT NULL
)
Go
```

Beachten Sie, dass die erste Einschränkung einen Namen hat, die zweite jedoch nicht und daher einen systemseitig zugewiesenen Namen erhält:



```
ALTER TABLE defaulttbl
ADD moneycol money NULL CONSTRAINT df_moneycol DEFAULT $2.00,
CONSTRAINT df_anumber DEFAULT 100 FOR anumber
Go
```

Starten Sie zur Überprüfung, ob die Einschränkungen nun für die Tabelle wirksam sind, die Prozedur `sp_help defaulttbl`, die Ihnen im Abschnitt des Berichts über Einschränkungen etwa die folgende Ausgabe liefert:

constraint_type	constraint_name	constraint_keys
DEFAULT on column char5col	DF__defaulttta__char5__6F9499E4	('Hello')
DEFAULT on column anumber	df_anumber	(100)
DEFAULT on column intcol	df_intcol	(0)
DEFAULT on column moneycol	df_moneycol	(2.00)

Beachten Sie den systemseitig zugewiesenen Namen für die Einschränkung, der Sie keinen Namen verliehen haben.

CHECK-Einschränkungen

CHECK-Einschränkungen arbeiten ähnlich wie Regeln und stellen einen Mechanismus zur Erzwingung der Domänenintegrität für Ihre Spalten zur Verfügung. Im Gegensatz zu anderen ANSI-Einschränkungen können Sie beliebig viele CHECK-Einschränkungen auf eine einzige Spalte legen. Sie weisen dieselben Restriktionen wie die DEFAULT-Einschränkungen auf, zum Beispiel in Bezug auf Spalten mit dem Datentyp `timestamp` oder mit der `IDENTITY`-Eigenschaft. CHECK-Einschränkungen werden wie Regeln bei Einfügungen und Aktualisierungen überprüft.

Mit CHECK-Einschränkungen lässt sich darüber hinaus etwas realisieren, was mit Regeln nicht möglich ist: Man kann in den Einschränkungsbedingungen der CHECK-Einschränkungen auf andere Spalten derselben Tabelle verweisen. Allerdings steht Ihnen diese Möglichkeit nur bei Einschränkungen auf Tabellenebene zur Verfügung.



Spaltenebene:

```
[CONSTRAINT Einschränkungsnamen]
CHECK [NOT FOR REPLICATION] (Ausdruck)
```

Tabellenebene:

```
[CONSTRAINT Einschränkungsnamen]
CHECK [NOT FOR REPLICATION] (Ausdruck)
```

Die Syntaxelemente haben folgende Bedeutung:

- *CONSTRAINT Einschränkungsnamen* ist wie schon bei den DEFAULT-Einschränkungen optional.
- *CHECK* legt fest, dass Sie eine CHECK-Einschränkung erstellen möchten. Der Ausdruck kann wie bei Regeln jeder beliebige Ausdruck sein, wobei Sie nun allerdings auch andere Spalten in derselben Tabelle referenzieren können.
- *NOT FOR REPLICATION* verhindert, dass die Einschränkung bei Einfügungen oder Aktualisierungen durch interne Replikationsvorgänge erzwungen wird.

Wie schon oben erwähnt, können nur Einschränkungen auf Tabellenebene Referenzen auf mehrere Spalten besitzen.

Durch einige Beispiele verstehen Sie die Arbeitsweise von CHECK-Einschränkungen am besten:



```
CREATE TABLE checktable
(col1 int not null CONSTRAINT ck_col1
CHECK (col1 between 1 and 100),
col2 char(5) null,
zip_code char(5) null,
col4 int not null,
CONSTRAINT ck_col4 CHECK (col4 > col1),
CONSTRAINT ck_zip_code CHECK
(zip_code like '[0-9][0-9][0-9][0-9][0-9]'))
)

ALTER TABLE checktable
ADD CONSTRAINT ck_col2 CHECK (col2 like 'H%')
Go
```

Beachten Sie, dass die Regeln nun erzwungen werden. Die Anweisung

```
INSERT checktable VALUES (1,'Howdy','99901',2)
```

funktioniert, während die Anweisung

```
INSERT checktable VALUES (2,'Howdy','8834A',3)
```

zu folgender Fehlermeldung führt:

Server: Nachr.-Nr. 547, Schweregrad 16, Status 1, Zeile 1

Die INSERT-Anweisung verstieß gegen die COLUMN CHECK-Einschränkung 'ck_zip_code'. Der Konflikt trat in der pubs-Datenbank, Tabelle 'checktable', column 'zip_code' auf.

Die Anweisung wurde beendet.

Primärschlüsseleinschränkungen

Primärschlüsseleinschränkungen werden für eine Kombination von referenzieller und Entitätsintegrität eingesetzt. Jede für einen Primärschlüssel verwendete Spalte muss mit dem Attribut NOT NULL definiert werden, wobei in einer Tabelle nur eine einzige Primärschlüsseleinschränkung existieren kann. Die Primärschlüsseleinschränkung kann dabei von Fremdschlüsseleinschränkungen referenziert werden. Manche Vorgänge wie etwa Replikationen oder ODBC-Anwendungen erfordern unter Umständen deklarierte ANSI-Primärschlüssel.

Primärschlüsseleinschränkungen sind eine implizite Form eines eindeutigen Index. Standardmäßig wird hierbei ein eindeutiger gruppierter Index erstellt.



Spaltenebene:

```
[CONSTRAINT Einschränkungsnamen]
  [ PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]
  [ WITH [ FILLFACTOR = Füllfaktor ] ] [ ON { Dateigruppe | DEFAULT } ]
```

Tabellenebene:

```
[CONSTRAINT Einschränkungsnamen]
  [ PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]
  { ( Spaltenname[,...n] ) } [ WITH [ FILLFACTOR = Füllfaktor ] ]
  [ ON { Dateigruppe | DEFAULT } ]
```

In dieser Syntax erzeugt PRIMARY KEY den (eindeutigen) Primärschlüsselindex. An dieser Stelle sind alle Indexoptionen einschließlich der Änderung des Index zu einem nicht gruppierten Index, der Anwendung von Füllfaktoren usw. gültig. Diese Option erzeugt standardmäßig einen gruppierten Index.

Einen funktionellen Unterschied zwischen Spaltenebenen- und Tabellenebeneinschränkungen gibt es an dieser Stelle per se nicht.

Einen Primärschlüssel können Sie für eine einzige Spalte oder bis zu 16 Spalten erstellen, sofern die Gesamtlänge der Spalten 900 Byte nicht übersteigt.

Das folgende Beispiel erstellt einen eindeutigen gruppierten Index für *col1* der Tabelle *pktable*:

```
CREATE TABLE pktable
(col1 int not null CONSTRAINT pk_col1 PRIMARY KEY,
 col2 char(5) null
)
```

Das nächste Beispiel erzeugt einen eindeutigen nicht gruppierten Index für *col1* der Tabelle *pktable2*:

```
CREATE TABLE pktable2
(col1 int not null CONSTRAINT pk2_col1
 PRIMARY KEY nonclustered (col1),
 col2 char(5) null
)
```

Das letzte Beispiel erzeugt einen eindeutigen gruppierten Index für (*col1*, *col2*) in der Tabelle *pktable3*:

```
CREATE TABLE pktable3
(col1 int not null,
 col2 char(2) not null,
 col3 int null,
 CONSTRAINT pk3_col1col2 PRIMARY KEY (col1, col2)
)
```

In allen Fällen können Sie den Index zwar betrachten, aber nicht direkt manipulieren. Falls Sie versuchen, den Index zu löschen, wie beispielsweise in der letztgenannten Tabelle, erhalten Sie etwa die folgende Fehlermeldung:

```
Server: Nachr.-Nr. 3723, Schweregrad 16, Status 4, Zeile 1
Ein explizites DROP INDEX ist für den pktable3.pk3_col1col2-Index nicht zulässig.
Er wird für die Durchsetzung von PRIMARY KEY-Einschränkungen verwendet.
```

UNIQUE-Einschränkungen

UNIQUE-Einschränkungen erlauben Ihnen wie Primärschlüssel die Erstellung eindeutiger Indizes, aber mit mehr Flexibilität. Typischerweise erstellen Sie UNIQUE-Einschränkungen, wenn Sie mehr als eine Spalte oder einen Satz von Spalten haben, die gültige Primärschlüssel sein könnten, was zwei Zwecken dient: Die Wahl der potentiellen Schlüssel wird dokumentiert und Fremdschlüsseln anderer Tabellen wird Gelegenheit gegeben, die UNIQUE-Einschränkungen zu referenzieren (zusätzlich zu der Möglichkeit, Primärschlüssel-einschränkungen referenzieren zu können).

UNIQUE-Einschränkungen können ebenfalls für Spalten erstellt werden, die NULL-Werte zulassen. Sie können darüber hinaus mehr als eine eindeutige Einschränkung in einer Tabelle haben.



Spaltenebene:

```
[CONSTRAINT Einschränkungsnamen] [ UNIQUE [ CLUSTERED | NONCLUSTERED ]
    [ WITH [ FILLFACTOR = Füllfaktor ] ] [ ON { Dateigruppe | DEFAULT } ]
```

Tabellenebene:

```
[CONSTRAINT Einschränkungsnamen]
    [ PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]
    { ( Spalte[,...n] ) } [ WITH [ FILLFACTOR = Füllfaktor ] ]
    [ ON { Dateigruppe | DEFAULT } ]
```

Wie schon weiter oben ist auch hier der Name der Einschränkung optional. Außerdem, wie auch bei Primärschlüsseleinschränkungen, brauchen Sie auf der Spaltenebene keine Spalten anzugeben, da die Spalte, für die Sie die Einschränkung erstellen, als maßgebliche Spalte angenommen wird.

Das folgende Beispiel erstellt einen Primärschlüssel und eine UNIQUE-Einschränkung. Beide sind eindeutige Indizes in der Tabelle *myuniquetable*.

```
CREATE TABLE myuniquetable
(col1 int not null CONSTRAINT pk_myuniquetable PRIMARY KEY,
 col2 char(20) NOT NULL CONSTRAINT u_myuniquetable UNIQUE
)
```

Fremdschlüsseleinschränkungen

Fremdschlüsseleinschränkungen bewahren die referenzielle Integrität zwischen Tabellen. Dazu erstellen Sie einen Fremdschlüssel in einer Tabelle, die ihrerseits den Primärschlüssel oder die UNIQUE-Einschränkung einer anderen Tabelle referenziert. Diese Einschränkung realisiert eines der folgenden drei Dinge:

- Unterbindet Datenmodifikationen gegenüber der Tabelle mit dem Primärschlüssel, solange die Tabelle mit den Fremdschlüsseln referenzierte Zeilen enthält.
- Löscht alle in Beziehung stehenden Zeilen in der Tabelle mit den Fremdschlüsseln.
- Aktualisiert den Primärschlüsselwert in allen referenzierten Zeilen in der Tabelle mit den Fremdschlüsseln.

Außerdem wird verhindert, dass Daten in die Tabelle mit der Fremdschlüsseleinschränkung eingefügt (oder aktualisiert) werden, die zu ungültigen Daten in den referenzierten Tabellen führen würden.

Durch Erstellen eines Fremdschlüssels wird kein Index für die Tabelle erstellt; es ist jedoch wahrscheinlich, dass es sich hierbei um einen guten Kandidaten für einen Index handelt. Aus diesem Grunde werden Sie normalerweise die Erstellung von Tabellen mit Fremdschlüsseln durch `CREATE INDEX`-Anweisungen ergänzen müssen. Auf Tabellen in derselben Datenbank können Sie sich nur dann beziehen, wenn Sie Fremdschlüsseleinschränkungen erstellen. Zur Bezugnahme auf eine solche Tabelle benötigen Sie die erforderliche Berechtigung (`SELECT` oder `REFERENCES`) für diese Tabelle, wobei eine einzelne Tabelle ein Maximum von 253 Fremdschlüsseln, die auf sie zeigen, haben kann. Eine Erweiterung dieser Grenze ist nicht möglich.



Spaltenebene:

```
[CONSTRAINT Einschränkungsnamen] [FOREIGN KEY] REFERENCES Ref_Tabelle
    ( ( Ref_Spalte ) ) [NOT FOR REPLICATION]
    [ ON UPDATE { CASCADE | NO ACTION } ]
    [ ON DELETE { CASCADE | NO ACTION } ]
    [NOT FOR REPLICATION]
```

Tabellenebene:

```
[CONSTRAINT Einschränkungsnamen] FOREIGN KEY [(Spalte[,...n])]
    REFERENCES Ref_Tabelle [(Ref_Spalte[,...n])]
    [ ON UPDATE { CASCADE | NO ACTION } ]
    [ ON DELETE { CASCADE | NO ACTION } ]
    [NOT FOR REPLICATION]
```

Es sei noch einmal darauf hingewiesen, dass Einschränkungsnamen optional sind. Wie bei den anderen referenziellen Einschränkungen benötigen Sie keine lokale Referenzierung des Spaltennamens, wenn es sich um eine Einschränkung für eine Einzelspalte handelt. Außerdem brauchen Sie die Spalte der anderen Tabelle nicht zu benennen, wenn die Spalten den gleichen Namen tragen.

Wenn Sie weder die Option `ON UPDATE` noch `ON DELETE` angeben, werden beide auf `NO ACTION` gesetzt, um mit vorherigen `SQL Server`-Versionen kompatibel zu sein. Das bedeutet, dass jede Löscho- oder Aktualisierungsoperation eines Primärschlüsselwertes verhindert wird, wenn irgendwelche Zeilen den Primärschlüssel in der/den Tabelle/n mit dem Fremdschlüssel referenzieren.

Wenn Sie die Option `CASCADE` angeben, werden die Zeilen in der/den Tabelle/n mit dem Fremdschlüssel gelöscht oder deren referenzierte Spaltenwerte werden aktualisiert.

Falls Sie eine mehrspaltige Primärschlüssel-/`UNIQUE`-Einschränkung referenzieren, müssen Sie darauf achten, dass Sie sie in derselben Reihenfolge in den `FOREIGN KEY`- und `REFERENCES`-Listen aufführen. Selbstverweise werden unterstützt, so dass Sie in der Lage sind, die Tabelle selbst zu referenzieren (mit einer anderen Spalte).

Der folgende Code erstellt die Tabelle *employee* (Mitarbeiter) und dazu eine Tabelle *order* (Bestellung), deren Daten von einem Mitarbeiter eingegeben werden sollen. Um sicherzustellen, dass die Bestellung von einem gültigen Mitarbeiter eingegeben wird, können Sie die entsprechende Funktionalität nun entweder programmieren oder über Fremdschlüssel deklarieren. Wenn dann jemand einen Mitarbeiter zu löschen versucht, wird dies solange nicht erlaubt, wie es noch Bestellungen für diesen Mitarbeiter gibt.



```
CREATE TABLE emp
(emp_id int not null CONSTRAINT pk_emp PRIMARY KEY,
 emp_name char(30) not null)
Go
CREATE TABLE orders
(order_id int not null CONSTRAINT pk_order PRIMARY KEY,
 emp_id int not null CONSTRAINT fk_order
FOREIGN KEY (emp_id) REFERENCES emp (emp_id)
)
Go
INSERT emp VALUES (1,'Joe Smith')
INSERT emp VALUES (2,'Ann Jones')
INSERT orders VALUES (1,1)
INSERT orders VALUES (2,2)
Go
```

Bis hierhin ist alles in Ordnung. Versuchen Sie nun eine Bestellung (*order*) für einen Mitarbeiter (*employee*) einzugeben, der nicht existiert:



```
INSERT orders VALUES (3,3)
Go
```



```
Server: Nachr.-Nr. 547, Schweregrad 16, Status 1, Zeile 1
Die INSERT-Anweisung verstieß gegen die COLUMN FOREIGN KEY-
Einschränkung 'fk_order'. Der Konflikt trat in der pubs-Datenbank,
Tabelle 'emp', column 'emp_id' auf.
Die Anweisung wurde beendet.
```

Okay! Versuchen Sie nun, einen Mitarbeiter zu löschen, dem eine Bestellung zugeordnet ist:



```
DELETE emp WHERE emp_id = 1
Go
```



Server: Nachr.-Nr. 547, Schweregrad 16, Status 1, Zeile 1
Die DELETE-Anweisung verstieß gegen die COLUMN REFERENCE-Einschränkung 'fk_order'. Der Konflikt trat in der pubs-Datenbank, Tabelle 'orders', column 'emp_id' auf.
Die Anweisung wurde beendet.

Der folgende Code zeigt ein Beispiel für eine Selbstreferenz, was bedeutet, dass jeder Manager auch ein gültiger Mitarbeiter sein muss:



```
CREATE TABLE emp_manager
(emp_id int not null CONSTRAINT pk_emp_mgr PRIMARY KEY,
 mgr_id int not null CONSTRAINT fk_emp_mgr FOREIGN KEY
REFERENCES emp_manager (emp_id),
 emp_name char(30) not null)
```

Die folgenden beiden Anweisungen lassen sich fehlerfrei ausführen, weil sie gültige Daten referenzieren:

```
INSERT emp_manager VALUES (1,1,'Ann Jones')
INSERT emp_manager VALUES (2,1,'Tom Smith')
```

Versuchen Sie jetzt aber einmal, jemanden zu referenzieren, den es noch nicht gibt:

```
INSERT emp_manager VALUES (3,4,'Bob Newett')
```

Sie erhalten nun eine ähnliche Meldung wie zuvor; nämlich dass die Fremdschlüsseleinschränkung verletzt worden ist. Eine derartige Lösung kann sich in der Praxis als außerordentlich nützlich erweisen.

Der Code in Listing 14.1 fügt der Gleichung eine Auftragsdetailtabelle hinzu, die die neuen Fähigkeiten von SQL Server 2000 zum kaskadierenden Löschen und Aktualisieren verwendet.

Listing 14.1: Kaskadierende Einschränkungen

```
CREATE TABLE order_detail
 (order_id int not null CONSTRAINT pk_order_detail PRIMARY KEY
  (order_id, line_no),
 line_no int not null,
 part_no int not null,
 price money not null,
 CONSTRAINT fk_order_detail FOREIGN KEY (order_id) REFERENCES orders
  (order_id) ON DELETE CASCADE ON UPDATE CASCADE
)

-- Einige Beispieldaten in die Tabelle laden
INSERT order_detail VALUES (1,1,1,$5.00)
INSERT order_detail VALUES (1,2,2,$15.00)
```

```
INSERT order_detail VALUES (1,3,9,$2.95)
INSERT order_detail VALUES (2,1,1,$5.00)
INSERT order_detail VALUES (2,2,8,$8.00)
INSERT order_detail VALUES (2,3,4,$29.00)

-- Jetzt Auftrag Nummer 2 löschen:
DELETE orders where order_id = 2

-- select * from order_detail, um nachzuweisen, dass die Zeilen für
-- Auftrag #2 gelöscht wurden:
select * from order_detail

-- Jetzt Auftrag #1 als Auftrag #5 aktualisieren und beobachten, wie
-- er an order_detail weitergegeben wird:
update orders set order_id = 5 where order_id = 1

-- select * from order_detail, um nachzuweisen, dass die Zeilen für
-- Auftrag #1 als Auftrag #5 aktualisiert wurden:
select * from order_detail
```

Einschränkungen löschen

Eine Einschränkung können Sie über die Anweisung `ALTER TABLE` löschen. Um zum Beispiel die Fremdschlüsseleinschränkung des Beispiels weiter oben zu löschen, führen Sie die folgende Anweisung aus, womit dann die Fremdschlüsseleinschränkung gelöscht ist:



```
ALTER TABLE emp_manager DROP CONSTRAINT fk_emp_mgr
```

Eine Primärschlüsseleinschränkung (oder eine `UNIQUE`-Einschränkung), die noch Fremdschlüsseleinschränkungen besitzt, werden Sie jedoch nicht ohne weiteres löschen können. Wenn Sie beispielsweise mit dem nun folgenden Code die Primärschlüsseleinschränkung der Tabelle *emp_manager* zu löschen versuchen (ohne zuvor den Fremdschlüssel zu löschen):

```
ALTER TABLE emp_manager DROP CONSTRAINT pk_emp_mgr
```

werden Sie die folgende Fehlermeldung erhalten:

```
Server: Nachr.-Nr. 3725, Schweregrad 16, Status 1, Zeile 1
Auf die pk_emp_mgr-Einschränkung wird von der emp_manager-Tabelle, fk_emp_mgr-
Fremdschlüsseleinschränkung verwiesen.
Server: Nachr.-Nr. 3727, Schweregrad 16, Status 1, Zeile 1
Die Einschränkung konnte nicht gelöscht werden. Siehe vorherige Fehler.
```

14.5 Datenintegrität und Enterprise Manager

Die gesamte Funktionalität können Sie ebenso auch über den SQL Server Enterprise Manager bekommen. Markieren Sie dazu in Ihrer Datenbank den Ordner *Tabellen* und klicken Sie im Detailbereich mit der rechten Maustaste auf eine Tabelle. Wählen Sie anschließend aus dem Kontextmenü **TABELLE BEARBEITEN**. Damit gelangen Sie zum Dialogfeld **TABELLE BEARBEITEN** (siehe Abbildung 14.7).

Zum Erstellen des Primärschlüssels für eine Tabelle markieren Sie einfach nur die Spalte bzw. die Spalten, die Sie als Primärschlüssel festlegen möchten, und klicken dann auf das gelbe Schlüsselssymbol in der Symbolleiste. Nachdem Sie dies getan und danach durch Klicken auf das Diskettensymbol die Änderung gespeichert haben, ist Ihre Primärschlüsseleinschränkung erstellt. Beachten Sie, dass jeder Spalte in Ihrer Tabelle, die Teil des Primärschlüssels ist, das Schlüsselssymbol vorangestellt worden ist.

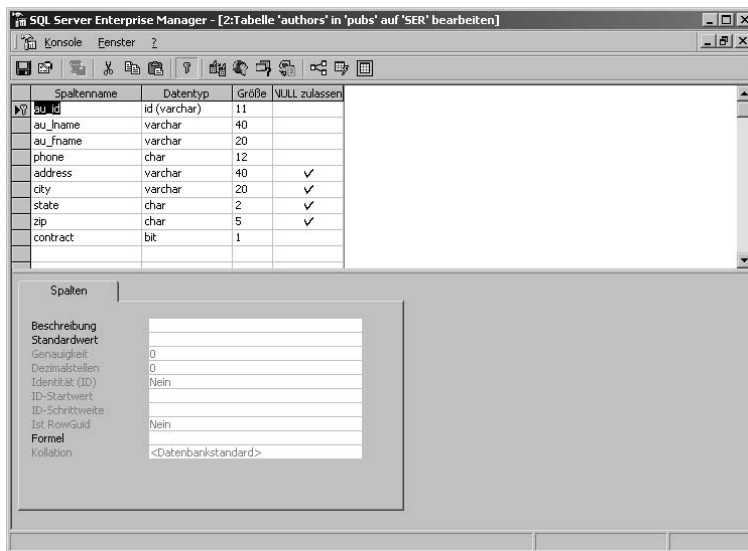


Abbildung 14.7:
Das Fenster zum
Bearbeiten einer
Tabelle

Zum Erstellen einer **UNIQUE**-Einschränkung für eine Tabelle klicken Sie auf das Symbol **TABELLEN- UND INDEXEIGENSCHAFTEN** in der Symbolleiste oder klicken mit der rechten Maustaste an einer beliebigen Stelle in der Tabelle und wählen dann **EIGENSCHAFTEN**. Gehen Sie auf die Registerkarte **INDIZES/SCHLÜSSEL** und füllen Sie das Dialogfeld aus. Diese Dialogfeld haben Sie bereits gestern kennen gelernt. Vergessen Sie nicht, nach der Festlegung der Spalten das Optionsfeld *Einschränkung* und das Kontrollkästchen **UNIQUE ERSTELLEN** zu markieren.

Erstellen Sie zur Veranschaulichung dieses Dialogfelds und zur Untersuchung von CHECK- und Fremdschlüsseinschränkungen sowie von Primärschlüssel- und UNIQUE-Einschränkungen in der Datenbank *pubs* eine Tabelle namens *ConstraintTab*. Nachdem Sie die Tabelle erstellt haben, fügen Sie in der Entwurfsansicht die folgenden Spalten hinzu:

Spaltenname	Datentyp	Größe	NULL zulassen
pkcol	int		
fkcol	int		eingeschaltet
checkcol	char	10	
defcol	int		
altpkcol	int		

Markieren Sie die Spalte *pkcol* und klicken Sie zum Hinzufügen einer Primärschlüsseinschränkung auf das gelbe Schlüsselsymbol. Verwandeln Sie *pkcol* zusätzlich in eine Identitätsspalte; klicken Sie dazu mit der rechten Maustaste auf die Tabelle, wählen Sie EIGENSCHAFTEN aus dem Kontextmenü und wählen Sie dann *pkcol* im Listenfeld TABELLENIDENTITÄTSSPALTE aus (übernehmen Sie die Standardwerte für Startwert und Schrittweite). Nach Beendigung der Aktion sollte die Tabelle Abbildung 14.8 entsprechen.

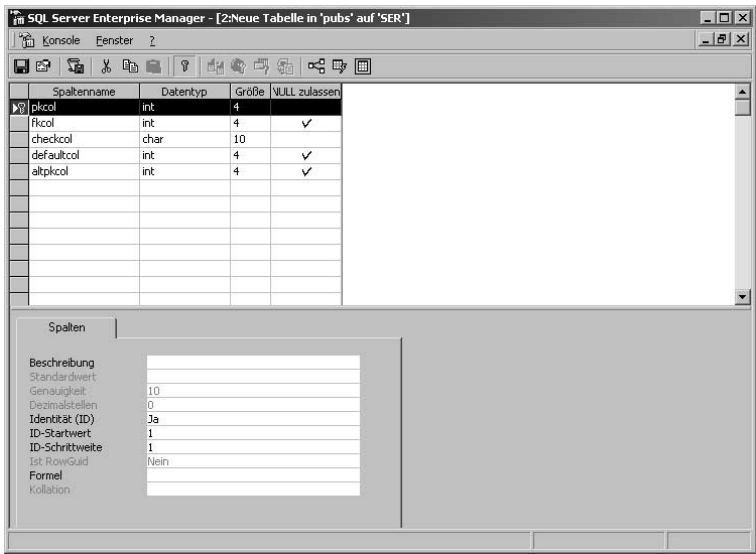


Abbildung 14.8:
Das Entwurfswen-
ster für die Tabelle
ConstraintTab

Bis zu dieser Stelle haben Sie eine Tabelle mit einer DEFAULT-Einschränkung für die Spalte *defaultcol* und einem Primärschlüssel für die Spalte *pkcol* erstellt. Erstellen Sie jetzt eine weitere Tabelle namens *rehtab* (steht für Relationstabelle) mit einer einzigen Spalte

namens *pkcol* mit dem Datentyp *int not null* und machen Sie diese zum Primärschlüssel. Klicken Sie nun mit der rechten Maustaste auf den Ordner *Diagramme* und wählen Sie **NEUES DATENBANKDIAGRAMM**, woraufhin der Datenbank-Diagrammerstellungs-Assistent erscheint. Klicken Sie auf **WEITER** und wählen Sie die Tabellen *ConstraintTab* und *rehtab* aus (siehe Abbildung 14.9).

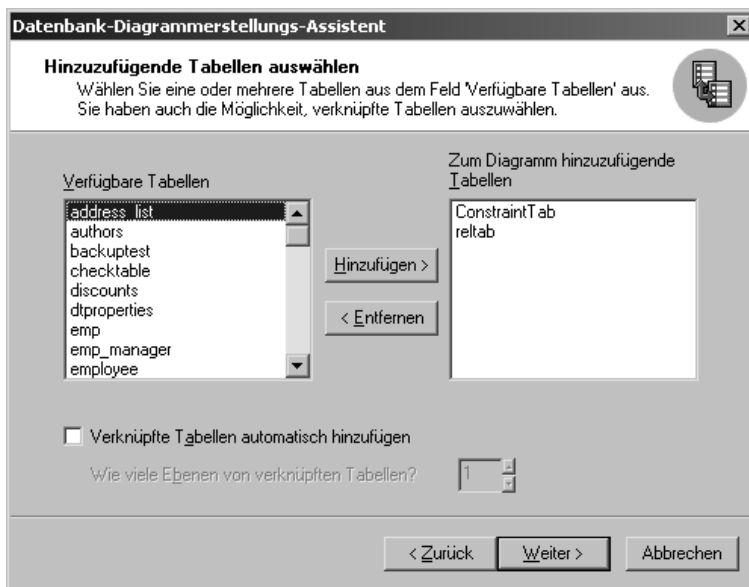


Abbildung 14.9:
Der Datenbank-
Diagrammerstel-
lungs-Assistent

Klicken Sie auf **WEITER** und dann auf **FERTIG STELLEN**. Der SQL Server Enterprise Manager weist Sie nun darauf hin, dass die von Ihnen angegebenen Tabellen in das Datenbankdiagramm aufgenommen worden sind. Klicken Sie auf **OK**, um das neue Datenbankdiagramm anzuzeigen (siehe Abbildung 14.10).

Klicken Sie auf die Spalte *fkcol* in der Tabelle *ConstraintTab* und halten Sie die Maustaste gedrückt. Ziehen Sie die Maus über die Tabelle *rehtab* und lassen Sie die Maustaste los. Es erscheint nun das in Abbildung 14.11 dargestellte Dialogfeld, das die Beziehung zwischen dem von Ihnen erstellten Primär- und Fremdschlüssel anzeigt. Übernehmen Sie die Standardwerte und klicken Sie auf **OK**.

Beachten Sie, dass sich das Diagramm geändert hat – die Beziehung wird nun über eine Linie zwischen den Tabellen wiedergegeben (siehe Abbildung 14.12). Um die Beziehung zu ändern, klicken Sie mit der rechten Maustaste auf die Linie und wählen den zutreffenden Befehl aus dem Kontextmenü. Die Tabellen können Sie beliebig verschieben.

Klicken Sie auf das Diskettensymbol in der Symbolleiste, um das Diagramm zu speichern, und übernehmen Sie den vorgegebenen Namen (die erste von Ihnen erstellte Tabelle wird

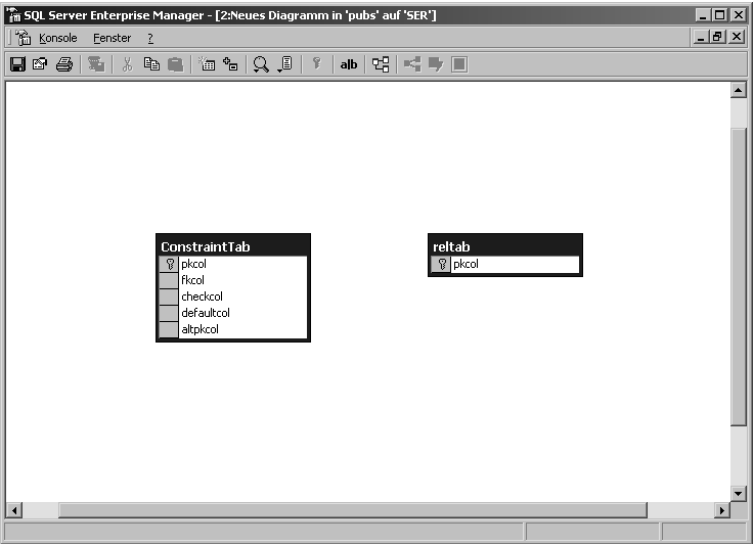


Abbildung 14.10:
Das anfängliche
Datenbank-
diagramm

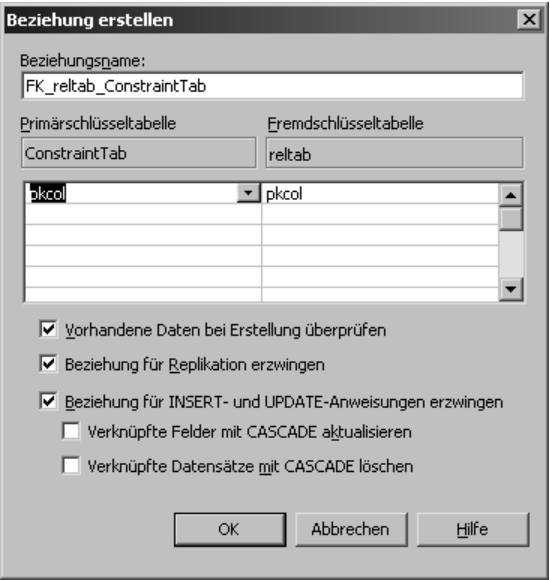


Abbildung 14.11:
Die neu zu erstellende Beziehung

DIAGRAM1 heißen). Der SQL Server Enterprise Manager zeigt Ihnen dann eine Liste der von Ihnen bearbeiteten Tabellen an und fragt Sie danach, ob Sie diese speichern möchten. Klicken Sie zum Speichern Ihrer Änderungen an der referenziellen Integrität auf JA.

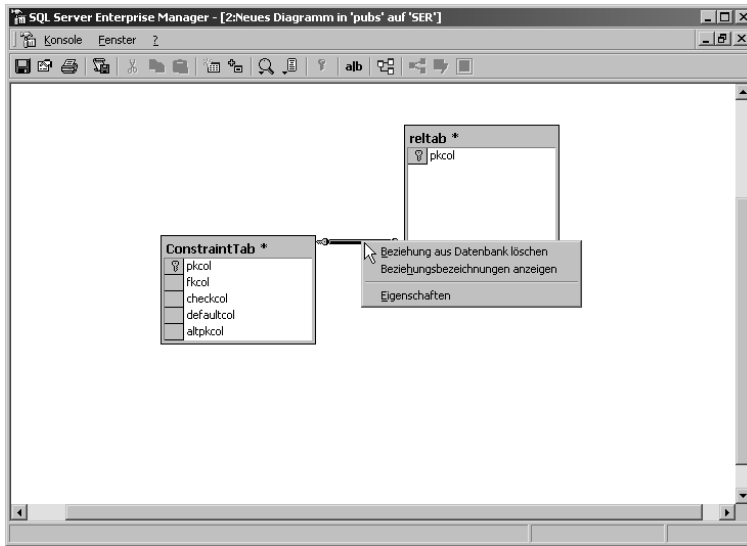


Abbildung 14.12:
Das Datenbank-
diagramm zeigt die
neu erstellte Bezie-
hung.

Eigenschaften

Tabellen | Spalten | **Beziehungen** | Indizes/Schlüssel | CHECK-Einsch

Tabellenname: ConstraintTab

Ausgewählte Beziehung: FK_reltab_ConstraintTab

Beziehungsname: FK_reltab_ConstraintTab

Primärschlüsseltabelle: ConstraintTab

Fremdschlüsseltabelle: reltab

pkcol	pkcol

☐ Vorhandene Daten bei Erstellung überprüfen

☒ Beziehung für Replikation erzwingen

☒ Beziehung für INSERT- und UPDATE-Anweisung

☐ Verknüpfte Felder mit CASCADE aktualisieren

☐ Verknüpfte Datensätze mit CASCADE löschen

Schließen Hilfe

Abbildung 14.13:
Das fertig gestellte Dialogfeld für
die Fremdschlüsseinschränkung

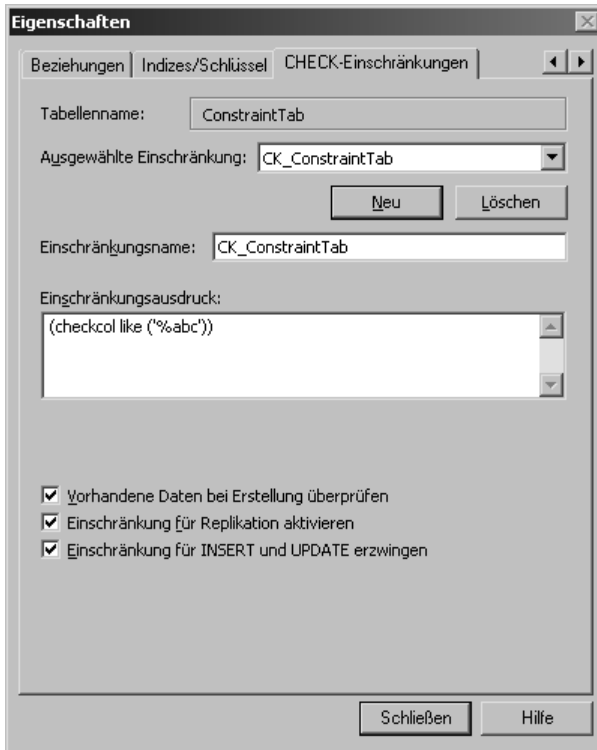


Abbildung 14.14:
Das fertig gestellte Dialogfeld für
die CHECK-Einschränkung

Gehen Sie zur Tabelle *ConstraintTab* zurück und wechseln Sie in die Entwurfsansicht (klicken Sie dazu mit der rechten Maustaste auf die Tabelle und wählen Sie TABELLE BEARBEITEN). Klicken Sie mit der rechten Maustaste auf die Spalte *fkcol* und wählen Sie EIGENSCHAFTEN. Wechseln Sie zur Registerkarte BEZIEHUNGEN und schauen Sie sich die Beziehungen an, die Sie zur Tabelle *reltab* erstellt haben (siehe Abbildung 14.13).

Wechseln Sie jetzt zur Registerkarte CHECK-EINSCHRÄNKUNGEN und klicken Sie auf die Schaltfläche NEU, um eine neue CHECK-Einschränkung zu erstellen. Nehmen Sie die in Abbildung 14.14 gezeigten Eingaben zum Erstellen der CHECK-Einschränkung vor.

Fügen Sie zum Schluss eine der vorher beschriebenen UNIQUE-Einschränkungen hinzu, indem Sie auf der Registerkarte INDIZES/SCHLÜSSEL einen neuen Index auswählen und das Dialogfeld gemäß Abbildung 14.15 ausfüllen.

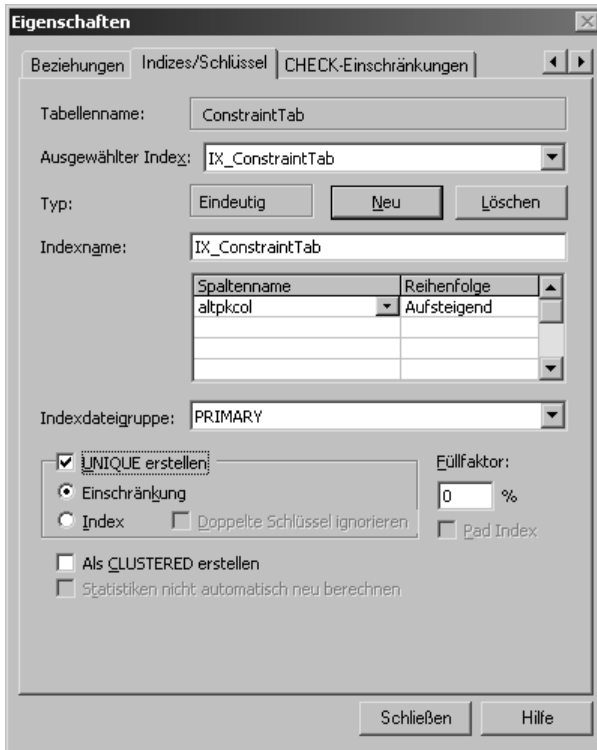


Abbildung 14.15:
Das fertig gestellte Dialogfeld für
eine UNIQUE-Einschränkung

Starten Sie jetzt im SQL Server Query Analyzer die gespeicherte Systemprozedur `sp_help` und sehen Sie sich die Einschränkungen an:



EXEC sp_help constrainttab

Beachten Sie, dass Sie jeden möglichen Einschränkungstyp für diese Tabelle erstellt und auch die IDENTITY-Eigenschaft verwendet haben. Im SQL Server Enterprise Manager steht Ihnen die volle Funktionalität der Einschränkungen und der IDENTITY-Eigenschaft zur Verfügung.

Einschränkungen zurückweisen

Fremdschlüssel- und CHECK-Einschränkungen können Sie nach der Erstellung auch abweisen. Primärschlüssel-, UNIQUE- und DEFAULT-Einschränkungen können dagegen nicht abgewiesen werden (weil sie im Fall der Primärschlüssel- und UNIQUE-Einschränkungen Indizes

erstellen; außerdem werden DEFAULT-Einschränkungen hinsichtlich früherer Daten nie überprüft). Wenn Sie eine Tabelle ändern, eine dieser Einschränkungen hinzufügen und dazu die Option NOCHECK angeben, werden die vorhandenen Daten nicht validiert. Wenn Sie dagegen eine Tabelle ändern und eine CHECK-Einschränkung hinzufügen, werden die vorhandenen Daten validiert. Dazu folgendes Beispiel:



```
CREATE TABLE mytesttable
(col1 char(5) not null
)
Go
INSERT mytesttable VALUES ('Howdy')
INSERT mytesttable VALUES ('Grant')
```

Versuchen Sie nun eine Einschränkung zu erstellen, die wegen einiger Daten fehlschlagen muss:



```
ALTER TABLE mytesttable
ADD CONSTRAINT ck_mytest CHECK (col1 like 'h%')
```

Dazu die Fehlermeldung:



```
Server: Nachr.-Nr. 547, Schweregrad 16, Status 1, Zeile 1
Die ALTER TABLE-Anweisung verstieß gegen die COLUMN CHECK-Einschränkung
'ck_mytest'. Der Konflikt trat in der pubs-Datenbank, Tabelle
'mytesttable', column 'col1' auf.
```

Falls Sie jedoch die gleiche Einschränkung mit der Option NOCHECK spezifizieren, funktioniert sie:



```
ALTER TABLE mytesttable
WITH NOCHECK
ADD CONSTRAINT ck_mytest CHECK (col1 like 'h%')
```

Dasselbe gilt für Fremdschlüsseleinschränkungen. Seien Sie hinsichtlich der Verwendung dieser Option vorsichtig, weil Sie damit praktisch die Existenz ungültiger Daten in Ihrer Datenbank erlauben.

Einschränkungen abschalten

Einschränkungen können Sie temporär auch abschalten, so dass Sie beispielsweise auch ungültige Daten, die andernfalls Ihre CHECK-Einschränkungen oder Ihre Fremdschlüsseleinschränkungen auslösen würden, hinzufügen können. Hierzu führen Sie die Anweisung

ALTER TABLE mit der Option WITH NOCHECK CONSTRAINT <Einschränkungsname> oder WITH NOCHECK ALL zum Abschalten sämtlicher CHECK- und Fremdschlüsseinschränkungen aus.

Der folgende Code, der auf dem Beispiel oben aufsetzt, schaltet sämtliche Einschränkungen ab:



```
ALTER TABLE mytesttable
NOCHECK CONSTRAINT ALL
```

Anschließend können Sie ungültige Daten hinzufügen (Daten, die die CHECK-Einschränkung verletzen würden) und die Einschränkung wieder einschalten:



```
ALTER TABLE mytesttable
CHECK CONSTRAINT ALL
```

14.6 Zusammenfassung

Die Datenintegrität können Sie auf mehreren verschiedenen Wegen erzwingen. Die »traditionellen« Optionen von SQL Server zur Erzwingung der Integrität drehen sich um Regeln, Standards und benutzerdefinierte Datentypen. Datenintegrität und referenzielle Integrität können Sie auch über ANSI-Einschränkungen erzwingen. Die richtige Wahl hängt hierbei gänzlich von den individuellen Anforderungen Ihrer Anwendung ab. ANSI-Einschränkungen und deklarative referenzielle Integrität werden typischerweise bevorzugt, weil sie besser arbeiten und in die Definition von Objekten integriert sind. In SQL Server 2000 sind jetzt kaskadierende referenzielle Aktionen (Löschen und Aktualisieren) verfügbar, und Sie brauchen diese Funktionalität nicht mehr mit Triggern und anderer Programmlogik nachzubilden.

Darüber hinaus haben Sie die IDENTITY-Eigenschaft kennen gelernt, mit der sich Primärschlüssel bilden lassen, falls die Spalten der Tabelle allein nicht genügen, um einen Primärschlüssel zu erstellen. Automatische Werte sind in vielen Datenbankentwürfen außerordentlich hilfreich.

14.7 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** Welche Optionen stehen zur Durchsetzung der Entitätsintegrität zur Verfügung?
- A** Eindeutige Indizes, Primärschlüssel und UNIQUE-Einschränkungen.
- F** Welche Auswirkungen hat die IDENTITY-Eigenschaft auf eine Spalte?
- A** Sie generiert einen automatischen Wert für eine Ganzzahlspalte.
- F** Kann eine Regel eine andere Spalte in einer Tabelle referenzieren?
- A** Nein. Nur Einschränkungen auf Tabellenebene können andere Spalten referenzieren.

Kontrollfragen

1. Ist dies eine gültige Definition für einen Standardwert?
`Create default mydefault as 'UNKNOWN'`
2. Können Sie eine Regel löschen, die noch an eine Spalte gebunden ist, selbst wenn sich keinerlei Daten mehr in der Tabelle befinden?
3. Ist ein numerischer Datentyp für IDENTITY-Spalten erlaubt?
4. Kann ein Fremdschlüssel auf eine Tabelle in einer anderen Datenbank verweisen?
5. Kann eine UNIQUE-Einschränkung abgewiesen oder abgeschaltet werden?
6. Können Sie die von bestimmten Einschränkungen erzeugten Indizes löschen?



Übungen

1. Erstellen Sie drei Tabellen mit jeweils kompatiblen Datentypen. Erstellen Sie zwischen ihnen mit den Datenbankdiagrammfunktionen des SQL Server Enterprise Managers Primärschlüssel/Fremdschlüssel-Beziehungen.
2. Erstellen Sie mehrere Tabellen mit der `IDENTITY`-Eigenschaft. Probieren Sie verschiedene Ausgangswerte und Schrittweiten aus und fügen Sie Daten hinzu, um zu sehen, ob sie sich wie erwartet verhalten. Probieren Sie außerdem die Anweisung `TRUNCATE TABLE` aus, um zu erleben, wie Ihre `IDENTITY`-Optionen auf den ursprünglichen Ausgangswert »zurückgesetzt« werden.

Tag 1	Relationale Datenbanken und Microsoft SQL Server 2000	31
Tag 2	SQL Server 2000 installieren	45
Tag 3	SQL Server 2000 Tools und Utilities	95
Tag 4	Datenbanken und Datendateien erstellen und verwalten	151
Tag 5	Sicherheit	187
Tag 6	Berechtigungen	233
Tag 7	Sicherungen von Datenbanken	275

W O C H E

1

Tag 8	Wiederherstellen von Datenbanken	333
Tag 9	Datentypen und Tabellen	367
Tag 10	Daten mit SELECT abrufen	397
Tag 11	Daten bearbeiten	459
Tag 12	Transact-SQL-Erweiterungen	481
Tag 13	Indizes	523
Tag 14	Datenintegrität	573

W O C H E

2

Tag 15	Sichten, Trigger, gespeicherte Prozeduren und Funktionen	627
Tag 16	Replikation – Entwurf	665
Tag 17	Replikation – Implementierung	701
Tag 18	Aufträge mit dem SQL Server-Agent planen	759
Tag 19	Daten zwischen Servern migrieren	805
Tag 20	Leistungsüberwachung	829
Tag 21	XML und SQL Server 2000	865

W O C H E

3

Woche 3 – Vorschau

In der dritten Woche lernen Sie mehr über die Programmierung in SQL Server. Die Lektionen in Transact-SQL umfassen Sichten, benutzerdefinierte Funktionen, gespeicherte Prozeduren und Trigger. Die administrativen Aufgaben können Sie durch Automatisierungsverfahren vereinfachen. Die Replikation ist ein Instrument, mit dem Sie Daten im gesamten Unternehmen verteilen können. Außerdem lernen Sie, wie Sie Daten nach SQL Server importieren bzw. aus SQL Server exportieren und dabei in andere Formate umwandeln. Ein wichtiges Thema ist auch die Feinabstimmung und Optimierung der Systemleistung von SQL Server. Der Einsatz von SQL Server mit den neuen XML-Fähigkeiten schließt diese Woche ab.

Der 15. Tag führt Sie in die Verwendung von Sichten, benutzerdefinierten Funktionen, gespeicherten Prozeduren und Triggern ein, die man aus Gründen der Geschwindigkeit, der leichteren Handhabbarkeit und der Datenbankkonsistenz implementiert.

Der 16. Tag untersucht die verfügbaren Typen der Replikation in SQL Server und führt Sie in die einschlägige Terminologie ein. Tag 17 setzt mit diesem Thema fort und zeigt, wie Sie die verschiedenen Typen der Replikation implementieren.

Der 18. Tag ist für Sie besonders aus der Sicht des Datenbankadministrators interessant. Sie erfahren, wie Sie die alltäglichen Aufgaben, die bei Betrieb und Wartung der Datenbank anfallen, automatisieren können. Außerdem werden Sie es so einrichten, daß Sie im Fehler- oder Erfolgsfall per E-Mail benachrichtigt werden. Alle diese Aufgaben werden über den SQL-Server-Agent erledigt.

Tag 19 untersucht die Migration von Daten aus anderen Systemen nach SQL Server bzw. aus SQL Server heraus. Dabei machen Sie sich mit den Data Transformation Services (DTS) vertraut; außerdem lernen Sie einige Transact-SQL-Befehle und Befehlszeilendienstprogramme für die Datenmigration kennen.

Die Optimierung Ihrer Datenbank auf bestmögliche Leistung und Datendurchsatz ist für jeden Datenbankadministrator ein Muss. Wie Sie ein Feintuning Ihrer Datenbank vornehmen können, erfahren Sie am 20. Tag.

XML ist ein brandaktuelles Thema auch in Bezug auf Datenbanken und Tag 21 bringt eine Einführung, wie Sie SQL Server mit XML verwenden.

Wie gewohnt erwarten Sie am Ende jeder Lektion einige Übungen, die nicht nur auf Ihrem bis dahin gelernten Wissen aufbauen, sondern Ihnen ebenso die Gelegenheit bieten, Ihr Verständnis des Lernstoffs testen zu können.



**Sichten, Trigger,
gespeicherte Prozeduren
und Funktionen**

Die heutige Lektion macht Sie mit Sichten, gespeicherten Prozeduren, Triggern und benutzerdefinierten Funktionen bekannt. Diese Datenbankkomponenten schirmen den Benutzer von der Komplexität der Transact-SQL-Anweisungen ab.

Mit Sichten, die man auch als gespeicherte Abfragen verstehen kann, können Sie genau spezifizieren, wie der Benutzer die Daten sieht. Gespeicherte Prozeduren sind vorkompilierte SQL-Anweisungen. Deshalb werden sie schneller ausgeführt als äquivalente Abfragen.

Trigger sind ein ziemlich kompliziertes Thema; es gibt viele Regeln, wie und wann man Trigger erstellt. Mit Triggern lässt sich die Datenintegrität, Domänenintegrität und referenzielle Integrität in einer Datenbank durchsetzen. Sie lernen zwei Kategorien von Triggern – AFTER und INSTEAD OF – sowie drei Triggerarten – INSERT, UPDATE und DELETE – kennen.

Der letzte Abschnitt dieser Lektion ist den benutzerdefinierten Funktionen gewidmet, einem neuen Merkmal von SQL Server. Benutzerdefinierte Funktionen sind gespeicherten Prozeduren ähnlich, lassen sich im Unterschied dazu aber auch in Anweisungen der Datenmanipulationssprache (wie INSERT, UPDATE und DELETE) verwenden.

15.1 Erstellen und Bearbeiten von Sichten

Mit Sichten können Sie die Informationen aus einer oder mehreren Tabellen in der Datenbank horizontal oder vertikal aufteilen; anders ausgedrückt: Sie zeigen dem Benutzer nur ausgewählte Felder und Zeilen. In Abbildung 15.1 ist eine Sicht zu sehen, die dem Benutzer nur die Felder für Nachname und Vorname eines Autors (vertikale Teilung) und nur Autoren, deren Nachname mit M beginnt (horizontale Teilung), anzeigt.

Nachname	Vorname	Stadt	Staat	PLZ
Caloney	Sarah	Fresno	CA	90225
Deitrich	Johanna	Plano	TX	76503
Dominic	Anthony	Bend	OR	97922
Ferrous	Eric	Towns	ME	02566
MacFeather	Stearns	Bowie	UT	82331
McDonald	Stephanie	London	MO	55823
Oreland	Lars	Reno	NV	89509
Spinola	Michael	Moreno	NM	73220

Abbildung 15.1:
Eine Sicht, die aus einer horizontal und vertikal geteilten Tabelle erzeugt wird.

au_lname	au_fname
MacFeather	Stearns
McDonald	Stephanie



Diese Sicht lässt sich mit folgendem Code erstellen:

```
CREATE VIEW dbo.vwAuthors AS  
SELECT au_lname, au_fname FROM authors  
WHERE au_lname LIKE 'M%'
```

Sichten können außerdem zur Anzeige abgeleiteter Informationen benutzt werden. So können Sie beispielsweise eine Sicht erzeugen, die den Nachnamen, Vornamen und Buchtitel eines Autors und dann ein berechnetes oder abgeleitetes Feld anzeigt, das die Anzahl der verkauften Bücher multipliziert mit den Autorentantiemen enthält.

Sichten bringen folgende Vorteile:

- Sie ermöglichen eine Kontrolle über das, was der Benutzer sehen darf, was sowohl aus Sicherheits- wie auch Bedienungsgründen nützlich ist. Der Benutzer braucht sich nicht mehr mit Zusatzinformationen zu befassen, die ihn nicht interessieren.
- Sie vereinfachen die Bedienerführung bei oft verwendeten Abfragen. Hierdurch ist der Benutzer in der Lage, die Sicht mit einer einfachen Anweisung auszuführen, statt jedes Mal eine Unmenge von Parametern bereitstellen zu müssen.
- Sie bieten Sicherheit. Die Benutzer können nur das kontrollieren, was sie auch sehen dürfen. Dabei kann es sich um eine Untermenge von Zeilen oder Spalten, statistische Informationen oder eine Untermenge von Informationen aus anderen Sichten handeln.
- Da eine Sicht ein Datenbankobjekt darstellt, können Sie ihr Benutzerberechtigungen zuweisen, was viel effizienter ist, als dieselben Berechtigungen einzelnen Spalten in einer Tabelle zuzuweisen.
- Mit SQL Server 2000 ist es jetzt möglich, Indizes für Sichten zu erstellen. Damit bieten Sichten ein sehr effizientes Verfahren, um auf häufig angeforderte Daten zuzugreifen.
- Die Daten aus einer Sicht können über das Dienstprogramm BCP exportiert werden.

Sichten stellen außerordentlich leistungsfähige und nützliche Objekte in Ihrem Datenbankentwurf dar. Zwar können Sie den Benutzern auch den expliziten Zugriff auf bestimmte Spalten und Zeilen innerhalb Ihrer Tabelle geben, jedoch ist eine Sicht viel einfacher als die verschiedenen Benutzerberechtigungen für Spalten zu verwalten. Sie ist daher die bevorzugte Verfahrensweise beim Aufteilen von Informationen. Außerdem spielen Überlegungen zur Systemleistung eine Rolle, denn wenn Sie den Spalten in einer Tabelle Berechtigungen zuweisen, müssen diese bei jedem Zugriff auf die Spalte erst überprüft werden.

Sichten erstellen

Sichten können Sie über den SQL Query Analyzer oder über den SQL Server Enterprise Manager erstellen. Dieser Abschnitt zeigt einige Beispiele, die Sie im SQL Query Analyzer ausführen können.

Die Syntax der CREATE VIEW-Anweisung lautet:



```
CREATE VIEW [Besitzer.]Sichtname [(Spaltenname [, Spaltenname...])]
[WITH Attribute]
AS Select_Anweisung
[WITH CHECK OPTION]
```

Attribute ::= { ENCRYPTION | SCHEMABINDING | VIEW_METADATA }

Eine einfache Anweisung zum Erstellen einer Sicht kann folgendermaßen aussehen:

```
CREATE VIEW all_authors
AS SELECT * FROM authors
```

Wie in den folgenden Beispielen gezeigt, können Sie diese Sicht ganz unterschiedlich verwenden:

```
SELECT * FROM all_authors
```

```
SELECT au_fname, au_lname
FROM all_authors
```

```
SELECT au_lname
FROM all_authors
WHERE au_lname like 'M%'
```

In der Anweisung CREATE VIEW können Sie verschiedene Optionen spezifizieren: WITH CHECK OPTION, WITH ENCRYPTION, WITH SCHEMABINDING und WITH VIEW_METADATA.

Die über eine Sicht vorgenommenen Datenänderungen werden standardmäßig nicht darauf geprüft, ob die betroffenen Zeilen auch innerhalb der Definition der Sicht liegen. Anders ausgedrückt heißt das: Es können auch dann Einfügungen und Aktualisierungen an Daten der zugrunde liegenden Tabelle vorgenommen werden, wenn die Sicht diese überhaupt nicht enthält. So könnte eine Sicht beispielsweise eine Tabelle horizontal teilen und Ihnen nur diejenigen Datensätze anzeigen, in denen Autoren vorkommen, deren Nachnamen mit dem Buchstaben F beginnen. Ohne die Option WITH CHECK OPTION wären Sie nun in der Lage, eine neue Zeile mit dem Nachnamen Meredith hinzuzufügen. Mit der Option WITH CHECK OPTION müssen dagegen sämtliche über die Sicht angewandten Datenänderungen den Kriterien genügen, die durch die SELECT-Anweisung in der Definition der Sicht vorgegeben sind. Durch die Option WITH ENCRYPTION wird die Anweisung CREATE VIEW in der Systemtabelle *syscomments* verschlüsselt. Nach der Verschlüsselung

einer Sichtdefinition kann diese von niemand anderem mehr eingesehen werden. Der einzige Weg zur Entschlüsselung einer Sicht besteht darin, sie zu löschen und neu zu erstellen.

Die beiden folgenden Sichten werden mit den Optionen `WITH CHECK OPTION` und `WITH ENCRYPTION` erstellt:

```
CREATE VIEW myCheck
AS select * from authors
WITH CHECK OPTION
```

```
CREATE VIEW myEncrypt
WITH ENCRYPTION
AS select * from authors
```

Beim Erstellen von Sichten sollten Sie immer die zugrunde liegende `SELECT`-Anweisung testen, bevor Sie sie zu einem Bestandteil der Sicht machen, was Ihnen die Überraschung durch unerwartete Ergebnisse erspart. Wenn Sie beispielsweise eine Sicht auf der Grundlage einer Tabelle erstellen, die lediglich 500 Datensätze enthält, jedoch in zwei Jahren schon 10.000 Datensätze enthalten könnte, kann man dann die zugrunde liegende `SELECT`-Anweisung noch als geeignet bezeichnen?



Beim Erstellen einer Sicht erzeugen Sie ein Datenbankobjekt. Vermeiden Sie dabei das Erzeugen von unterbrochenen Besitzketten, indem Sie beispielsweise dafür sorgen, dass es nur einen Entwickler gibt, der sämtliche Objekte in Ihrer Datenbank erzeugen darf. In der Regel ist dies der *dbo* oder ein Entwickler mit einem Aliasnamen für *dbo*. Der *dbo* kann dann den jeweiligen Datenbankbenutzern und Datenbankgruppen die erforderlichen Berechtigungen zum Gebrauch der Sicht zuweisen.

Die Option `WITH SCHEMABINDING` verlangt, dass die Sicht in der `SELECT`-Anweisung für Verweise auf die Basistabellen, Sichten oder benutzerdefinierte Funktionen zweiteilige Namen (*Besitzer.Objekt*) verwendet. In diesem Fall lässt sich das Schema der betreffenden Basisobjekte weder ändern noch löschen, sofern man nicht die Sicht löscht oder ändert. Wenn Sie beispielsweise eine Sicht basierend auf den Spalten *au_lname* und *au_fname* der Tabelle *authors* mit der Option `WITH SCHEMABINDING` erstellen, können Sie die Tabelle *authors* nicht löschen. Zuerst müssen Sie die Sicht löschen, erst dann ist es möglich, die Tabelle *authors* zu löschen.

Die Option `VIEW_METADATA` legt fest, dass SQL Server 2000 anstelle der Ergebnismenge der `SELECT`-Anweisung Informationen über die Sicht (Metadaten) zurückgibt, wenn Sie Metadaten des Durchsuchenmodus für eine Abfrage anfordern. Dieses Feature unterstützen derzeit die DBLIB-APIs und die OLE DB-Bibliotheken.

Die folgenden Regeln und Restriktionen sollten Sie beim Erstellen von Sichten beachten:

- `CREATE VIEW`-Anweisungen dürfen nicht mit anderen `SQL`-Anweisungen in einem Stapel zusammengefasst werden.
- Wenn Sie eine Sicht erstellen, werden sämtliche Datenbankobjekte, die von der Sicht referenziert werden, zum Zeitpunkt der Erstellung der Sicht überprüft.
- Beim Ausführen einer Sicht müssen Sie die `SELECT`-Berechtigung für die in der Sichtdefinition referenzierten Objekte haben; es sei denn, die Objekte haben keinen expliziten Besitzer. Dies bedeutet, dass Sie unter Umständen eine Sicht erzeugen, die überhaupt nicht lauffähig ist. Berechtigungen für eine Sicht werden nicht zum Zeitpunkt der Erstellung, sondern beim Ausführen der Sicht geprüft.
- Die `SELECT`-Anweisung innerhalb einer Sicht können Sie nicht zusammen mit den Klauseln `ORDER BY`, `COMPUTE` oder `COMPUTE BY` verwenden.
- Wenn Sie Objekte löschen, die in der Sicht referenziert werden (es sei denn, Sie haben die Option `SCHEMABINDING` spezifiziert), bleibt die Sicht nach wie vor existent – Sie erhalten lediglich beim nächsten Ausführen der Sicht eine Fehlermeldung. Wenn Sie beispielsweise eine Sicht über das Verknüpfen von Daten aus zwei Basistabellen erzeugen und anschließend eine dieser Tabellen löschen, bleibt die Sicht bestehen, ist aber nicht mehr ausführbar. Zur Vermeidung dieses Problems können Sie vor dem Löschen die gespeicherte Systemprozedur `sp_depends` aufrufen, die Ihnen anzeigt, welche abhängigen Objekte die Tabelle enthält.
- Temporäre Tabellen können in einer Sicht nicht referenziert werden, weshalb Sie in einer Sicht auch nicht die Anweisung `SELECT INTO` verwenden können.
- Falls Ihre Sicht die Anweisung `SELECT *` verwendet und die zugrunde liegende Tabelle, die in der `SELECT`-Anweisung referenziert wird, in der Zwischenzeit neue Spalten bekommen hat, werden diese neuen Spalten nicht in der Sicht angezeigt. Die durch das Sternchen referenzierten Felder werden bei der Erzeugung der Sicht in eine feste Feldliste aufgelöst. Zur Anzeige dieser neu hinzugekommenen Spalten müssen Sie die Sicht ändern.
- Wenn Sie eine abgeleitete Sicht auf der Grundlage einer übergeordneten Sicht erstellen, sollten Sie sich dessen bewusst sein, was die übergeordnete Sicht eigentlich genau macht. Bei umfangreichen und komplexen übergeordneten Sichten könnten Sie Schwierigkeiten bekommen.
- Die Daten in einer Sicht werden nicht separat gespeichert, was bedeutet, dass Sie bei Änderungen der Daten in einer Sicht immer direkt die Daten in den zugrunde liegenden Tabellen ändern.
- In einer Sicht können Sie nicht mehr als 1.024 Spalten referenzieren.
- Für alle abgeleiteten Spalten in Ihrer Sicht müssen Sie Namen angeben.

Informationen über Sichten

Den in der Anweisung `CREATE VIEW` enthaltenen Text erhalten Sie über den SQL Server Enterprise Manager oder durch Aufruf der gespeicherten Systemprozedur `sp_helptext`, der Sie den Namen der Sicht als Parameter übergeben:



```
sp_helptext myCheck
```



```
CREATE VIEW myCheck AS  
SELECT * FROM authors WITH CHECK OPTION
```



Falls eine Sicht mit aktivierter Option `ENCRYPTION` erstellt worden ist, können Sie `sp_helptext` (oder den SQL Server Enterprise Manager) zur Anzeige des in der Sicht enthaltenen Textes nicht benutzen. Bevor die Sicht gelesen werden kann, muss sie gelöscht und neu erstellt werden.



Die gespeicherte Systemprozedur `sp_depends` liefert einen Bericht über die Tabellen und Sichten, auf denen eine Sicht basiert, sowie über die Objekte, die von Ihrer Sicht abhängig sind. `sp_depends` können Sie für Tabellen, Sichten, gespeicherte Prozeduren und Trigger ausführen.

Zum Start von `sp_depends` für die obige Sicht führen Sie folgende Anweisung aus:



```
exec sp_depends myCheck
```



In der aktuellen Datenbank verweist das angegebene Objekt auf:

name	type	updated	selected	column
dbo.authors	user table	no	yes	au_id
dbo.authors	user table	no	yes	au_lname
dbo.authors	user table	no	yes	au_fname
dbo.authors	user table	no	yes	phone
dbo.authors	user table	no	yes	address
dbo.authors	user table	no	yes	city
dbo.authors	user table	no	yes	state
dbo.authors	user table	no	yes	zip
dbo.authors	user table	no	yes	contract

(9 row(s) affected)

Informationen über eine Sicht erhalten Sie auch durch eine Abfrage der folgenden Systemtabellen:

- *syscolumns* liefert die Spalten in einer Sicht zurück.
- *syscomments* liefert den Text der CREATE VIEW-Anweisung zurück.
- *sysdepends* liefert die Sichtabhängigkeiten zurück.
- *sysobjects* liefert den Sichtnamen zurück.

Sichttypen

Es gibt verschiedene Typen von Sichten, die jeweils vom Typ der SELECT-Anweisung abhängen, die zur Erstellung der Sicht verwendet wird. Werfen Sie dazu nun einen Blick auf Projektionen, Verknüpfungen, Aggregate, berechnete Spalten und Sichten, die auf anderen Sichten basieren.

Projektionen



Der einfachste Typ einer Sicht wird *Projektion* genannt. Eine Projektion ist einfach nur eine Untermenge der Spalten in einer Tabelle.



```
CREATE VIEW my_view
AS SELECT au_lname, au_fname
FROM authors
```

Zum Start der neu erstellen Sicht führen Sie die folgende Anweisung aus:



```
SELECT * FROM my_view
```



au_fname	au_lname
-----	-----
Bennet	Abraham
Blotchet-Halls	Reginald
Carson	Cheryl
DeFrance	Michel
...	
Stringer	Dirk
White	Johnson

Yokomoto

Akiko

(23 row(s) affected)

Verknüpfungen



Verknüpfungen verbinden Zeilen aus zwei oder mehr Tabellen durch Vergleich von Werten in vorher festgelegten Spalten. Nehmen Sie als Beispiel einmal an, dass Sie den Benutzern eine Liste der Autoren aus der Tabelle *authors* und der Titel der Bücher, die diese geschrieben haben, aus der Tabelle *titles* geben möchten. Beide Tabellen weisen in der Datenbank *pubs* eine n:n-Beziehung auf und verwenden daher, wie in Abbildung 15.2 zu sehen ist, die Tabelle *titleauthor* zur Erstellung zweier 1:n-Beziehungen.

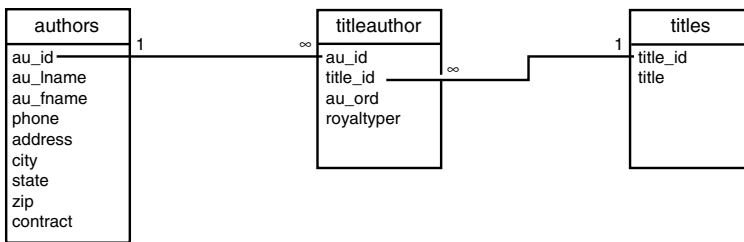


Abbildung 15.2:
Die Tabellen
authors und *titles*
sind durch die
Tabelle *titleauthor*
verknüpft.

Man kann eine Verknüpfung erzeugen, um Daten aus den Tabellen *authors*, *titles* und *titleauthor* zusammenzutragen, wobei es aber effizienter ist, eine Sicht wie folgt zu erstellen:



```

CREATE VIEW authors_titles
AS SELECT authors.au_lname, authors.au_fname, titles.title
FROM authors, titleauthor, titles
WHERE titleauthor.au_id = authors.au_id
AND titles.title_id = titleauthor.title_id
  
```

Führen Sie jetzt die folgende Anweisung aus, um Nachname, Vorname und Buchtitel von Autoren, deren Nachname mit M beginnt, aufzulisten:



```

SELECT * FROM authors_titles
WHERE au_lname like 'M%'
  
```



au_lname	au_fname	Title
MacFeather	Stearns	Cooking with Computers:...
MacFeather	Stearns	Computer Phobic AND Non-Phobic...

(2 row(s) affected)

Andere Arten von Sichten

Sichten können Sie auch unter Verwendung von Aggregatfunktionen, berechneten Spalten und Sichten auf der Grundlage anderer Sichten erstellen. In Aggregaten werden Aggregatfunktionen wie etwa AVG, COUNT und SUM verwendet. Mit berechneten Spalten erstellen Sie zusammenfassende Daten für Ihre Benutzer, während Sie eine Sicht auf eine Sicht dazu verwenden können, die ursprüngliche Sicht weiter zu verfeinern. Wenn Ihre ursprüngliche Sicht beispielsweise Informationen über sämtliche Produkte einschließlich ihrer Preise und Verkaufsmengen enthält, dann könnten Sie jetzt eine neue Sicht auf der Grundlage dieser Sicht erstellen, die anhand der für eine bestimmte Artikelnummer verkauften Einheiten die Mengen berechnet.

Es folgt nun eine Sicht mit einer berechneten Spalte:



```
CREATE VIEW book_totals AS
SELECT title, (price * ytd_sales) 'Total'
FROM titles
```

Diese Sicht können Sie nun verwenden, um alle Bücher in der Tabelle *titles* zurückzugeben, wobei eine Spalte namens *Total* den Preis multipliziert mit den Jahresverkaufszahlen (year-to-date sales) enthält:



```
Select * from book_totals
```



title	Total
The Busy Executive's Database Guide	81,859.05
Cooking with Computers: Surreptitious Balance...	46,318.20
You can combat computer Stress!	55,978.78
Straight Talk About Computers	81,859.05
Silicon Valley Gastronomic Treats	40,619.68
The Gourmet Microwave	66,515.54
The Psychology of Computer Cooking	(null)
...	
Fifty Years in Buckingham Palace Kitchens	180,397.20

Sushi, Anyone?

61,384.05

(18 row(s) affected)

Datenänderungen mit Sichten

Die Daten in einer Sicht können Sie auch ändern, wobei Sie nicht vergessen dürfen, dass Sie damit gleichzeitig die Daten in den zugrunde liegenden Tabellen selbst ändern. Beim Bearbeiten von Daten über eine Sicht sind mehrere Regeln zu beachten:

- Änderungen können immer nur eine zugrunde liegende Tabelle betreffen. Falls Ihre Verknüpfungen Daten aus mehr als einer Tabelle enthalten, dann können Sie jeweils nur die Daten in einer dieser Tabellen bearbeiten.
- Sie können nur die Daten in einer zugrunde liegenden Tabelle ändern; Aggregate, berechnete Spalten und Spalten mit eingebauten Funktionen können nicht geändert werden.
- Falls Sie beim Erstellen oder Ändern Ihrer Sichten die Option `WITH CHECK` benutzen, müssen sämtliche Daten, die über die Sicht eingefügt oder geändert werden sollen, den Restriktionen der `SELECT`-Anweisung entsprechen, die zur Erstellung der Sicht Verwendung fand. Falls Ihre `SELECT`-Anweisung beispielsweise die Klausel `WHERE au_lname LIKE "M%"` enthält, dann können Sie auch nur diejenigen Spalten hinzufügen oder ändern, die dieser `WHERE`-Klausel entsprechen.
- `NOT NULL`-Spalten, die in der zugrunde liegenden Tabelle definiert, aber nicht Teil der Sicht sind, müssen Standardwerte zugewiesen werden, damit Sie eine neue Zeile über die Sicht einfügen können.
- Falls eine Sicht einen `UNION`-Operator enthält, handelt es sich um eine so genannte *partitionierte Sicht*, die man aber für Einfüge- und Aktualisierungsoperationen unter sehr strengen Bedingungen nicht verwenden kann.

Sichten ändern

Eine Sicht können Sie über die Anweisung `ALTER VIEW` ändern. Die Syntax für `ALTER VIEW` entspricht dabei der `CREATE VIEW`-Anweisung, wobei Sie die Stellen mit `CREATE` einfach durch `ALTER` ersetzen. Die Syntax der Anweisung `ALTER VIEW` lautet also wie folgt:



```
ALTER VIEW [Besitzer.]Sichtname [(Spaltenname [, Spaltenname...])]
[WITH { ENCRYPTION | SCHEMABINDING | VIEW_METADATA }]
AS Select_Anweisung
[WITH CHECK OPTION]
```

Der Vorteil der ALTER VIEW-Anweisung besteht darin, dass sämtliche Berechtigungen, die der Sicht zugewiesen sind, trotz der Änderung beibehalten werden, was nicht zuträfe, wenn Sie die Sicht löschen und neu erstellen würden. Wenn Sie aber auf neue Objekte zugreifen, für die ein Besitzer spezifiziert ist, müssen auch die Benutzer der geänderten Sicht über die erforderlichen Berechtigungen für die neu referenzierten Objekte verfügen.

Entfernen von Sichten

Sichten entfernen Sie aus Ihrer Datenbank, indem Sie sie im SQL Server Enterprise Manager auswählen, mit der rechten Maustaste darauf klicken und anschließend im Kontextmenü LÖSCHEN wählen. Alternativ dazu können Sie auch die Anweisung DROP VIEW verwenden.

Die Syntax der Anweisung DROP VIEW lautet:



```
DROP VIEW [Besitzer.]Sichthname[, [Besitzer.]Sichthname...]
```

Zum Löschen einer Sicht namens *myCheck*, die dem Besitzer *dbo* gehört, schreiben Sie die folgende Anweisung:



```
DROP VIEW dbo.myCheck
```



Befehl(e) erfolgreich abgeschlossen.

15.2 Gespeicherte Prozeduren



Gespeicherte Prozeduren stellen vorkompilierte Transact-SQL-Anweisungen dar, die in einer SQL Server-Datenbank gespeichert werden. Da sie vorkompiliert sind, bieten sie unter den verschiedenen Typen von Abfragen auch die beste Leistung. Für die Anzeige von Informationen aus den Systemtabellen stehen zahlreiche gespeicherte Systemprozeduren zur Verfügung, deren Name mit *sp_* anfängt und die auch insbesondere bei der Administration sehr hilfreich sind. Außerdem können Sie natürlich auch Ihre eigenen gespeicherten Prozeduren erstellen.

Was macht eigentlich gespeicherte Prozeduren zu so markanten Objekten von SQL Server? Zunächst einmal stellen gespeicherte Prozeduren sehr schnell ablaufende Sätze von Transact-SQL-Anweisungen dar, die in einer SQL Server-Datenbank gespeichert sind. Wenn Sie eine gespeicherte Prozedur das erste Mal aufrufen, werden alle fünf nachstehend genannten Schritte durchlaufen. Dann wird der Ausführungsplan der gespeicherten Prozedur im Speicher abgelegt. Für darauffolgende Aufrufe derselben gespeicherten Prozedur ist nur noch Schritt 5 erforderlich.

1. Die Prozedur wird auf ihre Bestandteile hin analysiert.
2. Die Bestandteile, die andere Objekte in der Datenbank referenzieren (Tabellen, Sichten usw.), werden auf ihr Vorhandensein hin untersucht (dies ist auch unter der Bezeichnung *Auflösung* bekannt).
3. Nach dem Abschluss der Objektauflösung wird der Name der Prozedur in der Tabelle *sysobjects* und der Code zum Erstellen der Prozedur in *syscomments* gespeichert.
4. Im Verlauf der weiteren Kompilierung wird ein vorläufiger Plan erstellt, wie die Abfrage ausgeführt werden soll. Dieser Plan ist auch unter der Bezeichnung *normalisierter Plan* oder *Abfragestruktur* bekannt. Die Abfragestruktur wird in der Tabelle *sysprocedures* gespeichert.
5. Bei erstmaliger Ausführung der gespeicherten Prozedur wird die Abfragestruktur gelesen, im Rahmen eines Ausführungsplans für die Prozedur optimiert und dann ausgeführt. Hierdurch wird der Zeitaufwand vermieden, der sonst bei jedem Start der gespeicherten Prozedur durch die immer wieder neue Analyse, Auflösung und Kompilierung einer Abfragestruktur erforderlich wäre.

Ein weiterer Vorteil einer gespeicherten Prozedur besteht darin, dass nach der Ausführung der Ausführungsplan im Prozedurcache gespeichert wird. Dies bedeutet, dass die gespeicherte Prozedur bei der nächsten Ausführung innerhalb derselben Sitzung direkt in den Cache geladen und ausgeführt wird, was Ihnen einen großen Leistungsvorteil gegenüber einer immer wieder ablaufenden standardmäßigen SQL-Abfrage verschafft.

Darüber hinaus bieten gespeicherte Prozeduren folgende Vorteile:

- Mit gespeicherten Prozeduren lassen sich Geschäftsregeln verkapseln. Nach der Einkapselung können sich mehrere Anwendungen dieser Geschäftsregeln bedienen und Ihnen damit den Vorteil einer konsistenten Datenschnittstelle verschaffen. Falls Sie die Funktionalität ändern müssen, ist das nur noch an einer Stelle und nicht mehr in jeder einzelnen Anwendung erforderlich.
- Generell ist die Systemleistung schon bei normalen gespeicherten Prozeduren sehr hoch. Die Bilanz verbessert sich, je öfter eine gespeicherte Prozedur läuft, weil die Abfragestruktur im Prozedurcache gespeichert ist.

- Sie können gespeicherten Prozeduren Argumente übergeben und von ihnen Daten zurückliefern lassen.
- Gespeicherte Prozeduren können so eingerichtet werden, dass sie beim Start von SQL Server automatisch ausgeführt werden.
- Gespeicherte Prozeduren können zum Extrahieren oder Bearbeiten von Daten (nicht gleichzeitig) verwendet werden.
- Gespeicherte Prozeduren werden explizit aufgerufen. Anders als Trigger werden gespeicherte Prozeduren über Ihre Anwendung, Ihr Skript, Ihren Stapel oder Ihre Aufgabe aufgerufen.

Gespeicherte Prozeduren stellen eine außerordentlich leistungsfähige Datenbankkomponente dar. Gespeicherte Systemprozeduren sind nützlich bei der Datenbankverwaltung und -wartung. Benutzerdefinierte gespeicherte Prozeduren können Sie für die vielfältigsten Zwecke einsetzen. Gegenüber Sichten und Abfragen besitzen sie den Vorteil, dass sie vorkompiliert sind und ihr Ausführungsplan nach der ersten Ausführung im Prozedurcache – d.h. im Arbeitsspeicher – abgelegt wird. Ein weiterer Vorteil gespeicherter Prozeduren besteht darin, dass Sie einem Benutzer die Berechtigung zur Ausführung einer gespeicherten Prozedur selbst dann gewähren können, wenn dieser keinerlei Berechtigungen für die zugrunde liegenden Tabellen hat.



Wenn Sie sich für den Code von gespeicherten Prozeduren interessieren, können Sie ihn mit der gespeicherten Systemprozedur `sp_helptext` für die gespeicherten Prozeduren in der *master*-Datenbank anzeigen lassen (zum Beispiel `exec sp_helptext sp_helpdevice`).

Gespeicherte Prozeduren erstellen

Gespeicherte Prozeduren erstellen Sie mit der Anweisung `CREATE PROCEDURE`. Die Prozeduren werden in der aktuellen Datenbank gespeichert, außer wenn Sie temporäre gespeicherte Prozeduren in *tempdb* erstellen. Zum Erstellen einer gespeicherten Prozedur benötigen Sie die entsprechende Berechtigung.

Beim Erstellen einer gespeicherten Prozedur sind folgende Regeln zu beachten:

- Der Name muss den Regeln für Bezeichner entsprechen.
- Referenzierte Objekte müssen erst beim Ablauf Ihrer gespeicherten Prozedur vorhanden sein. Dieses Merkmal ist unter der Bezeichnung *verzögerte Namensauflösung* bekannt. Diese Funktion erlaubt Ihnen die Referenzierung von Objekten, die zur Kompilierzeit noch gar nicht vorhanden sind. Auf diese Weise können Sie mit Ihrer gespeicherten Prozedur temporäre Objekte erstellen, die Sie erst später in derselben gespeicherten Prozedur referenzieren.

- In einer einzelnen gespeicherten Prozedur können Sie keine Objekte mit demselben Namen erstellen und anschließend löschen und erneut erstellen.
- Sie können bis zu 1.024 Parameter einsetzen.
- Innerhalb Ihrer gespeicherten Prozedur können Sie temporäre Tabellen referenzieren. Lokale temporäre Tabellen werden am Ende der Prozedur gelöscht.
- Gespeicherte Prozeduren dürfen keine der folgenden Transact-SQL-Anweisungen enthalten: CREATE DEFAULT, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER und CREATE VIEW.
- Sie dürfen Prozeduren schachteln (bis zu einer Tiefe von 32 Ebenen).
- Wie bei Sichten werden auch in einer gespeicherten Prozedur mit einer SELECT *-Anweisung neue Spalten in der zugrunde liegenden Tabelle bei Ausführung der Prozedur nicht angezeigt. Sie müssen dazu die Anweisung ALTER PROCEDURE verwenden und die gespeicherte Prozedur neu kompilieren.



Testen Sie die SQL-Anweisungen, bevor Sie die gespeicherte Prozedur erstellen. Dadurch vermeiden Sie unerwartete Ergebnisse.

Die Syntax der Anweisung CREATE PROCEDURE lautet:



```
CREATE PROC[EDURE] Prozedurname {;Nummer}
    [{@Parameter Datentyp} [VARYING] [= default] [OUTPUT]][,...n]
    [WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}]
    [FOR REPLICATION]
    AS Sql_Anweisung[...n]
```

Schauen Sie sich zunächst einige Beispiele für die Anweisung CREATE PROCEDURE an. Dann erfahren Sie mehr über die Komponenten ;Nummer, @parameter, RECOMPILE und ENCRYPTION.



```
CREATE PROCEDURE spAuthors
AS SELECT au_fname, au_lname
FROM authors
ORDER BY au_lname DESC
```



Befehl(e) erfolgreich erstellt.

Diese Prozedur können Sie im Query Analyzer aufrufen:



EXEC spAuthors



au_fname	au_lname

Akiko	Yokomoto
Johnson	White
Dirk	Stringer
Dean	Straight
...	
Cheryl	Carson
Reginald	Blotchett-Halls
Abraham	Bennet

(23 row(s) affected)

Das Ergebnis ist eine zweispaltige Tabelle mit den Vornamen (*au_fname*) und Nachnamen (*au_lname*), die absteigend nach den Nachnamen sortiert ist.

Informationen über gespeicherte Prozeduren

Den Text in der Anweisung `CREATE PROCEDURE` erhalten Sie über den SQL Enterprise Manager oder über den Aufruf der gespeicherten Prozedur `sp_helptext`, der Sie als Parameter den Prozedurnamen übergeben.



sp_helptext spAuthors



Text

```
CREATE PROCEDURE spAuthors
AS SELECT au_fname, au_lname
FROM authors
ORDER BY au_lname DESC
```



Wie bei einer Sicht können Sie `sp_helptext` (oder den Enterprise Manager) nicht dazu einsetzen, den Text anzuzeigen, der bei der Erstellung der gespeicherten Prozedur erzeugt wurde, wenn diese mit der Option `ENCRYPTION` erstellt worden ist. Sie müssen dazu die Prozedur löschen und erneut erstellen.

Für einen Bericht über die Objekte, von denen eine gespeicherte Prozedur abhängig ist, können Sie auch die gespeicherte Prozedur `sp_depends` aufrufen. Um `sp_depends` für die oben erstellte Prozedur anzuwenden, führen Sie folgende Anweisung aus:



```
sp_depends spAuthors
```



In der aktuellen Datenbank verweist das angegebene Objekt auf:

name	type	updated	selected	column
dbo.authors	user table	no	yes	au_id
dbo.authors	user table	no	yes	au_lname
dbo.authors	user table	no	yes	au_fname
dbo.authors	user table	no	yes	phone
dbo.authors	user table	no	yes	address
dbo.authors	user table	no	yes	city
dbo.authors	user table	no	yes	state
dbo.authors	user table	no	yes	zip
dbo.authors	user table	no	yes	contract

(9 row(s) affected)

Erstellen einer Prozedurengruppe

Von den bereits erwähnten Optionen behandelt dieser Abschnitt zunächst die Option `;Nummer`. Über die Angabe eines Semikolons und einer Nummer erstellen Sie eine Gruppe gespeicherter Prozeduren. Gruppen gespeicherter Prozeduren werden oftmals für den Einsatz in ein und derselben Anwendung benutzt. Diese Prozeduren lassen sich dann einfacher verwalten, weil alle Prozeduren einer bestimmten Anwendung auf dieselbe Gruppe verweisen. Es folgt nun ein Beispiel für die Erstellung einer Gruppe von Prozeduren:

```
CREATE PROC group_sp;1
AS SELECT * FROM authors
GO
CREATE PROC group_sp;2
AS SELECT au_lname FROM authors
GO
CREATE PROC group_sp;3
AS SELECT DISTINCT city FROM authors
GO
```

Dieser Anweisungstapel erstellt eine einzelne Prozedur namens `group_sp`, die drei verschiedene Prozeduren als Gruppe enthält. Die Referenzierung der einzelnen Prozeduren

geschieht über den Teil ;Nummer in ihrem Namen. Zum Beispiel zeigt die folgende Anweisung alle Wohnorte von Autoren an:



```
EXEC group_sp;3
```



City

```
-----
Ann Arbor
Berkeley
Corvallis
Covelo
...
Vacaville
Walnut Creek
(16 row(s) affected)
```

Zum Löschen gruppierter Prozeduren brauchen Sie nur den Prozedurennamen zu löschen, wodurch alle Prozeduren, die Bestandteil der Gruppe sind, ebenfalls gelöscht werden:

```
DROP PROCEDURE dbo.group_sp
```



Bei der Arbeit mit einer Gruppe von Prozeduren können Sie nicht einzelne davon löschen. Sie müssen anstelle dessen die ganze Gruppe löschen und anschließend mit der entsprechenden Änderung neu erstellen.

Parameter in gespeicherten Prozeduren

Mit Parametern erstellen Sie gespeicherte Prozeduren, die sich bei jedem Aufruf anders verhalten können. Beispielsweise ist der Einsatz einer gespeicherten Prozedur möglich, die einen Mittelwert aus verschiedenen Testergebnissen berechnet, die Sie ihr übergeben. Welche Testergebnisse der Prozedur übergeben werden, wissen Sie bei der Erstellung der Prozedur noch nicht – trotzdem kann diese jetzt bei jeder Ausführung einen neuen Mittelwert zurückliefern. Die Syntax für den Parameterteil von `CREATE PROCEDURE` verdient eine nähere Betrachtung:



```
@Parametername Datentyp [= default|NULL] [VARYING] [OUTPUT]
```

@*Parametername* legt den Namen des Parameters innerhalb der Prozedur fest. Innerhalb einer einzigen gespeicherten Prozedur können Sie bis zu 1.024 Parameter verwenden. Der Datentyp des Parameters kann jeder systemseitig zur Verfügung gestellte oder benutzerdefinierte Datentyp außer *image* sein. *default* legt einen Standardwert für diesen Parameter fest. *VARYING* bezieht sich auf den zurückgelieferten Cursor (bzw. Recordset) und kann im Rahmen dieses Buches leider nicht weiter erörtert werden. *OUTPUT* bedeutet, dass dieser Parameter ein Rückgabeparameter sein soll.

Über die Option *OUTPUT* können Sie Daten aus der Prozedur heraus in die aufrufende Prozedur zurückliefern, was im Folgenden mit einem Beispielcode veranschaulicht wird, in dem eine gespeicherte Prozedur fünf Parameter übernimmt, deren Durchschnittswert berechnet und schließlich zurückgibt:



```
CREATE PROCEDURE scores
@score1 smallint,
@score2 smallint,
@score3 smallint,
@score4 smallint,
@score5 smallint,
@myAvg smallint OUTPUT
AS SELECT @myAvg =
    (@score1 + @score2 + @score3 + @score4 + @score5) / 5
```



Jeder Parameter in einer gespeicherten Prozedur muss, damit die Prozedur ablaufen kann, mit einem Wert versorgt werden. Hat ein Parameter einen zugewiesenen Standardwert, dann braucht der Benutzer nur dann einen Wert für diesen Parameter mitzugeben, wenn es sich um einen anderen als den Standardwert handeln soll. Wenn Sie eine Prozedur haben, die vier Parameter übernimmt, die alle über Standardwerte versorgt sind, brauchen Sie beispielsweise beim Aufruf der Prozedur nur die ersten zwei Parameter bereitzustellen. Dagegen ist es nicht möglich, beim Aufruf der Prozedur den ersten und den dritten Parameter anzugeben und den zweiten frei zu lassen (sofern Sie die Parameter nach der Reihenfolge übergeben). In SQL Server ist es auch möglich, *benannte Parameter* zu übergeben, was bedeutet, dass Sie den Parameter über die Kombination *name = parameter* bereitstellen. Hierdurch erhalten Sie die Möglichkeit, Parameter unabhängig von der Reihenfolge zu übergeben.

Um den Wert *myAvg* aus dieser Prozedur zu übernehmen, müssen Sie als Erstes eine Variable deklarieren und dann die Prozedur ausführen. Beachten Sie, dass bei dieser Verfahrensweise die Parameter nach Position übergeben werden:



```
DECLARE @AvgScore smallint
EXEC scores 10, 9, 8, 8, 10, @AvgScore OUTPUT
SELECT 'The Average Score is: ', @AvgScore
GO
```



```
-----
The Average score is: 9
(1 row(s) affected)
```



Schauen wir uns noch einmal den Ablauf an: Sie haben als Erstes die Prozedur `scores` mit der OUTPUT-Variablen `myAvg` deklariert. Anschließend haben Sie eine temporäre Variable namens `AvgScore` als OUTPUT-Variable deklariert und über diesen Parameter den Durchschnittswert `myAvg` übergeben. Der Wert von `myAvg` wurde nun von der gespeicherten Prozedur der Variablen `AvgScore` außerhalb der Prozedur übergeben. Über eine `SELECT`-Anweisung haben Sie den Wert von `AvgScore` ausgegeben. Beachten Sie, dass Sie die Werte an die gespeicherte Prozedur in einer bestimmten Reihenfolge – *nach Position* – übergeben haben. Eine Übergabe als *benannte Parameter* ist ebenfalls in der Form `Parametername = Wert` möglich und erlaubt eine Übergabe der Parameter in einer beliebigen Reihenfolge. Zum Beispiel übergibt der folgende Code die Werte in einer willkürlichen Reihenfolge:



```
DECLARE @AvgScore smallint
EXEC scores
@score1 = 10, @score3 = 9,
@score2 = 8, @score4 = 8,
@score5 = 10, @myAvg = @AvgScore OUTPUT
SELECT 'The Average score is: ', @AvgScore
GO
```



Wenn Sie die Übergabe in Form benannter Parameter durchführen, müssen Sie das für den gesamten Prozeduraufruf beibehalten. Eine gemischte Übergabe – nach Position und als benannte Parameter – ist nicht möglich.

Eine andere Möglichkeit zur Rückgabe von Daten aus der aufgerufenen Prozedur besteht in der Verwendung des Schlüsselwortes `RETURN`. Hierdurch wird der Wert einer Variablen direkt an die aufrufende Prozedur zurückgeliefert; dabei sind die `OUTPUT`-Anweisungen weder in der Definition noch im Aufruf der gespeicherten Prozedur notwendig. Schauen Sie sich nun den folgenden Beispielcode mit dem Schlüsselwort `RETURN` an.



```
CREATE PROC MyReturn
@t1 smallint, @t2 smallint, @retval smallint
AS SELECT @retval = @t1 + @t2
RETURN @retval
```

Nach der Erstellung dieser Prozedur geben Sie zum Start die folgenden Anweisungen ein:



```
DECLARE @myReturnValue smallint
EXEC @myReturnValue = MyReturn 9, 9, 0
SELECT 'The return value is: ', @myReturnValue
```



```
-----
The return value is:      18
(1 row(s) affected)
```

Verwenden der Option WITH RECOMPILE

Die Option `WITH RECOMPILE` fügen Sie entweder Ihrer Anweisung `CREATE PROCEDURE` oder `EXEC PROCEDURE` hinzu. Die Verarbeitung und der Ablauf der gespeicherten Prozedur hängen dabei davon ab, an welcher Stelle diese Option steht.

WITH RECOMPILE in der Anweisung CREATE PROCEDURE

Wenn Sie `WITH RECOMPILE` in `CREATE PROCEDURE` einsetzen, wird der Ausführungsplan für die Prozedur nicht im Prozedurcache gespeichert. Die gesamte Prozedur wird dagegen bei jedem Ablauf neu kompiliert, was der Verarbeitung einer Standardabfrage ähnelt. Sinnvoll kann dies in gespeicherten Prozeduren mit Parametern sein, die auf der Basis des normalen Ausführungsplanes eine schlechte Leistung zeigen. Da die Prozedur bei jedem Aufruf neu kompiliert wird, lässt sich die Prozedur für die neuen Parameter optimieren. Das folgende Beispiel zeigt eine gespeicherte Prozedur mit der Option `WITH RECOMPILE`:

```
CREATE PROCEDURE MyRecompileProc
WITH RECOMPILE
AS SELECT * FROM authors
ORDER BY au_lname
```

WITH RECOMPILE in der Anweisung EXEC PROCEDURE

Die Option `WITH RECOMPILE` können Sie ebenfalls in der Anweisung `EXEC PROCEDURE` verwenden. Hierdurch wird die gespeicherte Prozedur einmal bei der Ausführung kompiliert und der neue Ausführungsplan anschließend für nachfolgende Aufrufe von `EXEC PROCEDURE`

DURE im Prozedurcache gespeichert. Das folgende Beispiel verwendet die Option WITH RECOMPILE in einer EXEC PROCEDURE-Anweisung:



```
EXEC spAuthors
WITH RECOMPILE
```



au_fname	au_lname
-----	-----
Akiko	Yokomoto
Johnson	White
Dirk	Stringer
Dean	Straight
...	
Cheryl	Carson
Reginald	Blotchet-Halls
Abraham	Bennet

(23 row(s) affected)

Das Ergebnis ist eine zweispaltige Tabelle mit den Vornamen (*au_fname*) und Nachnamen (*au_lname*), die absteigend nach den Nachnamen sortiert ist. Die Prozedur wurde erneut kompiliert und der neue Ausführungsplan im Prozedurcache gespeichert.

Neukompilierung aller gespeicherten Prozeduren erzwingen

Mit der gespeicherten Prozedur `sp_recompile` erzwingen Sie, dass alle gespeicherten Prozeduren und Trigger, die auf eine bestimmte Tabelle verweisen, bei ihrem nächsten Aufruf neu kompiliert werden:

```
EXEC sp_recompile authors
```

Gespeicherte Prozeduren automatisch beim Start von SQL Server ausführen

Gespeicherte Prozeduren lassen sich auch automatisch beim Start von SQL Server ausführen. Die Ausführung dieser Prozeduren beginnt, nachdem die letzte Datenbank während des Startvorgangs wiederhergestellt worden ist.



Es lassen sich beliebig viele automatisch ausgeführte gespeicherte Prozeduren festlegen, wobei aber jede einzelne gespeicherte Prozedur eine Benutzerverbindung belegt. Wenn Sie von einer gespeicherten Prozedur andere gespeicherte Prozeduren aufrufen, ist nur noch eine einzige Benutzerverbindung zum Server erforderlich.

Zum Erstellen dieser automatisch ausgeführten gespeicherten Prozeduren verwenden Sie die gespeicherte Prozedur `sp_procoption`. Weitere Informationen entnehmen Sie bitte der Onlinedokumentation von SQL Server.

Die Option WITH ENCRYPTION

Die Option `WITH ENCRYPTION` verschlüsselt die zur Erzeugung der Prozedur eingesetzten SQL-Anweisungen und speichert den verschlüsselten Text in der Tabelle `syscomments`.

Das folgende Beispiel verwendet die Option `WITH ENCRYPTION`:

```
CREATE PROC encrypted_proc
WITH ENCRYPTION
AS SELECT * FROM authors
```

Dieser Code leistet das Gleiche wie die zusammen mit der Anweisung `CREATE VIEW` verwendete Option `WITH ENCRYPTION`. Mit der Prozedur `sp_helptext` oder mit dem SQL Enterprise Manager lässt sich der Text der gespeicherten Prozedur nun nicht mehr anzeigen.

Remote gespeicherte Prozeduren

Sie können auch Prozeduren – so genannte *remote gespeicherte Prozeduren* – implementieren und ausführen, die sich auf anderen SQL Servern befinden. Um die Verwendung von remote gespeicherten Prozeduren zu aktivieren, ist Folgendes erforderlich:

- Der Remoteserver muss den Remotezugriff erlauben, was bei der Installation von SQL Server standardmäßig der Fall ist. Sofern Sie daher Ihren Server nicht ohne diese Option neu konfiguriert haben, brauchen Sie sich um die Aktivierung dieser Option nicht zu kümmern.
- Beide Server müssen sich gegenseitig in ihren `sys.servers`-Tabellen registrieren.
- Beide Server müssen ihre Anmelde-IDs in der Tabelle `sys.slogins` führen.

Nachdem Sie den Remotezugriff eingerichtet haben, können Sie die gespeicherte Prozedur auf eine ähnliche Art und Weise ausführen, wie Sie dies schon lokal getan haben. Der Unterschied besteht darin, dass Sie den Aufruf der gespeicherten Prozedur etwa folgendermaßen qualifizieren müssen:



```
EXEC Servername.dbname.Besitzer.gespeicherteProzedur
```

Falls Sie beispielsweise die gespeicherte Systemprozedur `sp_addlogin` auf dem Server *Accounting* ausführen möchten, führen Sie den folgenden Code aus, um dem Server die Anmelde-ID Muriel hinzuzufügen:

```
EXEC Accounting.master.dbo.sp_addlogin Muriel
```

15.3 Trigger



Ein *Trigger* ist ein spezieller Typ einer gespeicherten Prozedur, die automatisch aufgerufen wird, sobald Sie Daten modifizieren, die als geschützt gelten. Trigger unterstützen die Integrität Ihrer Daten, indem sie unbefugte oder inkonsistente Änderungen verhindern. Nehmen wir als Beispiel zwei Tabellen namens *Kunden* und *Bestellungen* an. Sie können nun einen Trigger erstellen, der dafür sorgt, dass jede neue Bestellung auch eine gültige Kundennummer erhält. Ähnlich dazu können Sie den Trigger auch so formulieren, dass er bei jedem Versuch, einen Kunden aus der Kundentabelle zu löschen, nach Bestellungen Ausschau hält, die es für diesen Kunden noch gibt, und den Löschvorgang gegebenenfalls abbricht. Natürlich kann man diese Funktionalität auch über die deklarative referenzielle Integrität realisieren.

Mit Triggern lassen sich auch komplexere Geschäftsregeln durchsetzen, die man mit Einschränkungen nicht ohne weiteres realisieren kann. Beispielsweise kann ein *INSERT*-Trigger auslösen, wenn ein neuer Datensatz in die Tabelle *Bestellungen* hinzugefügt wird. Der Trigger kann die Zahlungsmoral des Kunden prüfen und daraufhin einen passenden Zahlungstermin vorschlagen.

Trigger übernehmen keine Parameter und lassen sich nicht explizit aufrufen. Das heißt, dass Sie Daten modifizieren müssen, um einen Trigger auszulösen.

Trigger kann man bis zu einer Tiefe von 32 Ebenen verschachteln. Die Funktionsweise von geschachtelten Triggern sieht etwa folgendermaßen aus: Ein Trigger für die Tabelle *Bestellungen* kann einen Eintrag in der Tabelle *Zahlungseingang* hinzufügen und die Tabelle *Zahlungseingang* löst ihrerseits einen Trigger aus, der nach überfälligen Zahlungen des Kunden Ausschau hält und Sie entsprechend darüber unterrichtet. Der eine Trigger aktualisiert eine andere Tabelle, die wiederum ihren eigenen Trigger auslöst.

Standardmäßig lösen alle Triggerarten – *INSERT*, *UPDATE*, *DELETE* – nach einer Datenmodifikation aus. In den vorherigen Versionen von SQL Server ist dieser *AFTER*-Trigger der einzige verfügbare Typ gewesen. SQL Server 2000 führt den *INSTEAD OF*-Trigger ein, der anstelle der auslösenden Aktion – d. h. anstelle der vorgesehenen Datenmodifikation – ausgeführt wird.

Aus der Sicht der Systemleistung bringen Trigger nur einen relativ geringen Verwaltungsaufwand mit sich. Die Verarbeitungszeit eines Triggers resultiert größtenteils aus Verweisen auf andere Tabellen. Das geschieht recht schnell, falls sich die Tabellen bereits im Speicher befinden, oder etwas langsamer, falls die Tabellen erst von der Festplatte gelesen werden müssen.

Trigger werden immer als Bestandteil einer Transaktion angesehen. Falls ein Trigger oder ein anderer Bestandteil einer Transaktion fehlschlägt, wird die Transaktion zurückgesetzt.

In der Vergangenheit stellten Trigger das einzige Mittel dar, um die referenzielle Integrität durchzusetzen. Beginnend mit SQL Server 7.0 können Sie die deklarative referenzielle Integrität nutzen, wodurch die meisten Trigger überflüssig geworden sind.

Die Tabellen inserted und deleted

Trigger benutzen die Tabellen `inserted` und `deleted`, die beide dieselbe Struktur wie die zugrunde liegende Tabelle bzw. die »Triggertabelle«, in der der Trigger erstellt worden ist, aufweisen. Die `inserted`- und `deleted`-Tabellen liegen im RAM, da es sich um logische Tabellen handelt. Beim Einfügen eines neuen Datensatzes in die zugrunde liegende Tabelle wird der Datensatz sowohl in die Basistabelle wie auch in die `inserted`-Tabelle eingefügt. Dadurch, dass die Werte des Datensatzes in der `inserted`-Tabelle zur Verfügung stehen, brauchen Sie keine Variablen mehr zum Speichern der Werte zu erzeugen. Wenn Sie einen Datensatz löschen, wird der gelöschte Datensatz in der `deleted`-Tabelle gespeichert. Eine Aktualisierung ähnelt dagegen einer Löschoperation mit anschließendem Einfügen: Bei der Aktualisierung des Datensatzes wird das Original in der `deleted`-Tabelle und der geänderte Datensatz sowohl in der zugrunde liegenden Tabelle als auch in der `inserted`-Tabelle abgelegt.

Erstellen von Triggern mit der Anweisung CREATE TRIGGER

Trigger erstellen Sie mit der Anweisung `CREATE TRIGGER`:



```
CREATE TRIGGER [Besitzer.]Triggername
ON [Besitzer.]Tabellenname | Sichtname
[FOR | AFTER | INSTEAD OF] {INSERT | UPDATE | DELETE}
[WITH ENCRYPTION]
AS Sql_Anweisungen
```

Eine Tabelle darf beliebig viele Trigger für die folgenden drei Triggeraktionen enthalten: `INSERT`, `UPDATE` oder `DELETE`. Sie können auch Standardtrigger erstellen, die nach Abschluss der Datenmodifikation auslösen, oder Sie können Trigger erstellen, die anstelle der Datenmodifikation auslösen.

Jede Aktion kann in einem einzelnen oder in mehreren Triggern gespeichert werden. Wird eine Aktion in verschiedenen Triggern untergebracht, muss jeder Triggernamen eindeutig sein. Beispielsweise könnten Sie einen Trigger namens `trInsUpdAuthors` für die Tabelle *authors* für INSERT- und UPDATE-Aktionen und anschließend einen zusätzlichen Trigger namens `trDelAuthors` mit der DELETE-Aktion definieren.

Falls Sie nun den Trigger `trInsUpdAuthors` ändern wollen, müssen Sie den ganzen Trigger löschen und neu erstellen oder aber die Anweisung `ALTER TRIGGER` verwenden.



Wenn Sie einen Trigger mit der Anweisung `ALTER TRIGGER` bearbeiten, wird der alte Trigger komplett durch den neuen ersetzt. Wenn Sie eine Tabelle mit zugeordneten Triggern löschen, werden automatisch auch die Trigger gelöscht.

Es gibt noch weitere Regeln für Trigger, die zu beachten sind:

- Es können keine Trigger für temporäre Tabellen erstellt werden. Trigger können aber temporäre Tabellen referenzieren.
- Trigger können keine Resultsets zurückliefern, weshalb Sie beim Einbinden von `SELECT`-Anweisungen vorsichtig sein sollten. In der Praxis ist es üblich, zusammen mit der `SELECT`-Anweisung die Klausel `IF EXISTS` im Triggercode zu verwenden.
- Mit Triggern sollte man die Datenintegrität und die referenzielle Integrität durchsetzen sowie Geschäftsregeln kapseln.
- Über die Angabe der Option `WITH ENCRYPTION` können Trigger in der Tabelle *syscomments* verschlüsselt werden.
- `WRITETEXT`-Anweisungen können Trigger nicht aktivieren. `WRITETEXT` wird zur Verarbeitung von Text oder Grafikdaten eingesetzt und stellt eine nicht protokollierte Transaktion dar.
- Die folgenden SQL-Anweisungen dürfen in einem Trigger nicht eingesetzt werden: sämtliche `CREATE`-Anweisungen, sämtliche `DROP`-Anweisungen, `ALTER TABLE` und `ALTER DATABASE`, `TRUNCATE TABLE`, `GRANT` und `REVOKE`, `RECONFIGURE`, `LOAD DATABASE` oder `TRANSACTION`, `UPDATE STATISTICS`, `SELECT INTO` und sämtliche `DISK`-Anweisungen.
- Anweisungen zum Rückgängigmachen von Transaktionen in einem Trigger können unerwartete Ergebnisse in Ihren aufrufenden Programmen zur Folge haben.

Trigger beim Einfügen und Aktualisieren auslösen

Schauen Sie sich nun ein Beispiel für einen Trigger mit INSERT- und UPDATE-Anweisungen für eine Tabelle an. Die Standardeinstellung für Trigger ist `FOR...AFTER` (das Schlüsselwort `AFTER` brauchen Sie nicht anzugeben):



```
CREATE TRIGGER trAddAuthor
ON authors
FOR INSERT, UPDATE
AS raiserror ('"%d Zeilen wurden geändert"', 0, 1, @@rowcount)
RETURN
```



Befehl(e) erfolgreich abgeschlossen.

Damit haben Sie einen Trigger erstellt, der jedes Mal dann ausgelöst wird, wenn Sie einen Datensatz in die Tabelle *authors* einfügen oder ändern. Der Trigger sendet eine Meldung, wie viele Zeilen bearbeitet worden sind. Versuchen Sie einfach einmal einen neuen Autor (*author*) einzufügen und warten Sie das Ergebnis ab:



```
INSERT authors
(au_id, au_lname, au_fname, phone, address, city, state, zip, contract)
VALUES
('555-66-7777', 'Leap', 'Frog',
'800 444-5656', '123 Sesame Street',
'West EastBrooke', 'CA', '90221', 0)
```



1 Zeilen wurden geändert
(1 row(s) affected)

Sie haben gerade erfolgreich den Datensatz »Leap Frog« in die Tabelle *authors* eingefügt! Die Meldung 1 Zeilen wurden geändert weist darauf hin, dass der Trigger tatsächlich ausgelöst worden ist.

Trigger beim Löschen auslösen

Bei Verwendung einer DELETE-Aktion sollten Sie sich darüber im Klaren sein, dass dieser Trigger nicht bei Ausführung der Anweisung `TRUNCATE TABLE` ausgelöst wird. `TRUNCATE TABLE` löscht sämtliche Zeilen in der Tabelle.

Zum Test der DELETE-Aktion fügen Sie als Erstes einen weiteren Datensatz in Ihre Tabelle ein, der dem zuerst von Ihnen erstellten Datensatz ähnelt. Dazu werden Sie nur das Schlüsselfeld Ihres Datensatzes »Leap Frog« wie folgt ändern:



```
INSERT authors
(au_id, au_lname, au_fname, phone, address, city, state, zip, contract)
VALUES
('444-55-6666', 'Leap', 'Frog', '800 444-5656',
'123 Sesame Street', 'West EastBrooke', 'CA',
'90221', 0)
```



1 Zeilen wurden geändert
(1 row(s) affected)

Damit haben Sie erfolgreich einen zweiten Datensatz »Leap Frog« in die Tabelle *authors* eingefügt. Beachten Sie, dass das Feld *au_id* auf den Wert 444-55-6666 geändert wurde.

Erstellen Sie nun einen DELETE-Trigger, der Ihnen mitteilt, wie viele Zeilen bei Ablauf dieses Triggers in der Tabelle gelöscht werden:



```
CREATE TRIGGER trDelAuthors
ON authors
FOR DELETE AS raiserror
('%d Zeilen werden aus dieser Tabelle gelöscht!',
0, 1, @@rowcount)
```



Befehl(e) erfolgreich abgeschlossen.

Löschen Sie nun sämtliche Datensätze mit dem Vornamen (*au_fname*) »Leap«:



```
DELETE FROM authors
WHERE au_fname = "Leap"
```



2 Zeilen werden aus dieser Tabelle gelöscht!
(2 row(s) affected)

Erzwingen der Datenintegrität mit Triggern

Mit Triggern lässt sich die Datenintegrität in einer Datenbank durchsetzen. Bisher waren Trigger die einzige Möglichkeit, um die referenzielle Integrität zu erzwingen. In neueren Versionen von SQL Server können Sie dafür auch Einschränkungen verwenden. Trigger bleiben jedoch trotzdem insofern sinnvoll, als man mit ihnen Geschäftsregeln kapseln und

kaskadierende Änderungen in der Datenbank realisieren kann. Nehmen Sie als Beispiel einmal an, dass eine bestimmte Buchhandlung aufgelöst werden soll: Durch einen kaskadierenden Trigger können Sie dafür sorgen, dass die Buchhandlung aus der Buchhandlungstabelle (*stores*) entfernt und sämtliche über die ID der Buchhandlung (*store_id*) verknüpften Aufträge aus der Auftragsstabelle (*sales*) gelöscht werden. Dazu können Sie den folgenden DELETE-Trigger für kaskadierendes Löschen erstellen.

Erstellen Sie zunächst einige Hilfstabellen, mit denen Sie arbeiten:



```
sp_dboption pubs, 'Select Into', TRUE
go
SELECT * INTO tblStores from pubs..stores
SELECT * INTO tblSales from pubs..sales
```



```
(6 row(s) affected)
(21 row(s) affected)
```

Führen Sie nun eine SELECT-Anweisung aus, um den derzeitigen Inhalt anzuzeigen:



```
SELECT sa.stor_id, st.stor_name
FROM tblStores st, tblSales sa
WHERE st.stor_id = sa.stor_id
```



```
stor_id stor_name
-----
6380    Eric the Read Books
6380    Eric the Read Books
7066    Barnum's
7066    Barnum's
7067    News & Brews
7067    News & Brews
7067    News & Brews
...
7131    Doc-U-Mat: Quality Laundry and Books
8042    Bookbeat
```

```
(21 row(s) affected)
```

Es sollen nun die vier Einträge mit der *stor_id* gleich 7067 gelöscht werden. Als Nächstes erstellen Sie den Trigger für die Tabelle *tblSales*, der Ihnen Auskunft darüber gibt, wie viele Aufträge (*sales*) gelöscht werden, wenn die damit in Verbindung stehende Buchhandlung (*store*) aus der Tabelle *tblStores* gelöscht wird:



```
CREATE TRIGGER trDelSales
ON tblSales
FOR DELETE AS
raiserror('%d Zeilen werden aus der sales-Tabelle gelöscht!'
, 0, 1, @@rowcount)
```



Befehl(e) erfolgreich abgeschlossen.

Erstellen Sie jetzt den DELETE-Trigger für *tblStores*:



```
CREATE TRIGGER trDelStore
ON tblStores
FOR DELETE AS
DELETE tblSales FROM deleted where deleted.stor_id = tblSales.stor_id
```



Befehl(e) erfolgreich abgeschlossen.

Löschen Sie zum Schluss die *stor_id* 7067, die für die Buchhandlung »News & Brews« steht:



```
DELETE FROM tblStores
WHERE tblStores.stor_id = '7067'
```



4 Zeilen werden aus der sales-Tabelle gelöscht!
(1 row(s) affected)

Der DELETE-Trigger für *tblStores* wird jetzt ausgelöst und löscht sämtliche verknüpften Datensätze in *tblSales*. Wenn Sie die obige SELECT-Anweisung noch einmal ausführen, werden Sie sehen, dass »News & Brews« nicht mehr vorhanden ist und statt 21 nur noch 17 Zeilen erscheinen.

Kapseln von Geschäftsregeln

Das Kapseln von Geschäftsregeln geschieht in der Regel über Einschränkungen, Standards, Datentypen und Regeln. Sie können zu diesem Zweck aber auch Trigger einsetzen. Trigger sind insbesondere dann sinnvoll, wenn Sie andere Tabellen referenzieren müssen, weil diese Aktion bei Einschränkungen, Standards, Datentypen und Regeln nicht erlaubt

ist. Als Anknüpfung an die Beispiele oben werden Sie nun Ihren Trigger `trDelSales` für die Tabelle `tblSales` dahingehend ändern, dass er als Geschäftsregel arbeiten kann. Dabei unterbinden Sie das Löschen einer Buchhandlung (*store*), wenn die Aufträge (*sales*) größer oder gleich 20 sind.

Bevor Sie diesen Trigger neu erstellen können, müssen Sie ihn unter Umständen erst löschen:



```
DROP TRIGGER trDelSales
GO
CREATE TRIGGER trDelSales
ON tblSales
FOR DELETE AS
IF (SELECT COUNT(*) FROM deleted
WHERE deleted.qty >= 20) > 0
BEGIN
PRINT 'Diese Buchhandlungen können nicht gelöscht werden.'
PRINT 'Einige haben mehr als 20 Aufträge!'
PRINT 'Transaktion wird rückgängig gemacht!'
ROLLBACK TRANSACTION
END
```



Befehl(e) erfolgreich abgeschlossen.

Testen Sie jetzt den neuen Trigger:



```
DELETE FROM tblSales
WHERE stor_id = '7066'
```



Diese Buchhandlungen können nicht gelöscht werden.
Einige haben mehr als 20 Aufträge!
Transaktion wird rückgängig gemacht!

Es gab also mindestens eine Buchhandlung mit der *stor_id* 7066, die mehr als 20 Aufträge hat. Überprüfen Sie das mit der folgenden `SELECT`-Anweisung:



```
SELECT stor_id, qty FROM tblSales
```



stor_id	qty
-----	-----
6380	5
6380	3
7066	50
7066	75
...	
8042	10
8042	25
8042	30

(17 row(s) affected)

Führen Sie noch einmal die DELETE-Anweisung mit der *stor_id* 6380 aus. Wie Sie aus dem obigen Code erkennen können, löscht die Anweisung jetzt die Buchhandlungen, weil kein Eintrag mit einem *qty*-Wert größer als 20 vorhanden ist.



```
DELETE FROM tblSales
WHERE stor_id = '6380'
```



(2 rows(s) affected)

Führen Sie die SELECT-Anweisung aus und Sie werden sehen, dass beide Buchhandlungen mit der ID 6380 verschwunden und nur noch 15 statt 17 Datenzeilen vorhanden sind.

Erzwingen der referenziellen Integrität

Mit Triggern lässt sich auch die referenzielle Integrität erzwingen. Letztendlich ist das ihr Hauptzweck in einer Datenbank. Insbesondere lassen sie sich leicht für kaskadierende Änderungen und Löschungen einsetzen. Trigger werden immer dann geprüft, wenn Datenmodifikationen auftreten, wobei zuerst die Einschränkungen für die Triggertabelle untersucht werden. Wurde eine Einschränkung verletzt, wird der Trigger auch nicht ausgelöst.

Es folgt nun ein Beispiel eines Triggers, der die referenzielle Integrität erzwingt. Er stellt sicher, dass vor dem Einfügen eines Auftrags in die Tabelle *sales* eine gültige Buchhandlungs-ID (*stor_ID*) in der Buchhandlungstabelle (*stores*) existiert.

```
CREATE TRIGGER trInsUpdSales
ON tblSales
FOR INSERT, UPDATE AS
IF (SELECT COUNT(*) FROM tblStores, inserted
```



```
WHERE tblStores.stor_id = inserted.stor_id) = 0
BEGIN
PRINT 'Diese stor_ID existiert nicht in der stores-Tabelle!'
ROLLBACK TRANSACTION
END
```

Dieser Trigger wird nun bei jeder einzelnen UPDATE- oder INSERT-Anweisung für die Tabelle *tblSales* ausgeführt. Er stellt sicher, dass sich in *tblStore* eine gültige *stor_id* befindet. Unter Umständen wird dieser Trigger jedoch bei einem SELECT INTO nicht korrekt ausgelöst. Sobald Sie mehrere Zeilen haben, die Sie bearbeiten möchten, sollten Sie immer überprüfen, ob der Zeilenzähler von eingefügten *stor_ids* auch gleich der Anzahl der eingefügten Aufträge ist. Es folgt nun ein Beispiel für die Codierung dieses Triggers zur Bearbeitung von mehreren Zeilen (auch hier müssen Sie den Trigger gegebenenfalls erst löschen, bevor Sie ihn neu erstellen können):

```
DROP TRIGGER trInsUpdSales
GO
CREATE TRIGGER trInsUpdSales
ON tblSales
FOR INSERT, UPDATE AS
DECLARE @rc int
SELECT @rc = @@rowcount
IF (SELECT COUNT(*) FROM tblStores, inserted
WHERE tblStores.stor_id = inserted.stor_id) = 0
BEGIN
PRINT 'Diese stor_ID existiert nicht in der stores-Tabelle!'
ROLLBACK TRANSACTION
END
IF (SELECT COUNT(*) FROM tblSales, inserted
WHERE tblSales.stor_id = inserted.stor_id) <> @rc
BEGIN
PRINT 'Nicht alle Aufträge haben eine gültige'
PRINT 'stor_id in der stores-Tabelle!'
ROLLBACK TRANSACTION
END
```

INSTEAD OF-Trigger

Wenn Sie einen INSTEAD OF-Trigger verwenden, wird der ursprüngliche Code, der eine Modifikation an der Tabelle vorgenommen hat, nicht ausgeführt; statt dessen wird der Triggercode ausgeführt. Beispielsweise können Sie einen Trigger für die *authors*-Tabelle erstellen, um die Benutzer darüber zu benachrichtigen, dass Autoren nicht gelöscht werden können. Prinzipiell lässt sich das Gleiche auch mit einem AFTER-Trigger durchführen; allerdings ist es dazu erforderlich, dass die Daten tatsächlich erst modifiziert worden sind

und dass Sie dann die Aktionen (d.h. die Transaktion) aus dem Triggercode heraus rückgängig machen. Wenn Sie gleich einen `INSTEAD OF`-Trigger einsetzen, findet die Aktualisierung gar nicht erst statt. Diese Lösung ist also effizienter als der vorher beschriebene Trigger. Der folgende Code zeigt ein Beispiel für den `INSTEAD OF`-Trigger:

```
USE pubs
GO
CREATE TRIGGER trIO_DelAuthors
ON authors INSTEAD OF DELETE AS
PRINT 'Aus der authors-Tabelle können keine Autoren gelöscht werden.'
```

Um diesen Trigger zu testen, versuchen Sie einfach, einen Autor mit dem Nachnamen White zu löschen:



```
USE pubs
GO
DELETE authors WHERE au_lname = 'White'
```



Aus der authors-Tabelle können keine Autoren gelöscht werden.
(1 row(s) affected)

Informationen über Trigger

Wie für die anderen Objekte, die Sie in der heutigen Lektion kennen gelernt haben, können Sie auch für den Text einer Triggeranweisung die gespeicherte Systemprozedur `sp_helptext` ausführen. Natürlich besitzen verschlüsselte Trigger keine Einträge in *syscomments*, die Sie sich im Klartext ansehen können. Bei einer Aktualisierung Ihrer Datenbank müssen solche verschlüsselten Objekte gelöscht und neu erstellt werden. Unverschlüsselte Objekte werden dagegen automatisch auf die nächst höhere Version aktualisiert.

15.4 Benutzerdefinierte Funktionen

Ein leistungsfähiges neues Merkmal von SQL Server 2000 sind die benutzerdefinierten Funktionen (UDF, User Defined Functions). Eine benutzerdefinierte Funktion können Sie erstellen, um entweder einen Skalarwert (einen Einzelwert) oder eine Tabelle zurückzugeben. Die `RETURNS`-Klausel in der benutzerdefinierten Funktion bestimmt, ob es sich um eine Skalarfunktion oder eine Funktion mit Tabellenrückgabe handelt. Es dauert etwas, bis man mit dem Einsatz von benutzerdefinierten Funktionen vertraut ist, es lässt sich aber voraussagen, dass benutzerdefinierte Funktionen zu den sehr häufig gebrauchten Erweiterungen von SQL Server 2000 gehören werden.

Man kann zwei Arten von Funktionen mit Tabellenrückgabe verwenden:

- Wenn Sie eine Tabelle aus einer einzelnen SELECT-Anweisung zurückgeben, handelt es sich um eine so genannte *Inlinefunktion mit Tabellenrückgabe*. Wenn Sie eine Tabelle auf diese Weise erstellen, bedeutet das, dass alle Spaltennamen und Datentypen durch die SELECT-Anweisung bestimmt werden.
- Wenn Sie neue Spaltennamen und andere Datentypen als Teil Ihrer Funktionsdefinition spezifizieren, handelt es sich um eine *Funktion mit mehreren Anweisungen und Tabellenrückgabe*.

Bei Funktionen mit mehreren Anweisungen können Sie die folgenden Anweisungen einbinden. Alle nicht in der Liste aufgeführten Anweisungen sind in benutzerdefinierten Funktionen auch nicht zulässig.

- Zuweisungsanweisungen
- Anweisungen, die den Programmfluss steuern
- Cursoroperationen, die lokal zur Funktion sind
- Deklarationsanweisungen für Variablen und Cursor, die lokal zur Funktion sind
- INSERT, UPDATE und DELETE-Anweisungen, die lokal zur Funktion sind
- SET-/SELECT-Anweisungen, mit denen Werte an lokale Funktionsvariablen zugewiesen werden

Die nicht deterministischen Funktionen von SQL Server 2000 sind nicht innerhalb einer Funktion zulässig, weil diese Funktionen unterschiedliche Werte zurückgeben können, selbst wenn man ihnen die gleichen Eingabeparameter übergibt. Zu dieser Kategorie gehören Funktionen wie @@CONNECTIONS, GETDATE und RAND.

Skalarfunktionen

Am Einfachsten lassen sich Skalarfunktionen erstellen und einsetzen. Die grundsätzliche Syntax sieht folgendermaßen aus:



```
CREATE FUNCTION [Besitzername.]Funktionsname
([{@Parametername Skalarparameterdatentyp [ = Standard]}[...n]])
RETURNS Skalarrückgabedatentyp
[WITH ENCRYPTION | SCHEMABINDING]
[AS]
RETURN [(] Auswahlanweisung [)]
```



Die Elemente in dieser Syntax haben folgende Bedeutung:

- *Besitzername* spezifiziert den Objektbesitzer.
- *Funktionsname* spezifiziert den Namen der benutzerdefinierten Funktion. Der Name muss den Regeln für SQL Server-Bezeichner entsprechen.
- *@Parametername* kennzeichnet alle Parameter, die an die Funktion zu übergeben sind. Es lassen sich bis zu 1024 Parameter übergeben. Das Verfahren ist mit den bereits in dieser Lektion behandelten gespeicherten Prozeduren vergleichbar.
- *Skalarparameterdatatype* ist der Datentyp für den Rückgabewert der Funktion. Nicht skalare Datentypen (die mehrere Werte umfassen) werden nicht unterstützt; dazu gehören Tabellen und Cursor. Außerdem werden die Datentypen *rowversion* und *timestamp* nicht unterstützt. Anstelle dieser Typen können Sie *uniqueidentifier* und *binary(8)* verwenden.

Gerüstet mit diesem Wissen erstellen Sie nun Ihre erste benutzerdefinierte Funktion. Das Beispiel in Listing 15.1 gibt den letzten Tag des Monats zurück, wenn Sie ein Datum eingeben. Die Programmlogik zerlegt das Datum in seine Bestandteile, addiert einen Monat zum Datum, geht zum ersten Tag des nächsten Monats und subtrahiert dann einen Tag. Das Ergebnis ist der letzte Tag dieses Monats.

Listing 15.1: Eine benutzerdefinierte Funktion für den letzten Tag des Monats

```
CREATE FUNCTION dbo.fnLastDayOfMonth (@mydate datetime)
RETURNS datetime
AS
BEGIN
    -- Variablen deklarieren, um die einzelnen Datumskomponenten
    -- aufzunehmen:
    DECLARE @yy int
    DECLARE @mm int
    DECLARE @dd int
    DECLARE @tempdate datetime
    DECLARE @foo varchar(10)

    -- Datumskomponenten speichern:
    SET @yy = DATEPART(yy, @mydate)
    SET @mm = DATEPART(mm, @mydate)
    SET @dd = DATEPART(dd, @mydate)

    -- Wenn der Monat Dezember ist, dann Monat auf Januar und das Jahr
    -- auf Jahr + 1 setzen, andernfalls Monat um 1 inkrementieren:
    IF @mm = 12
    BEGIN
```



```

        SET @mm = 1
        SET @yy = @yy + 1
    END
ELSE
    BEGIN
        SET @mm = @mm + 1
    END

-- Den Tag der temp. Zeichenfolge auf das Datum setzen, wobei der Tag
-- auf den ersten Tag des folgenden Monats gesetzt ist.
-- Dann die temp. Zeichenfolge zu einem echten Datum konvertieren.
-- Dann einen Tag vom Datum subtrahieren und diesen Wert zurückgeben.
    SET @foo = CONVERT(varchar(4), @yy) + '/01/' +
        CONVERT(varchar(2), @mm)
    SET @tempdate = CONVERT(datetime, @foo)
    SET @tempdate = DATEADD(dd, -1 , @tempdate)

    RETURN @tempdate
END

```

Die neue Funktion können Sie mit den folgenden Anweisungen testen:



```

SELECT dbo.fnLastDayOfMonth('12.12.1999')
SELECT dbo.fnLastDayOfMonth('02.02.1999')
SELECT dbo.fnLastDayOfMonth('02.02.2000')

```

Die Ergebnisse sollten wie folgt aussehen:



```

-----
1999-12-31 00:00:00.000

```

(1 row(s) affected)

```

-----
1999-02-28 00:00:00.000

```

(1 row(s) affected)

```

-----
2000-02-29 00:00:00.000

```

(1 row(s) affected)

Inlinefunktionen mit Tabellenrückgabe

Von den beiden Funktionsarten mit Tabellenrückgabe sind die Inlinefunktionen relativ einfach:



```
CREATE FUNCTION [Besitzername.]Funktionsname
([{@Parametername Skalarparameterdatentyp [ = Standard]}[...n]])
RETURNS TABLE
[WITH ENCRYPTION | SCHEMABINDING]
[AS]
RETURN [()] Auswahlanweisung [()]
```

Die Parameter entsprechen der Syntax für die Skalarfunktion mit Ausnahme der *Auswahl-anweisung*, die eine normale einzelne SELECT-Anweisung ist. Denken Sie daran, dass alle Spalten benannt sein müssen, was auch auf berechnete Spalten zutrifft.

Das Beispiel in Listing 15.2 gibt eine Tabelle mit den pro Autor berechneten Tantiemen zurück.

Listing 15.2: Eine benutzerdefinierte Funktion für die Anzeige der Tantiemen pro Autor

```
CREATE FUNCTION dbo.fnRoyaltiesByAuthor()
RETURNS TABLE AS
RETURN (SELECT a.au_fname, a.au_lname, t.title,
'RoyToDate' = (t.ytd_sales * t.price * t.royalty / 100)
FROM authors a, titles t, titleauthor ta
WHERE a.au_id = ta.au_id AND t.title_id = ta.title_id)
GO
```

```
SELECT * FROM dbo.fnRoyaltiesByAuthor()
SELECT au_fname, RoyToDate FROM dbo.fnRoyaltiesByAuthor()
```

Funktionen mit mehreren Anweisungen und Tabellenrückgabe

Die Funktionen mit mehreren Anweisungen und Tabellenrückgabe setzt man selbstverständlich ein, um eine Tabelle zurückzugeben:



```
CREATE FUNCTION [Besitzername.]Funktionsname
([{@Parametername Skalarparameterdatentyp [ = Standard]}[...n]])
RETURNS @Rückgabeveriable TABLE Tabellentypdefinition
[WITH ENCRYPTION | SCHEMABINDING]
[AS]
BEGIN
```



```

Funktionsrumpf
RETURN [([] Auswahlanweisung [])]
END

```

Die zurückgegebene Tabelle muss nicht unbedingt durch eine einzelne SELECT-Anweisung erzeugt werden (daher die Bezeichnung *Funktionen mit mehreren Anweisungen*). Zum Beispiel können Sie die Daten verarbeiten und eine temporäre Tabelle erzeugen. Dann können Sie zusätzliche Verarbeitungsschritte basierend auf dieser Tabelle durchführen und schließlich die neue Tabelle mit den Ergebnissen zurückgeben.

Das folgende Beispiel erweitert das vorher gezeigte, bei dem die Funktion `RoyaltyByAuthor` die Tantiemen basierend auf den verkauften Büchern zurückgegeben hat. Das Problem bei dieser Funktion besteht darin, dass viele Bücher mehr als einen Autor haben können. Wenn die Tantiemen bei Büchern mit mehreren Autoren aufgeteilt werden, erhalten Sie falsche Ergebnisse zurück.

Um dieses Problem zu beseitigen, muss man die Tantiemen pro Titel ermitteln, diese Tantiemen durch die Anzahl der Autoren für jeden Titel dividieren und dann eine Tabelle mit dem Namen, dem Titel und den neu berechneten Tantiemen für den Autor zurückgeben. Wir gehen davon aus, dass die Tantiemen gleichmäßig unter den Autoren aufgeteilt werden.

Obwohl dieses Beispiel mehrere Anweisungen und eine temporäre Tabelle verwendet, könnte man das Gleiche auch mit einer komplexen korrelierten Unterabfrage realisieren:

```

CREATE FUNCTION dbo.fnRoyaltySplitByAuthor()
RETURNS @RoyByAuthor TABLE
(
    au_lname varchar(40) NOT NULL,
    au_fname varchar(20) NULL,
    title varchar(80) NOT NULL,
    RoyToDate float)
AS
BEGIN
    -- Eine temp. Tabelle erstellen, um Autorenzahl pro Titel zu speichern
    DECLARE @temptbl TABLE (title_id varchar(6) NOT NULL,
        num_authors tinyint NOT NULL)
    INSERT @temptbl
        SELECT title_id, COUNT(au_id) FROM titleauthor
        GROUP BY title_id

    -- Rückgabetable mit einer Verknüpfung zwischen temp. Tabelle und
    -- allen anderen zutreffenden Tabellen bilden.

    INSERT @RoyByAuthor
        SELECT a.au_fname, a.au_lname, t.title,
            'RoyToDate' = ((t.ytd_sales * t.price * t.royalty / 100)
                / tt.num_authors)

```



```
FROM authors a, titles t, titleauthor ta, @temptbl tt
WHERE a.au_id = ta.au_id AND t.title_id = ta.title_id
      AND t.title_id = tt.title_id
RETURN
END

GO

SELECT * FROM dbo.fnRoyaltySplitByAuthor()
```

15.5 Zusammenfassung

In der Lektion des heutigen Tages haben Sie vier leistungsfähige Datenbankobjekte kennen gelernt: Sichten, gespeicherte Prozeduren, Trigger und benutzerdefinierte Funktionen. Sichten geben Ihnen die Möglichkeit, Ihren Benutzern ausschließlich bestimmte Spalten und Zeilen aus Ihren Tabellen zur Verfügung zu stellen. Gespeicherte Systemprozeduren sind für die Datenbankverwaltung und -wartung nützlich. Mit benutzerdefinierten gespeicherten Prozeduren können Sie Ihre eigenen Vorstellungen umsetzen. Gespeicherte Prozeduren haben den Vorteil, vorkompiliert zu sein und daher viel schneller als Sichten abzulaufen. Trigger stellen einen speziellen Typ einer gespeicherten Prozedur dar, der bei der Änderung von Daten in der Triggertabelle automatisch ausgeführt wird. Mit Triggern wird die Sicherstellung der Integrität Ihrer Datenbank unterstützt, indem unbefugte oder inkonsistente Änderungen verhindert werden. Ein neues Merkmal von SQL Server 2000 sind die benutzerdefinierten Funktionen, die dem Datenbankentwickler ein hohes Maß an Flexibilität bieten, um SQL-Skripts zu erstellen.

15.6 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F *Was wird schneller ausgeführt: Trigger, Sichten oder gespeicherte Prozeduren?*

A Gespeicherte Prozeduren und Trigger sind beide schneller als Sichten. Gespeicherte Prozeduren und Trigger stellen im Grunde vorkompilierte SQL-Anweisungen dar. Sie laufen daher etwa mit der gleichen Geschwindigkeit ab und erfordern generell den gleichen Verwaltungsaufwand. In der Ablaufreihenfolge von SQL Server werden zuerst die gespeicherten Prozeduren und Sichten und anschließend die Trigger ausgeführt. Wenn Sie also Probleme mit einer gespeicherten Prozedur lösen können, brauchen Sie im Sinne der Effizienz gar nicht bis zur Triggerebene hinabsteigen, nur um alles das rückgängig zu machen, was bereits erledigt ist.

F *Wie kann ich eine Liste sämtlicher gespeicherter Prozeduren auf meinem Server erhalten?*

A Die gespeicherten Prozeduren pro Datenbank erhalten Sie über die folgende Abfrage:

```
SELECT name
FROM sysobjects
WHERE type = 'P'
```

F *Wie kann ich eine Liste sämtlicher Trigger auf meinem Server bekommen?*

A Die Trigger pro Datenbank erhalten Sie über die folgende Abfrage, die Ihnen jede Tabelle mit den verknüpften Triggern zurückliefert:

```
SELECT name,
'INSERT' = object_name(instrig),
'UPDATE' = object_name(updtrig),
'DELETE' = object_name(deltrig)
FROM sysobjects
WHERE type = 'U'
AND (instrig <> 0 OR updtrig <> 0 OR deltrig <> 0)
```

A Das zweite Beispiel liefert die Triggerinformationen für eine einzelne Tabelle:

```
EXEC sp_helptrigger authors
```

F *Wie kann ich eine Liste sämtlicher Objekte in meiner Datenbank erhalten?*

A Führen Sie die folgende SELECT-Anweisung aus, um die Objektinformationen aus der Systemtabelle sysobjects der Datenbank abzurufen:

```
SELECT * FROM sysobjects
```

F Wie kann ich den Text der Transact-SQL-Anweisungen erhalten, mit denen Trigger, gespeicherte Prozeduren, Sichten oder benutzerdefinierte Funktionen erstellt werden?

A Am Einfachsten lässt sich das mit der gespeicherten Prozedur `sp_helptext` realisieren, wie es die folgenden Beispiele zeigen:

```
EXEC sp_helptext fnRoyaltySplitByAuthor
EXEC sp_helptext trInsInsteadOfAuthors
EXEC sp_helptext titleview
```

Kontrollfragen

1. Was wird durch Trigger erzwungen?
2. Pro Tabelle können Sie _____ Trigger festlegen.
3. *Richtig oder falsch:* Sichten können auf die benötigten Daten beschränkt werden, bieten Sicherheit und erlauben Änderungen an den zugrunde liegenden Tabellen. Außerdem sind sie schneller als gespeicherte Prozeduren.
4. *Richtig oder falsch:* Innerhalb einer Sicht können Sie mehrere zugrunde liegende Tabellen aktualisieren.
5. *Richtig oder falsch:* Sie können gespeicherte Prozeduren dazu einsetzen, Daten aus den zugrunde liegenden Tabellen zurückzuliefern, für die Sie keine Berechtigung haben.
6. *Richtig oder falsch:* Eine Sicht können Sie dazu verwenden, Daten aus zugrunde liegenden Tabellen zurückzuliefern, für die Sie keine Berechtigung haben.
7. *Richtig oder falsch:* Durch die deklarative referenzielle Integrität (DRI), Einschränkungen, Datentypen, Standards und Regeln sind Trigger überflüssig geworden.
8. *Richtig oder falsch:* Welche der folgenden Datenbankobjekte werden beim Löschen einer Tabelle ebenfalls gelöscht: Sichten, gespeicherte Prozeduren oder Trigger?
9. *Richtig oder falsch:* Man kann eine Tabelle löschen, für die eine Sicht mit der Option `SCHEMABINDING` erstellt wurde.
10. *Richtig oder falsch:* Benutzerdefinierte Funktionen können Skalarwerte und Tabellen zurückgeben. Außerdem kann man benutzerdefinierte Funktionen als Teil der `SELECT`-Anweisung oder in der `FROM`-Klausel einer Abfrage einsetzen.

Übungen

1. Erstellen Sie eine Sicht, die zeigt, welche Autoren welche Bücher geschrieben haben.
2. Erstellen Sie einen Trigger, der verhindert, dass ein neuer Buchtitel hinzugefügt wird, dem kein gültiger Verleger (*publisher*) zugeordnet ist.
3. Erstellen Sie eine gespeicherte Prozedur, die Ihnen anzeigt, welche Bücher über welche Buchhandlungen verkauft werden. (*Hinweis:* Um die Aufgabe zu lösen, müssen Sie drei Tabellen miteinander verknüpfen.)
4. Erstellen Sie eine benutzerdefinierte Funktion, die den Durchschnittspreis aller bis jetzt verkauften Bücher berechnet.



Replikation –
Entwurf

Die heutige Lektion beschäftigt sich mit der Replikation in SQL Server. Dabei erfahren Sie, was Replikation ausmacht, warum man Replikation verwendet und was man publizieren kann. Sie werden sich auch die unterschiedlichen Methoden der Verteilung von Daten ansehen und bestimmen, was Ihren Bedürfnissen am nächsten kommt. Replikation verwendet das Modell »Publizieren / Abonnieren« mit Artikeln und Publikationen. Sie lernen etwas über die verschiedenen Agents, mit denen SQL Server Ihre Daten von einem Standort zu einem anderen verschiebt. Dann sehen Sie sich die verschiedenen Replikationsszenarios mit ihren Vorteilen an. Am Ende beschäftigt sich diese Lektion noch mit dem Replizieren von Daten in heterogenen Umgebungen.

Die Grundlagen zur Replikation, die Sie sich hier aneignen, sind unbedingt erforderlich, damit Sie sich in der morgigen Lektion mit der eigentlichen Implementierung der Replikation vertraut machen können.

16.1 Was ist Replikation?

Replikation erzeugt eine Umgebung, die erlaubt, dass mehrere Kopien derselben Informationen an mehrere Datenbanken über Ihr Unternehmen verteilt werden. Replikation hat folgende Vorteile:

- Die Daten befinden sich näher beim Benutzer.
- Sie beseitigt den Einfluss von leseintensiven OLAP-Umgebungen (Online Analytical Processing, analytische Onlineverarbeitung) aus den transaktionsintensiven OLTP-Umgebungen (Online Transaction Processing, Online-Transaktionsverarbeitung).
- Sie verringert Konflikte zwischen mehreren Standorten, die versuchen, mit denselben Informationen zu arbeiten.
- Datenbankstandorte können autonom mit den replizierten Daten umgehen. Das heißt, dass sie ihre eigenen Regeln, Prozeduren und Sichten mit ihren Kopien der Daten definieren können.

Die beiden grundlegenden Typen der Replikation sind Replikation und verteilte Transaktionen. Beide Typen erlauben Ihnen, die unterschiedlichen Kopien Ihrer Daten aktuell zu halten. Sie können sogar beide Strategien gleichzeitig in Ihrer Umgebung einsetzen.



Replikation dupliziert und verteilt Kopien Ihrer Daten an die unterschiedlichen Orte in Ihrer Umgebung. Diese Aktualisierungen finden periodisch statt, so dass Sie diese replizierten Standorte autonomer halten können. Die Datenbanken an jedem Standort müssen nicht die ganze Zeit mit dem Server verbunden sein, der die Daten publiziert. Die Replikation kann lediglich Transaktionen bewegen oder einen kompletten Snapshot der Daten verschieben.

16.2 Faktoren bei verteilten Daten

Bei der Entscheidung für eine bestimmte Verteilungsmethode der Daten sind verschiedene Faktoren zu beachten:

- *Eigenständigkeit des Standorts* hängt von den Antworten auf die folgenden Fragen ab: Wie viel Unabhängigkeit von der publizierenden Datenbank müssen Ihre Abonnentensites haben? Wie lange ist eine Version der Daten an der Abonnentensite brauchbar? Wie oft müssen Sie sich mit der publizierenden Site verbinden und Ihre Kopie der Daten aktualisieren?
- *Transaktionskonsistenz* bezieht sich auf die Transaktionen selbst. Müssen alle gespeicherten Transaktionen zur selben Zeit angewendet werden oder überhaupt nicht? Ist es akzeptabel, Transaktionen verzögert zu verarbeiten, sofern die Reihenfolge der Transaktionen gewahrt bleibt? Werden die Transaktionen in einer Weise angewendet, die nicht mit der Konsistenz Ihrer Daten im Konflikt steht?
- *Latenz* bezieht sich darauf, wann die Kopien der Daten angewendet werden. Müssen Ihre Daten in 100 % der Zeit 100 % synchron sein oder ist es akzeptabel, die Daten periodisch aktualisieren zu lassen? Wenn Sie eine gewisse Latenz tolerieren können, wie groß ist die akzeptable Wartezeit?

Bevor Sie sich in die verschiedenen Verteilungsmethoden vertiefen, behalten Sie das Folgende im Gedächtnis, wenn Sie entscheiden, wo und was publiziert werden soll:

- Was werde ich publizieren? Erhalten die Abonnenten die gesamten Daten oder nur Untermengen meiner Daten? Sollten meine Daten nach Bereichswerten oder Postleitzahlen unterteilt werden? Sollte ich Abonnenten meiner Daten erlauben, mir Aktualisierungen zu schicken? Wenn ich Aktualisierungen zulasse, wie sollten sie implementiert werden?
- Wer kann Zugriff auf meine Daten haben? Sind diese Benutzer online oder offline? Sind sie quer über das Land verteilt und über teure Telefonleitungen verbunden?
- Wie oft sollte ich meine Daten mit den Abonnenten synchronisieren? Wie oft werden Änderungen an sie geschickt?
- Wie sieht das Netzwerk aus? Ist das Netzwerk schnell? Sollte ich die Daten stärker partitionieren, um den mit der Replikation verbundenen Datenverkehr zu minimieren? Ist das Netzwerk zuverlässig? Sind alle Knoten im Netzwerk jederzeit verfügbar?

Sehen Sie sich jetzt die unterschiedlichen Verteilungsmethoden an, die diese Faktoren in der Praxis berücksichtigen.

16.3 Verteilungsmethoden

Wenn man Latenz, Transaktionskonsistenz und Siteautonomie berücksichtigt, gibt es etliche unterschiedliche Methoden, um Replikation in SQL Server zu implementieren. In diesem Abschnitt können Sie sich Verteilungsmethoden näher ansehen und bestimmen, welche Methodik auf ein gegebenes Szenario am besten anzuwenden ist.

Drei Grundtypen der Replikation lassen sich in unterschiedlichem Ausmaß kombinieren und ergeben insgesamt sieben verschiedene Methoden. Die Replikationstypen sind Transaktionsreplikation, Snapshotreplikation und Mergereplikation. Wenn Sie Latenz, Konsistenz und Eigenständigkeit hinzunehmen, erhalten Sie die folgenden Methoden der Datenverteilung:

- Mergereplikation
- Snapshotreplikation
- Transaktionsreplikation
- Aktualisierbare Abonnements
- Snapshotreplikation mit aktualisierenden Abonnenten
- Transaktionsreplikation mit aktualisierenden Abonnenten
- Verteilte Transaktionen

Mergereplikation

Mergereplikation hat den höchsten Wert der Siteautonomie. Sie kann auch die größte Latenz bieten, hat jedoch die niedrigste Transaktionskonsistenz.

Mergereplikation erlaubt jedem Standort, Veränderungen an der lokalen Kopie der replizierten Daten vorzunehmen. Zu einem bestimmten Zeitpunkt werden die Änderungen von der Site zur publizierenden Datenbank geschickt, wo sie mit Veränderungen aus anderen Sites verschmolzen werden. Früher oder später erhalten alle Sites die Aktualisierungen von allen anderen Sites. Dies ist als *Datenkonvergenz* bekannt. Die Änderungen aller Sites konvergieren – und früher oder später haben alle Stellen dieselben Informationen (siehe dazu die Abbildungen 16.1 und 16.2).

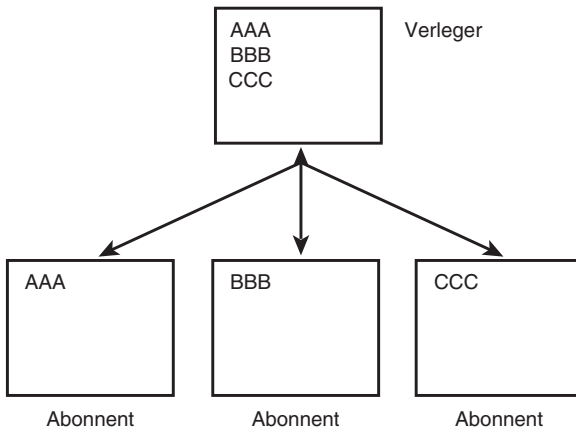


Abbildung 16.1:
Geänderte Datensätze konvergieren
beim publizierenden Server.

Transaktionskonsistenz geht hier über Bord, weil unterschiedliche Sites Daten zu unterschiedlichen Zeiten aktualisieren können. Eine spezielle Site wartet nicht darauf, dass ihre Aktualisierungen zu jeder anderen Site geschickt werden, bevor sie ihre Arbeit fortsetzt. Mit anderen Worten wird garantiert, dass jede Site zu den gleichen Ergebnismengen konvergiert, jedoch nicht unbedingt gleichzeitig.

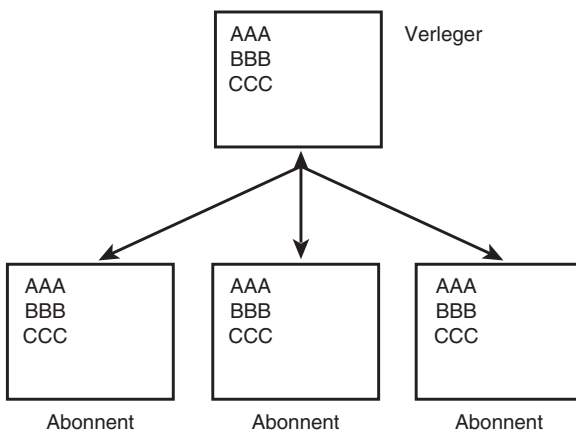


Abbildung 16.2:
Konvergierte Datensätze werden an
alle Abonnenten zurückgeschickt.

Mergereplikation kann manchmal zu Konflikten führen. Beispielsweise nimmt Site A einige Änderungen an Datensatznummer 27 vor und Site B nimmt ebenfalls einige Änderungen an Datensatznummer 27 vor. Beide schicken ihre Versionen der Daten zurück an den publizierenden Server. Der publizierende Server sieht nun, dass beide Sites Änderungen an Datensatznummer 27 vorgenommen haben. Welche Version von Datensatz 27 ist die richtige? Ist es die Version, die zuerst bearbeitet wurde? Ist es die Version, die zuletzt bearbeitet wurde? Wie ist es mit anderen Abhängigkeiten? Wenn beispielsweise Änderun-

gen in Datensatz 27 auf Daten basieren, die sich in Datensatz 26 befinden, und Datensatz 26 auf beiden Rechnern verschieden ist, welche ist richtig? Die Bestimmung der richtigen Antwort kann eine komplexe Aufgabe sein. Sie werden dies etwas später in diesem Kapitel näher betrachten.

Wer sollte Mergereplikation verwenden? Gute Frage. Wegen der potentiellen Konflikte ist Mergereplikation besser geeignet für Umgebungen, in denen die Wahrscheinlichkeiten für solche Konflikte minimiert sind. Gute Kandidaten für Mergereplikation sind beispielsweise Sites, die vorwiegend Änderungen nur an ihren eigenen Datensätzen vornehmen (angezeigt durch eine Standort-ID in jedem Datensatz), jedoch Informationen von allen anderen Standorten brauchen.

Sie können beispielsweise eine Datenbank erstellen, die die Kriminalgeschichte von Individuen verfolgt. Ein großer Staat wie Alaska, in dem jede Kleinstadt möglicherweise eine Kopie dieser Kriminalgeschichte haben möchte, aber sich nicht den ständigen Kontakt zu einer zentralen Datenbank leisten kann, könnte ein hervorragender Ort sein, um Mergereplikation zu implementieren. Jede Stadt könnte autonom sein und die Latenz dürfte sehr hoch sein. Die örtliche Polizei oder der Sheriff dürften der Datenbank neue Kriminalinformationen hinzufügen und dann an die Hauptquartiere zurückschicken, damit sie mit Daten aus vielen anderen Städten verschmolzen werden können. Es könnte noch Konflikte geben, wenn sich ein Krimineller von Stadt zu Stadt bewegt und Probleme macht; diese Konflikte können jedoch aufgespürt, die entsprechenden Datensätze aktualisiert – sprich konvergiert – und zu allen Kleinstädten zurückgeschickt werden.

Snapshotreplikation

Bei der Snapshotreplikation wird eine vollständige Kopie der zu replizierenden Elemente vom publizierenden Server auf die abonnierende Datenbank kopiert (siehe Abbildung 16.3). Dieser Replikationstyp ist am leichtesten einzurichten und zu pflegen. Snapshotreplikation hat ein hohes Niveau der Standortautonomie. Transaktionskonsistenz ist auch garantiert, weil alle Transaktionen nur auf den Publikationsserver angewendet werden. Die Siteautonomie kann sehr brauchbar für Standorte sein, die Nur-Lese-Versionen der Daten brauchen und eine größere Latenz tolerieren. Wenn Sie Snapshotreplikation verwenden, sollte die Abonentendatenbank die replizierten Daten als Nur-Lese-Daten ansehen. Das liegt daran, dass keine Änderungen, die an den Daten vorgenommen werden, zurück an die Publikationsdatenbank geschickt werden. Zusätzlich werden alle Änderungen an den Daten zunichte gemacht, wenn der nächste Snapshot heruntergeladen wird.

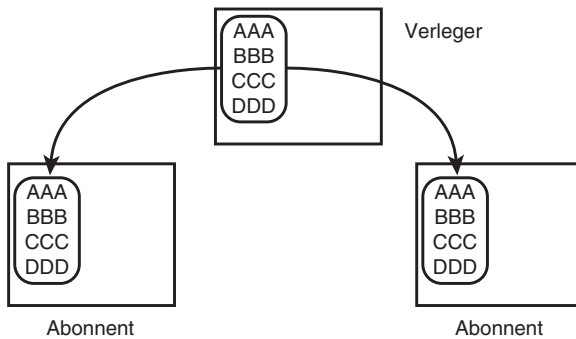


Abbildung 16.3:
Snapshots der gesamten Datenmenge werden an jeden Abonnenten geschickt. Änderungen an den Daten können nur beim publizierenden Server durchgeführt werden.

OLAP-Server sind hervorragende Kandidaten für Snapshotreplikation. Die Ad-hoc-Abfragen, die Managementinformationssystemadministratoren (MIS-Administratoren) üblicherweise auf Daten anwenden, werden allgemein nur gelesen, und Daten, die mehrere Stunden oder sogar mehrere Tage alt sind, beeinträchtigen die Abfragen nicht. Beispielsweise möchte vielleicht eine MIS-Abteilung einer Firma Erkundigungen über die demographischen Faktoren von Waren einholen, die vor zwei Monaten verkauft wurden. Die Informationen von letzter Woche oder sogar von heute fallen bei diesen Abfragen nicht ins Gewicht. Zudem plant die Abteilung keine Veränderungen an den Daten, sie braucht nur das Datendepot. Die Siteautonomie gestattet es der MIS-Abteilung, zusätzliche Indizes auf den Daten zu implementieren, ohne die OLTP-Publikationsdatenbank zu beeinflussen.

Transaktionsreplikation

Bei der Transaktionsreplikation werden die Transaktionen vom Verleger zu den Abonnenten geschickt. Dieser Replikationstyp ist eine Einbahnstraße. Der Abonnent kann Daten nur direkt an der publizierenden Datenbank ändern. Die Änderungen werden dann bei der nächsten Synchronisierung zurück zu den Abonnenten repliziert (siehe Abbildung 16.4).

Dieser Replikationstyp erlaubt einen mittleren Autonomiegrad. Der Abonnent sollte die replizierten Daten als Nur-Lese-Daten behandeln. Das ist wichtig, weil Änderungen an den replizierten Daten die Durchführung der zukünftigen replizierten Transaktionen möglicherweise nicht zulassen. Nehmen wir an, dass Sie das Feld *Land* in der Tabelle *Lieferanten* beim Abonnenten von USA in Korea ändern. Wenn Sie dann eine Transaktion auf dem Verleger durchführen, die alle Lieferanten aus den USA aktualisiert, kommt Ihr neuer Korea-Datensatz nicht in den Genuss der Aktualisierung, die alle anderen Abonnenten erhalten haben. Das führt zu Konsistenzproblemen in Ihrer Datenbank und sollte vermieden werden. Bei diesem Replikationstyp ist die Latenzzeit auf einem mittleren Niveau angesiedelt. Der Abonnent muss nicht immer in Kontakt mit dem Verleger sein, aber regelmäßige Synchronisierungen sind nützlich und die Menge der bewegten Daten ist

relativ klein. Denken Sie daran, dass Snapshotreplikation alle publizierten Daten vom Verleger zum Abonnenten bewegen muss (ob sie bearbeitet worden sind oder nicht). Bei der Transaktionsreplikation werden nur die durchgeführten Transaktionen zu den Abonnenten geschickt. Dieser Replikationstyp wurde hauptsächlich in SQL Server 6.5 eingesetzt, bei dem es nicht die Möglichkeit gab, irgendeinen Typ von Mergereplikation zu verwenden.

Transaktionsreplikation bietet sich vor allem für Szenarios an, in denen die Abonnenten ihre Daten als Nur-Lese-Daten behandeln können, aber Änderungen an den Daten mit einer minimalen Latenz brauchen. Ein hervorragendes Beispiel für Transaktionsreplikation findet sich in Warenwirtschaftssystemen. In diesem Szenario haben Sie möglicherweise etliche unterschiedliche publizierende Stellen, die Warenbestellungen entgegennehmen. Diese Bestellungen werden dann zu einem zentralen Verteilungsdepot repliziert, wo Entnahmebelege erstellt und die Bestellungen ausgeführt und versandt werden. Das Depot kann die Daten als Nur-Lese-Daten behandeln und braucht rechtzeitig neue Informationen.

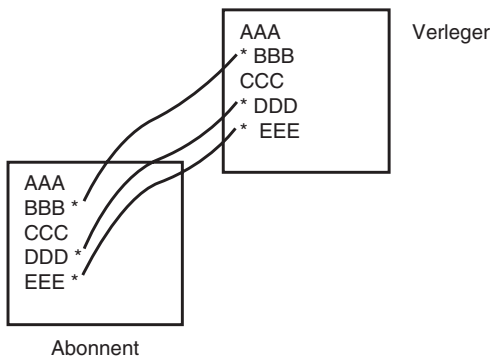


Abbildung 16.4:
Bei der Transaktionsreplikation können Änderungen nur beim Verleger durchgeführt werden. Die Änderungen (Transaktionen) werden dann von jedem Abonnenten übernommen.

Aktualisierbare Abonnements

Obwohl die replizierten Daten im Allgemeinen als schreibgeschützt behandelt werden, hat SQL Server 7.0 die Möglichkeit geboten, Abonnenten die Aktualisierung der Daten beim Verleger zu erlauben. Diese Optionen haben die Fähigkeit der Abonnenten erweitert, mit ihren Daten autonom zu arbeiten. Die Aktualisierung der Abonnenten hat in SQL Server 7.0 das Zweiphasencommit-Protokoll (2PC) verwendet, um die Transaktions- und Datenkonsistenz zu verwalten. Diese Lösung ist dadurch beschränkt, dass eine beständige Netzwerkverbindung zwischen Verleger und Abonnent existieren muss. SQL Server 2000 begegnet diesen Einschränkungen mit der neuen Option *Aktualisieren über eine Warteschlange*.

Für aktualisierbare Abonnements sind jetzt drei Optionen verfügbar, je nachdem, ob Sie Snapshotreplikation oder Transaktionsreplikation konfigurieren:

- Sofortiges Aktualisieren
- Verzögertes Aktualisieren über eine Warteschlange
- Sofortiges Aktualisieren mit verzögertem Aktualisieren über eine Warteschlange als Failover

Sofortiges Aktualisieren

Das sofortige Aktualisieren von Abonnenten verwendet das Zweiphasencommit-Protokoll (2PC), um simultane Änderungen am Verleger und Abonnenten durchzuführen, wie es in SQL Server 7.0 geschehen ist. Diese Option erfordert, dass Verleger und Abonnent verfügbar und verbunden sind. Bei Bestätigung einer Änderung wird die Aktualisierung an die zusätzlichen Abonnenten über Snapshot- oder Transaktionsreplikation weitergegeben.

Aktualisieren über eine Warteschlange

Diese Option ist neu in SQL Server 2000 und nutzt die Vorteile der Mechanismen zum Speichern und Weiterleiten, wie sie in Nachrichtenwarteschlangen realisiert sind. In diesem Fall werden die Modifikationen an eine Warteschlange gesendet, wo sie bei einer verfügbaren Netzwerkverbindung angewandt werden können. Die asynchrone Anwendung von Aktualisierungen auf den Verleger kann zu Konflikten führen, wenn ein anderer Abonnent seine Aktualisierungen bereits angewandt hat. Um diese Situation in den Griff zu bekommen, müssen Sie Konfliktauflösungsrichtlinien für den Warteschlangenlese-Agent Ihrer Publikation erstellen.

Die Konfliktauflösung für Aktualisieren über eine Warteschlange wird anders als bei der Mergereplikation behandelt. Die Mergereplikation löst Konflikte auf der Zeilenebene basierend auf einer Richtlinie, die Sie für die Publikation konfigurieren. Dagegen löst das Aktualisieren über eine Warteschlange Konflikte auf der Transaktionsebene, um die Unteilbarkeit der Transaktionen zu gewährleisten. Der Konfliktlöser hat die Wahl unter drei Möglichkeiten, wenn ein Konflikt erkannt wird:

- Verleger gewinnt und das Abonnent wird erneut initialisiert.
- Verleger gewinnt.
- Abonnent gewinnt.

Wenn man die Option zur erneuten Initialisierung des Abonnenten konfiguriert, ist der höchste Grad an Transaktionskonsistenz sichergestellt, während die Konfiguration als »Abonnent gewinnt« zum geringsten Grad der Konsistenz führt. Der goldene Mittelweg ist die Option »Verleger gewinnt«.

Die Warteschlange selbst ist als Tabelle (*MSReplication_queue*) in der Abonnentendatenbank realisiert und wird als SQL Server 2000-Warteschlange bezeichnet. Der Vorteil dieses Warteschlangentyps besteht darin, dass er mit allen SQL Server-Plattformen arbeitet, einschließlich der Desktop-Edition für Windows 98. Bei Verwendung einer SQL Server 2000-Warteschlange läuft der Warteschlangenlese-Agent auf dem Abonnenten, der die Warteschlange liest und leert, wenn eine Netzwerkverbindung mit dem Verleger wiederhergestellt wird.

Wenn Verteiler und Abonnenten unter Windows 2000 laufen, können Sie die Abonnenten so konfigurieren, dass sie die von Microsoft Message Queue 2.0 (MSMQ) bereitgestellten Warteschlangenmechanismen nutzen. Mit dieser Konfiguration werden Aktualisierungen als Nachrichten verpackt und mittels 2PC an den Verteiler gesendet. Wenn die Abonnenten offline sind, werden ihre Änderungen in der lokalen Warteschlange zwischengespeichert, bis sie sich an den Verteiler zustellen lassen. Obwohl diese Lösung komplizierter zu implementieren ist, verfügt man über die Vorteile erweiterter Nachrichtenweiterleitung, zentralisierter Warteschlangenverwaltung und besserer Überwachung, was sich mit SQL Server 2000-Warteschlangen nicht erreichen lässt.

Sofortige Aktualisierung mit verzögertem Aktualisieren über eine Warteschlange als Failover

Diese Option ist die Luxusvariante, bei der sich die Weiterleitung der Änderungen an den Verleger und andere Abonnenten auf die sofortige Aktualisierung mittels 2PC stützt. Wenn die Netzwerkverbindung verloren geht, können Sie die verzögerte Bereitstellung über eine Warteschlange aktivieren, so dass die beabsichtigten Aktualisierungen nicht unbedingt verloren gehen. Sobald die Rückkehr zur sofortigen Aktualisierung möglich ist, können Sie die Netzwerkverbindung wiederherstellen und mit der sofortigen Aktualisierung beginnen, nachdem die Warteschlange gelöscht ist. Diese Option ist sehr nützlich, wenn Sie einen Verleger zu Wartungszwecken in den Offline-Status versetzen müssen.

Snapshotreplikation mit aktualisierenden Abonnenten

Snapshotreplikation mit aktualisierenden Abonnenten ist eine interessante Erscheinung. Sie kombiniert die bereits beschriebene Snapshotreplikation mit einer Option der aktualisierbaren Abonnenten.

Bei dieser Methode haben Sie eine gewisse Autonomie, weil die Abonnentendatenbank nicht ständig in Kontakt mit der publizierenden Datenbank stehen muss. Der einzige Zeitpunkt, an dem der Abonnent mit dem Verleger arbeitet, ist dann, wenn ein Snapshot heruntergeladen wird oder der Abonnent eine der Aktualisierungsoptionen verwendet, um

eine Transaktion sowohl auf der lokalen Site (Abonnentensite) als auch der publizierenden Datenbank vorzunehmen. Abbildung 16.5 zeigt ein Beispiel der Snapshotreplikation mit der Option der sofortigen Abonnentenaktualisierung.

Dies hält auch einen hohen Grad an Transaktionskonsistenz aufrecht, weil Transaktionen sowohl auf die Abonentendatenbank, als auch auf die Verlegerdatenbank angewendet werden müssen. Die Tatsache, dass andere Abonnenten Änderungen an der Verlegerdatenbank vorgenommen haben könnten, seit Ihr Abonnent zuletzt einen Snapshot heruntergeladen hat, ist bedeutungslos. Die Konsistenz bleibt durch die Tatsache erhalten, dass die Transaktion auf dem Verlegerserver fehlschlagen wird, wenn die bearbeiteten Daten beim Verleger und beim Abonnenten unterschiedlich sind. Das funktioniert sogar bei der Snapshotreplikation, die mit Latenz behaftet ist.

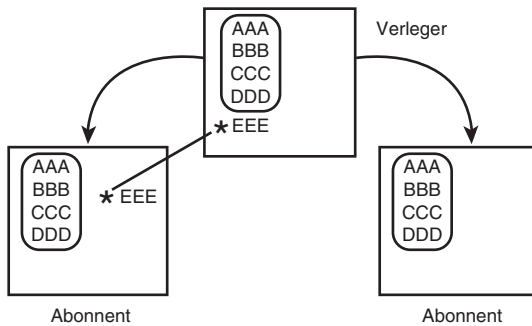


Abbildung 16.5:

Snapshots werden in regelmäßigen Abständen an jeden Abonnenten geschickt. Damit ein Abonnent eine lokale Änderung durchführen kann, müssen die gleichen Änderungen erfolgreich beim publizierenden Server durchgeführt werden.

Der Abonnementsserver kann unmittelbar die Arbeit mit den geänderten Daten aufnehmen, weil er weiß, dass er den Publikationsserver erfolgreich aktualisiert hat. Der Publikationsserver konvergiert die Informationen, und alle an der Replikation teilhabenden Server erhalten rechtzeitig die Veränderungen.

Diese Methode ist nützlich in Szenarios, in denen nur gelegentliche Bearbeitungen an Abonentenservern stattfinden. Je länger die Latenz ist, um so wahrscheinlicher ist es, dass Ihr 2PC fehlschlägt. Das liegt daran, dass es eine höhere Wahrscheinlichkeit dafür gibt, dass die Daten beim Verleger geändert worden sind, aber noch nicht an den Abonnenten repliziert worden sind.

Transaktionsreplikation mit aktualisierenden Abonnenten

Bei Transaktionsreplikation mit aktualisierenden Abonnenten geben Sie sogar noch mehr Autonomie an den Abonnentenstandorten auf, aber Sie minimieren die Latenz. Bei dieser Methode verwenden Sie die weiter vorn beschriebene Transaktionsreplikation.

Bei der Aktualisierung einer Abonnentendatenbank wird die Aktualisierung sowohl beim Abonnenten als auch beim Verleger mithilfe von 2PC durchgeführt. Das heißt, dass die Änderung gleichzeitig sowohl beim Abonnenten als auch beim Verleger geschrieben wird. Deshalb haben Sie garantierte Transaktionskonsistenz. Die Änderung wird mit anderen aktualisierenden Abonnenten konvertiert und dann zurück an alle Abonnentendatenbanken geschickt. Wie der Abschnitt zu den Optionen für aktualisierbare Abonnements erläutert hat, löst das Aktualisieren über eine Warteschlange Konflikte auf der Transaktionsebene.

Transaktionsreplikation mit aktualisierenden Abonnenten hat eine geringere Latenz als die Verwendung von Snapshotreplikation mit aktualisierenden Abonnenten, weil die replizierten Transaktionen viel kleiner (und schneller zu bewegen) sind als das Synchronisieren eines Snapshots der Daten.

Nützliche Szenarios für diesen Replikationstyp schließen Reservierungssysteme mit geringem Volumen ein. Bei diesem Systemtyp kann ein Abonnent sich einen Zeitplan der Verfügbarkeit ansehen und dann versuchen, eine Reservierung vorzunehmen. Nachdem die Reservierung auf den Plan gesetzt wurde, kann sie innerhalb einiger Minuten (bzw. innerhalb einer festgelegten Zeitspanne) an alle anderen Abonnentendatenbanken repliziert werden. Diese aktualisieren alle ihre Zeitpläne. Vielleicht taucht bei Ihnen jetzt folgende Frage auf: »Was passiert, wenn ich eine Reservierung vornehmen will, die jemand anderes bereits gebucht hat, aber die Buchung noch nicht an diesen Abonnenten repliziert worden ist?« In diesem Fall wäre es besser, sofortiges Aktualisieren zu verwenden, weil dieser Replikationstyp mit 2PC arbeitet und Sie den Stand Ihrer Reservierung erfahren: Wird sie erfolgreich bestätigt, ist sie verfügbar, und Sie überschreiben die Reservierung einer anderen Person nicht.

Verteilte Transaktionen

Die in SQL Server 6.5 und SQL Server 7.0 verfügbaren verteilten Transaktionen stehen weiterhin in SQL Server 2000 zur Verfügung. Verteilte Transaktionen – oder 2PC – haben fast keine Autonomie, keine Latenz und garantierte Transaktionskonsistenz.

Wie Sie bereits gelernt haben, werden Änderungen entweder alle gleichzeitig oder gar nicht repliziert. Dieser Replikationstyp nutzt MS DTC (Microsoft Distributed Transaction Coordinator), der das Zweiphasencommit-Protokoll implementiert und Transaktionen auf alle Sites gleichzeitig anwendet.

Weil sich jeder Abonnent jederzeit in Kontakt mit dem Verleger und allen anderen Abonnenten befinden muss, lässt dies wenig Autonomie und keine Latenz zu. Dieser Replikationstyp ist nützlich in Szenarios, in denen jeder Echtzeitversionen der Daten haben muss.

Reservierungssysteme mit hohem Volumen können diesen Replikationstyp vorteilhaft nutzen. Das trifft zum Beispiel auf ein Fluglinienreservierungssystem zu. Abonnenten sind ausgelagert und vermindern einen Teil der Arbeitslast auf dem Publikationsserver. Die garantierte Transaktionskonsistenz zeigt sich, wenn eine Person in New York versucht, Platz 2A auf Flug 2400 zu bekommen, und eine andere Person in Idaho versucht, denselben Platz für denselben Flug zu buchen. Nur eine von ihnen wird den Platz buchen können, und die andere Person, die die Buchung nicht erhält, erfährt das unmittelbar.

16.4 Das Verleger/Abonnent-Modell

Wie bereits in der Einführung erwähnt, verwendet die Replikation in SQL Server das Modell »Verleger / Abonnent«. Wie bei einem echten Verleger spielen in diesem Modell mehrere Aspekte eine Rolle:

- Der *Verleger* ist der Besitzer der Quelldatenbankinformationen. Er stellt Daten für die Replikation zur Verfügung und verschickt Änderungen der publizierten Daten an den Verteiler.
- Die *Abonentendatenbank* erhält Kopien der Daten (Snapshotreplikation) oder der Transaktionen, die in der Verteilungsdatenbank gespeichert sind.
- Der *Verteiler* empfängt alle Änderungen, die an den publizierten Daten vorgenommen wurden. Er speichert dann die Daten und leitet sie zum geeigneten Zeitpunkt an die Abonnenten weiter. Ein einziger Verteilungsserver kann mehrere Verleger und mehrere Abonnenten gleichzeitig unterstützen.

Es gibt einen oder mehrere Verleger und Abonnenten sowie einen Verteiler. Ein SQL Server im Unternehmen kann eine beliebige oder alle diese Rollen gleichzeitig übernehmen. Die Art und Weise, wie Sie diese Rollen implementieren, bringt Sie zu den verschiedenen Replikationsmodellen, auf die diese Lektion in Kürze eingeht.

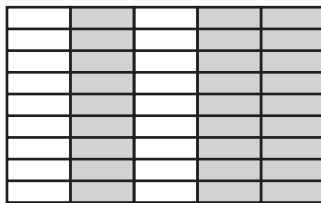
Um mit dem Publizieren/Abonnieren-Modell fortzusetzen: Eine publizierende Datenbank erstellt Artikel. Die Artikel werden dann gebündelt, um eine Publikation zu erstellen.

Artikel

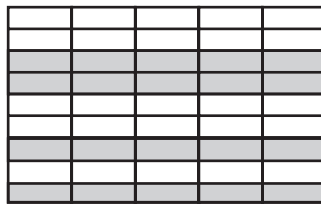


Ein *Artikel* ist eine Einzeltabelle oder eine Untermenge von Daten in einer Tabelle. Artikel werden zu Publikationen gebündelt. Zum Publizieren einer Untermenge der Daten in einem Artikel müssen Sie einen bestimmten Filter verwenden, um die Daten in Ihrer Tabelle zu partitionieren. Sie können vertikale Partitionen, horizontale Partitionen oder beide verwenden (siehe Abbildung

16.6). Eine vertikale Partition wählt nur bestimmte Spalten, eine horizontale Partition nur bestimmte Zeilen aus der Tabelle aus. Artikel können Sie sowohl mit einzelnen SQL-Anweisungen als auch mit gespeicherten Prozeduren erstellen.



vertikale
Unterteilung



horizontale
Unterteilung

Abbildung 16.6:
Datenpartitionierung

Tabellen können Sie horizontal und/oder vertikal partitionieren. Jede dieser partitionierten Tabellen wird zu einem einzigen Artikel, der dann abonniert werden kann. Manchmal ist es sinnvoll, die physische Implementierung der Tabellen zu ändern. Beispielsweise könnten Sie eine Tabelle *sales_info* (Verkaufsinformationen) haben, die drei unterschiedliche Regionen bedient. Wenn Sie Ihre Datenbank physisch neu entwerfen, können Sie drei getrennte Tabellen *sales_info* implementieren (eine für jede Region). Als Erweiterung zur Mergereplikation ist es sogar möglich, die Daten je nach Abonnent dynamisch zu filtern. Dynamische Filter sind Zeilenfilter, die entweder als SQL Server-Funktionen oder benutzerdefinierte Funktionen implementiert sind. Wenn Sie Daten basierend auf den Daten in anderen Tabellen filtern wollen, können Sie auch einen Verknüpfungsfiler implementieren.

SQL Server 2000 erlaubt es auch, Datenbankobjekte wie gespeicherte Prozeduren, Sichten, indizierte Sichten und benutzerdefinierte Funktionen zu publizieren. In der Tat können Sie sogar Schemaobjekte wie Indizes (gruppiert und nicht gruppiert), Schlüssel einschränkungen, Benutzertrigger und Sortierungen replizieren.

Überlegungen zur Datendefinition

Wenn Sie Ihre Artikel erstellen, sind einige Regeln zu beachten:

- **IDENTITY**-Spalten werden repliziert, die Eigenschaft aber nicht. Wenn eine anfängliche Synchronisierung stattfindet, werden die Werte der **IDENTITY**-Spalten als Ganzzahlwerte kopiert.

- Zeitstempeldatentypen (im Unterschied zu Datum/Zeit-Datentypen) spiegeln die Reihenfolge der SQL Server-Aktivitäten für eine Datenzeile wider und geben damit die Herkunft der Änderungen für diese Zeile an. Diese Werte werden als binäre Datentypen repliziert.
- Uniqueidentifizier erstellt GUIDs (global eindeutige Bezeichner). Diese sind 128-Bit-Bezeichner und werden mit Mergereplikation verwendet. Sie ermöglichen die eindeutige Identifizierung jeder Zeile, selbst wenn Sie Zeilen aus zwei getrennten Tabellen konvergieren. Sie können die Funktion NEWID verwenden, um eine GUID zu erstellen. Der Datentyp wird repliziert, aber die Funktion nicht.
- Benutzerdefinierte Datentypen können nur repliziert werden, wenn es sie auch in der Abonnentendatenbank gibt.

Arbeiten mit der Eigenschaft IDENTITY

Es gibt einige spezielle Überlegungen, wenn Sie mit der Eigenschaft IDENTITY arbeiten. Wie bereits erwähnt, werden automatisch inkrementierte Spalten bei den Abonnenten in Binärzahlen geändert (die keine automatischen Werte generieren). Diese Änderung findet statt, damit die Informationen von Abonnent und Verleger übereinstimmen. Wenn Sie Ihre Tabellen auf der Abonnentensite partitionieren wollen, sollten Sie eine neue IDENTITY-Spalte hinzufügen. Dabei müssen Sie Startwerte und CHECK-Einschränkungen verwenden, um Konflikte mit eingehenden Transaktionen zu vermeiden. Sie können auch den Datentyp uniqueidentifizier mit der Funktion NEWID verwenden, um eindeutige Schlüssel zu garantieren.

Bei Szenarios, in denen Sie mehrere Verleger und Abonnenten derselben Informationen haben, möchten sie möglicherweise Abonnenten erlauben, nur Daten aus ihrem lokalen Bereich zu ändern, aber dennoch Daten aus anderen Regionen zu sehen. Das erreichen Sie mit der Anweisung NOT FOR REPLICATION, wenn Sie eine IDENTITY-Eigenschaft, eine CHECK-Einschränkung oder einen Trigger neu erstellen.

Die Anweisung NOT FOR REPLICATION können Sie verwenden, wenn Sie beim Abonnenten Daten hinzufügen. Die CHECK-Einschränkung oder der Trigger kann ausgelöst werden und sicherstellen, dass die Daten, die Sie beim Abonnenten bearbeiten, nur diesem speziellen Abonnenten gehören. Auf diese Weise können Sie die referenzielle Integrität aufrechterhalten, weil ein Abonnent einen lokalen Datensatz mit einem gültigen IDENTITY-Spaltenwert hinzufügen kann. Dieser IDENTITY-Spaltenwert wird dann als Binärzahl reproduziert.

Publikationen

Publikationen bestehen aus einem oder mehreren Artikeln und bilden die Basis für ein Abonnement. Wenn Sie ein Abonnement erstellen, abonnieren Sie eine ganze Publikation. Wenn Sie eine Publikation abonniert haben, können Sie danach einzelne Artikel »lesen«. Mit Publikationen können Sie die referenzielle Integrität durchsetzen, weil alle Artikel in einer einzigen Publikation zum gleichen Zeitpunkt synchronisiert werden. Sie können beliebig viele Publikationen pro Datenbank erstellen.

Wenn Sie mit Publikationen und Abonnements arbeiten, können Sie entweder Push- oder Pullabonnements einrichten. Beide Methoden der Datenübertragung unterscheiden sich vor allem darin, wo die Verwaltung des Abonnements stattfindet.



Sie können auch auf Nicht-SQL Server publizieren und sogar die Replikation für Artikel implementieren, die eine andere Sortierreihenfolge und andere Datentypen haben; allerdings ist das nicht zu empfehlen.

Darüber hinaus sind folgende Publizierungsbeschränkungen zu beachten:

- Tabellen müssen einen Primärschlüssel haben, um die Integrität sicherzustellen (die Ausnahme besteht in der Verwendung von Snapshotreplikation.)
- Die Datenbanken *master*, *model*, *msdb*, *tempdb* und beliebige Verteilungsdatenbanken können Sie nicht replizieren.
- Publikationen dürfen nicht mehrere Datenbanken überspannen. Jede Publikation kann nur Artikel aus einer Datenbank enthalten.
- image-, text- und ntext-Daten erhalten begrenzte Unterstützung. Transaktions- und Mergereplikation können Veränderungen in diesen Werten nicht entdecken, weil sie getrennt von den Tabellen gespeichert und Änderungen nicht protokolliert werden. Die Snapshotreplikation eignet sich am besten für derartige Datentypen. Für die Snapshots Ihrer Tabellen (einschließlich dieser Felder) können Sie einen Zeitplan erstellen, damit sie regelmäßig stattfinden. Dieser Vorgang heißt auch *geplante Tabellenaktualisierung*.

Pushabonnements

Wenn Sie ein Abonnement gleichzeitig mit der Erstellung Ihrer Publikationen einrichten, erstellen Sie praktisch ein Pushabonnement. Dieser Abonnementtyp unterstützt die Zentralisierung der Abonnementverwaltung, weil das Abonnement beim Verleger zusammen mit dem Synchronisierungszeitplan des Abonnenten definiert ist. Der Verleger übernimmt die gesamte Verwaltung der Abonnements. Die Daten werden zum Abonnenten »geschoben« (englisch: push), wenn der Verleger beschließt, sie zu senden.



Wenn Sie mit Pushabonnements arbeiten, können Sie mehrere Abonnenten gleichzeitig einrichten.

Pushabonnements sind am nützlichsten, wenn Ihre Abonnenten die Zusendung von Aktualisierungen brauchen, sobald sie erfolgt sind. Pushabonnements erlauben auch eine höhere Sicherheitsebene, da der Verleger bestimmt, wer wann abonnieren darf. Pushabonnements erfordern eine zusätzliche Mehrarbeit an der Verteilerdatenbank, weil sie die Replikation verwaltet.

Pullabonnements

Ein Pullabonnement wird von jedem einzelnen Abonnenten eingerichtet. Der Abonnent leitet die Informationsübertragung zeitabhängig ein. Dies ist nützlich für Anwendungen, die eine niedrigere Sicherheitsebene zulassen können. Der Verleger kann bestimmten Abonnenten erlauben, Informationen abzuholen, oder er kann anonyme Abonnements zulassen. Pullabonnements sind auch nützlich in Situationen, in denen es eine große Anzahl an Abonnenten geben kann. Lösungen auf Internetbasis sind gute Kandidaten für Pullabonnements.

Anonyme Abonnements

Ein anonymes Abonnement ist ein Pullabonnementstyp, für den keine detaillierten Informationen über das Abonnement und den Abonnenten beim Verleger gespeichert werden. Die Abonnenten müssen selbst dafür sorgen, dass ihre Abonnements synchronisiert bleiben. Bei anonymen Abonnements muss der Abonnent dem Verleger nicht namentlich bekannt gemacht werden. Trotzdem gelten alle Regeln für Pullabonnements. Dieser Abonnementstyp eignet sich, wenn Sie den zusätzlichen Verwaltungsaufwand für Abonnenteninformationen beim Verleger und beim Verteiler vermeiden möchten. Anwendungen mit einer großen Zahl von Abonnenten und Internet-Abonnenten eignen sich am besten für diese Abonnementoption.

16.5 Wie arbeitet Replikation?

Auf den ersten Blick scheint die Replikation sehr kompliziert zu sein, wenn Sie sie jedoch in ihre Komponenten zerlegen, ist es halb so schlimm. Im Wesentlichen wird die Replikation von fünf Agents mit jeweils genau umrissenem Aufgabenbereich gehandhabt. Wenn Sie alle Agents zusammenfügen, erhalten Sie als Ergebnis die Replikation:

- Ein *Verteilungs-Agent* verschiebt Informationen von der Verteilungsdatenbank zu den Abonnenten.
- Ein *Protokolllese-Agent* überwacht das Transaktionsprotokoll aller publizierter Datenbanken, die es zur Replikation verwenden. Wenn er Transaktionen findet, die Teil einer Publikation sind, kopiert er sie auf die Verteilerdatenbank, wo sie dann durch den Verteilungs-Agent auf die Abonnenten angewendet werden können.
- Ein *Merge-Agent* mischt die Modifikationen von mehreren Sites.
- Ein *Snapshot-Agent* verschiebt einen Snapshot der Daten, bevor die Replikation beginnen kann. Dieser Datentransfer ist notwendig. Wenn kein Snapshot der Daten beim Abonnenten existiert, können Sie auf den Abonnenten keine Transaktionen anwenden. Er wird auch für die verschiedenen Typen der Snapshotreplikation eingesetzt.
- Ein *Warteschlangenlese-Agent* liest Nachrichten von der SQL Server-Warteschlange auf jedem Abonnenten und wendet die Transaktionen auf die Publikation an. Wenn Sie mit MSMQ 2.0 arbeiten, liest der Warteschlangenlese-Agent die Nachrichten, die in der zentral auf dem Verteiler angelegten Warteschlange gespeichert sind.

Sehen Sie sich jetzt näher an, wie sich jeder Agent in seiner Aufgabe unterscheidet, wenn er mit den weiter oben erläuterten Replikationsmethoden konfrontiert wird. Denken Sie daran, dass eine Replikationsmethode auf der individuellen Publikationsebene angewendet wird und dass jeder Abonnent die Vorteile unterschiedlicher Methoden nutzen kann, die auf verschiedene Publikationen angewendet worden sind.

Wie arbeitet Mergereplikation?

Bei der Mergereplikation kann der Merge-Agent auf dem Verteilungsserver oder auf jedem Abonnentenserver angesiedelt sein. In einem Pushszenario befindet sich der Merge-Agent auf dem Verteilungsserver. In Pullszenarios befindet sich der Agent auf jedem Abonnenten. Abbildung 16.7 gibt einen Überblick über den in den folgenden Schritten dargestellten Replikationsprozess:

1. Der Snapshot-Agent (der sich auf dem Verteilungsserver befindet) nimmt einen Anfangssnapshot der Daten und verschiebt sie zu den Abonnenten. Denken Sie daran, dass die Abonnenten zuerst mit den Verlegern synchronisiert werden müssen, damit die Replikation beginnen kann (mit Ausnahme der Snapshotreplikation).
2. Ein Verteilungsarbeitsordner wird auf dem Verteilungsserver erstellt, um Mergevorgänge zu behandeln.
3. Die Replikation beginnt.
4. Der Merge-Agent übernimmt Modifikationen von den Verlegern und wendet sie auf die Abonnenten an.

5. Der Merge-Agent übernimmt Modifikationen von den Abonnenten und wendet sie auf die Verleger an.
6. Der Merge-Agent empfängt alle Aktualisierungskonflikte und unternimmt die geeignete Aktion.

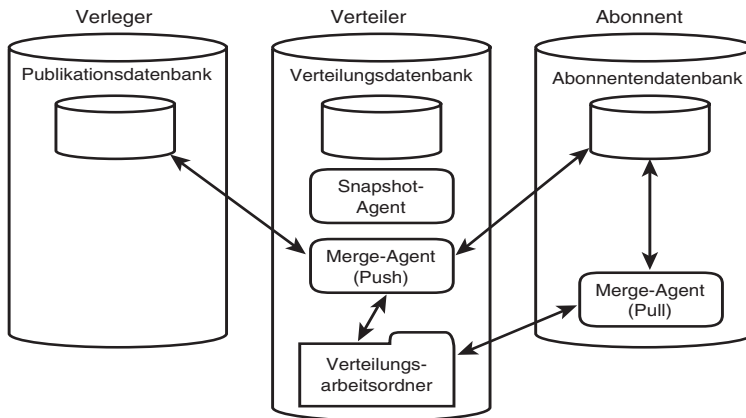


Abbildung 16.7:
Der Vorgang der
Mergereplikation



Mergereplikation wird am besten dann eingesetzt, wenn es wenig wahrscheinlich ist, dass Veränderungen an denselben Datensätzen durch verschiedene Standorte vorgenommen werden.

Horizontal segmentierte Tabellen sind bei Mergereplikation nützlich. Basierend auf einem Regionalcode oder einem anderen Unterscheidungskriterium können Sie getrennte Publikationen erstellen und abonnieren lassen.

Damit Mergereplikation richtig arbeitet, werden einige Änderungen am Tabellenschema sowie an der Verteilungsdatenbank vorgenommen, damit sich SQL Server um die Konfliktlösung kümmern kann. Denken Sie an die folgenden Schemaänderungen, wenn Sie sich zum Einsatz der Mergereplikation entschließen:

- Dem Verteilungsarbeitsordner werden Systemtabellen hinzugefügt; sie zeichnen Änderungen auf, die während der Synchronisation sowie zur Konfliktlösung verwendet werden.
- SQL Server erstellt Trigger sowohl auf den publizierenden Servern als auch auf den Abonnentenservern, die an der Mergereplikation teilhaben. Diese Trigger werden ausgelöst, wenn eine Datenbearbeitung in einer der Tabellen stattfindet, die an der Replikation beteiligt ist. Die Informationen über die Veränderung werden in den Systemtabellen gespeichert, die dem Verteilungsarbeitsordner hinzugefügt wurden. Diese

gespeicherten Veränderungen ermöglichen Ihnen die Rückverfolgung von Veränderungen bis auf jede Zeile oder Spalte der bearbeiteten Informationen.

- SQL Server erstellt eine neue *uniqueidentifier*-Spalte für jede Zeile in replizierten Tabellen. Eine GUID oder ROWGUID wird dann hinzugefügt, um diese Zeile eindeutig zu kennzeichnen. Auf diese Weise können die verschiedenen Aktualisierungen unterschieden werden, wenn ein Datensatz an unterschiedlichen Quellen aktualisiert wird.



Denken Sie daran, dass Sie Information nicht vertikal partitionieren können, wenn sie Mergereplikation einsetzen.

Wenn Sie Aktualisierungen an den gleichen Daten an verschiedenen Standorten zulassen, riskieren Sie Konflikte. Um diese Konflikte zu lösen, geht SQL Server wie folgt vor:



Die Systemtabellen, die im Verteilungsarbeitsordner gespeichert sind, werden verwendet, um jede Änderung an einer Zeile zurückzuverfolgen. Die Veränderungen an jeder Zeile sind aufgelistet und jeder Eintrag hat eine ROWGUID. Diese Geschichte der Bearbeitungen eines Datensatzes wird als *Datensatzherkunft* bezeichnet – eine Geschichte der Änderungen an einer Zeile, die an der Mergereplikation beteiligt ist. Änderungen werden durch eine ROWGUID identifiziert, die der Zeile zugeordnet und im Verteilungsarbeitsordner auf dem Verteilungsserver gespeichert wird.

Anhand der Herkunft kann der Merge-Agent sowohl die aktuellen als auch die ankommenden Daten bewerten und automatisch Konflikte basierend auf Prioritäten lösen, die Sie zugewiesen haben. Sie können diese Prioritäten, die als Trigger gespeichert sind, anpassen, um Ihre eigenen Konfliktauflösungsverfahren zu erstellen. Mehr darüber lernen Sie in der morgigen Lektion.



Denken Sie daran, dass SQL Server 2000 die Definition mehrerer Trigger in einer einzigen Tabelle zulässt. Ihr Konfliktlösungsschema kann sehr komplex sein.

Es empfiehlt sich, dass Sie einen Datensatz als aktuellen Datensatz nehmen und dann beide Datensätze zur manuellen Behandlung in einer eigenen Tabelle speichern. Auf diese Weise können Sie die Qualität Ihres Konfliktlösungsprozesses testen.

Nachdem Sie einen Datensatz herausgegriffen haben, wird das Synchronisationsverfahren fortgesetzt und der gesamte Zyklus wiederholt sich.

Interna der Snapshotreplikation

Wie Sie wissen, kopiert die Snapshotreplikation den gesamten Artikel oder die gesamte Publikation en gros von den Verlegern auf die Abonnenten. Dies schließt Snapshotreplikation mit aktualisierenden Abonnenten ein. Die Aktualisierungen werden sowohl am Abonnenten als auch am Verleger vorgenommen, wenn jedoch eine Synchronisation stattfindet, werden die Daten des Abonnenten vollständig durch den hereinkommenden replizierten Artikel überschrieben.

Bei der Snapshotreplikation gibt es keinen Merge-Agent; allerdings kommt der Verteilungs-Agent zum Einsatz. Wenn Sie ein Pullabonnement verwenden, befindet sich der Verteilungs-Agent auf dem Abonnementserver. Wenn Sie ein Pushabonnement verwenden, ist der Verteilungs-Agent auf dem Verteilungsserver angeordnet. Die Verwendung eines Push- oder Pullabonnements in diesem Szenario hängt von vielen Faktoren ab, nicht zuletzt davon, wie beschäftigt Ihr Verteilungsserver ist und wie sie die Abonnenten verwalten. Wollen Sie Abonnements zentral verwalten (Pushabonnement) oder wollen Sie Abonnements an jedem Abonnenten verwalten (Pullabonnements)?

Abbildung 16.8 gibt einen Überblick über den Vorgang der Snapshotreplikation entsprechend der folgenden Schritte:

1. Der Snapshot-Agent liest den publizierten Artikel und erstellt das Tabellenschema und die Daten im Verteilungsarbeitsordner.
2. Der Verteilungs-Agent liest dieses Schema und baut die Tabellen auf dem Abonnenten neu auf.
3. Der Verteilungs-Agent verschiebt die Daten dann in die neu erstellten Tabellen auf dem Abonnenten.
4. Indizes (falls verwendet) werden dann auf der neu synchronisierten Abonentendatenbank neu erstellt.



Snapshotreplikation findet auf Anforderung statt. Das heißt, dass der Snapshot der Daten nicht in der Verteilungsdatenbank gespeichert ist, wie es mit dem Speichern-und-Weiterleiten-Algorithmus (store-and-forward) passiert, der in der Replikation auf Transaktionsbasis verwendet wird. In der Verteilungsdatenbank werden nur Statusinformationen zu einem Snapshot gespeichert. Der Snapshot-Agent und der Verteilungs-Agent verrichten die gesamte Arbeit in der Zeit, in der die Synchronisation eingeleitet wird.

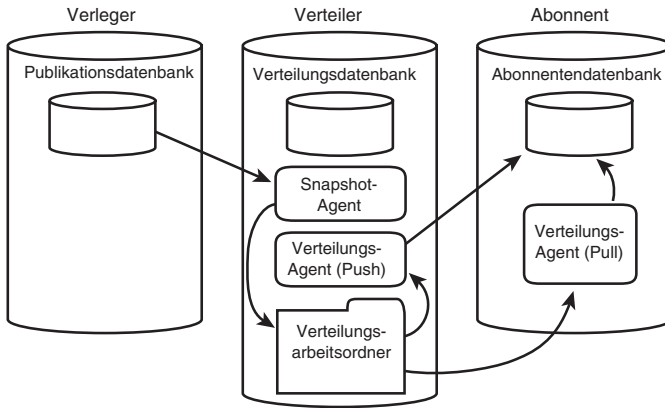


Abbildung 16.8:
Der Vorgang der Snapshotreplikation

Interna der Transaktionsreplikation

Denken Sie daran, dass die Replikation auf Transaktionsbasis nur die Transaktionen auf die Verteilerdatenbank kopiert, die in den publizierten Datenbanken stattgefunden haben. Die Aktualisierungen werden dann – im Allgemeinen sobald sie stattfinden – auf die Abonnentendatenbank angewendet, was die Latenz verringert. Die Abonnentendatenbank sollte man sich als Nur-Lese-Datenbank denken, weil dieser Replikationstyp eine Einbahnstraße ist. Änderungen an den Daten können nur beim Verleger erfolgen.



Obwohl Sie sich die Abonnentendatenbank als Nur-Lese-Daten vorstellen sollten, setzen Sie keinesfalls die Datenbankoption *read-only* (Nur-Lesen) auf wahr (true). Wenn Sie das tun, können Sie keine replizierten Transaktionen auf sie anwenden.

Die Transaktionsreplikation verwendet keinen Merge-Agent, sondern einen Protokolllese-Agent. Nach wie vor ist auch der Snapshot-Agent aktiv. Sie brauchen eine Basis, auf die Sie Ihre Transaktionen anwenden können, und der Snapshot-Agent erledigt dies für Sie. Wie bei der Snapshotreplikation wird der Verteilungs-Agent verwendet. Wenn Sie ein Pullabonnement einsetzen, befindet sich der Verteilungs-Agent auf dem Abonnementserver. Bei einem Pushabonnement residiert der Verteilungs-Agent auf dem Verteilungsserver.

Abbildung 16.9 gibt einen Überblick über den Vorgang der Transaktionsreplikation entsprechend der folgenden Schritte:

1. Der Snapshot-Agent liest den publizierten Artikel und erstellt das Tabellenschema und die Daten im Verteilungsarbeitsordner in der gleichen Weise wie es Abbildung 16.8 gezeigt hat.

2. Der Verteilungs-Agent liest diese Schemas und baut die Tabellen bei den Abonnenten auf.
3. Der Verteilungs-Agent verschiebt dann die Daten zu den neu erstellten Tabellen bei den Abonnenten.
4. In der neu synchronisierten Abonnentendatenbank werden dann die Indizes (falls verwendet) neu erstellt.
5. Die normale Transaktionsreplikation kann beginnen.
6. Der Protokolllese-Agent beobachtet die Transaktionsprotokolle der publizierenden Datenbank. Wenn er eine Transaktion findet, kopiert er sie in die Verteilungsdatenbank. Dort wird die Transaktion gespeichert, bis der nächste Synchronisationsprozess beginnt.
7. Wenn der Synchronisationsprozess aufgerufen wird (entweder durch ein Push seitens des Verteilers oder ein Pull seitens des Abonnenten) wird die Transaktion vom Verteilungs-Agent gelesen und dann auf die Abonnentendatenbank angewendet.

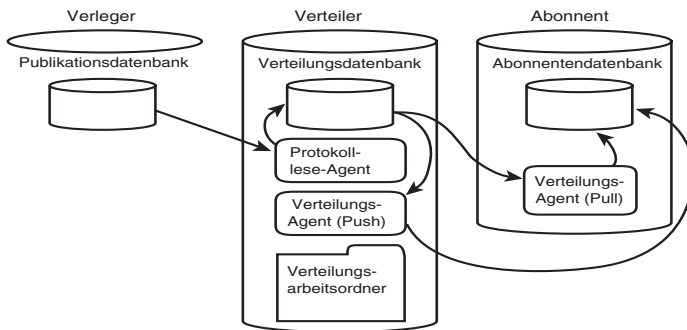


Abbildung 16.9:
Der Vorgang der Transaktionsreplikation



Die Transaktionsreplikation setzt man üblicherweise ein, wenn man eine geringe Latenz anstrebt und sich die Abonnentendatenbank als schreibgeschützt betrachten lässt. Dieses Szenario ist nützlich, wenn Sie Ihre OLTP-Server vom Einfluss der OLAP-Dienste freihalten wollen.

Funktionsweise beim Aktualisieren über eine Warteschlange

Wenn eine Publikation für das Aktualisieren über eine Warteschlange aktiviert ist, werden die beim Abonnenten erfolgten Datenänderungen in der Warteschlange gespeichert, bis sie sich auf den Verleger anwenden lassen. Abbildung 16.10 gibt einen Überblick zum Vorgang beim Aktualisieren über eine Warteschlange entsprechend den folgenden Schritten:

1. Trigger für die Abonententabellen erfassen Aktualisierungen, die beim Abonnenten stattfinden. Diese als Teil des Abonnementvorganges erstellten Trigger speichern die Aktualisierungen in einer Warteschlange, bei der es sich standardmäßig um eine SQL Server-Warteschlange handelt.
2. Bei SQL Server 2000-Warteschlangen werden die Aktualisierungen in einer Tabelle namens `MSReplication_queue` abgelegt, die ebenfalls während des Abonnementvorganges erstellt wird. Bei MSMQ 2.0-Warteschlangen werden die Aktualisierungen in einer zentralen Warteschlange auf dem Verteiler gespeichert. Wird der Abonnent vom Netzwerk getrennt, werden die Aktualisierungen lokal gespeichert und automatisch an die Warteschlange auf dem Verteiler weitergeleitet, nachdem die Netzwerkverbindung wieder hergestellt ist.
3. Der Warteschlangenlese-Agent wendet Warteschlangentransaktionen auf die passende Publikation an. Bei SQL Server-Warteschlangen werden die Aktualisierungen direkt aus der Warteschlange des Abonnenten gelesen, bei MSMQ 2.0-Warteschlangen aus der zentralen Warteschlange, die auf dem Verteiler gespeichert ist.

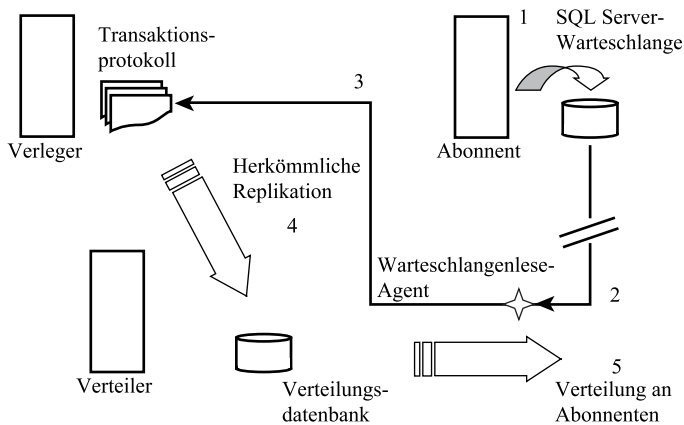


Abbildung 16.10:
Der Vorgang der Warteschlangenaktualisierung

4. Alle Konflikte, die während der Anwendung der Warteschlangentransaktionen erkannt wurden, werden entsprechend der beim Erstellen der Publikation festgelegten Konflik-

trichtlinie aufgelöst. Die konfliktverursachenden Transaktionen werden dann mit dem normalen Transaktionsverteilungsvorgang beim Abonnenten, der die Transaktionen gesendet hat, rückgängig gemacht.

5. Die beim Verleger vorgenommenen Änderungen werden an alle anderen Abonnenten entsprechend der konfigurierten Zeitpläne des Verteilungs-Agents weitergeleitet.

16.6 Replikationsszenarios

Die Replikation können Sie in verschiedenen Szenarios implementieren. Jedes Szenario hat für eine gegebene Geschäftssituation Vorteile.

Zentrale Verleger

Im Szenario des zentralen Verlegers (siehe Abbildung 16.11) können Sie den Einfluss von OLAP-Diensten von der OLTP-Umgebung verringern. In diesem Szenario ist der publizierende Server auch der Verteilungsserver. Beachten Sie, dass der Einfluss auf den Verteilungsserver um so größer sein wird, je mehr Abonnenten Sie haben. Dieser Einfluss lässt sich etwas verringern, indem Sie ein Pullabonnement verwenden, bei dem sich der Verteilungs-Agent auf jedem Abonnenten befindet.

Obwohl sich die Verteilungsdatenbank auf dem Publikationsserver befindet, kann sie mehrere Publikationsserver und mehrere Abonnementserver unterstützen.

Zentraler Verleger mit Remoteverteiler

Im Szenario des zentralen Verlegers mit Remoteverteiler können Sie den Einfluss der Verteilungsdatenbank auf Ihre OLTP-Umgebung mit hohen Volumina weiter verringern, indem Sie sie auf ihren eigenen Server verschieben (siehe Abbildung 16.12). Wie zuvor kann ein einziger Verteilungsserver mehrere Verleger und mehrere Abonnenten unterstützen. Dies ist das bevorzugte Szenario, wenn Sie Situationen mit mehreren Verlegern und Abonnenten haben.

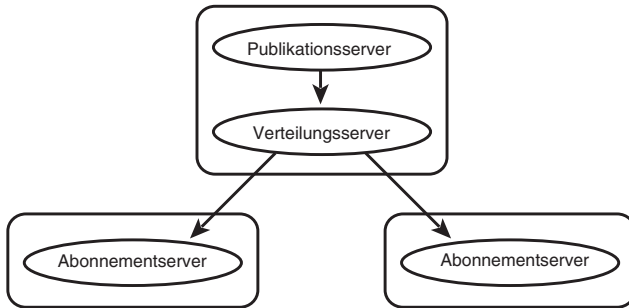


Abbildung 16.11:
Zentraler Verleger

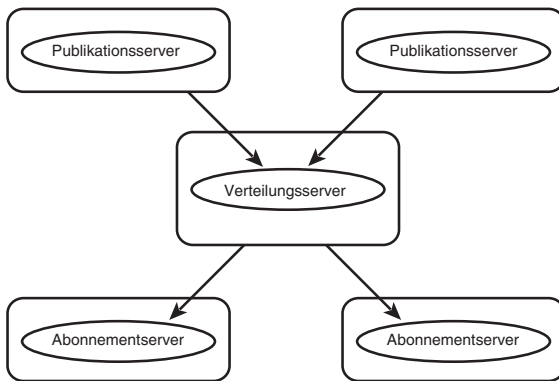


Abbildung 16.12:
Zentraler Verleger mit Remoteverteiler

Neuverleger (Publizierender Abonnent)

Im Szenario mit Neuverleger ist Ihr Abonnementsserver auch ein publizierender Server. Dies kann sehr günstig sein, wenn Sie eine langsame und preisgünstige Anbindung (reguläre Telefonleitungen) zwischen dem Hauptverleger und einer Gruppe von Abonnenten haben (siehe Abbildung 16.13). Bei diesem Szenario handeln die Abonnenten für eine Reihe von Abonnementdatenbanken auch als Verleger. Dadurch können Sie die Kosten Ihrer langsamen Anbindung minimieren, indem Sie einen einzigen Abonnenten aktualisieren und dann die neu synchronisierten Informationen wieder an den Rest der Abonnementsserver publizieren.

In Abbildung 16.13 befindet sich Ihre hauptpublizierende Datenbank in New York City. Sie können jetzt nach einem Zeitplan synchronisieren, der die Telefonleitungskosten zum Hauptabonnenten in Hamburg minimiert. Hamburg kann dann die Daten im näheren Bereich nach Berlin und München weiterverteilen.

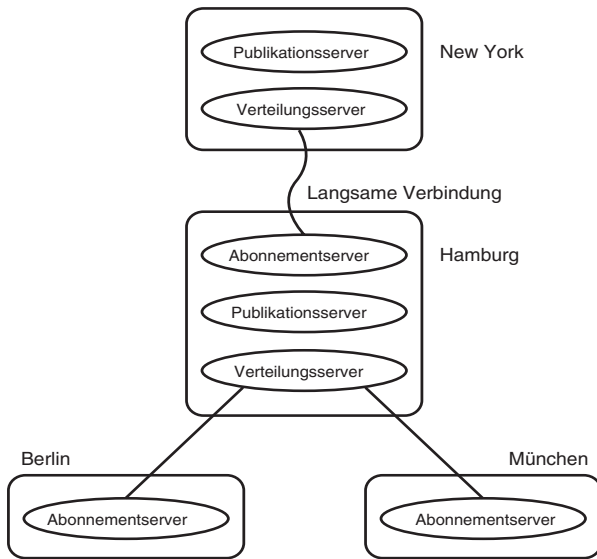


Abbildung 16.13:
Neuverleger (Publizierender Ab-
onent)

Zentraler Abonnent

Im Szenario mit zentralem Abonnenten ist Ihr Abonnementsserver zentral lokalisiert und ein oder mehrere publizierende Server schicken ihm Aktualisierungen. Dies ist nützlich für Zusammenfassungsberichte oder zentrale Verarbeitung. Beispielsweise zeigt Abbildung 16.14 ein zentrales Verlegerszenario, bei dem Zweigstellen in Seattle, Los Angeles und Detroit (die Verleger) Bestellungen an den Standort Denver schicken. Vom Standort Denver aus werden diese Bestellungen dann verarbeitet und das Produkt wird versandt.

Um zwischen Aufträgen zu unterscheiden, die von verschiedenen Standorten kommen, sollten Sie einige Schemaänderungen an Ihren Daten vornehmen:

- Fügen Sie ihrer Auftragsstabelle eine Spalte *Bereichscode* (RegionCode) hinzu.
- Machen Sie die neue Spalte *Bereichscode* zum Teil des Primärschlüssels.

Indem eine Spalte *Bereichscode* erstellt und zum Teil Ihres Primärschlüssels gemacht wird, können Sie Konflikte vermeiden. Was würde beispielsweise passieren, wenn Los Angeles eine Auftragsnummer 2000 hat und Detroit ebenfalls? Wenn sie beide auf den Abonnenten angewendet werden, gibt es einen Konflikt. Durch Hinzufügen des Bereichscodes haben Sie nun einen Primärschlüssel, der beispielsweise wie 2 2000 für Los Angeles und 3 2000 für Detroit aussehen könnte (siehe Abbildung 16.15).

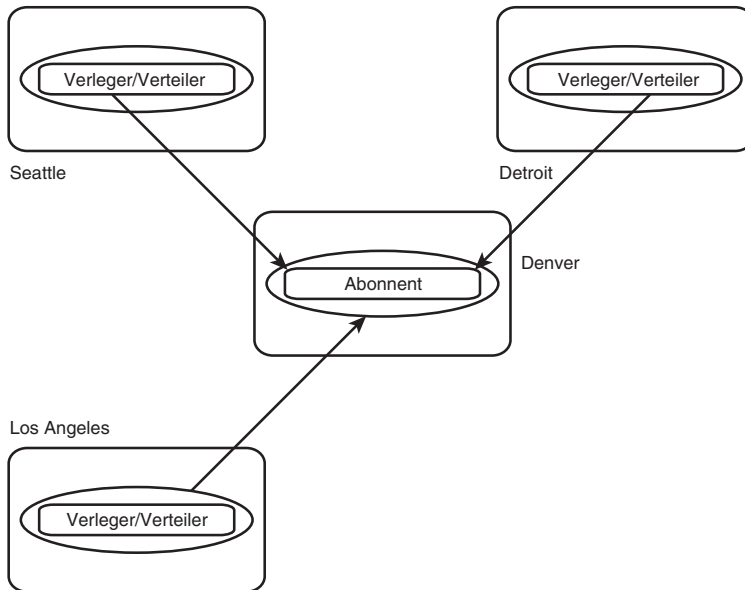


Abbildung 16.14:
Zentraler Abonnent

Bereich Seattle				Bereich Los Angeles			
Primärschlüssel				Primärschlüssel			
Bereichscode	Auftragsnr.	Teil	Stadt	Bereichscode	Auftragsnr.	Teil	Stadt
1	2000	565	335	2	2000	323	890
1	2001	786	100	2	2001	443	500

Bereich Denver			
Primärschlüssel			
Bereichscode	Auftragsnr.	Teil	Stadt
1	2000	565	335
1	2001	786	100
2	2000	323	890
2	2001	443	500
3	2000	233	100
3	2001	418	500

Bereich Detroit			
Primärschlüssel			
Bereichscode	Auftragsnr.	Teil	Stadt
3	2000	233	100
3	2001	418	500

Abbildung 16.15:
Mit einem Bereichs-
code lassen sich
Datenzeilen von ver-
schiedenen Quellen
unterscheiden.

Das zentrale Abonnementenszenario führt von selbst zu einer Replikation auf Transaktionsbasis. Jeder Standort, der einen Artikel bestellt, lässt die Artikelbestellung auf den zentralen Standort replizieren, von wo aus der Artikel verarbeitet und versandt wird.

Mehrere Verleger und Abonnenten

Mehrere Verleger und Abonnenten sind nützlich für Systeme vom Reservierungstyp oder beliebige andere Systeme, in denen Informationen aus anderen Regionen in jeder Region zur Verfügung stehen müssen. Nehmen wir als Beispiel ein Callcenter für 0800er Rufnummern. Wenn Sie eine 0800-Nummer anrufen, um ein spezielles Produkt zu bestellen oder den Status einer Bestellung zu überprüfen, könnte Ihr Anruf auf irgendeines der geografisch verteilten Callcenter im Land verteilt werden. Abbildung 16.16 zeigt Callcenter in Bend, Boulder und Atlanta. Jedes dieser Callcenter muss Informationen haben, die bei einem anderen Callcenter gesammelt wurden, um Kunden zu unterstützen, die den Status von Bestellungen überprüfen wollen.

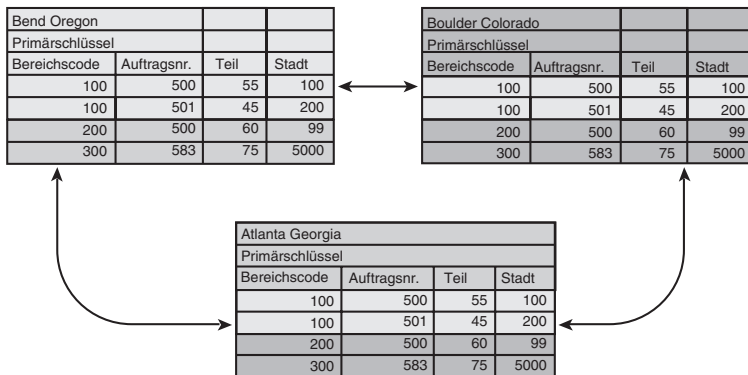


Abbildung 16.16:
Mehrere Verleger
und Abonnenten

In einem System vom Reservierungstyp können Sie diese Replikationsform ebenfalls implementieren. Hier empfehlen sich Transaktionen, so dass alle Standorte die gleichen Daten zur gleichen Zeit haben. (Sie würden keine doppelte Buchung einer Flugreservierung oder eines Hotelzimmers wünschen, oder?)

Wie Abbildung 16.16 zeigt, ist jeder Standort sowohl Verleger als auch Abonnent von Informationen. Jede Verkaufsstelle abonniert die publizierten Artikel der anderen Verkaufsstellen. Jede Verkaufsstelle sollte nur Artikel publizieren, die ihr gehören. Verwenden Sie also ein Feld *Bereichscode*, um zwischen Daten zu unterscheiden, die einer oder mehreren Verkaufsstellen gehören. So vermeiden sie die Konflikte, die der Abschnitt »Zentraler Abonnent« diskutiert hat.

Damit dieses Szenario richtig funktioniert, sollte jede Verkaufsstelle nur Daten von ihrem eigenen Standort aktualisieren. Wenn Sie hier einen Szenariotyp mit aktualisierendem Abonnenten verwenden, stellen Sie sicher, dass Sie alle Ihre Aktualisierungen an allen Stellen gleichzeitig vornehmen, um Konflikte zu vermeiden.

Publizierte Daten transformieren

Bei Snapshot- und Transaktionsreplikation erlaubt es SQL Server 2000, auf die Data Transformation Services (DTS) zurückzugreifen, um die Publikationen je nach Abonnent anzupassen. Auch diese Option können Sie festlegen, wenn Sie eine Publikation erstellen. Wenn Sie das Transformieren publizierter Daten in der Publikation aktiviert haben, können Sie ein Replikations-DTS-Paket mithilfe des Assistenten zum Transformieren publizierter Daten oder im DTS-Designer erstellen. Das DTS-Paket ist als Teil des Abonnementprozesses anzufügen. Allerdings ist diese Funktionalität auf SQL Server 2000 und OLE DB-Abonnenten beschränkt.

16.7 SQL Server 2000-Datenbanken im Internet publizieren

Um das Publizieren Ihrer SQL Server-Datenbank im Internet zu ermöglichen, müssen Sie einige zusätzliche Konfigurationsänderungen vornehmen. Tag 21 geht näher darauf ein. Allerdings sollten Sie im Rahmen der Replikation bereits folgende Punkte beachten, wenn Sie sowohl mit Push- als auch mit Pullabonnements arbeiten:

- TCP/IP muss aktiviert und auf allen Computern installiert sein, auf denen der Verteilungs-Agent und der Merge-Agent laufen. Das Gleiche trifft auf Computer zu, die mit den Computern verbunden sind, auf denen diese Agents laufen.
- Publikations- und Verteilungsserver sollten über eine direkte Netzwerkverbindung und nicht über das Internet verbunden sein. Das empfiehlt sich nicht nur aus Sicherheitsgründen, sondern auch, um mögliche Probleme mit der Latenz zu vermeiden.
- Publikations- und Verteilungsserver sollten sich auf derselben – vorzugsweise der sicheren – Seite der Firewall befinden.

Publizieren über ein VPN

Das sicherste Verfahren für das Publizieren über das Internet stellt die VPN-Technologie (Virtual Private Network) dar. Damit können sich Remotebenutzer zu einem Firmennetzwerk über sichere Kommunikationskanäle verbinden. Ein VPN realisiert die Verbindung zweier Netzwerke über das Internet als WAN- (Wide Area Network) Verbindung zwischen den Standorten. Für diesen Zweck setzt man häufig das über Windows NT 4.0 oder Windows 2000 verfügbare Point-to-Point-Tunneling-Protokoll (PPTP) oder das über Windows

2000 verfügbare Layer-2-Tunneling-Protokoll (L2TP) ein. Nachdem ein VPN konfiguriert ist, lässt sich die Replikation genauso konfigurieren, als würden sich Verleger und Abonnent im selben LAN befinden.

Publizieren über Microsoft Proxy Server

Die Replikation über das Internet in SQL Server 2000 lässt sich so konfigurieren, dass sie die von Microsoft Proxy Server gebotene Sicherheit nutzt. In dieser Lösung werden Internet-Pullabonnements konfiguriert, um die Verbindung zum Proxy Server statt direkt zu Ihrem SQL Server herzustellen. Im Proxy Server kann man nicht autorisierten Zugriff auf die internen Netzwerkressourcen unterbinden.

Publizieren mithilfe von FTP

Wenn Sie Pullabonnements aus dem Internet zulassen, müssen Sie an Ihren Servern die folgenden zusätzlichen Konfigurationsänderungen vornehmen:

- IIS (Internet Information Server) muss auf demselben Server wie Ihr Verteilungsserver installiert sein.
- Der Merge-Agent und der Verteilungs-Agent müssen mit der richtigen FTP-Adresse konfiguriert sein. Diese Adresse lässt sich über Dienstprogramm Verteiler-Agent oder von einer Befehlszeile auf dem Verteilungsserver konfigurieren.
- Die FTP-Pfade und Anschlüsse unter den Publikationseigenschaften müssen mit einer FTP-Site konfiguriert sein, die für den Standort des Snapshotordners verwendet wird. Sie können auch das FTP-Stammverzeichnis als Snapshotordner einrichten (in der Voreinstellung `\Microsoft SQL Server\MSSQL\Repldata\FTP`) und dieses Stammverzeichnis als FTP-Site konfigurieren.
- Der Snapshotordner muss Ihren Abonnementservern zur Verfügung stehen.

16.8 Empfehlungen

Wenn Sie den Typ der Replikation und das zu implementierende Szenario bestimmen, sollten Sie die folgenden Punkte beachten, um die beste Leistung zu erzielen:

- Stellen Sie Netzwerkgeschwindigkeit und -verfügbarkeit sicher.
- Sofort aktualisierende Abonnenten können nützlich sein, wenn Sie ausreichend Netzwerkbandbreite zur Verfügung haben.

- Publizieren Sie partitionierte Daten. Es ist nicht nötig, immer alle Informationen gleichzeitig zu verschicken. Geben Sie den Abonnenten, was sie brauchen, und verringern Sie überflüssigen Netzwerk- und Verteilungsaufwand.
- Pullabonnements können einen großen Teil der Arbeitslast vom Verteilungsserver abziehen. Denken Sie daran, dass Sie Ihre zentralisierte Replikationsverwaltung verlieren, jedoch einiges an Leistung gewinnen.
- Verwenden Sie in Ihren Szenarios einen Remoteverteiler. Damit entlasten Sie Ihre OLTP-Datenbank von Replikationsaufgaben. Dies ist auch ein nützliches Werkzeug, wenn Sie eine langsame und teure Verbindung zwischen Ihrem Verteilungsserver und Ihren Abonnementservern haben.
- Minimieren Sie Aktualisierungskonflikte durch Minimieren der Wahrscheinlichkeit ihres Auftretens, wenn Sie Mergereplikation oder aktualisierende Abonnenten verwenden.
- Stellen Sie bei Mergereplikation oder aktualisierenden Abonnenten sicher, dass Sie einen Gebietscode als Teil Ihres Primärschlüssels einschließen.

16.9 Replikation in einer heterogenen Umgebung

SQL Server 2000 unterstützt Replikation in einer heterogenen Umgebung. Das heißt, dass Sie einen SQL Server 2000-Verleger verwenden können, um über ODBC-Treiber und OLE DB-Provider Publikationen an Abonnenten zu verschicken. In einigen Fällen können Sie sogar SQL Server 2000 als Abonnenten eines Nicht-SQL Server-Verlegers verwenden.

Replikation wird auf den Plattformen Windows 9x und Windows NT unterstützt. Replikation zu anderen Plattformen wird über geeignete ODBC-Treiber oder OLE DB-Provider und Kommunikationssoftware unterstützt. SQL Server 2000 unterstützt Replikation zu Microsoft Access, Oracle und jede andere Datenbank, die mit den ODBC-Abonnentenanforderungen von SQL Server konform ist. SQL Server wird mit Treibern für Oracle, Access und das Datenprotokoll DRDA (Distributed Relational Database Architecture) von IBM ausgeliefert.

Der Treiber für ODBC-Abonnenten muss:

- ODBC-Level-1-Konformität unterstützen
- die Datendefinitionssprache (DDL) von Transact-SQL unterstützen
- im 32-Bit-Modus laufen
- Thread-fähig sein
- Aktualisierungen zulassen

Publizieren von Daten an Nicht-SQL Server 2000-Datenbanken

Beachten Sie folgende Punkte, wenn Sie die Replikation auf Nicht-SQL Server 2000 ODBC-Abonnenten durchführen:

- SQL Server 2000 unterstützt keine Pullabonnements. Sie müssen Pushabonnements von einem SQL Server 2000-Verleger aus verwenden.
- Stapelanweisungen an ODBC-Abonnenten werden nicht unterstützt.
- Datentypen werden auf den ähnlichsten verfügbaren Datentyp abgebildet, der auf der Abonentendatenbank zur Verfügung steht. Beispielsweise kennt Microsoft Access kein Konzept für ein Textfeld. Access würde ein repliziertes Textfeld als Access-Datentyp Memo erhalten.
- Der ODBC DSN (Datenquellenname) muss den Regeln für SQL Server-Bezeichner entsprechen.
- Wenn Sie mit Snapshots arbeiten, verwenden die zu übertragenden Daten das Zeichenformat des Programms zum Massenkopieren (BCP). Tag 19 geht detailliert auf das Dienstprogramm BCP ein.

Nicht-SQL Server 2000-Datenbanken an SQL Server 2000 publizieren

In der SQL Server 2000-Umgebung ist es möglich, dass Datenbanken von Drittanbietern die Rolle von Verlegern übernehmen. Um dies zu bewirken, müssen Sie ein Programm schreiben, das die Vorzüge von SQL-DMO (SQL Database Management Objects) nutzt. Nachdem Sie die entsprechenden Programme erstellt haben, können Sie alle Replikationsfeatures unterstützen, die SQL Server 2000 bietet.

SQL-DMO legt die folgenden COM-Objekte für die Replikation offen:

- Objekte für die Replikationsverwaltung
- Objekte für die Replikationüberwachung
- Objekte, die die Vorteile des Verteilungs-Agents für Weiterleitungstransaktionen zu Abonentenservern nutzen
- Objekte zum Arbeiten mit der Verteilungsdatenbank zur Speicherung von replizierten Transaktionen
- Datenquellen des SQL Server-Systemmonitors, von denen mehrere zu den SQL Server-Systemmonitorobjekten gruppiert sind und im SQL Server-Systemmonitor angezeigt werden

- Objekte, um Ihren Verleger für den SQL Server Enterprise Manager zur grafischen Verwaltung offen zu legen

16.10 Zusammenfassung

Die heutige Lektion hat sich ausschließlich den Grundlagen der Replikation gewidmet. Dabei haben Sie die verschiedenen Replikationstypen kennen gelernt und wissen jetzt, wann man welchen Typ in welchem Szenario einsetzt. Dabei sind die Faktoren Autonomie für die Abonnenten, Transaktionskonsistenz und Latenz zu berücksichtigen.

Als Replikationstypen haben Sie Mergereplikation, Snapshotreplikation und Transaktionsreplikation kennen gelernt. Außerdem hat diese Lektion die Optionen für die Aktualisierung von Abonnenten erläutert.

Zentrales Prinzip der Replikation ist das Modell »Publizieren und Abonnieren«. Für die Funktionsweise der Replikation sind fünf Agents verantwortlich: Verteilungs-, Protokolllese-, Merge-, Snapshot- und Warteschlangenlese-Agent.

16.11 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F Was ist das Wichtigste: Transaktionskonsistenz, Latenz oder Autonomie?

- A** Allgemein gesagt ist Transaktionskonsistenz von höchster Bedeutung. Über Latenz und Siteautonomie müssen Sie von Fall zu Fall entscheiden, aber die korrekten Daten zu haben, sollte der wichtigste Punkt sein. Denken Sie daran, dass es nur bei der Mergereplikation die Möglichkeit gibt, die Transaktionskonsistenz zu verlieren. Alle anderen Replikationstypen unterstützen die Transaktionskonsistenz vollständig.

F *Welchen Replikationstyp sehen Sie am häufigsten?*

A Zur Zeit ist Transaktionsreplikation am häufigsten. Das liegt daran, dass SQL Server 6.5 nur Transaktionsreplikation und verteilte Transaktionen unterstützt. Transaktionsreplikation mit aktualisierenden Abonnenten wird vermutlich die beliebteste Form der Replikation werden, weil sie Transaktionskonsistenz, ein vernünftiges Maß an Siteautonomie und ein niedriges Latenzniveau einschließt.

F *Sollte ich Mergereplikation oder Aktualisieren über eine Warteschlange verwenden?*

A Diese Frage ist nicht so ohne weiteres zu beantworten. Mergereplikation behandelt Konflikte auf der Zeilenebene, während das Aktualisieren über eine Warteschlange die Konflikte auf Transaktionsebene auflöst. Beispielsweise kann bei einem umfangreichen Kassensystem die Warteschlangenaktualisierung sinnvoller sein als die Mergereplikation. In den Beispielen, die diese Lektion zur Mergereplikation angegeben hat, wäre die Warteschlangenaktualisierung weniger geeignet. Grundsätzlich hängt die Antwort davon ab, ob in Ihrer Anwendung eine Zeile oder eine Transaktion wichtiger ist.

F *Wie schwierig ist es, die Konfliktlösungstrigger zu bearbeiten?*

A Das ist nicht allzu schwierig. Es handelt sich nur um Trigger mit entsprechendem Code. Wenn Sie sich ausreichend fit fühlen, um in SQL Server zu programmieren, sollten diese Trigger eine Kleinigkeit sein. Sie können Ihre Trigger implementieren, um sowohl die gewählte Version eines Datensatzes als auch die nicht gewählte Version des konfliktverursachenden Datensatzes in eine separate Tabelle zu kopieren und zu bewerten, wie gut Ihr Triggercode arbeitet.

F *Was ist eine ROWGUID?*

A Eine ROWGUID ist ein eindeutiger 128-Bit-Bezeichner. Es handelt sich dabei um eine global eindeutige Zahl. SQL Server verwendet einen Algorithmus, der diese Zahlen basierend auf vielen Faktoren erstellt. ROWGUIDs werden üblicherweise Tabellen hinzugefügt, die für die Mergereplikation vorgemerkt sind. Mit diesen ROWGUIDs können Sie die Herkunft von Veränderungen verfolgen, die an einem speziellen Datensatz vorgenommen wurden.

F *Kann der Verteilungsserver alle unterschiedlichen Typen der Replikation gleichzeitig unterstützen?*

A Ja. In der Tat können Sie Publikationen und Abonnements auf denselben Servern einrichten, die unterschiedliche Typen von Replikation verwenden – und zwar auf der Basis einzelner Publikationen.

Kontrollfragen

1. Welcher Agent liest das Transaktionsprotokoll von publizierten Datenbanken und kopiert dann die Daten zur Verteilungsdatenbank?
2. Was macht der Verteilungs-Agent?
3. Welches Replikationsszenario ist am besten für Zusammenfassungsberichte geeignet?
4. Wo befindet sich der Merge-Agent bei einem Pushabonnement mit Mergereplikation?

Übung

1. Bestimmen Sie das geeignetste Replikationsszenario unter Berücksichtigung der folgenden Fakten:
 - ▶ Sie haben vier Standorte in den Vereinigten Staaten, die die Auftragsverarbeitung durchführen. In Wichita, Kansas, befindet sich ein zentrales Lager, das Auftragsbelege haben muss, die so erstellt sind, dass die Güter versandt werden können.
 - ▶ In Großbritannien befinden sich drei Standorte, die MIS-Aufgaben und andere Arten der Verwaltung realisieren. Die britischen Standorte müssen Kopien von allen Datensätzen der versandten Artikel und die hierfür angenommenen Aufträge haben.
 - ▶ Die britischen Niederlassungen möchten aus Kostengründen keine Standleitung in die Vereinigten Staaten unterhalten.



Replikation – Implementierung



Die gestrige Lektion war sehr umfangreich und enthielt zahlreiche Themen, die Neueinsteiger in SQL Server leicht überfordern können. Deshalb sollten Sie sich an dieser Stelle etwas Zeit für eine Wiederholung nehmen.

Der heutige Tag beginnt mit einem Überblick zu Sicherheitsproblemen. Dann aber geht es in medias res: Als Erstes richten Sie Ihre Server für die Replikation ein. Dann erfahren Sie, wie man Publikationen und Abonnements erstellt.

Der nächste Schwerpunkt dieser Lektion ist die Replikationsverwaltung. Dabei beschäftigen Sie sich mit Fragen in Bezug auf die Überwachung der Replikation, das Erstellen von Replikationsskripts, den Windows NT/2000-Systemmonitor, Probleme bei der Sicherung von Verteilungsdatenbanken und den Verlauf der Replikations-Agents.

Die Lektion schließt ab mit einem kurzen Abschnitt zur Fehlerbehebung.

17.1 Sicherheitsbetrachtungen

Für die Replikation sind einige Voraussetzungen zu erfüllen:

- Jeder Server, der in den Replikationsprozess einbezogen ist, muss im Enterprise Manager registriert sein.
- Wenn Sie unter Windows NT mit SQL Servern aus verschiedenen Domänen arbeiten, müssen zwischen den Domänen vertraute Beziehungen eingerichtet werden. Windows 2000 implementiert das Domänenmodell in anderer Form und bidirektionale, transitive Vertrauensstellungen sind der Standard.
- Die Replikation verwendet den gleichen Sicherheitskontext des SQL Server-Agent-Dienstes unter Windows NT/2000. Dieses Konto muss Administratorrechte haben und sollte ein Mitglied der lokalen Administratorengruppe sein.
- Das Konto muss als Dienst mit erweiterten Benutzerrechten konfiguriert sein.



Sie sollten auf jeden Fall ein einzelnes Windows 2000-Konto erstellen, das alle SQL Server-Agents gemeinsam nutzen. Damit schließen Sie mögliche Verbindungsprobleme von vornherein aus. Wenn Sie sich für alternative Konten entscheiden, brauchen diese Konten Zugriff auf das Arbeitsverzeichnis des Verteilungsservers. Diese Konten müssen Sie der lokalen Windows NT-Gruppe *Administratoren* auf dem Verteilungsserver hinzufügen. Bei Nicht-Windows NT-Computern können Sie auch SQL Server-Benutzerkonten verwenden.



Lokale Systemkonten haben *keinen* Netzwerkzugriff und können deshalb nicht an der Replikation teilnehmen.

17.2 Server einrichten

In diesem Abschnitt bereiten Sie Ihre Server mit dem Publizierungs- und Verteilungskonfigurations-Assistenten im Enterprise Manager vor.

Sie müssen zuerst den Verteilungsserver installieren, bevor Sie abhängige Publikationsserver einrichten können. Um den Verteiler zu erstellen, brauchen Sie *sysadmin*-Rechte. Nachdem Sie den Verteilungsdienst installiert haben, sehen Sie sowohl die lokalen Servereigenschaften als auch die Eigenschaften des Remoteverteilers. Die Verteilungsdatenbank wird automatisch installiert, wenn Sie Ihren Verteilungsserver anlegen. Sie können weitere Verteilungsdatenbanken sowohl auf einem lokalen als auch auf einem Remoteverteiler anlegen.

Außerdem müssen Sie den Zugriff auf das Arbeitsverzeichnis des Verteilungsservers sicherstellen. Normalerweise ist das Arbeitsverzeichnis unter `\\computer_name\C$\Programme\Microsoft SQL Server\MSSQL\Repldata` eingerichtet. Unter Windows 2000 wird dieser Ordner standardmäßig angelegt.

Schließlich ist zu gewährleisten, dass der Verteilungsserver genügend Speicher besitzt – mindestens 32 MB auf dem Server, wovon 16 MB für SQL Server reserviert sind, um im Rahmen der Replikation als Verteiler zu wirken. Wenn Sie viele Verleger oder viele Abonnenten unterstützen, sollten Sie zusätzlichen Speicher vorsehen.

Einen Verteilungsserver können Sie über den Publizierungs- und Verteilungsdeaktivierungs-Assistenten im Enterprise Managers deinstallieren. Beachten Sie das Folgende, wenn Sie einen Verteilungsserver entfernen:

- Alle Verleger, die den Verteilungsserver nutzen, werden deaktiviert.
- Alle Publikationen, die auf diesen Servern liegen, werden gelöscht.
- Alle Abonnements für diese Publikationen werden gelöscht.
- Alle Verteilungsdatenbanken auf dem zu deinstallierenden Verteilungsserver werden gelöscht.

Die nächsten Abschnitte zeigen, wie Sie Ihren Server für die Replikation installieren und konfigurieren. Sie können denselben Server als Verleger, Verteiler und Abonnent einsetzen. Bei mehreren Servern sind die hier beschriebenen Abläufe gleich. In jedem Abschnitt erhalten Sie bei Bedarf zusätzliche Hinweise.

17.3 Replikation installieren

Um die Replikation zu installieren und zu aktivieren, müssen Sie drei in sich abgeschlossene Schritte durchführen:

1. Einen Verteilungsserver installieren.
2. Publikationen erstellen.
3. Diese Publikationen abonnieren.

Mehr zu diesen Schritten erfahren Sie zum gegebenen Zeitpunkt, wenn Sie die betreffenden Abschnitte durcharbeiten.

Installation des Verteilungsservers

Ein Verteilungsserver erfordert zusätzlichen Festplattenplatz, um Snapshots und die Verteilungsdatenbank zu speichern. Die verschiedenen Replikations-Agents, die auf diesem Server laufen, benötigen außerdem zusätzliche Prozessorzeit. Wenn Sie Ihren Verteiler installieren, beachten Sie die folgenden Punkte:

- Stellen Sie sicher, dass genügend Festplattenspeicherplatz für den Arbeitsordner und die Datenbank des Verteilers vorhanden ist.
- Die Verteilungsdatenbank muss alle Transaktionen zwischen Verleger und Abonnent speichern, wenn Sie mit Transaktionsreplikation arbeiten.
- Sie müssen die Transaktionsprotokolldatei der Verteilungsdatenbank für die Transaktionsreplikation sorgfältig verwalten, weil jede publizierte Transaktion und jedes angewandte Abonnement protokolliert werden.



Snapshot- und Mergereplikation speichern Daten im Arbeitsordner des Verteilers. Die Verteilerdatenbank zeichnet bei diesen Verfahren der Mergereplikation nur den Status Ihrer Replikationsaufträge auf. Denken Sie daran, dass bei der Transaktionsreplikation alle Transaktionen in der Verteilungsdatenbank gespeichert werden.

Wenn Sie über die Größe Ihrer Verteilungsdatenbank etwas besorgt sind, sollten Sie sich im Hinblick auf die Größenanforderungen für Ihren Verteiler mit folgenden Fragen beschäftigen:

- Wie viele Artikel sind zu publizieren?
- Wie groß sind die zu publizierenden Artikel?

- Gibt es Artikel mit text- oder image-Datentypen? Wenn das der Fall ist, müssen Sie entweder eine geplante Tabellenaktualisierung oder eine Snapshotreplikation verwenden, weil diese Elemente während einer Transaktionsreplikation nicht repliziert werden.
- Lässt sich der Platzbedarf durch eine höhere Latenz zwischen den Synchronisierungen signifikant erhöhen? Transaktionsinformationen auf dem Verteiler werden zurückgehalten, bis sie auf den möglicherweise offline arbeitenden Abonnenten angewandt werden.
- Wie groß ist die Anzahl der INSERT- und UPDATE-Anweisungen bei Transaktionsreplikation? Denken Sie daran, dass diese Anweisungen tatsächliche Datenänderungen enthalten. Wenn Sie 500 Zeilen in einen Server einfügen, sind 500 Datenzeilen auf dem Verteilungsserver unterzubringen.
- Wie viele Transaktionen sammeln sich bei Transaktionsreplikation in einer bestimmten Zeiteinheit an? Zum Beispiel: Wie viele Transaktionen sind voraussichtlich in einer halben Stunde zu replizieren?

Unter der Annahme, dass keine Probleme in Bezug auf den Größenbedarf bestehen oder diese sich unkompliziert lösen lassen, können Sie sich nun der Installation eines Verteilungsservers zuwenden:

1. Stellen Sie die Verbindung zu Ihrem SQL Server über den SQL Enterprise Manager her.
2. Markieren Sie Ihren Server und wählen Sie EXTRAS / REPLIKATION / PUBLIZIERUNG, ABONNENTEN UND VERTEILUNG KONFIGURIEREN. Sie können auch den Server markieren und im Taskpad die Konfigurationsoption *Publizierung und Verteilung konfigurieren* wählen.
3. Es erscheint ein Willkommen-Bildschirm (siehe Abbildung 17.1). Klicken Sie auf WEITER.
4. Sie müssen jetzt einen Server als Verteilungsserver auswählen (siehe Abbildung 17.2). Standardmäßig wird der Computer, mit welchem Sie verbunden sind, als Verteiler ausgewählt. Um einen anderen Verteiler festzulegen, wählen Sie die Option *Den folgenden Server verwenden* und geben dann einen anderen Server an, der bereits in Ihrem Unternehmen registriert ist. Klicken Sie auf WEITER.
5. Es erscheint die Aufforderung, den Snapshotordner anzugeben. Übernehmen Sie die Standardeinstellung und klicken Sie auf WEITER. Wenn SQL Server eine Meldung anzeigt, dass der gewählte Snapshotordner nur durch einen Benutzernamen mit Administratorprivilegien zugänglich ist, klicken Sie auf OK um fortzusetzen.



Abbildung 17.1:
Willkommen-Bild-
schirm des Publizie-
rungs- und
Verteilungs-
konfigurations-Assis-
tenten

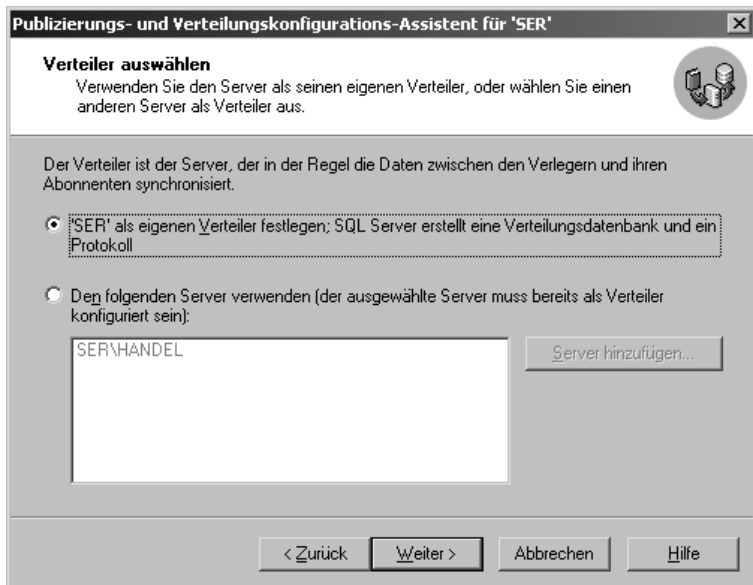


Abbildung 17.2:
Einen Verteiler aus-
wählen

- Legen Sie fest, ob Sie eine Standardkonfiguration verwenden oder die Einstellungen anpassen möchten. Wählen Sie für dieses Beispiel die Option *Ja, ich möchte den Namen und den Speicherort ... anpassen*. Die in Abbildung 17.3 gezeigten Standardein-

stellungen spezifizieren den Computer *SER* (mit dem alle Beispiele in diesem Buch entstanden sind) als Verteiler. Der Server *SER* kann von *SER* abonnieren und auf *SER* publizieren. Die Verteilungsdatenbank ist im Ordner `\MSSQL\Data` untergebracht. Alle anderen im Enterprise Manager registrierten SQL Server sind ebenfalls in dieser Liste aufgeführt. Haben Sie zum Beispiel einen SQL Server als *Leap* registriert, können Sie mit dem Computer *Leap* publizieren und abonnieren. Klicken Sie auf **WEITER**.

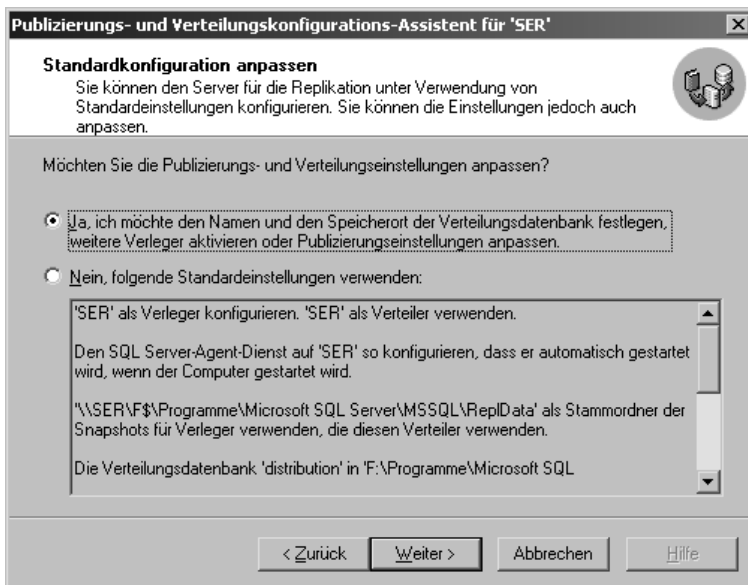


Abbildung 17.3:
Das Dialogfeld
Standardkonfigura-
tion anpassen

7. Im Dialogfeld **VERTEILUNGSDATENBANKINFORMATIONEN BEREITSTELLEN** (siehe Abbildung 17.4) spezifizieren Sie den Namen der Verteilungsdatenbank, den Ordner für die Datenbankdatei und den Ordner für die zugehörige Protokolldatei. Diese Dateien müssen sich auf dem Verteilungsserver befinden (im Beispiel *SER*). Klicken Sie auf **WEITER**.
8. Im Dialogfeld **VERLEGER AKTIVIEREN** können Sie Berechtigungen für andere Verleger erteilen, diesen Verteiler zu verwenden. Wählen Sie einfach den registrierten Server aus, dem Sie den Zugriff erteilen möchten, und klicken Sie auf die Schaltfläche mit den drei Punkten (...), um die Sicherheitsoptionen festzulegen. Wenn Ihr Server nicht aufgeführt ist, klicken Sie auf die Schaltfläche **NEU**, um einen neuen Server hinzuzufügen. Schalten Sie das Kontrollkästchen neben dem Namen Ihres Computers ein und klicken Sie dann auf **WEITER**.

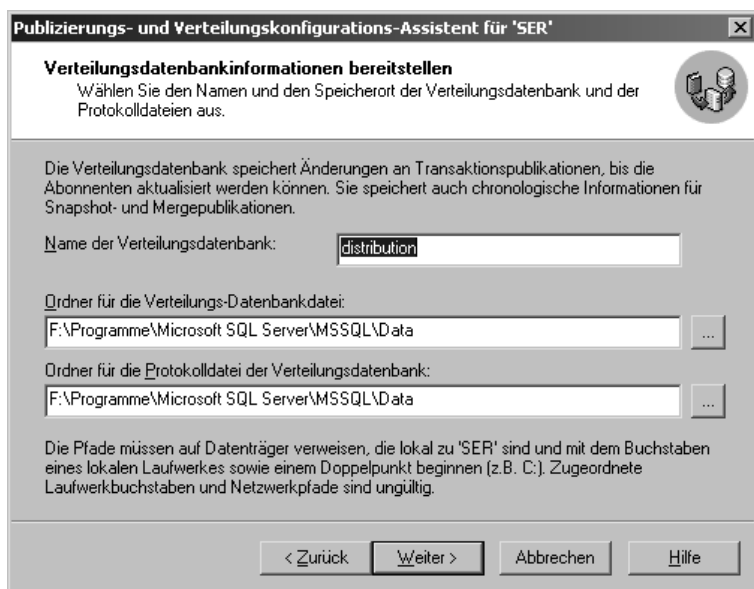


Abbildung 17.4:
Das Dialogfeld Ver-
teilungsdatenbank-
informationen
bereitstellen

9. Im Dialogfeld PUBLIKATIONS-DATENBANK AKTIVIEREN (siehe Abbildung 17.5) legen Sie fest, ob Sie Transaktionsreplikation (eingeschlossen Snapshot) oder Mergereplikation für Ihre Datenbanken nutzen wollen. Sie können die Replikationstypen entweder einzeln durch Klicken auf die Kontrollkästchen neben den Datenbanken oder insgesamt über die Schaltflächen auf der rechten Seite aktivieren bzw. deaktivieren. Schalten Sie für das Beispiel das Kontrollkästchen in der Spalte TRANS für die Datenbank *pubs* ein und klicken Sie dann auf WEITER.



Beachten Sie folgende Regeln, wenn Sie eine Publikationsdatenbank einrichten:

- ▶ Sie müssen Mitglied der festen Serverrolle *sysadmin* sein.
- ▶ Nachdem Sie eine Datenbank für das Publizieren eingerichtet haben, können Mitglieder der festen Datenbankrolle *db_owner* Publikationen erstellen.

10. Im Dialogfeld ABONNENTEN AKTIVIEREN (siehe Abbildung 17.6) sind alle registrierten Server als Abonnenten verfügbar. Über die Schaltfläche Neu können Sie auch hier zusätzliche Server registrieren. Um einen Server als Abonnenten zu aktivieren, schalten Sie das Kontrollkästchen neben dem Servernamen ein.
11. Über die Schaltfläche mit den drei Punkten neben dem Servernamen gelangen Sie zu einem Dialogfeld, in dem Sie Anmeldeinformationen und Standardzeitpläne für die Replikation einrichten können (siehe Abbildung 17.7).

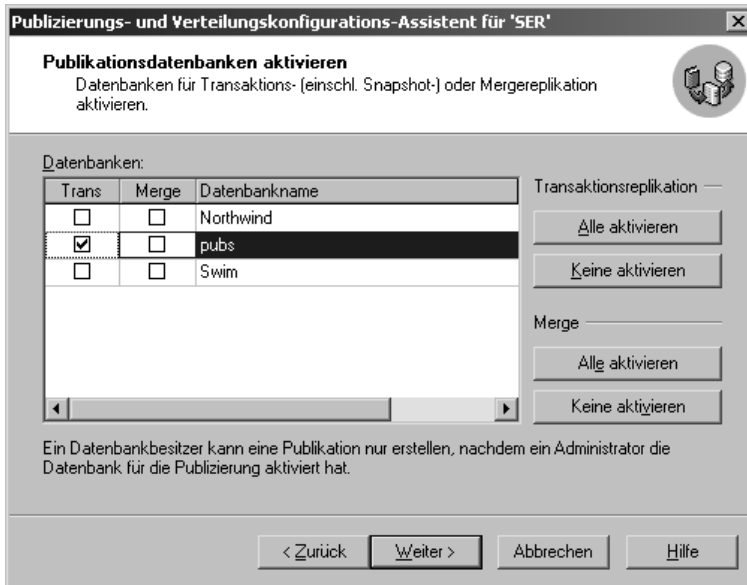


Abbildung 17.5:
Das Dialogfeld
Publikationsdaten-
banken aktivieren

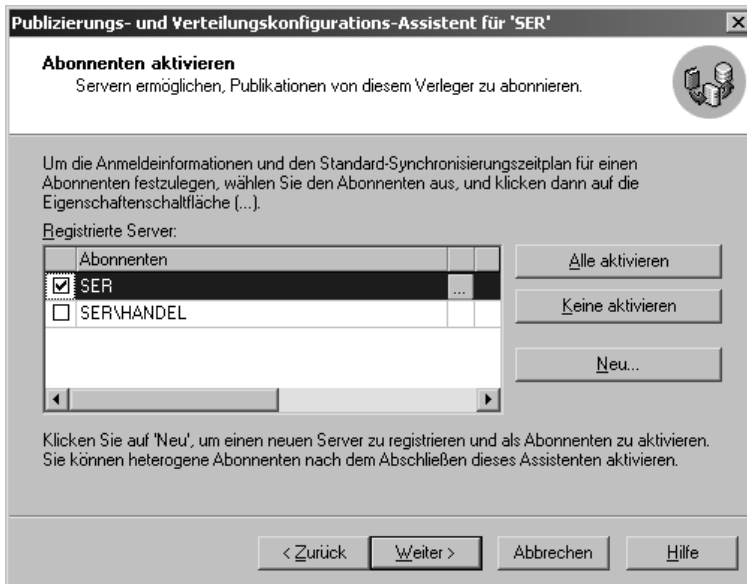


Abbildung 17.6:
Das Dialogfeld
Abonnenten aktivie-
ren



Über die in Abbildung 17.6 gezeigte Schaltfläche NEU können Sie nur Microsoft SQL Server registrieren. Um Nicht-Microsoft-SQL Server hinzuzufügen, wählen Sie PUBLIZIERUNG, ABONNENTEN UND VERTEILUNG KONFIGURIEREN aus dem Menü EXTRAS des Enterprise Managers. Mehr dazu erfahren Sie später in dieser Lektion im Abschnitt »Erstellen einer Publikation«.

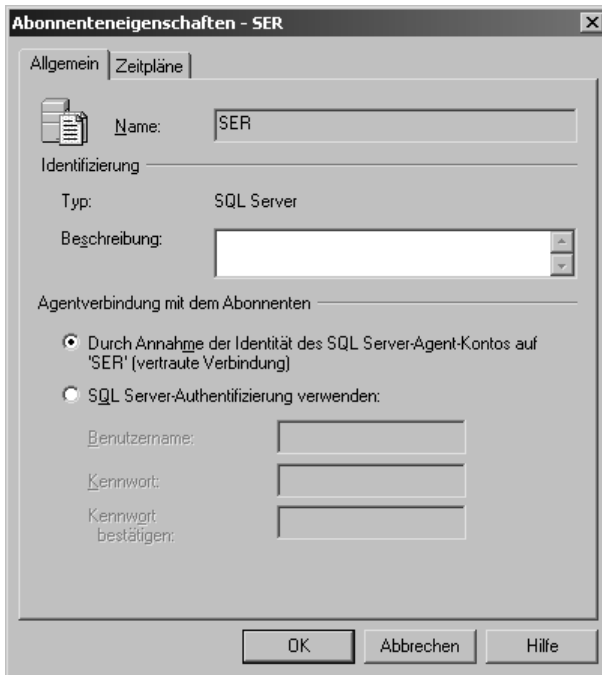


Abbildung 17.7:
Die Registerkarte Allgemein des
Dialogfelds Abonenteneigenschaften

12. Im Dialogfeld ABONNENTENEIGENSCHAFTEN können Sie festlegen, ob Sie die Verbindung zum Abonnenten mithilfe eines so genannten Identitätswechsels – hier durch Annahme der Identität des SQL Server-Agent-Kontos auf dem Computer des Abonnenten – herstellen möchten. Wenn Sie die alternative Option *SQL Server-Authentifizierung verwenden* wählen, müssen Sie einen Benutzernamen und ein Kennwort bereitstellen.
13. Auf der Registerkarte ZEITPLÄNE (siehe Abbildung 17.8) können Sie festlegen, ob der Verteilungs- und der Merge-Agent fortlaufend oder zu geplanten Zeiten ausgeführt werden sollen. Die Standardeinstellung für den Verteilungs-Agent ist *Fortlaufend*. Wenn Sie auf die Schaltfläche ÄNDERN klicken, erscheinen die Optionen zur Planung von Replikationszeiten. Sie werden sich später noch damit befassen. Vergewissern Sie sich, dass die Option *Fortlaufend* ausgewählt ist und klicken Sie auf OK, um das Dialogfeld zu schließen. Damit kommen Sie wieder zum Dialogfeld ABONNENTEN AKTIVIEREN (siehe Abbildung 17.6) zurück. Klicken Sie hier auf WEITER.

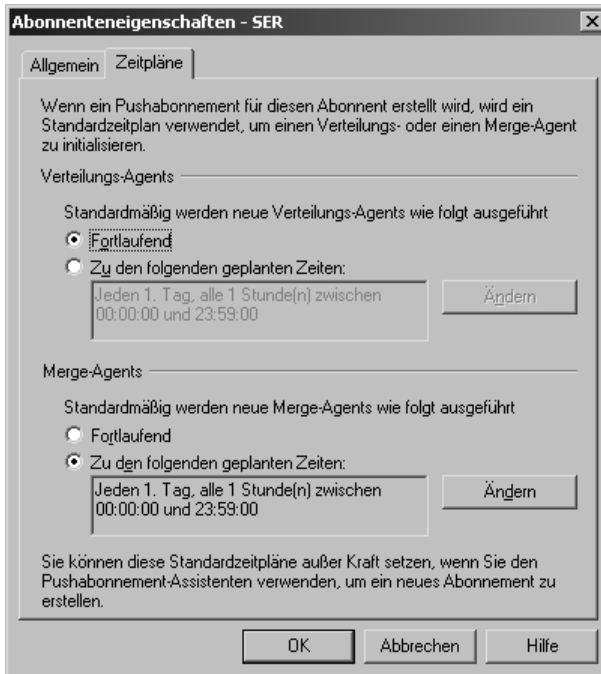


Abbildung 17.8:
Die Registerkarte Zeitpläne des
Dialogfelds Abonenteneigenschaf-
ten

14. Es erscheint das Dialogfeld PUBLIZIERUNGS- UND VERTEILUNGSKONFIGURATIONS-ASSISTENTEN ABSCHLIESSEN mit einer Zusammenfassung der von Ihnen vorgenommenen Einstellungen. Klicken Sie auf FERTIG STELLEN, um diesen Teil des Replikationskonfiguration abzuschließen.

Nachdem Sie im letzten Schritt auf FERTIG STELLEN geklickt haben, wird ein Fortschrittsdialogfeld eingeblendet, das die derzeit implementierten Abschnitte zeigt (siehe Abbildung 17.9). Das letzte Dialogfeld zeigt die Meldung, dass SQL Server Enterprise Manager den Computer erfolgreich als Verteiler für den spezifizierten Server aktiviert hat. Schließlich erscheint noch ein Bildschirm des Replikationsmonitors, das Sie durch Klicken auf OK schließen.



Wenn Ihr SQL Server-Agent-Dienst nicht so konfiguriert ist, dass er automatisch startet, erscheint ein Meldungsfeld mit den Schaltflächen JA und NEIN. Damit können Sie entscheiden, ob Sie den Agent für den automatischen Start konfigurieren möchten.

Falls Sie die Replikation auf diesem Server erstmalig installieren, erhalten Sie außerdem eine Information, dass Enterprise Manager den Replikationsmonitor zur Konsolenstruktur hinzugefügt hat. Standardmäßig verwendet jede Publikation einen separaten Publikations-Agent.

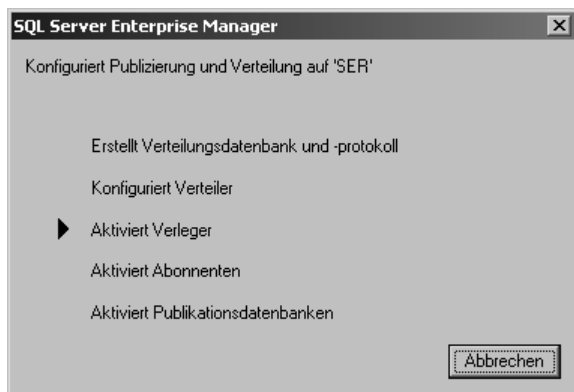


Abbildung 17.9:
Das Fortschrittsdialogfeld des Publizierungs- und Verteilungskonfigurations-Assistenten

Beachten Sie, dass jetzt ein Symbol für den Replikationsmonitor in der Konsolenstruktur des Enterprise Managers zu sehen ist. Dieser Eintrag erscheint erst dann, wenn auf dem betreffenden Server die Replikation konfiguriert ist.

Mit diesen Schritten haben Sie einen Verteilungsserver erfolgreich installiert. Dazu eine kurze Zusammenfassung:

- Sie haben die Verteilungsdatenbank auf Ihrem Server installiert.
- Sie haben festgelegt, welche Server auf diesen Server (Ihren Verteiler) als Remoteverteiler zugreifen können.
- Sie haben konfiguriert, welche Server Daten publizieren können.
- Sie haben Abonnenten aktiviert.
- Sie haben den Verteilerzeitplan für Ihre Abonnenten unter Verwendung der Pushreplikation mit der Methode der Transaktionsreplikation überprüft.
- Sie haben sichergestellt, dass Ihr Verleger genügend Speicher und Festplattenplatz besitzt. Eventuell benötigen Sie zusätzlichen Plattenplatz, falls Ihr Verteilungsserver nicht verfügbar sein sollte.

Erstellen einer Publikation

In diesem Abschnitt erstellen Sie eine Publikation, bestimmen die Snapshotanforderungen, legen fest, ob anonyme oder aktualisierende Abonnenten oder Pullabonnements erlaubt sind, und definieren Partitionen für Ihre Daten.

Publikationen können Sie nach verschiedenen Verfahren erstellen: mit Assistenten, über die Konsolenstruktur im Enterprise Manager oder über Menübefehle.

Der Publikationserstellungs-Assistent erstellt Publikationen und legt fest

- ob ein oder mehrere Artikel verfügbar sind,
- ob der Snapshot-Agent nach Zeitplan ausgeführt wird,
- ob ein Snapshot auf dem Verteilungsserver zu verwalten ist,
- welche Tabellen zu veröffentlichen sind,
- welche gespeicherten Prozeduren zu veröffentlichen sind,
- welche Publikationen Agents gemeinsam nutzen,
- ob anonyme Aktualisierungen von Abonnements erlaubt sind,
- ob Aktualisierungen von Abonnenten erlaubt sind,
- ob Pullabonnements erlaubt sind.

Im folgenden Beispiel erstellen Sie eine Publikation, die die Tabelle *authors* der Datenbank *pubs* verwendet. Dabei installieren und nutzen Sie die Transaktionsreplikation, um diese Tabelle auf eine neue Tabelle namens *rplAuthors* zu replizieren. Dann nehmen Sie Änderungen an der Originaltabelle vor und überprüfen, ob die Daten erfolgreich in die Tabelle *rplAuthors* repliziert wurden. Führen Sie dazu die folgenden Schritte aus:

1. Verbinden Sie sich mit Ihrem Server im Enterprise Manager. Markieren Sie Ihren Server und wählen Sie EXTRAS / REPLIKATION / PUBLIKATIONEN ERSTELLEN UND VERWALTEN. Daraufhin öffnet sich das Dialogfeld PUBLIKATIONEN AUF <SERVERNAME> ERSTELLEN UND VERWALTEN (siehe Abbildung 17.10). Klicken Sie auf die Schaltfläche PUBLIKATION ERSTELLEN und dann im Willkommen-Bildschirm auf WEITER. Damit gelangen Sie zu einem Dialogfeld entsprechend Abbildung 17.11, das die Datenbanken *Northwind*, *pubs* und alle von Ihnen erstellten Datenbanken anzeigt.
2. Markieren Sie die Datenbank *pubs* und klicken Sie auf WEITER.
3. Im Dialogfeld PUBLIKATIONSTYP AUSWÄHLEN (siehe Abbildung 17.12) stehen die Optionen *Snapshotpublikation*, *Transaktionspublikation* und *Mergepublikation* zur Wahl. Klicken Sie für das Beispiel auf *Transaktionspublikation* und dann auf WEITER.



In der Personal Edition von SQL Server können Sie keine Transaktionspublikation wählen, weil diese Edition nur für das Abonnieren und nicht für das Veröffentlichen von Publikation zugelassen ist.

4. Als Nächstes legen Sie fest, welche Typen von Abonnenten zu unterstützen sind (siehe Abbildung 17.13). Die Auswahl umfasst SQL Server 2000, SQL Server 7.0 und andere Produkte als SQL Server. Übernehmen Sie die Standardeinstellung SERVER, AUF DENEN SQL SERVER 2000 AUSGEFÜHRT WIRD und klicken Sie auf WEITER.



Abbildung 17.10:
Das erste Dialogfeld des Publikationserstellungs-Assistenten



Abbildung 17.11:
Hier wählen Sie eine Datenbank für die Publikation aus.

- Im nächsten Dialogfeld können Sie Artikel festlegen (siehe Abbildung 17.14). Auf der rechten Seite des Dialogfelds können Sie sowohl publizierte als auch nicht publizierte Artikel oder lediglich die publizierten Artikel auflisten. Außerdem können Sie Sichten und gespeicherte Prozeduren publizieren, indem Sie die betreffenden Kontrollkästchen im linken Teil des Dialogfelds aktivieren. Schalten Sie für dieses Beispiel das Kontrollkästchen für *dbo.authors* ein und klicken Sie dann auf WEITER.



Abbildung 17.12:
In diesem Dialog-
feld legen Sie den
Publikationstyp fest.



Abbildung 17.13:
Hier legen Sie einen
Abonententyp fest.

6. Geben Sie einen Publikationsnamen und gegebenenfalls eine Beschreibung ein (siehe Abbildung 17.15). Ändern Sie für das Beispiel den Namen der Publikation in pubs_authors. Klicken Sie dann auf WEITER.

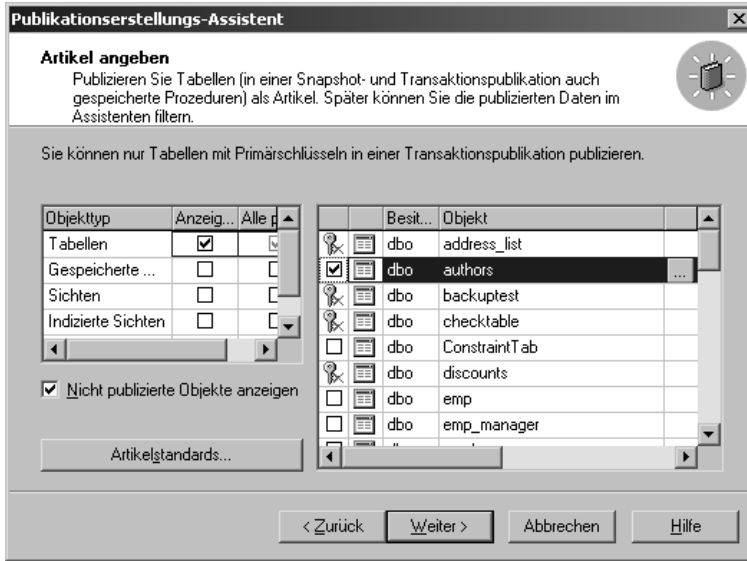


Abbildung 17.14:
In diesem Dialogfeld wählen Sie die zu publizierenden Artikel aus.

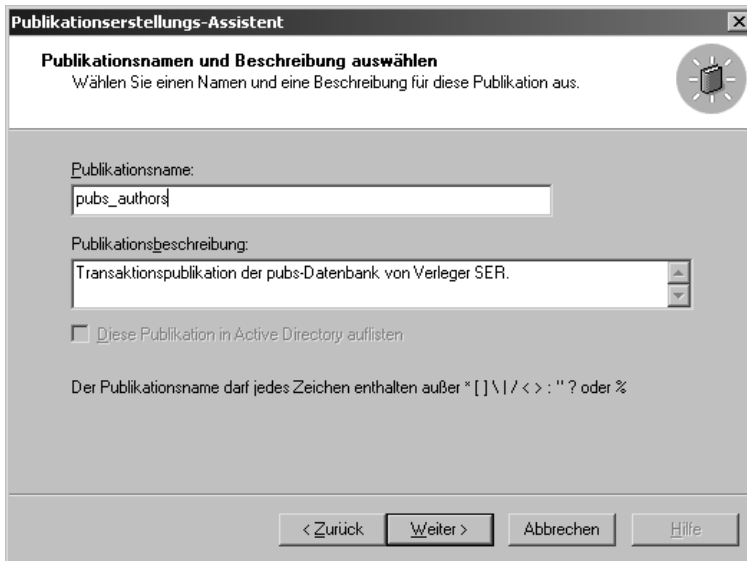


Abbildung 17.15:
Geben Sie einen Namen und eine Beschreibung für die Publikation ein.

- Im Dialogfeld **STANDARDEIGENSCHAFTEN DER PUBLIKATION ANPASSEN** (siehe Abbildung 17.16) können Sie Filter (Partitionen) für Ihre Daten festlegen oder die Publikation ohne Filter und mit den im Kasten angegebenen Eigenschaften erstellen. Wählen Sie für das Beispiel die Option *Ja, Datenfilter definieren, anonyme Abonnenten aktivieren oder andere Eigenschaften anpassen*. Klicken Sie dann auf **WEITER**.

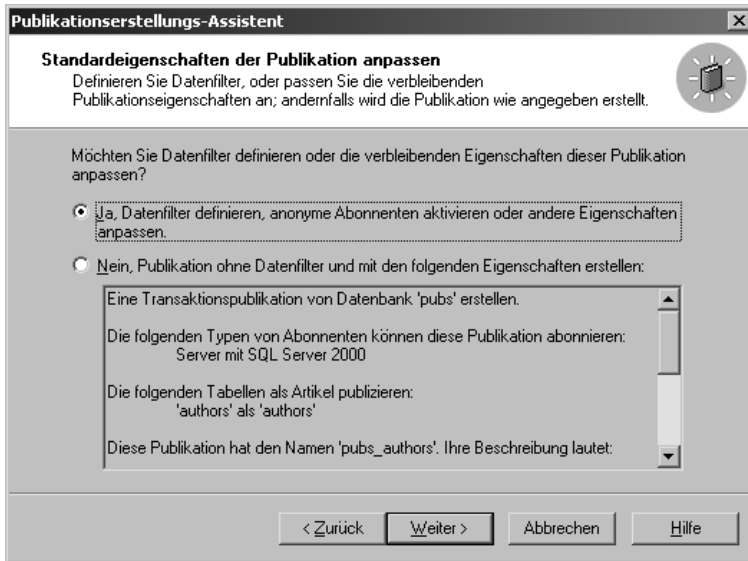


Abbildung 17.16:
Das Dialogfeld
Standardeigenschaften
der Publikation
anpassen

8. Als Nächstes erscheint die Frage, wie diese Publikation gefiltert werden soll. Wie die gestrige Lektion erläutert hat, filtern oder partitionieren Sie, wenn Sie mit `SELECT`-Anweisungen nur bestimmte Spalten (vertikale Partition) oder nur bestimmte Zeilen (horizontale Partition) zusammentragen. Lassen Sie für das Beispiel beide Kontrollkästchen ausgeschaltet und klicken Sie auf WEITER.
9. Im nächsten Dialogfeld können Sie festlegen, ob Sie anonyme Abonnements zulassen möchten. Wählen Sie die Option *Nein* und klicken Sie auf WEITER. Anonyme Abonnements werden in speziellen Pullabonnement-Situationen benutzt. Normalerweise sind die Angaben zu jedem Abonnenten beim Verleger gespeichert und die Leistungsinformationen zu den Abonnenten werden auf dem Verteiler abgelegt. Diesen Mehraufwand können Sie vermeiden, indem Sie anonyme Abonnements zulassen. Jeder Abonnent hält die Leistungsinformationen in seiner lokalen Abonentendatenbank. Anonyme Abonnements sind nützlich, wenn Ihre Publikation eine große Zahl von Abonnenten hat oder die Abonnenten über das Internet Zugang haben.
10. Als Nächstes erscheint das Dialogfeld ZEITPLAN FÜR SNAPSHOT-AGENT FESTLEGEN (siehe Abbildung 17.17). Hier ist der Standardzeitplan angegeben. Wenn Sie auf ÄNDERN klicken, erscheint ein Zeitplanungsdialogfeld, das Sie bereits aus anderen Lektionen dieses Buches kennen. Klicken Sie auf OK, um zum Dialogfeld ZEITPLAN FÜR SNAPSHOT-AGENT FESTLEGEN zurückzukehren und klicken Sie dort auf WEITER.
11. Damit haben Sie das abschließende Dialogfeld des Publikationserstellungs-Assistenten erreicht. Klicken Sie auf FERTIG STELLEN. Es erscheint nun eine Liste der Aufgaben, die SQL Server fertig stellt, und schließlich erhalten Sie eine Erfolgsmeldung.

12. Nachdem die Publikation erstellt ist, haben Sie die Gelegenheit die Eigenschaften der Publikation festzulegen (siehe Abbildung 17.18). Klicken Sie dazu auf die Schaltfläche PUBLIKATIONSEIGENSCHAFTEN.



Abbildung 17.17:
Das Dialogfeld Zeitplan für Snapshot-Agent festlegen

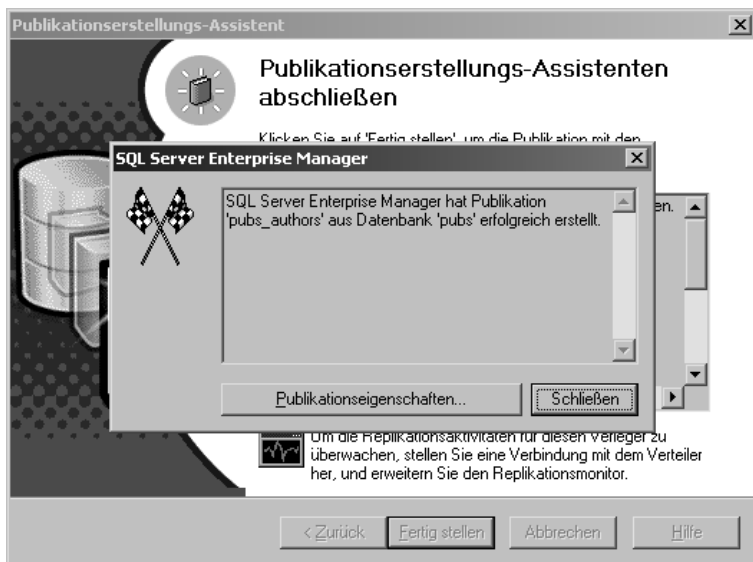


Abbildung 17.18:
Klicken Sie auf Publikationseigenschaften, um die Eigenschaften der Publikation festzulegen.

13. Klicken Sie auf die Registerkarte PUBLIKATIONSZUGRIFFSLISTE (siehe Abbildung 17.19). Hier lässt sich die Liste der Benutzer ändern, die Zugriff auf Ihre Publikation haben. Sie können die Liste anpassen oder die automatisch erstellte Liste übernehmen. Klicken Sie auf OK und dann auf SCHLIESSEN, um zum Enterprise Manager zurückzukehren.

Im Enterprise Manager können Sie sich jetzt davon überzeugen, dass die neu erstellte Publikation *pubs_authors* im Ordner *Replikation\Publikationen* aufgeführt ist. Vom Enterprise Manager aus können Sie ein neues Pushabonnement oder zusätzliche Publikation erstellen, die Eigenschaften des Verteilers und der Abonnements anzeigen oder Publikationen löschen. Mit einigen dieser Punkte haben Sie noch später in dieser Lektion zu tun.

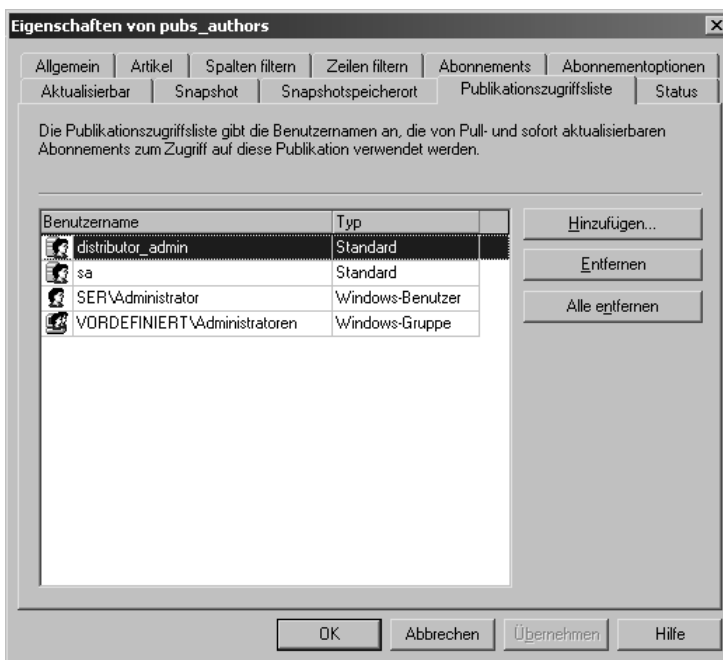


Abbildung 17.19:
Hier können Sie die
Liste der Benutzer mit
Zugriff auf Ihre Publi-
kation anpassen.

Erstellen eines Abonnements

In diesem Abschnitt erstellen Sie ein Abonnement, um die Informationen zu sammeln, die Sie eben publiziert haben. Wenn Sie das Abonnieren konfigurieren, können Sie folgende Spezifizierungen vornehmen:

- Den Verleger wählen, von welchem Sie abonnieren wollen.
- Die Zieldatenbank festlegen, die die replizierten Daten erhalten soll.

- Überprüfen, ob Sie ein gültiges Konto für den Zugriff auf den Verteilungsserver haben.
- Eine Beschreibung Ihres Abonnenten festlegen.
- Einen Standardzeitplan einrichten.
- Einen Sicherheitsmodus wählen.

Führen Sie die folgenden Schritte aus, um Ihren Abonnenten zu aktivieren und den Replikationsprozess abzuschließen:

1. In diesem Beispiel erstellen Sie ein Pullabonnement. Stellen Sie also im Enterprise Manager eine Verbindung zu Ihrem Server her. Markieren Sie Ihren Server und wählen Sie EXTRAS / REPLIKATION / PULLABONNEMENT FÜR 'SERVERNAME'. Daraufhin erscheint ein Dialogfeld, wie es Abbildung 17.20 zeigt.
2. Klicken Sie auf die Schaltfläche NEUES PULLABONNEMENT. Es erscheint das übliche Willkommen-Fenster. Klicken Sie auf WEITER.
3. Das nächste Dialogfeld fragt Sie, ob nach Publikationen gesucht werden soll. Übernehmen Sie die Standardeinstellung *Publikationen von registrierten Servern anzeigen* und klicken Sie auf WEITER.
4. Im Dialogfeld Publikation auswählen sollte Ihr Server zu sehen sein. Klicken Sie auf das Pluszeichen, um den Serverknoten zu erweitern und die verfügbaren Publikationen einzublenden (siehe Abbildung 17.21). Markieren Sie die Publikation *pubs_authors:pubs* und klicken Sie dann auf WEITER.



Abbildung 17.20:
Das Dialogfeld Pul-
labonnement auf
<Servername> (im
Beispiel SER)

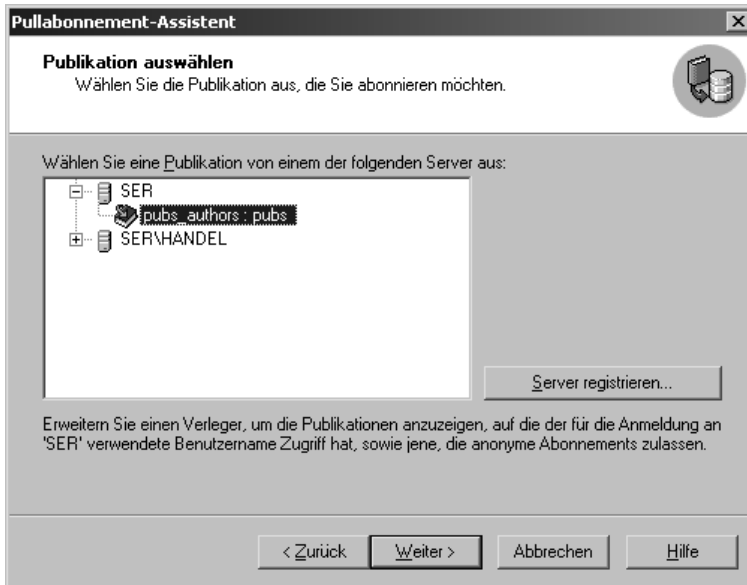


Abbildung 17.21:
Wählen Sie die
gewünschte Publika-
tion aus.

5. Als Nächstes sollte das Dialogfeld BENUTZERNAME FÜR SYNCHRONISIERUNGS-AGENT ANGEBEN erscheinen. Ist das der Fall, klicken Sie auf WEITER. Andernfalls gelangen Sie direkt zum Dialogfeld ZIELDATENBANK AUSWÄHLEN (siehe Abbildung 17.22). Da Sie auf einem einzelnen Server arbeiten, erstellen Sie eine neue Datenbank. Klicken Sie auf die Schaltfläche NEU. Geben Sie der Datenbank den Namen `rp1Pubs`. Klicken Sie im Dialogfeld DATENBANKEIGENSCHAFTEN auf OK, um für die übrigen Einstellungen die Standardwerte zu übernehmen und die Datenbank anzulegen. Wenn Sie sich wieder im Dialogfeld ZIELDATENBANK AUSWÄHLEN befinden, markieren Sie die Datenbank `rp1Pubs` und klicken dann auf WEITER.
6. Als Nächstes müssen Sie festlegen, wie die anfängliche Synchronisierung ablaufen soll (siehe Abbildung 17.23). Übernehmen Sie die Standardoption *Ja, Schema und Daten auf dem Abonnenten sofort initialisieren* und klicken Sie auf WEITER.
7. Im nächsten Dialogfeld legen Sie fest, wie die Snapshotübermittlung stattfinden soll. Übernehmen Sie die Standardoption *Snapshotdateien aus dem standardmäßigen Snapshotordner für diese Publikation verwenden* und klicken Sie auf WEITER.
8. Im nächsten Dialogfeld legen Sie fest, wie häufig der Verteilungs-Agent das Abonnement aktualisiert (siehe Abbildung 17.24). Das kann fortlaufend, nach einem festgelegten Zeitplan oder manuell bei Bedarf geschehen. Übernehmen Sie die Standardoption *Fortlaufend* und klicken Sie auf WEITER.



Abbildung 17.22:
Markieren Sie die
neu angelegte
Datenbank.



Abbildung 17.23:
Das Dialogfeld
Abonnement initia-
lisieren

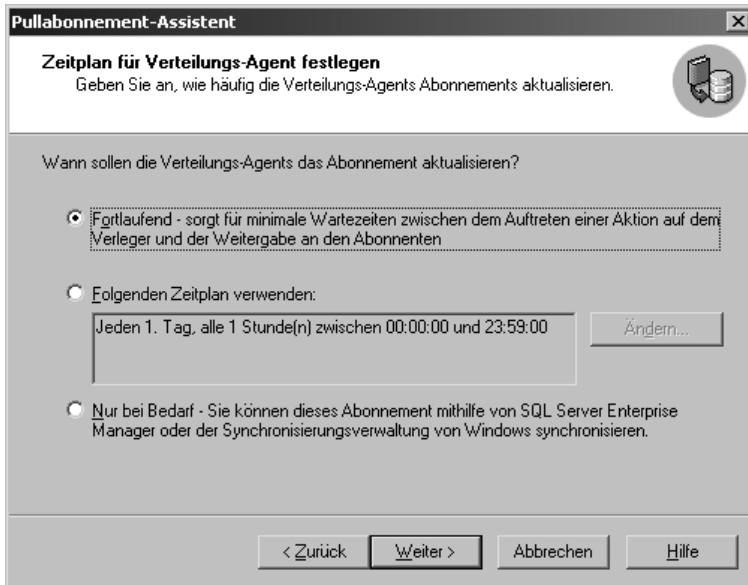


Abbildung 17.24:
Legen Sie fest, wie
häufig der Vertei-
lungs-Agent das
Abonnement aktua-
lisiert.

9. Sie gelangen nun zum Dialogfeld **ERFORDERLICHE DIENSTE STARTEN** (siehe Abbildung 17.25). Wenn der Dienst nicht gestartet ist, schalten Sie das Kontrollkästchen neben dem Dienstenamen ein, um den Dienst automatisch zu starten, nachdem das Abonnement erstellt ist. Vergewissern Sie sich, dass das Kontrollkästchen neben dem Dienst *SQLServerAgent* für Ihren Computer eingeschaltet ist. Klicken Sie dann auf **WEITER**.
10. Damit haben Sie das Dialogfeld **PULLABONNEMENT-ASSISTENTEN ABSCHLIESSEN** erreicht. Klicken Sie auf **FERTIG STELLEN**, um die Installation abzuschließen. Es erscheint auch hier eine Fortschrittsanzeige und danach eine Meldung über die erfolgreich verlaufene Installation. Klicken Sie auf **OK**, um das Dialogfeld zu schließen und den Vorgang zu beenden.
11. Jetzt sollten Sie sich wieder im Dialogfeld **PULLABONNEMENT AUF <SERVERNAME>** befinden. Wie Abbildung 17.26 zeigt, hat sich das Abonnement *pubs:pubs_authors* geändert. Die Abbonnementeigenschaften können Sie sich nun ansehen, indem Sie auf die Schaltfläche **EIGENSCHAFTEN** klicken.
12. Abbildung 17.27 zeigt die Registerkarte **ALLGEMEIN** des Dialogfelds **EIGENSCHAFTEN VON PULLABONNEMENT**. Hier finden Sie Publikationsinformationen (Verleger, Verlegerdatenbank und Beschreibung) und Abonnementinformationen (ob anonyme oder aktualisierbare Abonnements zugelassen sind). Auf der Registerkarte **SICHERHEIT** können Sie die Verteileranmeldung modifizieren.

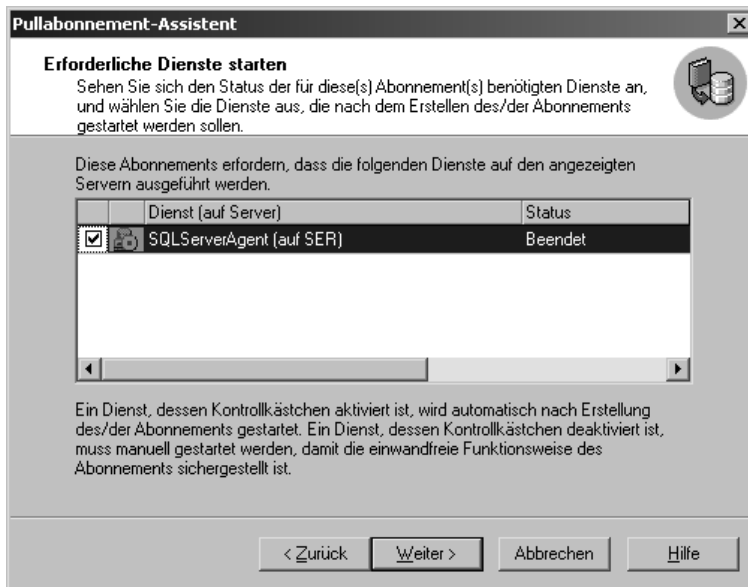


Abbildung 17.25:
Aktivieren Sie die
Dienste, die nach
dem Erstellen des
Abonnements auto-
matisch gestartet
werden sollen.

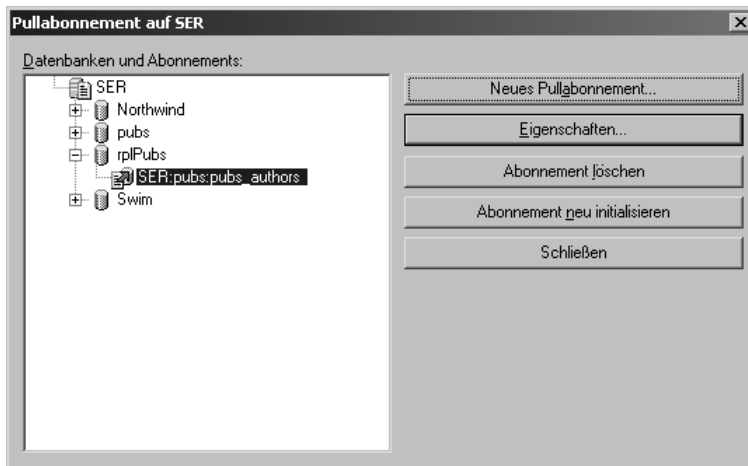


Abbildung 17.26:
Das Dialogfeld Pul-
labonnement mit
dem geänderten
Abonnement

13. Klicken Sie auf OK, um das Dialogfeld EIGENSCHAFTEN VON PULLABONNEMENT zu verlassen, und dann auf SCHLIESSEN, um das Dialogfeld PULLABONNEMENT zu schließen und zum Enterprise Manager zurückzukehren.



Abbildung 17.27:
Die Registerkarte Allgemein des
Dialogfelds Eigenschaften von Pul-
labonnement

Manuelle Synchronisation

Der Snapshot-Agent wird zu dem Zeitpunkt ausgeführt, den Sie spezifiziert haben. Damit beginnt der Replikationsprozess. Eine Synchronisierung können Sie auch erzwingen. Führen Sie die folgenden Schritte aus, um die anfängliche Situation durch Starten des Snapshot-Agents einzuleiten:

1. Öffnen Sie im Enterprise Manager den Knoten *Replikationsmonitor*, dann *Verleger* und danach Ihren Server. Markieren Sie den Eintrag *pubs_authors:pubs* (siehe Abbildung 17.28). Abhängig von Ihrer derzeitigen Konfiguration erscheint beim erstmaligen Zugriff auf den Replikationsmonitor eventuell die Frage, ob der Enterprise Manager den Replikationsmonitor automatisch aktualisieren soll. Wenn Sie die Option *Ja* wählen, können Sie die Aktualisierungsrate festlegen.
2. Klicken Sie mit der rechten Maustaste im Detailbereich auf das Snapshot-Symbol und wählen Sie *AGENT STARTEN*. Für kurze Zeit erscheint in der Statusspalte die Meldung *Wird ausgeführt*, dann wechselt die Anzeige zu *Erfolgreich beendet*. Das Symbol des Snapshot-Agents zeigt wieder das Stoppzeichen, nachdem der Agent den Snapshot übertragen hat.

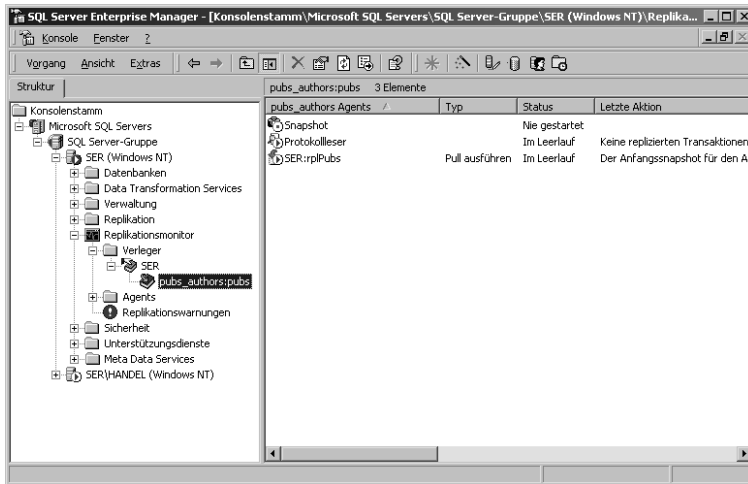


Abbildung 17.28:
Im Detailbereich
ist der Status des
Snapshot-Agents
zu sehen.

- Wenn Sie mit der rechten Maustaste auf den Snapshot-Agent klicken, nachdem er erfolgreich ausgeführt wurde, können Sie sich den Agentverlauf ansehen. Machen Sie sich kurz mit dem Dialogfeld SNAPSHOT-AGENT – VERLAUF bekannt.

Prüfen der Anfangssynchronisation

Ob der Snapshot-Agent seine Aufgaben erfüllt – d.h. das Datenbankschema und die Daten kopiert – hat, können Sie nachprüfen, indem Sie mithilfe von Abfragen die Daten in der neuen Tabelle anzeigen. Führen Sie dazu folgende Abfrage im Query Analyzer aus:



```
USE rp1Pubs
GO
SELECT * FROM authors
GO
```

Wenn Sie Daten zurückbekommen, wissen Sie, dass die Anfangssynchronisation korrekt gelaufen ist.

Test der Replikation

Jetzt können Sie über einen Test feststellen, ob die Replikation tatsächlich funktioniert. Dazu führen Sie mehrere Abfragen der Datenbanken *pubs* und *rp1pubs* aus.

Die erste Abfrage stellt fest, wie viele Zeilen in der Tabelle *authors* enthalten sind:



```
SELECT COUNT(*) FROM pubs..authors
GO
```



```
-----
23
(1 row(s) affected)
```

Vergleichen Sie das Ergebnis mit der Anzahl der Zeilen in der anderen Tabelle, indem Sie die Abfrage leicht modifizieren und erneut ausführen:



```
SELECT COUNT(*) FROM replPubs..authors
GO
```



```
-----
23
(1 rows(s) affected)
```

Fügen Sie nun der Tabelle *authors* eine Zeile hinzu:



```
USE pubs
GO
INSERT INTO authors VALUES
('555-55-5555', 'Johnson', 'Susan', '555 555-5555',
'123 Anystreet', 'Fresno', 'CA', '90000', '1')
GO
```



```
(1 row(s) affected)
```

Führen Sie jetzt die `SELECT COUNT`-Abfrage erneut für beide Datenbanken aus und überzeugen Sie sich davon, dass sowohl in der Tabelle *pubs..authors* als auch in der Tabelle *replPubs..authors* genau 24 Zeilen enthalten sind.

Noch eine weitere Kontrollmöglichkeit: Führen Sie die folgende Abfrage aus, um festzustellen, dass die Daten tatsächlich übertragen wurden:



```
SELECT * FROM rplPubs..authors
WHERE au_lname = 'Johnson'
GO
```

Sie sollten die oben angegebene Datenzeile zurückerhalten. Löschen Sie nun diesen Eintrag auf der Verlegerseite und überzeugen Sie sich, dass der gelöschte Datensatz repliziert wird. Starten Sie die folgende Abfrage, um den Eintrag von Susan Johnson aus der Tabelle *pubs..authors* zu löschen:



```
DELETE FROM pubs..authors
WHERE au_lname = 'Johnson'
GO
```



(1 row(s) affected)

Sie können nochmals die `SELECT COUNT`-Abfragen starten, um sich zu vergewissern, dass nun wieder 23 Datenzeilen sowohl in der Publikationsdatenbank (*pubs*) als auch in der Abonentendatenbank (*rplPubs*) vorhanden sind. Schließen Sie dann das Abfragefenster.

17.4 Überlegungen zur Replikation

Es gibt unterschiedliche Dinge zu beachten, wenn Sie mit Replikation in SQL Server arbeiten. Dazu gehören Publikationseinschränkungen, Überlegungen in Bezug auf aktualisierende Abonnenten und auch Leistungsbetrachtungen. Die folgenden Abschnitte gehen näher auf diese Punkte ein.

Probleme beim Publizieren

Bei der Mergereplikation müssen Sie gewährleisten, dass jede Datenzeile, die in den Replikationsprozess einbezogen ist, einen eindeutigen Bezeichner enthält. Das lässt sich mit der `uniqueidentifizier`-Einschränkung und der `ROWGUIDCOL`-Eigenschaft realisieren. Wenn SQL Server eine Spalte mit dieser Einschränkung bzw. Eigenschaft findet, verwendet er automatisch diese Spalte als Zeilenbezeichner für die replizierte Tabelle. Ist keine derartige Zeile vorhanden, erstellt sie SQL Server automatisch, wenn Sie den Assistenten für Mergereplikation verwenden.

Beachten Sie auch die folgenden Beschränkungen:

- Die Systemdatenbanken – *master*, *model*, *tempdb*, *msdb* und *distribution* (Verteilerdatenbanken) – können nicht repliziert werden.
- Alle an der Replikation beteiligten Tabellen oder Teile von Tabellen müssen einen Primärschlüssel besitzen, um die Entitätsintegrität sicherzustellen. Einzige Ausnahme ist die Snapshotreplikation, bei welcher der gesamte Inhalt als einzelne Einheit kopiert wird.
- Publikationen können sich nicht über mehrere Datenbanken erstrecken. Wenn Sie eine Publikation erstellen, müssen alle Daten von einer einzigen Datenbank kommen.

Beachten Sie diese Einschränkungen, wenn Sie Daten replizieren, die in nicht protokollierten Operationen verwendet werden. Das betrifft die Datentypen *text*, *ntext* und *image*. Die folgende Zusammenfassung beschreibt die drei Hauptmethoden der Replikation und wie sie von diesen Datentypen beeinflusst werden:

- Bei Mergereplikation werden die Veränderungen nicht erkannt und die Daten nicht repliziert.
- Transaktionsreplikation verwendet den Protokollleser, um Änderungen zu erkennen. Da es sich um nicht protokollierte Operationen handelt, kann der Protokollleser weder die Operationen erkennen noch die Daten replizieren.
- Weil die Snapshotreplikation den kompletten Inhalt des Artikels kopiert, werden auch die nicht protokollierten Werte repliziert.



Wenn Sie in Ihren Replikationsszenarios die Datentypen *text*, *ntext* oder *image* vorgesehen haben, sollten Sie entweder die Snapshotreplikation einsetzen oder den Snapshot-Agent regelmäßig ausführen, um diese Werte zu den Abonnenten zu übertragen.

Probleme beim Abonnieren

Wenn Sie die Option *Sofortiges Aktualisieren* der Abonnenten beim Erstellen einer Publikation verwenden, legen Sie fest, dass der Abonnent sowohl die lokale Kopie der Daten als auch die Verlegerkopie der Daten aktualisiert. Sobald diese Änderung wirksam ist, werden die Daten zu anderen Abonnenten bei der nächsten geplanten Synchronisierung repliziert. (Bei Mergereplikation ist das nicht möglich.)

In SQL Server 7.0 konnten Sie eine *timestamp*-Spalte (Zeitstempel) in Ihrem Artikel mitführen, um mögliche Konflikte zu vermeiden. SQL Server 2000 greift jedoch zur Konflikterkennung nicht mehr auf *timestamp*-Daten zurück. Bei Mergereplikation und Abonnenten mit Aktualisierung über eine Warteschlange wird zwar die *timestamp*-Spalte, nicht jedoch ihr literaler Wert repliziert. Clientanwendungen können Zeitstempel beim

Abonnenten verwenden, um zum Beispiel Sperren zu realisieren. Auf diese Weise lässt sich immer herausfinden, welcher Eintrag zuerst geändert wurde, so dass eine Anwendung die passende Aktion durchführen kann.

Leistungsprobleme

Die folgenden Vorschläge sollen Ihnen helfen, die zu implementierenden Methoden der Replikation hinsichtlich der Leistung zu optimieren:

- Benutzen Sie Pullabonnements, um Ihren Verteiler zu entlasten. Obwohl diese Option nicht so einfach zu verwalten ist, kann sie einen wesentlichen Einfluss auf die Leistungsfähigkeit des Verteilungsservers haben.
- Verwenden Sie einen Remoteverteiler, der mehrere Verleger und mehrere Abonnenten unterstützen kann. Indem Sie die Verteilungsaufgaben auf einen anderen SQL Server verschieben, können Sie den Einfluss der Verteilung auf Ihre OLTP-Server verringern.
- Replizieren Sie nur die erforderlichen Daten. Indem Sie die Daten partitionieren (filtern), können Sie die zu replizierenden Informationen minimieren und somit den Einfluss der Replikation auf Ihre SQL Server verringern.
- Denken Sie daran, dass die meisten Typen der Replikation den Netzverkehr steigern. Stellen Sie sicher, dass Ihr Netzwerk die entsprechende Bandbreite zulässt. SQL Server-Computer, die hochgradig mit Replikation belastet sind, sollte man in einem separaten physischen Teilnetz unterbringen.
- Verwenden Sie Primärschlüssel für alle Tabellen, die in die Replikation einbezogen sind, um relationale und Entitätsintegrität zu gewährleisten.
- Verwenden Sie dasselbe Domänenbenutzerkonto (SQL Server-Agent) für alle SQL Server, die in die Replikation einbezogen sind. Alle Server können zur Domäne gehören oder in einer Vertrauensbeziehung zur Domäne stehen und deshalb dasselbe SQL Server-Agent-Konto nutzen. Das erspart Ihnen eine Menge Probleme, wenn Sie die SQL Server-Sicherheit verwalten müssen.

17.5 Überwachen der Replikation

In den folgenden Abschnitten dringen Sie tiefer in die Überwachung und Verwaltung der Replikation ein. Mit dem Replikationsmonitor können Sie Ihre Verleger, Abonnenten und Publikationen verwalten. Sie können Agent-Profile überprüfen und deren Eigenschaften modifizieren. Außerdem können Sie Verlaufsinformationen abrufen und Replikationswarnungen erstellen.

Zur Replikationsverwaltung gehört es, die Größe der Verteilungsdatenbank zu überwachen sowie die Cleanup-Aufträge zu verfolgen, um ihre Ausführung zu den richtigen Zeitpunkten sicherzustellen. Natürlich sollten Sie auch auf mögliche Ausfälle der Replikationsserver vorbereitet sein. Mit einer durchdachten und konsequent angewandten Sicherungsstrategie für die Verteilerdatenbank können Sie garantieren, dass die Wiederherstellung und erneute Synchronisierung glatt verläuft.

Über den Enterprise Manager ist es möglich, mehrere Replikationsserver zu verwalten. Mit Replikationsskripts können Sie die Konfiguration der Replikation für mehrere Abonnenten vereinfachen.

Der Replikationsmonitor

Der Replikationsmonitor ist nur auf dem SQL Server verfügbar, der als Verteilungsserver arbeitet. Er wird benutzt, um Statusinformationen über die Verleger, Abonnenten und Publikationen abzufragen. Sie können ebenfalls geplante Replikations-Agents und deren Verlauf überprüfen. Dazu gehören Informationen über Einfügungen, Aktualisierungen und Transaktionen. Der Replikationsmonitor ermöglicht Ihnen ebenfalls das Konfigurieren der Agent-Profile und -Eigenschaften.

Um Informationen über die Replikations-Agents zu erhalten, führen Sie folgende Schritte aus:

1. Öffnen Sie den Enterprise-Manager auf dem SQL Server, der als Verteilungsserver dient (für die Beispiele in diesem Buch ist das derselbe Server wie der Verleger).
2. Stellen Sie die Verbindung zu Ihrem Server her und gehen Sie dann in der Konsolenstruktur über den Knoten *Replikationsmonitor* zum Knoten *Agents* und markieren Sie den Ordner *Snapshot-Agents* (siehe Abbildung 17.29).
3. Klicken Sie mit der rechten Maustaste im Detailbereich auf den Eintrag *pubs_authors* und wählen Sie AGENTVERLAUF, um das Dialogfeld SNAPSHOT-AGENT – VERLAUF zu öffnen. Die Liste lässt sich für die folgenden Sitzungen filtern:
 - ▶ Sitzungen mit Fehlern
 - ▶ Alle Sitzungen
 - ▶ Sitzungen der letzten 24 Stunden
 - ▶ Sitzungen der letzten 2 Tage
 - ▶ Sitzungen der letzten 7 Tage
 - ▶ Die letzten 100 Sitzungen

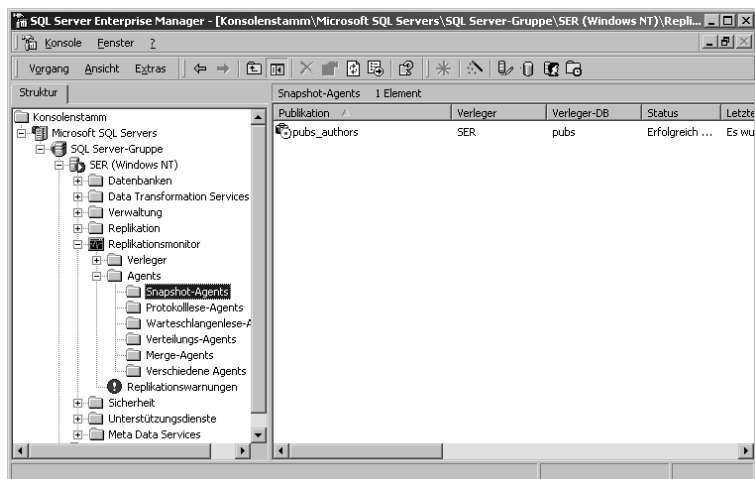


Abbildung 17.29:
Der Ordner Agents
im Knoten Replika-
tionsmonitor

- Wenn Sie die Sitzungen der letzten 100 Tage anzeigen, kann lediglich eine Liste von Versuchen erscheinen, aber es ist kein neues Abonnement verfügbar. Wenn Sie die Liste bis zum Ende durchblättern, finden Sie mindestens einen Eintrag mit der Sitzungsmeldung *Es wurde ein Snapshot mit 1 Artikel(n) generiert*. Damit haben Sie einen Nachweis für die stattgefundene Anfangssynchronisierung (siehe Abbildung 17.30).

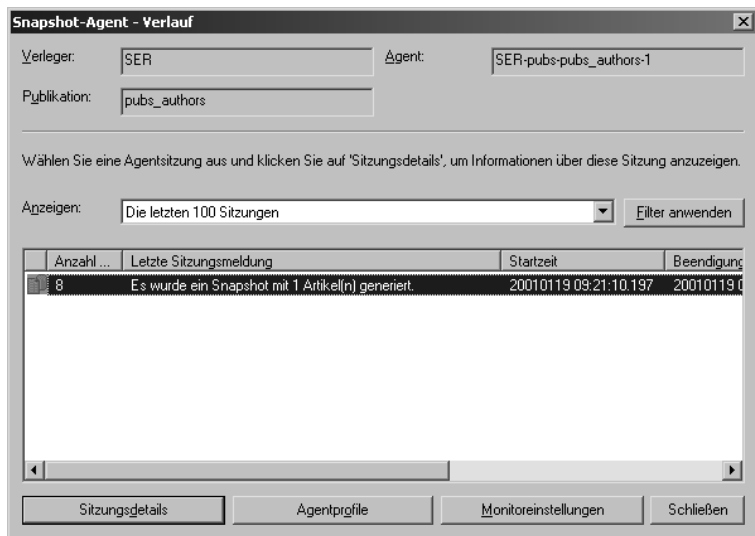


Abbildung 17.30:
Das Dialogfeld
Snapshot-Agent

- Markieren Sie diesen Eintrag in der Liste und klicken Sie auf die Schaltfläche SITZUNGSDETAILS, um zusätzliche Informationen für diesen Snapshot-Vorgang anzuzeigen (siehe Abbildung 17.31).

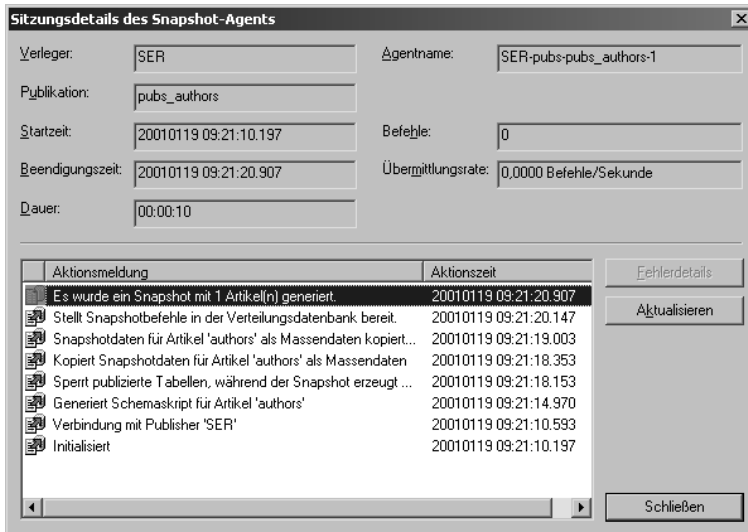


Abbildung 17.31:
Sitzungsdetails für
die Anfangssyn-
chronisierung

- Klicken Sie im Dialogfeld SITZUNGSDETAILS und dann auch im Dialogfeld SNAPSHOT-AGENT – VERLAUF auf SCHLIESSEN, um die Verlaufsanzeige zu beenden.
- Klicken Sie im Detailbereich erneut auf den Eintrag *pubs_authors* und wählen Sie dieses Mal den Befehl AGENTEIGENSCHAFTEN aus dem Kontextmenü. Daraufhin erscheint das Eigenschaftsfenster für Aufträge an diesen Agent (siehe Abbildung 17.32).

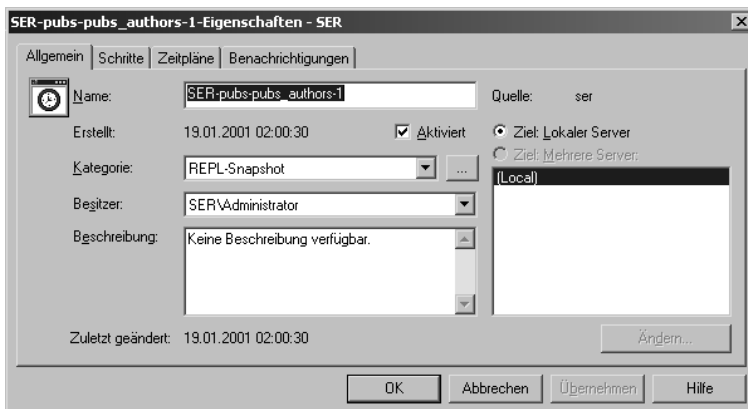


Abbildung 17.32:
Die Registerkarte
Allgemein des Auf-
tragsdialogfeldes für
diesen Replikations-
Agent

8. Klicken Sie auf die Registerkarte SCHRITTE. Hier erscheinen die Schritte, die an der Ausführung des betreffenden Auftrags beteiligt sind. Wie Abbildung 17.33 zeigt, hat der Auftrag des Snapshot-Agents drei Schritte: Protokollieren der Startmeldung, Ausführen des Replikations-Agents und Ermitteln des nicht protokollierten Herunterfahrens. Außerdem ist zu erkennen, ob jeder Auftrag erfolgreich gelaufen (grünes Häkchen) oder gescheitert (rotes X) ist. Sie können die Schritte umstellen, löschen oder zusätzliche Schritte einfügen.

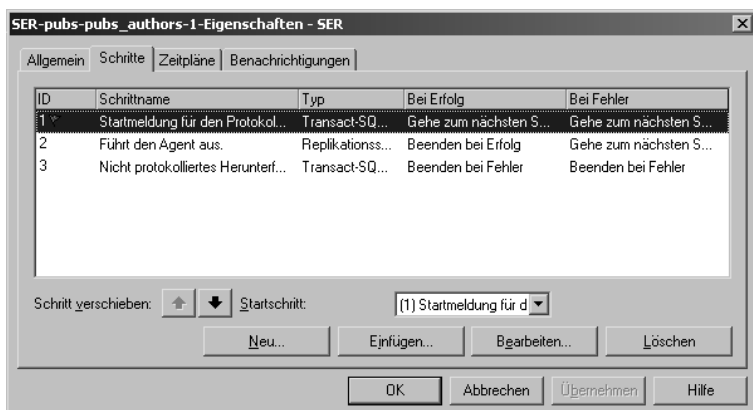


Abbildung 17.33:
Die Registerkarte
Schritte der Auftrag-
eigenschaften

9. Gehen Sie jetzt auf die Registerkarte ZEITPLÄNE, um die Zeitplanungsinformationen für die Replikations-Agents anzuzeigen (siehe Abbildung 17.34). Hier können Sie die Aufträge bearbeiten und löschen. Außerdem lassen sich neue Zeitpläne und Warnungen erstellen.

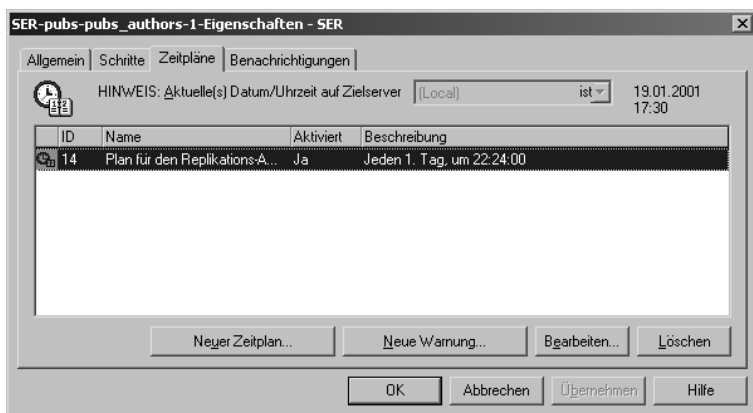


Abbildung 17.34:
Die Registerkarte
Zeitpläne der Auf-
tragseigenschaften

10. Klicken Sie auf die Registerkarte BENACHRICHTIGUNGEN (siehe Abbildung 17.35). Von hier aus können Sie E-Mail-, Pager- und NET SEND-Operatoren über die erfolgreiche oder gescheiterte Ausführung dieser Operation benachrichtigen. Außerdem ist es möglich, die Benachrichtigung in das Windows-Anwendungsereignisprotokoll schreiben zu lassen. Wenn Sie das letzte Kontrollkästchen einschalten, wird der Auftrag automatisch nach seiner Fertigstellung gelöscht. Klicken Sie auf OK, um das Dialogfeld der Auftrags-eigenschaften zu schließen.

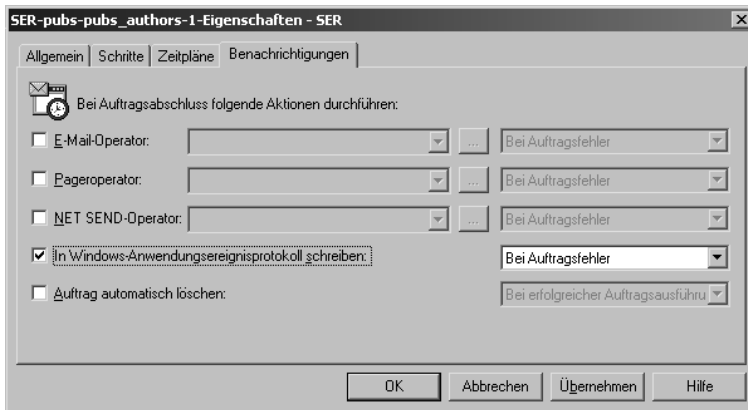


Abbildung 17.35:
Die Registerkarte
Benachrichtigun-
gen der Auftrags-
eigenschaften

Verteilungsserver verwalten

Jetzt ist es an der Zeit, einige Dinge in Bezug auf die Wartung der Replikation zu untersuchen. Sie sollten einen Plan entwickeln, um die Größe der Verteilungsdatenbank zu verwalten und die Sicherungsstrategien für diese Datenbank festzulegen.

Überwachen Sie die Größe Ihrer Verteilungsdatenbank, um zu gewährleisten, dass genügend freier Platz für alle Ihre Replikationsaufträge zur Verfügung steht. Dazu können Sie die Aufbewahrungsdauer für den Replikationsverlauf und die replizierten Transaktionen ermitteln. Beobachten Sie auch die Cleanup-Aufträge, die anfallen, wenn Ereignisse im Replikationsverlauf oder in den Replikationstransaktionen entfernt werden. Diese Optionen können Sie über das Dialogfeld VERLEGER- UND VERTEILEREIGENSCHAFTEN konfigurieren.



Mindestens ein Cleanup-Auftrag fällt auf jedem Verteiler für jeden Abonnenten an. Bei einem Verteilungsserver mit 10 Abonnenten hat dieser Verteilungsserver somit mindestens 10 Cleanup-Aufträge.

Führen Sie die folgenden Schritte aus, um die Verteilereigenschaften zu bearbeiten:

1. Klicken Sie in der Konsolenstruktur mit der rechten Maustaste auf den Knoten *Replikationsmonitor* und wählen Sie im Kontextmenü den Befehl VERTEILEREIGENSCHAFTEN. Daraufhin öffnet sich ein Dialogfeld wie in Abbildung 17.36 gezeigt.

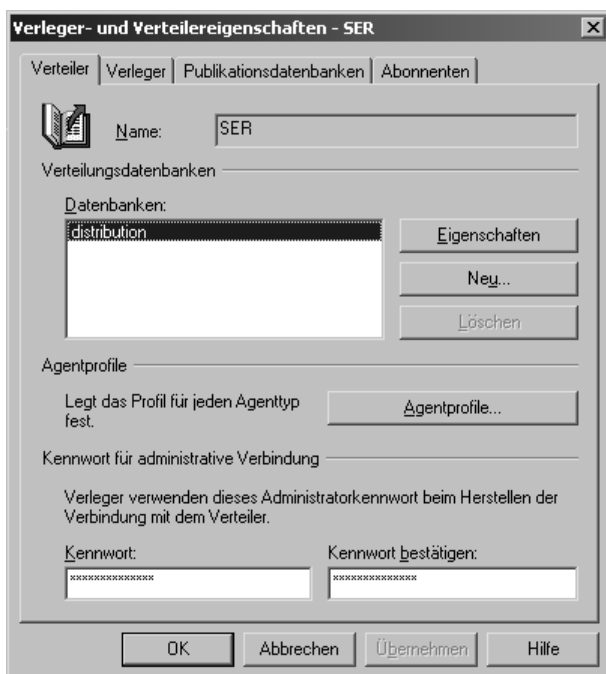


Abbildung 17.36:
Das Dialogfeld Verleger- und Verteilereigenschaften

2. Über dieses Dialogfeld können Sie sich alle Arten von Informationen ansehen. Nachdem Sie die Replikation installiert haben, können Sie von hier aus alle Aspekte Ihres Replikationsszenarios überwachen und verwalten. Klicken Sie auf die Verteilungsdatenbank und dann auf die Schaltfläche EIGENSCHAFTEN. Jetzt sollten Sie ein Dialogfeld wie in Abbildung 17.37 vor sich haben. Hier können Sie festlegen, dass Sie Transaktionsdatensätze mindestens für eine bestimmte Zeitdauer, aber nicht über eine zeitliche Obergrenze speichern möchten. Die Standardwerte betragen 0 bzw. 72 Stunden. Sie können auch eine minimale Zeitspanne einstellen, für die der Replikationsleistungsverlauf gespeichert wird. Wenn Sie alle Einstellungen vorgenommen haben, klicken Sie auf OK, um zum Dialogfeld VERLEGER- UND VERTEILEREIGENSCHAFTEN zurückzukehren.

Das Dialogfeld VERLEGER- UND VERTEILEREIGENSCHAFTEN ist verfügbar, nachdem die Verteilungsdatenbank installiert ist. Auf der Registerkarte VERLEGER spezifizieren Sie die verfügbaren Verleger. Auf der Registerkarte PUBLIKATIONSDATENBANKEN lassen sich die

Datenbanken modifizieren, die für die Publikation zur Verfügung stehen. Die Registerkarte ABONNENTEN ermöglicht das Bearbeiten und Hinzufügen von Abonnenten.

Die Cleanup-Aufträge können Sie sich im Ordner *Verwaltung / SQL Server-Agent / Aufträge ansehen*. (Schließen Sie vorher das Dialogfeld VERLEGER- UND VERTEILEREIGENSCHAFTEN.) Diese Cleanup-Aufträge werden nach einem normalen Zeitplan verarbeitet (siehe Abbildung 17.38).

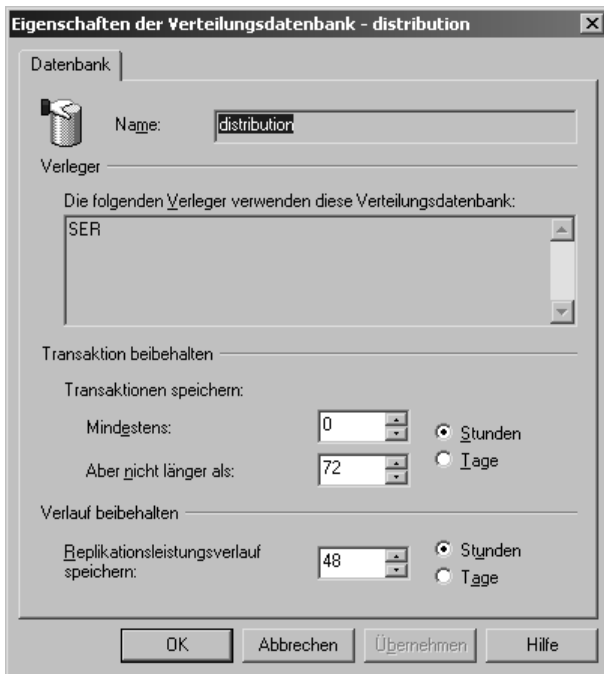


Abbildung 17.37:
Das Eigenschaftsdialogfeld der Verteilungsdatenbank

Man kann sich leicht vorstellen, dass die Verteilungsdatenbank und das zugehörige Transaktionsprotokoll mit allen Erweiterungen und Löschungen aus diesen Datenbanken sowie den regulär für die Replikation protokollierten Transaktionen sehr schnell wachsen können.

Auch wenn Sie keine eigentliche Sicherung Ihrer Verteilungsdatenbank durchführen, sollten Sie zumindest das Transaktionsprotokoll regelmäßig abschneiden. Wenn Sie Sicherungen der Verteilungsdatenbank besitzen und die Datenbank aus irgendwelchen Gründen ausfällt, müssen Sie nicht den gesamten Vorgang durchlaufen, um den Verteilungsserver, die Verleger und die Abonnenten neu zu erstellen. Sie stellen einfach die Datenbank wieder her und synchronisieren Ihre Daten neu.

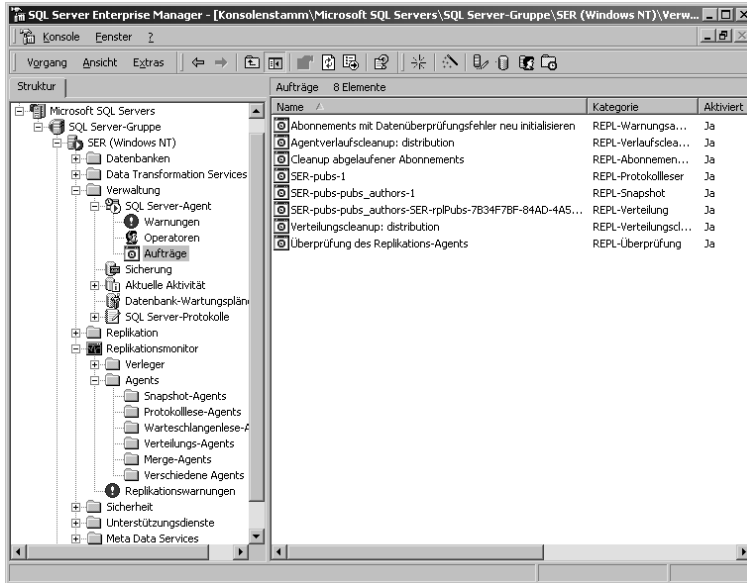


Abbildung 17.38:
Cleanup-Aufträge
für die Replikation



Falls Sie ein »Sicherungsmuffel« sind, sollte Ihnen das folgende Szenario zu denken geben. Vielleicht entschließen Sie sich dann doch, die Verteilungsdatenbank regelmäßig zu sichern:

Wenn die Verteilungsdatenbank »überläuft«, können Transaktionen, die auf ihre Veröffentlichung warten, nicht mehr aus dem Transaktionsprotokoll des Verlegers entfernt werden.

Das erscheint auf den ersten Blick unerheblich, aber überdenken Sie es einen Moment. Wenn die Einträge aus dem Transaktionsprotokoll des Verlegers nicht entfernt werden können, ist der Verleger nicht in der Lage, standardmäßig dieses Protokoll zu bereinigen, was bedeutet, dass früher oder später das Protokoll der Publikationsdatenbank die gesamte Kapazität beansprucht und Sie keine Modifikationen an dem Verleger vornehmen können.

Führen Sie deshalb regelmäßige Sicherungen Ihrer Verteilungsdatenbank durch oder benutzen Sie wenigstens die Anweisung `BACKUP TRANSACTION TRUNCATE_ONLY`, um die Protokolle zu bereinigen.

Replikationsskripts

Replikationsskripts enthalten die erforderlichen Transact-SQL-Anweisungen, um ein bestimmtes Replikationsszenario zu erstellen und zu konfigurieren. Skripts bieten folgende Vorteile:

- Mit einem Skript lassen sich viele Server identisch konfigurieren. Das erspart Ihnen den Aufwand, ständig die Assistenten zu durchlaufen.
- Sie können Skripts benutzen, um verschiedene Versionen Ihrer Replikationsumgebung zu protokollieren. Wenn Sie Änderungen an der Umgebung vornehmen, können Sie zusätzliche Skripts erstellen.
- Eine vorhandene Replikationsumgebung lässt sich schnell und einfach anpassen.
- Sie können Skripts in den Wiederherstellungsvorgang einbeziehen und somit praktisch alle Aspekte Ihrer Replikationsumgebung neu installieren.

Wenn Sie ein Skript für mehrere Server erstellen, müssen Sie gewisse Modifikationen vornehmen. Hauptsächlich betrifft das die Computernamen, auf die das Skript verweist. Obwohl diese Änderungen zeitaufwändig sein können, geht das immer noch schneller, als sich jedes Mal durch die Assistenten zu arbeiten.

Führen Sie die folgenden Schritte aus, um ein Replikationsskript zu erstellen:

1. Markieren Sie Ihren Server in der Konsolenstruktur und wählen Sie **EXTRAS / REPLIKATION / SQL-SKRIPT GENERIEREN**. Es erscheint das in Abbildung 17.39 dargestellte Dialogfeld.
2. Auf der Registerkarte **ALLGEMEIN** wählen Sie aus, welche Komponenten in das Skript einzubeziehen sind. Dazu gehören die Verteilereigenschaften und eine beliebige Anzahl von Publikationen. Außerdem können Sie festlegen, ob das Skript die ausgewählten Komponenten aktivieren oder erstellen bzw. deaktivieren oder löschen soll. Klicken Sie auf die Schaltfläche **VORSCHAU**, um sich das generierte Skript anzusehen (siehe Abbildung 17.40). Von hier aus können Sie einen Dateinamen und den Speicherort für Ihr Skript festlegen. Klicken Sie dann auf **SCHLIESSEN**, um das Dialogfeld zu schließen.
3. Wechseln Sie im Dialogfeld **SQL-SKRIPT GENERIEREN** zur Registerkarte **DATEIOPTIONEN**. Hier können Sie festlegen, in welchem Format das Skript zu speichern ist (siehe Abbildung 17.41). In der Standardeinstellung wird das Skript als Unicode gespeichert und an eine vorhandene Datei angehängt (falls Sie in eine vorhandene Datei speichern).

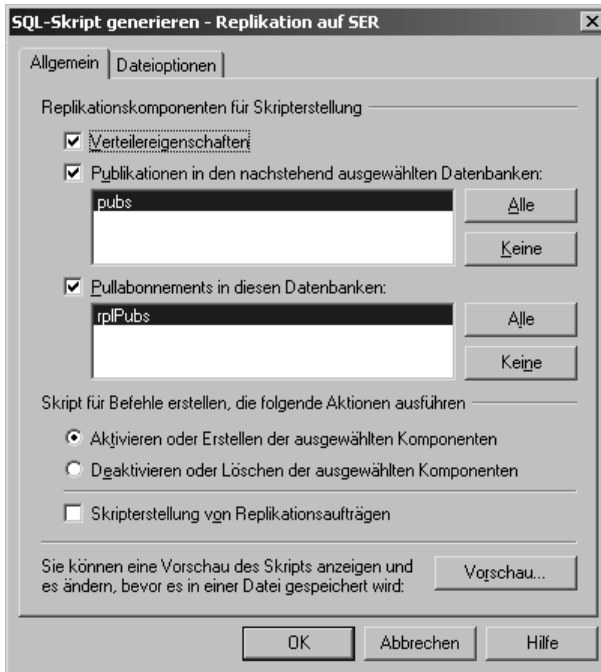


Abbildung 17.39:
Die Registerkarte Allgemein des
Dialogfelds SQL-Skript generieren

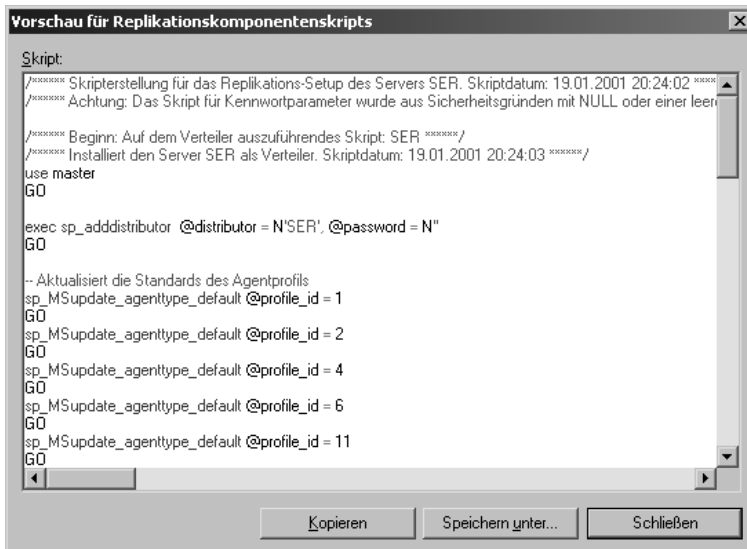


Abbildung 17.40:
Das Dialogfeld Vor-
schau für Replikati-
onskomponentens-
kripts

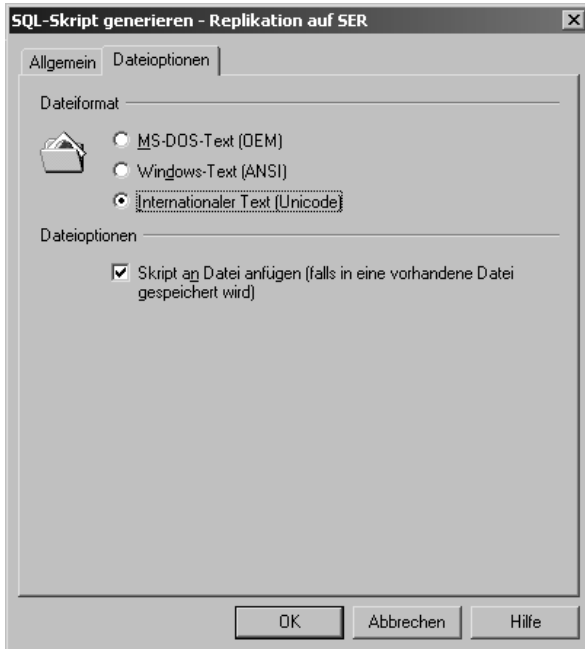


Abbildung 17.41:
Die Registerkarte *Dateioptionen* des
Dialogfelds *SQL-Skript generieren*

4. Klicken Sie auf OK. Daraufhin erscheint ein Standarddialogfeld, in das Sie den Standort und den Dateinamen für das neue Skript eingeben können. Es bietet sich der Standardstandort `\MSSQL\Install` an, weil dieser Ordner auch viele andere Skripts enthält, die während der Installation Ihres SQL Servers angelegt wurden. Übernehmen Sie den Standarddateinamen `replication.sql` und klicken Sie auf **SPEICHERN**.

Ein Replikationsskript können Sie im Query Analyzer ausführen. Öffnen Sie dort die Skriptdatei und führen Sie sie genau wie jedes andere gespeicherte Skript aus.

Systemmonitor und Replikation

SQL Server ist eng mit Windows NT/2000 verbunden, was sich unter anderem im Systemmonitor dokumentiert. SQL Server erzeugt Leistungsindikatoren, von denen mehrere zu einem Leistungsobjekt zusammengefasst sind. Ein Objekt des Windows NT/2000 ist zum Beispiel *Prozessor*. Dieses Objekt hat viele Leistungsindikatoren – beispielsweise *Prozessorzeit (%)*, *Benutzerzeit (%)* und *Interrupts/s*.

Die Replikation in SQL Server legt auch einige Leistungsobjekte und Leistungsindikatoren offen, mit denen Sie die Leistung der verschiedenen Aspekte der Replikation verfolgen können. Tabelle 17.1 listet eine Auswahl von Leistungsindikatoren für die Replikation und ihre zugehörigen Werte auf.

Datenobjekt	Leistungsindikator	Erklärung
SQL Server:Repl- ication Agents	Running	Anzahl der Replikations-Agents, die momentan ausgeführt werden, inklusive Verteilung-, Protokolllese-, Merge-, Warteschlangenlese- und Snapshot-Agent
SQL Server:Repl- ication Distributors	Dist:Delivered Cmds/sec	Anzahl der Verteilungsbefehle, die pro Sekunde an den Abonnenten übermittelt wurden.
	Dist:Delivered Trans/sec	Anzahl der Verteilungstransaktionen, die pro Sekunde an den Abonnenten übermittelt wurden.
	Delivered Latency	Verteilungswartezeit (in Millisekunden). Die Zeit, um Transaktionen an den Verteiler zu übermitteln und auf dem Abonnenten anzuwenden.
SQL Server:Repl- ication Logreader	LogReader:Deli- vered Cmds/sec	Anzahl der Befehle des Protokolllese-Agents, die pro Sekunde an den Verteiler übermittelt wurden.
	Logreader:Deli- vered Trans/sec	Anzahl der Transaktionen des Protokolllese-Agents, die pro Sekunde an den Verteiler übermittelt wurden.
	Logreader:Delivery Latency	Die aktuelle Zeitspanne (in Millisekunden), die vom Zeitpunkt der Anwendung von Transaktionen auf dem Verteiler bis zur Übermittlung der Transaktionen an den Verteiler verstrichen ist.
SQL Server:Repl- ication Merge	Conflicts/sec	Anzahl der Konflikte bei Mergereplikation pro Sekunde.
	Downloaded Chan- ges/sec	Anzahl der vom Verleger stammenden Zeilen, die pro Sekunde auf dem Abonnenten zusammengeführt wurden (eingefügte, aktualisierte und gelöschte Zeilen).
	Uploaded Changes/ sec	Anzahl der vom Abonnenten stammenden Zeilen, die pro Sekunde auf dem Verleger zusammengeführt wurden (eingefügte, aktualisierte und gelöschte Zeilen).
SQL Server:Repl- ication Snapshot	Snapshot:Deli- vered Cmds/sec	Anzahl der Befehle, die pro Sekunde an den Verteiler übermittelt wurden.
	Snapshot:Deli- vered Trans/sec	Anzahl der Transaktionen, die pro Sekunde an den Verteiler übermittelt wurden.

Gespeicherte Prozeduren einsetzen

Bisher haben Sie nur die Implementierung der Replikation über den Enterprise Manager kennen gelernt. Natürlich können Sie auch gespeicherte Prozeduren aufrufen, um die vorgestellten Replikationsszenarios zu implementieren. Vielleicht ist es dabei am Einfachsten, alles Erforderliche im Enterprise Manager zu erstellen und dann Replikationsskripts generieren zu lassen. Außerdem geht es über den Rahmen dieses Buches hinaus, genau zu erläutern

tern, wie Sie Ihre eigenen Replikationsszenarios ausschließlich über Skripts erzeugen. Allerdings gibt es verschiedene gespeicherte Prozeduren, um Informationen über den Replikationsvorgang auf einem bestimmten Server zusammenzutragen. Die folgende Kurzübersicht gibt gespeicherte Prozeduren an, mit denen Sie die SQL Server 2000-Replikation verwalten können. Ein doppeltes Sternchen (**) markiert gespeicherte Prozeduren, die Sie höchstwahrscheinlich einsetzen, um Informationen über Ihr Replikationsszenario zu erhalten.

sp_browserreplcmds	sp_check_for_sync_trigger
sp_help_agent_default	sp_help_agent_parameter
sp_help_agent_profile	sp_help_publication_access
sp_helparticle	sp_helparticlecolumns
sp_helpdistpublisher	sp_helpdistributiondb
sp_helpdistributor**	sp_helpmergearticle
sp_helpmergearticleconflicts	sp_helpmergeconflictrows
sp_helpmergedeleteconflictrows	sp_helpmergefilter
sp_helpmergepublication	sp_helpmergepullsubscription
sp_helpmergesubscription	sp_helppublication**
sp_helppublication_snapshot	sp_helppullsubscription
sp_helpremotelogin	sp_helpreplfailovermode
sp_helpreplcationdb	sp_helpreplcationdboption
sp_helpreplcationoption	sp_helpsubscriber**
sp_helpsubscriberinfo**	sp_helpsubscription**
sp_helpsubscription_properties	sp_link_publication
sp_linkedservers	sp_mergedummyupdate
sp_mergesubscription_cleanup	sp_publication_validation
sp_refreshsubscriptions	sp_reinitmergepullsubscription
sp_reinintmergesubscription	sp_reinitpullsubscription
sp_reinitsubscription	sp_removedbreplication
sp_replcmds**	sp_replcounters**
sp_repldone**	sp_replflush
sp_replication_agent_checkup	sp_replicationdboption
sp_replicationoption	sp_repltrans**
sp_resynch_targetserver	sp_revoke_publication_access
sp_subscription_cleanup	

Die Liste ließe sich noch fortsetzen; für viele Zwecke dürften Sie aber mit den hier genannten Prozeduren auskommen.

Überlegungen zur Verwaltung

Das Verwalten der Replikation kann eine ziemlich umfangreiche Aufgabe sein – muss es aber nicht. Die folgenden Vorschläge können helfen, die Aufgaben der Replikationsverwaltung in einem vernünftigen Rahmen zu halten:

- Erstellen und überwachen Sie Replikationswarnungen. Mit dem passenden Bestand an Warnmeldungen können Sie Ihre Replikation vorbeugend administrieren. Sie erkennen potentielle Probleme und unternehmen Korrekturen, bevor Ihnen diese aus der Hand gleiten können.
- Vergewissern Sie sich, dass die Replikationsaufträge und -Agents fehlerfrei laufen. Informationen darüber erhalten Sie bei regelmäßigen Kontrollen des Replikationsverlaufs und der Fehlerprotokolle.
- Überprüfen Sie, ob genügend Festplattenplatz vorhanden ist. Die Verteilungsdatenbank benötigt Platz, um Transaktionsinformationen und Informationen zum Replikationsprozess selbst zu speichern. Legen Sie Sicherungen der Verteilungsdatenbank an oder überwachen Sie zumindest das Transaktionsprotokoll der Verteilungsdatenbank genauestens.
- Erstellen Sie einen Plan zur Wiederherstellung und erneuten Synchronisierung. Erstellen und speichern Sie Replikationsskripts. Diese sind nützlich für die Wiederherstellung und Versionskontrolle sowie für das Duplizieren mehrerer identischer Server. Testen Sie den Wiederherstellungs- und Synchronisierungsplan und stellen Sie fest, wie viel Zeit dafür erforderlich ist.
- Sicherheit ist wahrscheinlich die häufigste Ursache für Fehler, denen Sie bei der SQL Server-Replikation begegnen. Verwenden Sie nach Möglichkeit für alle in die Replikation einbezogenen Server dasselbe SQL Server-Agent-Konto. So beugen Sie einer Vielzahl von potentiellen Sicherheitsproblemen vor.
- Der Netzwerkzugang ist eine weitere Fehlerquelle. Vergewissern Sie sich, dass Ihr Netzwerk genügend Bandbreite bietet, um die Replikationstasks zu unterstützen.
- Beseitigen Sie den Einfluss der Verteilungsdatenbank auf Ihren OLTP-Server mithilfe von Remoteverteilern.
- Wenn Sie Ihre Replikationsszenarios erstellen, versuchen Sie, als Erstes die Snapshotreplikation zu implementieren. Diese Form der Replikation ist am Einfachsten zu implementieren und zu warten. Wenn Sie sich überzeugt haben, dass es keine Netzwerkzugangs- und Sicherheitsprobleme gibt, ändern Sie den Replikationstyp so, wie Sie ihn wünschen.

Fehlerbehebung

Die Fehlerbehebung setzt grundlegende technische Kenntnisse und eine Menge Erfahrung voraus. Dieser Abschnitt führt Sie von der technischen Seite her an die Fehlerbehebung bei der Replikation heran und gibt einige Vorschläge, wie Sie die Fehlerquellen in Ihrem System einkreisen können.

Im Rahmen der Replikation können Probleme an mehreren Stellen auftreten:

- Sicherheit
- Protokolllese-Agent
- Verteilungs-Agent

Die folgenden Abschnitte erläutern, wie sich Fehler in diesen Elementen auf das System auswirken können.

Die Aufträge für Anfangssnapshots werden nicht auf den Abonnenten angewandt

Bevor die eigentliche Replikation stattfinden kann, muss ein Anfangssnapshot Ihrer Daten auf den Abonnenten angewandt werden. Wenn dies nicht geschieht, ist die häufigste Ursache der Snapshot-Agent selbst. Überprüfen Sie den Snapshot-Agentverlauf. Ist der Snapshot überhaupt geplant? Wenn er geplant ist, wurde versucht, einen Snapshot Ihrer Daten anzufertigen? Haben Ihr Snapshot-Agent und Ihr Abonnent die richtigen Sicherheitsmerkmale aktiviert?

Wenn der Snapshot-Agent keine Daten erhält und deshalb nicht versucht, einen Snapshot auf den Abonnenten anzuwenden, könnte ein Sicherheitsproblem zwischen dem Snapshot-Agent und der Publikationsdatenbank vorliegen.

Wenn der Snapshot-Agent überhaupt nicht reagiert, sollten Sie überprüfen, ob er eingerichtet ist und ob der Verteilungsserver online und verfügbar ist.

Kein Abonnement-Server empfängt replizierte Informationen

Wenn keiner Ihrer Abonnement-Server replizierte Informationen empfängt, ist es unwahrscheinlich, dass Ihr Verteilungs-Agent schuldig ist. Eher ist es möglich, dass der Protokolllese-Agent die Daten aus den Transaktionsprotokollen der veröffentlichten Datenbank – aus welchen Gründen auch immer – nicht mehr liest. Um herauszufinden, ob der Protokolllese-Agent oder der Verteilungs-Agent gescheitert ist, überprüfen Sie den Agentverlauf. Kontrollieren Sie, ob Aufträge von einem zum anderen übertragen werden.

Wenn es so aussieht, dass diese Aufgaben nicht laufen, ändern Sie ihre Zeitpläne geringfügig, um sich von den Auswirkungen ein Bild machen zu können.

Nur ein Abonnement-Server empfängt keine replizierten Informationen

Wenn nur ein Abonnement-Server (oder nur ein Teil der Server) keine replizierten Informationen erhält, wissen Sie, dass der Protokolllese-Agent richtig arbeitet (wie würden die anderen Abonnenten sonst ihre Informationen bekommen?). Sie sollten die folgenden Fehlermöglichkeiten überprüfen:

- Arbeitet der Verteilungs-Agent des fehlerhaften Servers korrekt? (Überprüfen Sie den Agentverlauf.)
- Die Abonentendatenbank ist nicht verfügbar. Die Datenbank könnte als schreibgeschützt markiert sein, als nur für den *dbo* zugänglich usw.
- Der Abonnementserver ist nicht verfügbar. Ist der Server heruntergefahren? Besteht die Netzwerkverbindung?
- Die Sicherheitsinformationen des SQL Server-Agent-Dienstes sind geändert worden.
- Der Verteilungs-Agent erwartet eine manuelle Synchronisation der Daten.

Probleme bei der Wiederherstellung der Replikation

Die konzeptionelle Gestaltung der SQL Server-Replikation erlaubt in den meisten Fällen eine Wiederherstellung aus eigener Kraft. Beachten Sie aber die folgenden Überlegungen in Bezug auf Publikationsserver, Verteilungsserver und Abonentenserver.

Wiederherstellung des Publikationsservers

Der Protokolllese-Agent benutzt Zeiger in jeder publizierten Datenbank um festzuhalten, bis wohin er die Transaktionen gelesen hat. Mit diesen Zeigern ist die Wiederherstellung ein relativ einfacher Vorgang, wenn der Publikationsserver aus bestimmten Gründen offline ist. Sobald die Publikationsdatenbank wieder online verfügbar ist, setzt der Protokolllese-Agent an der Stelle fort, wo er aufgehört hat.

Wenn aus irgendwelchen Gründen die Publikationsdatenbanken neu erstellt und mit neuen Daten gefüllt werden, sind die Zeiger nicht mehr gültig. Dann müssen Sie Ihre Verleger und Abonnenten erneut synchronisieren, um die Replikation fortsetzen zu können.

Wiederherstellung des Verteilungsservers

Verteilungsserver protokollieren im Allgemeinen die Position, wo sie das Senden von Aufträgen (Transaktionen) an die Abonnenten unterbrochen haben. Wenn der Verteiler offline ist und dann wieder online verfügbar wird, setzt er die Verarbeitung am Unterbrechungspunkt fort. Normalerweise ist dann alles wieder im Lot. Ist der Verteiler über ausgedehnte Zeiträume nicht verfügbar, müssen Sie gegebenenfalls Ihren Verteiler wiederherstellen. Für diesen Fall sollten Sie ein abgestimmtes Sicherungsschema haben, um zu gewährleisten, dass der Verteilungsserver gemeinsam mit den Publikationsservern gesichert wird. Dadurch bleibt der Protokolllese-Agent synchron mit der Publikationsdatenbank und es findet eine automatische Wiederherstellung statt.

Wenn keine Synchronisation mehr besteht und die Zeiger des Protokolllese-Agents nicht mehr gültig sind (bei Wiederherstellung der Datenbanken sind zwischen Verleger und Verteiler Konflikte aufgetreten), müssen Sie Verleger und Abonnenten erneut synchronisieren. Wenn diese Lösung nicht funktioniert, müssen Sie die Abonnements aufheben und die Datenbanken erneut abonnieren.

Wiederherstellung von Abonnementservern

Wenn ein Abonnementserver herunterfährt, müssen Sie ihn lediglich wieder in den Onlinezustand bringen. Die replizierten Transaktionen sind auf dem Verteiler gespeichert und werden angewandt, sobald der Abonnent wieder online ist. Wenn Sie Snapshotreplikation benutzen, wird der gesamte Snapshot zum geplanten Zeitpunkt gesendet. Bei Mergereplikation werden alle konvergierten Datensätze bei einer Synchronisation erneut gesendet. Diese Lösung funktioniert, sofern der Abonnent nur für eine kurze Zeit heruntergefahren ist.

Ist der Abonnent über einen längeren Zeitraum hinweg offline, können die auf den Abonnenten angewandten Transaktionen wegen Überalterung aus der Verteilungsdatenbank entfernt worden sein. Wenn bekannt ist, dass ein Abonnent für längere Zeit offline sein wird und Sie keinen Ärger mit einer erneuten Synchronisation haben wollen (eventuell haben Sie eine extrem große Datenbank und möchten diese nicht per Snapshot über das Netzwerk synchronisieren), können Sie die Cleanup-Aufträge bearbeiten, die mit diesem Snapshot-Auftrag verbunden sind. Sie können auch die Aufbewahrungsdauer für Transaktionen verändern, die mit diesem Abonnenten verbunden sind, wie es diese Lektion weiter vorn erläutert hat.



17.6 Zusammenfassung

Die heutige Lektion hat sich mit der Implementierung der Replikation beschäftigt. Nachdem einige Sicherheitsprobleme angesprochen wurden, haben Sie sich dem praktischen Teil zugewandt.

Als Erstes haben Sie Ihre Server für die Replikation eingerichtet. Im Anschluss daran haben Sie Publikationen und Abonnements erstellt sowie die Synchronisierung manuell durchgeführt.

Betrachtungen zu allgemeinen Problemen der Replikation und zur Replikationsverwaltung waren weitere Schwerpunkte dieser Lektion. In diesem Zusammenhang wurden auch Fragen in Bezug auf die Überwachung der Replikation, das Erstellen von Replikationskripts, den Windows NT/2000-Systemmonitor, Probleme bei der Sicherung von Verteilungsdatenbanken und den Verlauf der Replikations-Agents behandelt.

Abschließend hat die Lektion einen kurzen Überblick zur Fehlerbehebung gebracht.

17.7 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Welcher Replikationstyp ist der gebräuchlichste?*
- A** Transaktionsreplikation ist die gebräuchlichste Form der Replikation. Sie garantiert Transaktionskonsistenz und hat geringen bis mittleren Overhead.
- F** *Ist zu erwarten, dass die Mergereplikation zum gebräuchlichsten Replikationstyp avanciert?*
- A** In Anbetracht der zunehmenden Zahl von Remotebenutzern ist das durchaus möglich. Sie sollten versuchen, den Einfluss der Mergereplikation durch die Einführung von Regionalcodes in Ihren Tabellen zu minimieren. Dadurch ist es weniger wahrscheinlich, dass gleiche Daten an zwei unterschiedlichen Standorten gleichzeitig geändert werden.

F *Sollte ich Skripts zur Erstellung und Installation der Replikation benutzen?*

A Das kommt darauf an. Verwenden Sie für die Anfangsreplikation, die Publikation und das Abonnieren den Enterprise Manager; dann erstellen Sie Replikations-skripts mithilfe des Replikationsmonitors. Auf diese Art und Weise nehmen Sie alle erforderlichen Einstellungen über die komfortable Benutzeroberfläche vor und lassen dann Skripts erstellen, die sich für eine Wiederherstellung eignen.

F *Wie wichtig ist die Wiederherstellung der Replikation?*

A Das hängt wirklich von den Erfordernissen Ihres Unternehmens ab. Denken Sie daran, dass sich die Replikation in den meisten Fällen selbst wiederherstellt. Wenn ein Verleger-Server vorübergehend heruntergefahren wird, greift das automatische Wiederherstellen. Wenn ein Verteilungsserver für kurze Zeit nicht verfügbar ist, wird er einfach dort fortfahren, wo er aufgehört hat. Wenn ein Abonnentenserver für kurze Zeit offline ist, wird er wiederhergestellt, wenn er online geht. Das einzig wirkliche Problem, über das Sie sich in Zusammenhang mit der Wiederherstellung Gedanken machen müssen, besteht darin, dass die Transaktionen nicht automatisch aus den Replikations-Agents entfernt werden, bevor sie angewandt werden konnten. Außerdem ist es wichtig, dass Sie regelmäßig die Größe des Transaktionsprotokolls Ihrer Verteilungsdatenbank verwalten.

Kontrollfragen

Dieses Kapitel hatte mehr den Charakter eines Durchmarschs als einer Übung. Deshalb verzichten wir hier auf die Kontrollfragen. Nehmen Sie sich aber die Zeit, die Übung zu versuchen.

Übung

Modifizieren Sie die Publikation und die Abonnements, die Sie in der heutigen Lektion erstellt haben, so dass sie mit Mergereplikation anstelle der Transaktionsreplikation arbeiten.



Aufträge mit dem
SQL Server-Agent
planen

In der heutigen Lektion lernen Sie die Automatisierungsmöglichkeiten von SQL Server kennen. Dabei geht es vor allem um den SQL Server-Agent-Dienst, mit dem Sie unter anderem folgende Aufgaben ausführen können: zeitlich geplante Aufträge erstellen, SQL Server-Ereignisse überwachen und Warnungen einrichten, um einen Auftrag zu starten oder eine Benachrichtigung via E-Mail, Pager oder NET SEND-Befehl zu verschicken. Darüber hinaus ist die gesamte Replikation um die geplanten Aufträge des SQL Server-Agents herum aufgebaut.

18.1 Warum SQL Server automatisieren?

Vielleicht fragen Sie sich, warum man überhaupt SQL Server automatisieren sollte. Allein schon deshalb, weil man als Administrator nicht mitten in der Nacht wegen lapidarer Support-Aufgaben geweckt werden möchte. Sie sollten Sie also alles versuchen, derartigen Anfragen vorzubeugen. Wenn SQL Server korrekt konfiguriert ist, lässt sich der Server so einrichten, dass die meisten Probleme ohne Ihr Zutun automatisch abgefangen und gelöst werden. Sie können auch Routineaufgaben der Datenbankwartung automatisieren, wie zum Beispiel das Laden von Daten, die Überprüfung der Integrität und Datenbanksicherungen.

Der SQL Server-Agent-Dienst ermöglicht die vollständige Automatisierung dieser Verwaltungsaufgaben mithilfe von geplanten Aufträgen. Zu den Auftragsstypen gehören Datenbankwartung, Volltextindizierung, Webassistent oder verschiedene Replikationsaufträge. Man kann auch einen generischen Auftragsstyp verwenden, um die Ausführung von Transact-SQL-Anweisungen, Windows 2000-Befehlsdateien oder ActiveX-Skripts (VBScript oder JavaScript) zu unterstützen.

Es lassen sich Aufträge erstellen, die nach einem regulären Zeitplan ausgeführt werden oder die man explizit und nur bei Bedarf aufrufen kann. Das ist für alle hier erwähnten Auftragsstypen möglich.

Ein anderes Element der SQL Server-Automatisierung sind Warnungen, die bei einem bestimmten Ereignis oder einem bestimmten Fehler aktiv werden. Als Reaktion kann man zum Beispiel eine E-Mail generieren, einen Operator per Pager rufen oder einen Auftrag starten, der das Problem automatisch behebt.

Alle hier beschriebenen Funktionen können Sie in eigener Regie erstellen und ausführen, ohne sich auf den SQL Server-Agent-Dienst zu stützen. Allerdings ist es in einer Umgebung, in der nur ein SQL Server zum Einsatz kommt, sicherlich einfacher, verschiedene Aufgaben vom System durchführen zu lassen. Wenn Sie die Vorteile dieser eingebauten Funktionalität nutzen, können Sie sich auf das geeignete Design Ihrer Datenbanken konzentrieren und brauchen nur bei echten Abweichungen vom normalen Ablauf einzugreifen.

18.2 Welches Planungstool ist zu empfehlen?

Nachdem Sie sich entschieden haben, die Vorteile der Automatisierung von SQL Server zu nutzen, könnte Ihre nächste Frage sein, ob der SQL Server-Agent Dienst das richtige Werkzeug ist. Der SQL Server-Agent-Dienst ist viel mehr als ein einfaches Planungsinstrument; er erlaubt Ihnen, auf Fehlersituationen zu reagieren und E-Mails oder sogar Pagernachrichten zu versenden. Wenn Sie allerdings eher mit Windows 2000 zu tun haben und bereits andere Aufträge (wie zum Beispiel Sicherungen) mit dem Windows-Zeitplandienst eingerichtet haben, sollten Sie sich mit diesem Planungsinstrument vertraut machen, um SQL Server zu unterstützen.

Der Windows 2000-Zeitplandienst

Da der Windows-Zeitplandienst ein reines Zeitplanungsinstrument ist, könnten Sie versucht sein, ihn zusammen mit dem AT-Befehl zu verwenden. Der Befehl AT erlaubt es, die Ausführung von Befehlen und Programmen zu einem bestimmten Zeitpunkt (Uhrzeit und Datum) zu planen. Der Windows-Zeitplandienst muss laufen, damit Sie den AT-Befehl verwenden können.

Um Hilfe zum AT-Befehl zu erhalten, geben Sie an der Eingabeaufforderung Folgendes ein:



C:> AT /?

Dieser Befehl liefert folgende Ausgabe:



Mit dem Befehl AT können Befehle und Programme zu einem vorbestimmten Termin gestartet werden. Der Zeitplandienst muss gestartet sein, um den Befehl AT zu verwenden.

```
AT [\\Computername] [ [Kennung] [/DELETE] | /DELETE [/YES]]
AT [\\Computername] Zeit [/INTERACTIVE]
    [ /EVERY:Datum[,...]] | /NEXT:Datum[,...]] "Befehl"
```

\\Computername	Gibt einen Remotecomputer an. Ohne diesen Parameter werden
Kennung	die Befehle auf dem lokalen Computer ausgeführt. Eine Identifikationsnummer, die dem geplanten Befehl zugeteilt wird.
/DELETE	Löscht geplante Befehle. Ohne Kennung werden alle

geplanten	Befehle auf dem Computer gelöscht.
/YES	In Verbindung mit /DELETE werden die geplanten Befehle ohne weitere Bestätigung gelöscht.
Zeit werden soll.	Gibt die Zeit an, zu der ein Befehl ausgeführt wird.
/INTERACTIVE anzu-	Ermöglicht dem Auftrag, Eingaben von dem Benutzer nehmen, der angemeldet ist, wenn der Auftrag ausgeführt wird.
ausgeführt	/EVERY:Datum[,...] Führt den Befehl zu jedem der angegebenen Tage der Woche oder des Monats aus. Ohne Angabe eines Datums wird der aktuelle Tag des Monats angenommen.
	/NEXT:Datum[,...] Führt den Befehl am nächsten angegebenen Tag aus (z.B. nächsten Montag). Ohne Angabe eines Datums wird der aktuelle Tag des Monats angenommen.
"Befehl"	Ist der auszuführende Windows NT-Befehl oder Stapelprogramm.

Um zum Beispiel eine automatisierte Sicherung zu starten, können Sie eingeben:



```
C:> AT \\MYSERVER 12:00 /EVERY:M,T,W,TH,F
F:\Programme\Microsoft SQL Server\mssql\backup.cmd
```

Damit richten Sie einen Aufruf des Programms `backup.cmd` im SQL Server 2000-Installationsverzeichnis ein, wobei auch Aufrufe von `isql.exe` oder `osql.exe` zur Durchführung der Sicherungen enthalten sein können. Der angegebene Befehl bewirkt, dass das Sicherungsprogramm an jedem Wochentag um Mitternacht ausgeführt wird.

Dies ist sicherlich eine Möglichkeit, Sicherungen zu planen. Es ist auch eine gute Option, wenn Sie mit dem AT-Befehl und der Planung von Tasks auf diese Art und Weise vertraut sind. Sie können auch das Windows 2000 Ressource Kit kaufen und eine grafische Version der AT-Anweisung verwenden. Beide Versionen stützen sich auf den Windows-Zeitplandienst, um geplante Tasks abzarbeiten.

Der SQL Server-Agent-Dienst

Der SQL Server-Agent-Dienst erlaubt Ihnen, sowohl einfache als auch komplexe Aufträge mit mehreren Schritten zu erstellen. Sie können auf grafischem Weg die E-Mail-Integration einrichten, so dass Sie keine eigenen Aufrufe des E-Mail-Subsystems schreiben müssen. Die Aufträge können Windows-Befehlsdateien analog zum AT-Befehl aufrufen, Transact-SQL-Anweisungen direkt ausführen oder sogar die Optionen der ActiveX-Skriptsprachen verwenden. Die Bedienung ist zudem intuitiver als das Konfigurieren der AT-Anweisung und das Schreiben von Stapeldateien.

Der SQL Server-Agent-Dienst besteht aus einer Reihe von Modulen, die zusammenarbeiten, um SQL Server zu automatisieren:

- Das *Zeitplanmodul* entspricht dem Windows-Zeitplandienst. Es startet geplante Aufträge zu den festgelegten Zeitpunkten, basierend auf Tag und Zeit der Windows-Uhr auf dem SQL Server-Computer.
- Das *Warnungsmodul* überwacht Nachrichten, die in das Windows 2000-Ereignisprotokoll geschrieben werden, empfängt Ereignisse vom SQL Server-Ereignismodul und führt eine Aktion aus (wie zum Beispiel das Starten eines Auftrags oder das Senden einer E-Mail- oder Pagerbenachrichtigung). Es kann auch SQL Server-Leistungsindikatoren überwachen, die normalerweise der Windows 2000-Systemmonitor liest.
- Das *Ereignismodul* überwacht das Windows 2000-Anwendungsereignisprotokoll und fängt Nachrichten ab, die von SQL Server stammen. Dann leitet es diese Nachrichten an das Warnungsmodul weiter.
- Das *Auftragsmodul* startet Aufträge entweder zu einem vorbestimmten Zeitpunkt oder dynamisch auf Anforderung. Es zeichnet auf, ob eine Routine erfolgreich oder fehlerhaft gelaufen ist. Es kann ebenfalls eine E-Mail senden oder eine Nachricht in das Windows 2000-Ereignisprotokoll bei Erfolg oder Scheitern schreiben. Ein Auftrag besteht aus einem oder mehreren Schritten, wie Sie später in dieser Lektion sehen werden.

Tritt ein Ereignis ein, das in das Windows 2000-Anwendungsereignisprotokoll geschrieben wird, fängt das Ereignismodul die Nachricht ab und gibt sie an das Warnungsmodul weiter. Das Warnungsmodul bestimmt, ob es sich um eine Warnung handelt, für die eine Überwachung konfiguriert ist; in diesem Fall startet es beispielsweise einen Auftrag oder sendet eine E-Mail-Nachricht. Der Auftrag kann auch bewirken, dass eine Nachricht in das Windows 2000-Ereignisprotokoll geschrieben wird, so dass der Kreislauf erneut beginnt.

Konfiguration des SQL Server-Agent-Dienstes

Wenn Sie SQL Server erstmalig installieren, wählen Sie ein Dienstkonto, um die SQL Server-Dienste zu starten – einschließlich des SQL Server-Agent-Dienstes. Das ist aber nur eine der Konfigurationsoptionen, mit denen Sie Einfluss auf den SQL Server-Agent-Dienst nehmen können.

Starten Sie den SQL Server Enterprise Manager (falls er noch nicht läuft). Stellen Sie dann die Verbindung zu Ihrem Server her und erweitern Sie den Ordner *Verwaltung*. Klicken Sie mit der rechten Maustaste auf den Eintrag *SQL Server-Agent* und wählen Sie **STARTEN**, falls der Dienst noch nicht läuft.

Nachdem der Dienst gestartet ist und das Symbol den grünen Wiedergabepfeil zeigt, klicken Sie mit der rechten Maustaste auf das Symbol oder den Text und wählen **EIGENSCHAFTEN**, um das in Abbildung 18.1 dargestellte Dialogfeld zu öffnen.

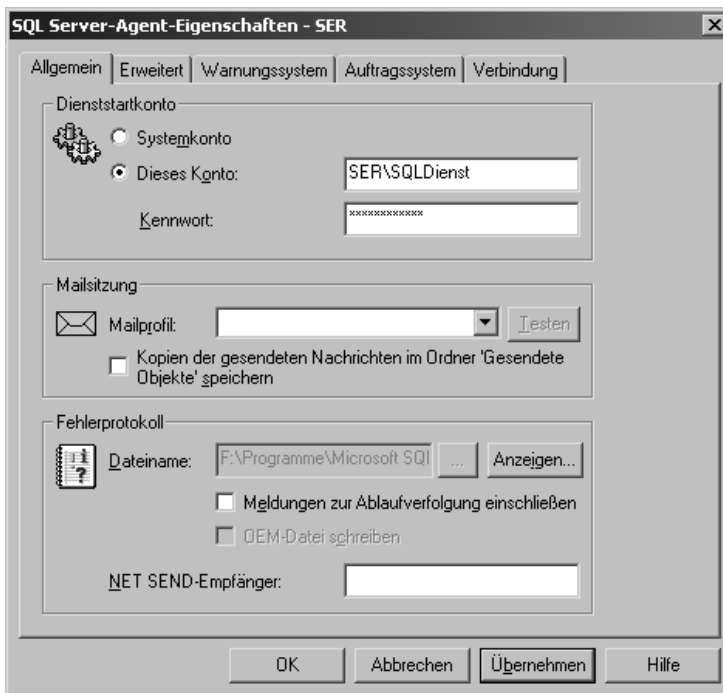


Abbildung 18.1:
Das Dialogfeld SQL
Server-Agent-Eigen-
schaften

Die Registerkarte Allgemein

Dienststartkonto: Die erste Option auf der Registerkarte ALLGEMEIN dieses Dialogfelds legt fest, unter welchem Windows-Konto der SQL Server-Agent-Dienst läuft (auf Windows 9x-Plattformen ist diese Option nicht verfügbar). Das Beispiel in Abbildung 18.1 zeigt, dass

der Dienst im Kontext eines Domänenbenutzers (im Dialogfeld zu sehen als *SER\SQL-Dienst*) als Konto auf dem Windows 2000-Server läuft. Wenn Sie kein Benutzerkonto verwenden (d.h., nicht die Option *Systemkonto* wählen), können Sie nicht die E-Mail-Features von SQL Server nutzen. Außerdem können Sie keine Aufträge ausführen, die andernfalls auf mehreren Servern laufen oder den Zugriff auf Ressourcen von Remoteservern benötigen (beispielsweise bei einer Sicherung auf einen Remotedateiserver).

Mailsitzung: Die nächste Option nach dem Dienststartkonto ist die Konfiguration der Mailsitzung. Hier wählen Sie das Mailprofil, das Sie eingerichtet haben. Das korrekte Einrichten von SQL Mail erfordert verschiedene Schritte, die in Kürze besprochen werden. Lassen Sie also den Eintrag für das Mailprofil erst einmal leer.

Fehlerprotokoll: Die letzte Option auf der Registerkarte ALLGEMEIN ist die Konfiguration des Fehlerprotokolls für den SQL Server-Agent. Dieses Fehlerprotokoll ist komplett unabhängig vom SQL Server-Fehlerprotokoll (dem Protokoll für den SQL Server-Dienst). Dieses Fehlerprotokoll dient hauptsächlich zum Aufzeichnen von Problemen mit Aufträgen. Abbildung 18.2 zeigt ein Beispiel für das Fehlerprotokoll des SQL Server-Agents. Um das Fehlerprotokoll anzuzeigen, klicken Sie mit der rechten Maustaste in der Konsolenstruktur des Enterprise Managers auf *SQL Server-Agent* und wählen FEHLERPROTOKOLL ANZEIGEN. (Auf der Registerkarte ALLGEMEIN des Dialogfelds SQL SERVER-AGENT-EIGENSCHAFTEN ist das Fehlerprotokoll auch über die Schaltfläche ANZEIGEN zugänglich.) Hier sind verschiedene Filteroptionen verfügbar: Sie können nur die Fehler, nur die Warnungen bzw. nur die Informationen anzeigen oder – wie es auch in Abbildung 18.2 zu sehen ist – alle Typen auf einmal auswählen.

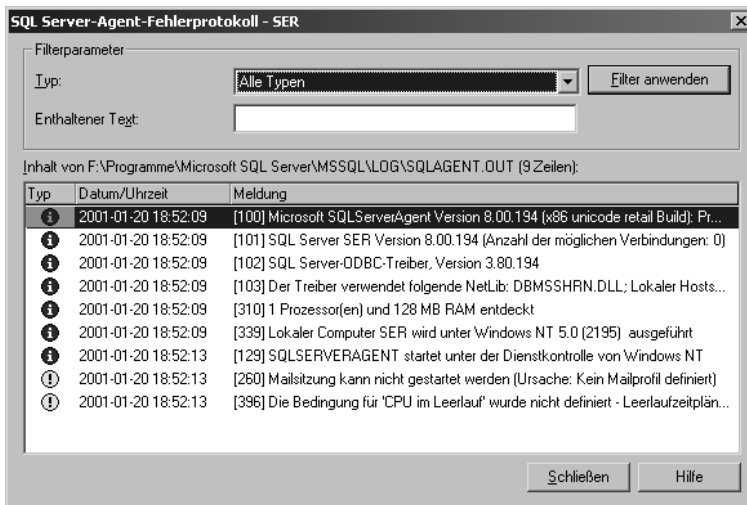


Abbildung 18.2:
Fehlerprotokoll des
SQL Server-Agents

Auf der Registerkarte ALLGEMEIN sind die folgenden Konfigurationsoptionen für das Fehlerprotokoll verfügbar:

- MELDUNGEN ZUR ABLAUFVERFOLGUNG EINSCHLIESSEN umfasst alle Details von Ereignissen des SQL Server-Agent-Dienstes. Sie werden diese Einstellung gewöhnlich aktivieren, wenn Sie Probleme mit dem SQL Server-Agent beseitigen wollen.
- OEM-DATEI SCHREIBEN: Das SQL Server-Fehlerprotokoll wird standardmäßig im Unicode-Format aufgezeichnet. Das macht das Protokoll weltweit nutzbar, verdoppelt aber dessen Größe. Wenn Sie Ihr Protokoll nicht in Unicode aufzeichnen möchten, wählen Sie diese Option.
- NET SEND-EMPFÄNGER spezifiziert den Windows 2000-Kontenname eines Operators, der bei einem Popup-Fehler zu benachrichtigen ist. Der Benutzer wird über einen NET SEND-Befehl in Windows 2000 benachrichtigt und auf allen Computern, an denen dieser Benutzer angemeldet ist, erscheint ein Meldungsfeld.

Die Registerkarte Erweitert

Nachdem Sie die allgemeinen Einstellungen vorgenommen haben, wechseln Sie zur Registerkarte ERWEITERT, die Abbildung 18.3 zeigt. Hier können Sie für den Fehlerfall den Restart Ihrer SQL Server-Dienste, das Weiterleiten von Ereignismeldungen an einen anderen SQL Server und das Setzen der CPU-Leerlauf-Konfiguration für Ihren Server konfigurieren.

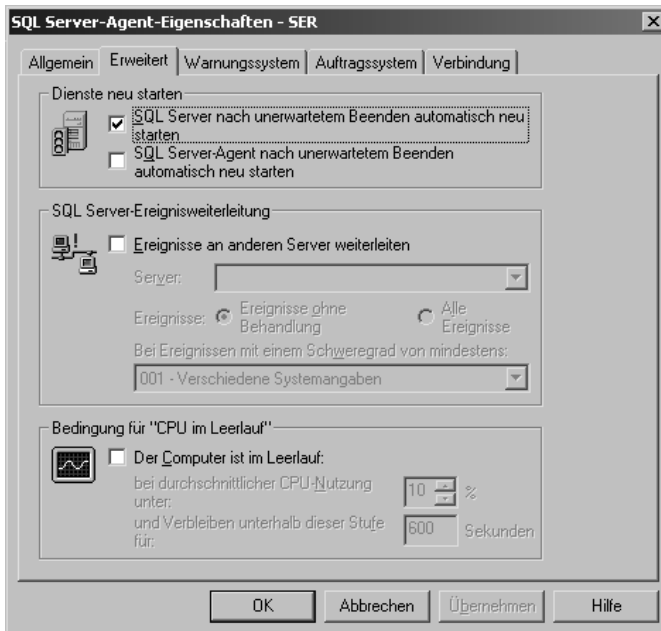


Abbildung 18.3:
Die Registerkarte Erweitert
des Dialogfelds SQL Server-
Agent-Eigenschaften

Dienste neu Starten: Sie können festlegen, ob der SQL Server-Dienst oder der SQL Server-Agent-Dienst bei einem unerwarteten Beenden neu gestartet werden soll. Im Abschnitt DIENSTE NEU STARTEN ist das Kontrollkästchen für den SQL Server-Dienst standardmäßig eingeschaltet. Wenn Sie diese Optionen aktivieren, können Sie Ihre SQL Server-Umgebung weiter automatisieren, um die erforderlichen manuellen Eingriffe zu minimieren. Wenn SQL Server aus irgendeinem Grund abstürzt, vermeiden Sie so unter Umständen einen Hilferuf mitten in der Nacht, nur um die Dienste einfach neu starten zu müssen.

SQL Server-Ereignisweiterleitung: Hier legen Sie einen anderen SQL Server fest, an den Ihre Ereignismeldungen gesendet werden, wenn Sie die Ereignisbehandlung nicht für den lokalen SQL Server konfigurieren möchten. Die Idee dahinter ist die, dass Sie die Ereignismeldungen an ein zentrales SQL Server System weiterleiten können. Der zentrale SQL Server lässt sich mit Warnungen für mehrere Server konfigurieren und kann dann zentral gesteuert werden. Einerseits lagert man so einen möglicherweise beträchtlichen Arbeitsumfang auf den zentralen Server aus, andererseits muss man aber mit längeren Reaktionszeiten für Ereignisse auf dem eigenen Server rechnen, weil sämtliche Ereignisse zuerst an den zentralen Server weiterzuleiten sind.

Wenn Sie einen zentralen Server für den Empfang von Ereignismeldungen konfigurieren, erhöht sich auch der Netzwerkverkehr. Deshalb sollten Sie die SQL Server-Ereignisweiterleitung möglichst nicht für einen Server spezifizieren, der bereits mit der Verwaltung einer Produktionsdatenbank beschäftigt ist.

Indem Sie die Option *Bei Ereignissen mit einem Schweregrad von mindestens* wählen, erfolgt die Weiterleitung nur für solche Fehler, die mindestens den spezifizierten Schweregrad aufweisen. Mit der Option *Ereignisse ohne Behandlung* leiten Sie nur Ereignisse weiter, die nicht lokal behandelt werden. Wenn Sie also als Reaktion auf ein lokales Ereignis eine Warnung konfigurieren, gelangt dieses Ereignis nicht zum spezifizierten Server.

Die folgende Übersicht erläutert die Klassifizierung der Schweregrade:

Schweregrad 0-10	Meldungen sind informeller Natur
Schweregrad 11-16	Verweisen auf Fehler, die der Benutzer beheben kann.
Schweregrad 17	Kennzeichnet unzureichende Ressourcen, z.B. nicht genügend Sperren
Schweregrad 18	Gibt an, dass ein interner Fehler aufgetreten ist. Der Fehler ist nicht schwerwiegend und die Verbindung zum Server bleibt erhalten.
Schweregrad 19	Zeigt an, dass ein interner nicht konfigurierbarer Grenzwert für eine Ressource überschritten worden ist.
Schweregrad 20	Kennzeichnet das Auftreten eines schweren Fehlers in Ihrer Verbindung.
Schweregrad 21	Kennzeichnet das Auftreten eines schweren Fehlers in Ihrer Verbindung, der alle Verbindungen zur Datenbank beeinflusst.
Schweregrad 22	Gibt an, dass die angegebene Tabelle oder der angegebene Index beschädigt wurde.

Schweregrad 23	Signalisiert, dass die verwendete Datenbank durch ein Hardware- oder Software-Problem beschädigt wurde.
Schweregrad 24	Kennzeichnet das Auftreten einiger Arten von Hardware-Fehlern
Schweregrad 25	Zeigt an, dass ein interner Systemfehler aufgetreten ist.

Wie Sie feststellen können, sind Fehler mit einem Schweregrad größer als 18 wirklich unangenehme Fehler. Diese Fehler erfordern oft einen Kontakt zu Microsofts SQL Server Product Support.

Bedingung für »CPU im Leerlauf«: SQL Server-Agent-Aufträge können ausgeführt werden, wenn die CPU im Leerlauf arbeitet. Wenn Sie das Kontrollkästchen DER COMPUTER IST IM LEERLAUF einschalten, können Sie den Leerlaufzustand durch den Prozentsatz und die Zeitdauer charakterisieren. In der Standardeinstellung gilt der Server als im Leerlaufzustand befindlich, wenn die CPU zu weniger als 10 Prozent ausgelastet ist und dieser Zustand mindestens 10 Minuten (d.h. 600 Sekunden) dauert. Falls Sie diese Einstellungen ändern möchten, geben Sie einfach neue Werte ein.

Überlegen Sie aber genau, ob Sie diese Standardwerte ändern wollen, denn normalerweise ruft man prozessor- und festplattenintensive Tasks nur im Leerlaufzustand auf und andere Benutzer sollen nicht behindert werden, wenn diese Aufträge laufen.

Die Registerkarte Verbindung

Auf die Registerkarten WARNUNGSSYSTEM und AUFTRAGSSYSTEM geht diese Lektion später ein. Gehen Sie jetzt auf die Registerkarte VERBINDUNG, um die anfängliche Konfiguration Ihres SQL Server-Agents fertig zu stellen (siehe Abbildung 18.4).

Für den SQL Server-Agent-Dienst können Sie die SQL Server-Authentifizierung verwenden. Standardmäßig wird die Verbindung zu SQL Server auf einem Windows 2000-Computer per Windows-Authentifizierung hergestellt. Wenn Sie die SQL Server-Authentifizierung verwenden möchten oder auf einer Windows 9x-Plattform arbeiten, wählen Sie einfach die Option *SQL Server-Authentifizierung verwenden* und geben Benutzernamen und Kennwort an. In jedem Fall sollte das verwendete Konto ein Mitglied der festen Serverrolle *sysadmin* sein.

Der Anmeldetimeout spezifiziert, wie lange der SQL Server-Agent nach einem Verbindungsversuch zu SQL Server wartet, ehe er annimmt, dass SQL Server nicht aktiv ist. Der Standardwert beträgt 30 Sekunden.

Außerdem können Sie auf der Registerkarte VERBINDUNG einen alternativen Namen für den Zugang zu SQL Server angeben. Wenn Sie mit dem SQL Server-Clientkonfigurationsprogramm einen Aliasnamen für den Server konfiguriert haben (zum Beispiel, um einen nicht standardmäßigen Anschluss für TCP/IP-Sockets zu erzwingen), geben Sie diesen Aliasnamen hier an. In den meisten Umgebungen brauchen Sie diese erweiterte Konfigurationsoption nicht zu ändern.

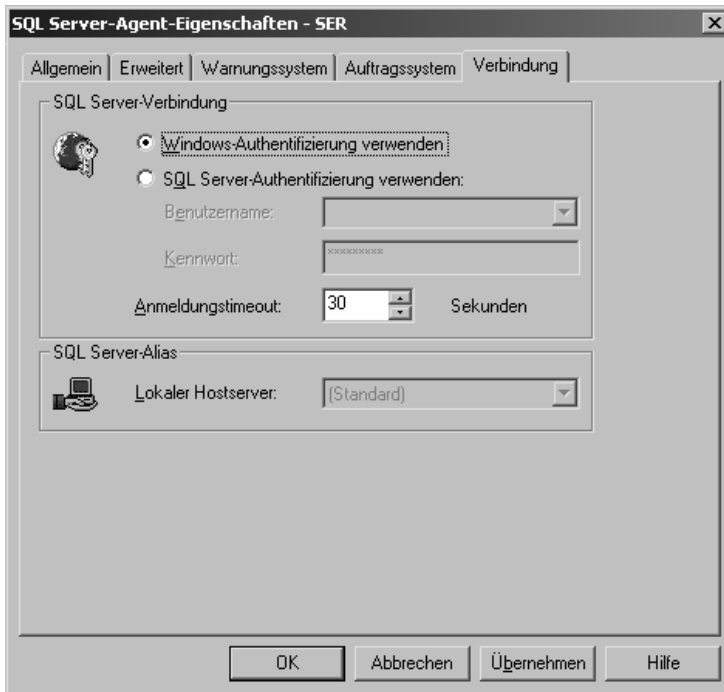


Abbildung 18.4:
Die Verbindungskonfiguration des SQL Server-Agents

SQL Mail-Integration

Wie Sie gesehen haben, sind verschiedene Konfigurationseinstellungen vorzunehmen, bevor man den SQL Server-Agent einsetzen kann. Für die meisten Fälle kann man die Standardeinstellungen übernehmen. Dennoch ist es von Vorteil, wenn man einen Überblick über die Optionen hat, bevor man mit dem SQL Server-Agent praktisch arbeitet.

Die Integration von E-Mail mit SQL Server wurde bis jetzt nur kurz erwähnt. Mit dieser Option, die aus historischen Gründen SQLMail heißt, können Sie ein Mailprofil konfigurieren, mit dem dann die SQL Server-Dienste E-Mail senden (und empfangen). Beispielsweise kann es nützlich sein, Berichte über die erfolgreiche Fertigstellung von Aufträgen zu erhalten. Mit SQL Server lassen sich auch Pagernachrichten versenden, was vor allem bei ernsthaften Problemen infrage kommt.

In SQL Server 2000 können Sie zwei unterschiedliche SQL Mail-Konfigurationen einrichten:

- Der SQL Server-Dienst kann erweiterte gespeicherte Prozeduren verwenden, um E-Mail zu senden und zu empfangen. Diese Mailkonfiguration ist die SQL Mail-Option, die Sie im Ordner *Unterstützungsdienste* des Enterprise Managers finden.

- Die andere E-Mail-Konfiguration basiert auf der Nutzung des SQL Server-Agent-Dienstes. Das ist die eigentlich interessante Option von SQLMail – die Fähigkeit von SQL Server, automatisch eine Benachrichtigung zu schicken, wenn die oben besprochenen Bedingungen zutreffen.

Wenn Ihre SQL Server- und SQL Server-Agent-Dienste unter derselben Windows 2000-Kontokonfiguration für ihre jeweiligen Dienste laufen, können Sie für beide Dienste ein einziges Mailprofil konfigurieren und nutzen. Das ist die empfohlene Konfiguration, und wird für die Beispiele im Buch auch angenommen.

Wenn Ihre SQL Server-Dienste das Konto »Lokales System« verwenden, sollten Sie das jetzt in ein Benutzerkonto umstellen. Sie haben gerade gelernt, wie Sie den SQL Server-Agent-Dienst anpassen; wie Sie das Dienstkonto für den SQL Server-Dienst ändern, hat Tag 2 behandelt. Wenn Sie das Windows 2000-Konto noch nicht eingerichtet haben, konsultieren Sie am besten noch einmal Lektion 2, um ein Konto für die Ausführung der SQL Server-Dienste einzurichten und zu konfigurieren.

Konfiguration eines Microsoft Outlook-Nachrichtenprofils

Wenn Sie Ihr Dienstkonto ausgewählt haben, melden Sie sich mit den Sicherheitsinformationen des Dienstkontos zuerst am Windows 2000-Computer an, auf dem SQL Server läuft. Haben Sie die Setupkonfiguration von Tag 2 übernommen, melden Sie sich als SQLDienst mit dem Kennwort kennwort an. Die SQL Server-Dienste brauchen Sie, um den Namen des zu verwendenden Mailprofils zu spezifizieren. Jedes gewählte Mailprofil beruht auf dem Profil des Benutzers, der für die Dienste konfiguriert ist. Deshalb müssen Sie sich mit diesem Konto anmelden.



Abbildung 18.5:
Das Standardeigenschaftsdialogfeld für Mail

Klicken Sie mit der rechten Maustaste auf das Symbol *Posteingang* oder auf das Symbol *Microsoft Outlook*. Dieses Buch bezieht sich auf Microsoft Outlook 2000, den derzeit neuesten Mailclient von Microsoft. Wählen Sie **EIGENSCHAFTEN**, um das Dialogfeld gemäß Abbildung 18.5 zu öffnen.

Klicken Sie nun auf die Schaltfläche **HINZUFÜGEN**, um festzulegen, ob Sie Internet-E-Mail oder Microsoft Exchange Server verwenden möchten (siehe Abbildung 18.6). Diese Optionen müssen Sie bei der Installation von Microsoft Outlook installiert haben, damit sie in diesem Dialogfeld erscheinen.



Abbildung 18.6:
Hier wählen Sie den Typ
des verfügbaren E-Mail-
Servers aus.

Microsoft Exchange Server

Wenn Sie Microsoft Exchange Server ausgewählt haben, erscheint das in Abbildung 18.7 dargestellte Dialogfeld. Tragen Sie einfach den Namen des Microsoft Exchange Server-Computers ein und geben Sie entweder Ihren vollen Namen oder Ihren E-Mail-Alias an.

Klicken Sie auf **WEITER**, um zum nächsten Dialogfeld (siehe Abbildung 18.8) zu gelangen. Wenn Sie die Frage *Verwenden Sie diesen Computer auf Reisen?* mit *Ja* beantworten, wird eine Kopie Ihrer Mail auf Ihrem Computer gespeichert. Wenn Sie *Nein* (Standard) wählen, bleibt Ihre Mail nur auf Ihrem Microsoft Exchange Server.

Klicken Sie auf **WEITER**, um die Installation abzuschließen. Das Dialogfeld gemäß Abbildung 18.9 zeigt, dass Sie Microsoft Exchange Server sowie das Outlook-Adressbuch erfolgreich konfiguriert haben. Outlook fügt das Outlook-Adressbuch automatisch hinzu.



Abbildung 18.7:
Microsoft Exchange Server-Konfiguration



Abbildung 18.8:
Verwenden Sie diesen Computer auf Reisen?

Klicken Sie auf **FERTIG STELLEN**, damit das Profil konfiguriert wird. Wenn Ihnen der von Outlook vorgegebene Standardname für das Profil nicht zusagt, klicken Sie auf **KOPIEREN** und geben als neuen Profilnamen zum Beispiel **SQLMail** ein. Klicken Sie dann auf **OK** und ändern Sie die Option unter *Beim Start von Microsoft Outlook dieses Profil verwenden* in den neu gewählten Namen **SQLMail**.



Abbildung 18.9:
Bestätigungsdialogfeld für
die Konfiguration

Internet-Mail

Nicht jeder baut auf Microsoft Exchange Server als E-Mail-Server. Eine andere populäre Möglichkeit ist es, einen Internet-Mail-Provider zu nutzen. Wenn Sie die Option Internet-E-Mail (siehe Abbildung 18.6) wählen und auf WEITER klicken, gelangen Sie zum Dialogfeld gemäß Abbildung 18.10. Klicken Sie auf die Schaltfläche E-MAIL-KONTO einrichten, um das Eigenschaftsdialogfeld entsprechend Abbildung 18.11 zu öffnen.



Abbildung 18.10:
Vorbereitung für die Kon-
figuration eines Internet-
E-Mail-Kontos

Mail Server Eigenschaften

Allgemein | Server | Verbindung | Erweitert

E-Mail-Konto

Geben Sie einen Namen für diesen Server ein. Zum Beispiel: "Arbeit" oder "Microsoft Mail Server".

Mail Server

Benutzerinfo

Name: Richard Waymire

Organisation:

E-Mail-Adresse: rwaymi@hotmail.com

Antwortadresse: rwaymi@hotmail.com

OK Abbrechen Übernehmen

Abbildung 18.11:
Geben Sie hier Ihre E-Mail-Kontoinformationen ein.

Abbildung 18.11 zeigt als Beispiel den Namen und die E-Mail-Adresse des Autors. Auf jeden Fall müssen Sie hier zunächst die Informationen für das Konto eingeben, für das Sie sich entschieden haben, bevor Sie auf die Registerkarte SERVER klicken. Geben Sie also erst die für Sie zutreffenden Informationen ein und klicken Sie dann auf die Registerkarte SERVER (siehe Abbildung 18.12).

Geben Sie hier den Namen Ihres SMTP-Servers für die ausgehende Post, des POP3-Servers für die eingehende Post und Ihre Anmeldeinformationen ein. Diese Angaben erhalten Sie von Ihrem Internetdienstanbieter (ISP, Internet Service Provider). Nachdem Sie diese Informationen eingetragen haben, können Sie auf die Registerkarte VERBINDUNG wechseln (siehe Abbildung 18.13).

Wenn Sie sich in einem lokalen Netzwerk (LAN) befinden, übernehmen Sie einfach die Standardoption. Arbeiten Sie von zu Hause aus und möchten, dass SQL Server automatisch Ihren Internetdienstanbieter anwählt, wenn eine Mail zu senden ist, wählen Sie die Option *Verbindung per Telefon herstellen*. Soll SQL Server die Verbindung nur herstellen, wenn Sie sich manuell in das Internet eingewählt haben, nehmen Sie die Option *Internet-Verbindung manuell herstellen*. Wenn Sie sich nicht in das Internet eingewählt haben, protokolliert SQL Server einen Fehler, wenn er die Verbindung zu Ihrem Mail-Server herzustellen versucht.

Mail Server Eigenschaften

Allgemein | Server | Verbindung | Erweitert

Serverinformationen

Posteingang (POP3):

Postausgang (SMTP):

Posteingangsserver

Benutzername:

Kennwort:

☒ Kennwort speichern

☐ Anmeldung durch gesicherte Kennwort-Authentifizierung

Postausgangsserver

☐ Server erfordert Authentifizierung

Abbildung 18.12:
Geben Sie hier die Informationen für Ihren E-Mail-Server ein.

Mail Server Eigenschaften

Allgemein | Server | Verbindung | Erweitert

Geben Sie die Art der Verbindung zum Senden und Empfangen von E-Mail an.

Verbindung

☒ LAN-Verbindung verwenden

☐ Verbindung per Telefon herstellen

☐ Internet-Verbindung manuell herstellen

Modem

Folgende D\FÜ-Verbindung verwenden:

Abbildung 18.13:
Die Verbindungsinformationen für Ihr Internet-E-Mail-Profil

Klicken Sie auf OK, wenn Sie die gewünschte Option gewählt haben. Gegebenenfalls erhalten Sie die Möglichkeit, Ihr persönliches Adressbuch zu konfigurieren. Ist das Adressbuch bereits vorhanden, geben Sie hier den Dateinamen ein, oder Sie übernehmen die Standardeinstellung und legen ein neues Adressbuch an.

Klicken Sie auf WEITER und Sie werden aufgefordert, einen vorhandenen Satz persönlicher Ordner auszuwählen oder einen neuen Satz zu erstellen. Ihre Internet-E-Mail wird in dem Ordner abgelegt, den Sie in diesem Konfigurationsfenster angegeben haben.

Klicken Sie erneut auf WEITER. Es erscheint ein Dialogfeld mit der Frage, ob Outlook bei jedem Anmelden in Windows gestartet werden soll. Bestätigen Sie das standardmäßige *Nein* und klicken Sie auf WEITER.

Damit haben Sie das letzte Dialogfeld erreicht. Es zeigt an, dass Sie Ihren Internet-E-Mail-Server sowie das Outlook-Adressbuch und die persönlichen Ordner erfolgreich konfiguriert haben. Outlook fügt ebenfalls Ihr Adressbuch hinzu.

Klicken Sie auf FERTIG STELLEN, um das Profil zu speichern. Beachten Sie, dass der Standardname *MS Exchange-Einstellungen* lautet. Hier können Sie wieder den Namen ändern, wie es weiter oben beschrieben wurde. Markieren Sie dazu das Profil *MS Exchange-Einstellungen* und klicken Sie auf die Schaltfläche KOPIEREN. Geben Sie einen Namen wie zum Beispiel *SQLMail* ein. Klicken Sie auf OK und schließen Sie das Dialogfeld MAIL. Wenn Sie jetzt nach einem E-Mail-Profilnamen gefragt werden, können Sie *SQLMail* eingeben.

SQL Mail für SQL Server konfigurieren

Nachdem Sie nun über ein Mailprofil verfügen, ist es an der Zeit, SQL Mail für SQL Server zu konfigurieren. Als Erstes konfigurieren Sie SQL Mail, um den Einsatz von mailbezogenen erweiterten gespeicherten Prozeduren zu unterstützen. Wechseln Sie zum SQL Server Enterprise Manager zurück, öffnen Sie den Ordner Unterstützungsdienste und klicken Sie mit der rechten Maustaste auf SQL Mail. Wählen Sie EIGENSCHAFTEN und vervollständigen Sie das Dialogfeld SQL Mail-Konfiguration (siehe Abbildung 18.14).

Geben Sie den SQL Mail-Profilnamen ein, wie ihn Abbildung 18.14 zeigt. Um zu testen, ob alles funktioniert, klicken Sie auf die Schaltfläche TESTEN.

Nachdem Sie die SQL Mail-Konfiguration abgeschlossen und erfolgreich Ihr Profil getestet haben, klicken Sie auf OK.

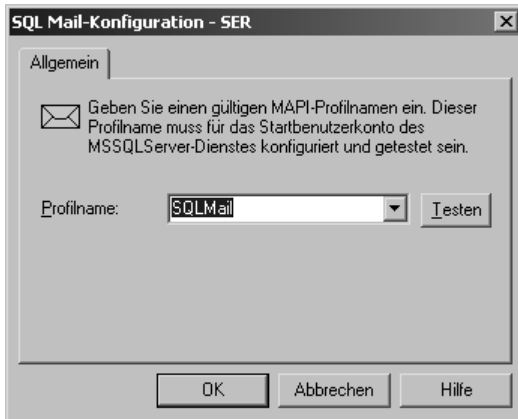


Abbildung 18.14:
Das fertig gestellte Dialogfeld SQL Mail-Konfiguration

SQL Mail für den SQL Server-Agent konfigurieren

Um das gleiche Profil für den SQL Server-Agent zu konfigurieren, klicken Sie mit der rechten Maustaste auf den Agent (im Ordner *Verwaltung*) und wählen **EIGENSCHAFTEN** aus dem Kontextmenü. Im Abschnitt **MAILSITZUNG** wählen Sie *SQLMail* aus der Drop-down-Liste und klicken auf die Schaltfläche **TESTEN**, um zu überprüfen, dass auch hier alles ordnungsgemäß funktioniert. Wenn Sie irgendeine Fehlermitteilung erhalten, überprüfen Sie, ob Ihr SQL Server-Agent-Dienst dasselbe Konto benutzt wie Ihr SQL Server-Dienst. Wenn Sie auf **OK** klicken, fordert Sie der Agent zum Neustart auf, damit die vorgenommenen Änderungen wirksam werden. Wählen Sie **JA**; daraufhin hält der SQL Server-Agent-Dienst an und startet neu. SQL Mail ist nun voll funktionstüchtig.

18.3 Das Senden von E-Mail vom SQL Server

Um E-Mails von SQL Server über eine Transact-SQL-Abfrage zu senden, verwenden Sie die erweiterte gespeicherte Prozedur `xp_sendmail`:



```
xp_sendmail [[@recipients =] 'recipients [;...n]']
[, [[@message =] 'message']]
[, [[@query =] 'query']]
[, [[@attachments =] attachments]]
[, [[@copy_recipients =] 'copy_recipients [;...n]']]
[, [[@blind_copy_recipients =] 'blind_copy_recipients [;...n]']]
[, [[@subject =] 'subject']]
[, [[@type =] 'type']]
```

```
[, [@attach_results =] {'true' | 'false' | 'NULL'}]
[, [@no_output =] {'true' | 'false'}]
[, [@no_header =] {'true' | 'false'}]
[, [@width =] width]
[, [@separator =] separator]
[, [@echo_error =] {'true' | 'false'}]
[, [@set_user =] 'user']
[, [@dbuse =] 'dbname']
```

Von den zahlreichen Optionen in dieser Syntax konzentrieren wir uns hier auf die wichtigsten Argumente:

- **@recipients= recipients** spezifiziert den Namen des E-Mail-Empfängers. Wenn Sie mehrere Empfänger festlegen wollen, setzen Sie einfach Semikolons dazwischen.
- **@message=message** enthält den Text, der im Rumpf der gesendeten Nachricht eingeschlossen ist.
- **@query = query** spezifiziert eine auszuführende Abfrage, deren Ergebnis in einer E-Mail an den Benutzer gesendet wird, der im Parameter **@recipients** spezifiziert ist.
- **@attachments = attachments** gibt eine Liste der Dateien (mit Pfad und Name) an, die an die E-Mail-Nachricht angehängt werden sollen.
- **@subject= subject** gibt den Betreff der E-Mail an.

Wie bei jeder gespeicherten Prozedur können Sie Parameter entweder durch ihren Namen oder über die Position übergeben. Zum Beispiel sendet die folgende Anweisung eine E-Mail mit dem Betreff »test mail« an den Empfänger »Ihre Mail«

```
xp_sendmail @recipients = 'Ihre Mail', @subject = 'test mail'
```

Die Nachricht können Sie auch ohne Angabe der Parameternamen senden, wobei Sie aber die einzelnen Werte in der richtigen Reihenfolge angeben müssen:

```
xp_sendmail 'Ihre Mail', 'Das ist eine Nachricht von SQL Server'
```

SQL Server kann auch E-Mail empfangen. Sie können eine Abfrage an SQL Server richten und die gespeicherte Prozedur **sp_processmail** aufrufen, um eingehende Mails zu verarbeiten. Wenn die Mail-Nachricht entsprechend formatiert ist, wird der Rumpf der Nachricht als Abfrage interpretiert, diese Abfrage wird ausgeführt und die Ergebnisse erhält der Benutzer als Textdateianhang zurück.



```
sp_processmail [[@subject =] 'subject'] [,[@filetype =] 'filetype']
[, [@separator =] 'separator'] [, [@set_user =] 'user']
[, [@dbuse =] 'dbname']
```

Am gebräuchlichsten sind die folgenden Parameter:

- `@subject = subject` gibt an, dass SQL Server nur E-Mails mit der spezifizierten Betreff-Information verarbeitet.
- `@filetype = filetype` legt die Dateierweiterung des Anhangs fest, der als Teil der Abfrage zurückgesandt wird. Standardmäßig wird eine `.txt`-Datei zurückgegeben.
- `@separator = separator` gibt die Art des Trennzeichens in der Abfrageausgabe an. Standard ist das Tabulatorzeichen.
- `@set_user = user` gibt den Sicherheitskontext der Abfrage an. Der Standardbenutzer ist `Guest`.
- `@dbuse = dbname` legt fest, in welcher Datenbank die Abfrage vorgenommen werden soll. Standard ist die `master`-Datenbank.

Wenn Sie also an Ihre SQL Server-Mailbox eine Nachricht mit dem Rumpf

```
select au_fname, au_lname from pubs..authors
```

geschickt haben, sich dann bei SQL Server als Mitglied der festen Serverrolle `sysadmin` anmelden und die Abfrage

```
exec sp_processmail
```

ausführen, wird die Nachricht verarbeitet, die Abfrage ausgeführt und eine E-Mail mit den Ergebnissen der Abfrage als angehängte Textdatei an Sie zurückgesendet.

18.4 Aufträge

Wie heute schon erwähnt, sind *Aufträge* SQL Server-Objekte, die Sie zum Abarbeiten von Tasks konfigurieren können. Gewöhnlich erstellt man einen Auftrag, wenn eine oder mehrere Gruppen von Befehlen wiederholt auszuführen sind.

Das Erstellen eines Auftrags erfolgt in vier Schritten: Sie legen den Auftrag an, erstellen die Auftragsschritte, richten einen Zeitplan für den Auftrag ein und legen dann die Benachrichtigungsoptionen für den Auftrag fest. Gehen Sie jetzt wieder zum SQL Server Enterprise Manager. Erweitern Sie den Ordner *Verwaltung* und dann den Knoten *SQL Server Agent*. Daraufhin erscheinen die Ordner *Warnungen*, *Operatoren* und *Aufträge*. Wenn Sie den Ordner *Aufträge* markieren, erscheint im Detailbereich eine Liste der bereits erstellten Aufträge.

Einen neuen Auftrag erstellen

Klicken Sie mit der rechten Maustaste auf das Symbol Aufträge und wählen Sie NEUER AUFTRAG aus dem Kontextmenü. Das Eigenschaftsdialogfeld für den neuen Auftrag erscheint (siehe Abbildung 18.15).

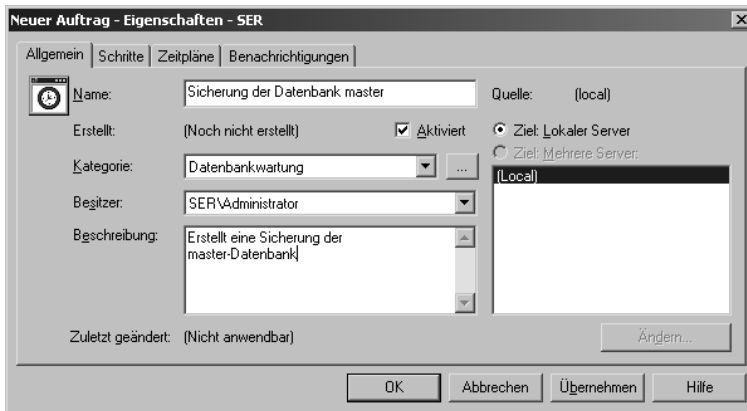


Abbildung 18.15:
Das Dialogfeld
Neuer Auftrag –
Eigenschaften

Als Erstes geben Sie dem neuen Auftrag einen Namen. Weil Sie höchstwahrscheinlich einen Auftrag für Sicherung der Datenbank *master* brauchen, nennen Sie den Auftrag Sicherung der Datenbank *master*. Aus der Dropdown-Liste KATEGORIE wählen Sie den Eintrag *Datenbankwartung* aus.

Als Besitzer des Auftrags ist standardmäßig der Benutzer eingetragen, unter dem Sie sich angemeldet haben. In SQL Server 2000 kann jeder einen Auftrag erstellen. Die Mitglieder der Rolle *sysadmin* können jedoch Aufträge erstellen, deren Besitzer andere Benutzer sind. Übernehmen Sie hier einfach den Standardwert.

Im Feld BESCHREIBUNG können Sie den Auftrag kurz erläutern.

Auf die Zielserveroption geht diese Lektion in Kürze ein; übernehmen Sie hier die Standardoption *Ziel: Lokaler Server*.

Auftragschritte

Klicken Sie auf die Registerkarte SCHRITTE, um die Erstellung der Auftragsschritte vorzubereiten (siehe Abbildung 18.16). Jeder Schritt ist ein separater Befehl oder ein Satz von Befehlen.

Klicken Sie auf die Schaltfläche NEU, um den ersten Schritt für diesen Auftrag zu erstellen. Das Dialogfeld NEUER AUFTRAGSSCHRITT wird geöffnet (siehe Abbildung 18.17).

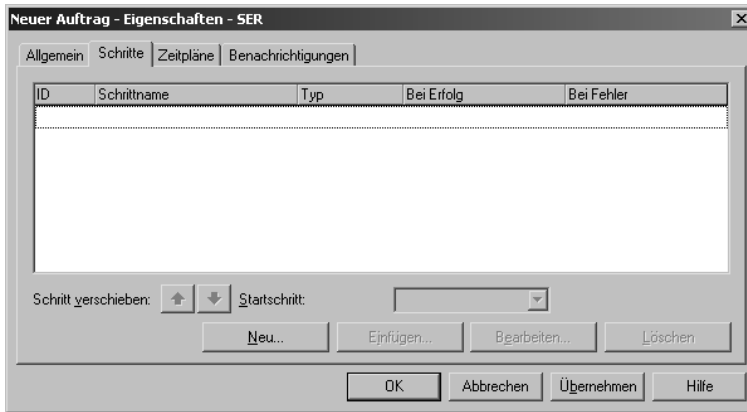


Abbildung 18.16:
Die Registerkarte
Schritte für den neu-
en Auftrag

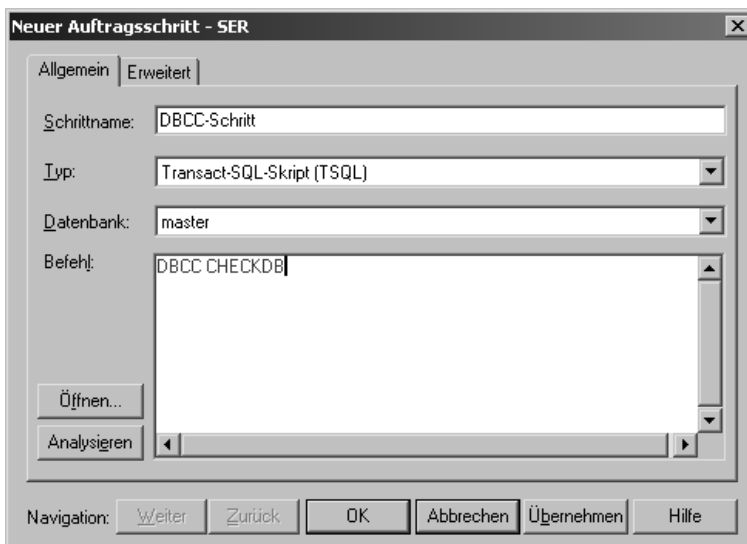


Abbildung 18.17:
Das Dialogfeld
Neuer Auftrags-
schritt

In diesem Auftrag führen Sie als Erstes `DBCC CHECKDB` für die Datenbank *master* aus. Daran schließt sich der Befehl `BACKUP DATABASE` an, der die Datenbank sichert. Allerdings soll die Sicherung nur dann ausgeführt werden, wenn die DBCC-Prüfung für die Datenbank *master* keine Fehlermeldungen geliefert hat.

Abbildung 18.17 zeigt bereits das ausgefüllte Dialogfeld für den DBCC-Schritt. Klicken Sie auf die Schaltfläche **ANALYSIEREN**, um die Syntax der eingegebenen Transact-SQL-Anweisungen zu überprüfen.

Gehen Sie nun auf die Registerkarte ERWEITERT (siehe Abbildung 18.18). Hier steuern Sie den Ablauf der Befehlsausführung abhängig von Erfolg oder Fehler und legen fest, ob die Ergebnisse der Transact-SQL-Anweisungen als Ausgabedatei zu speichern sind.

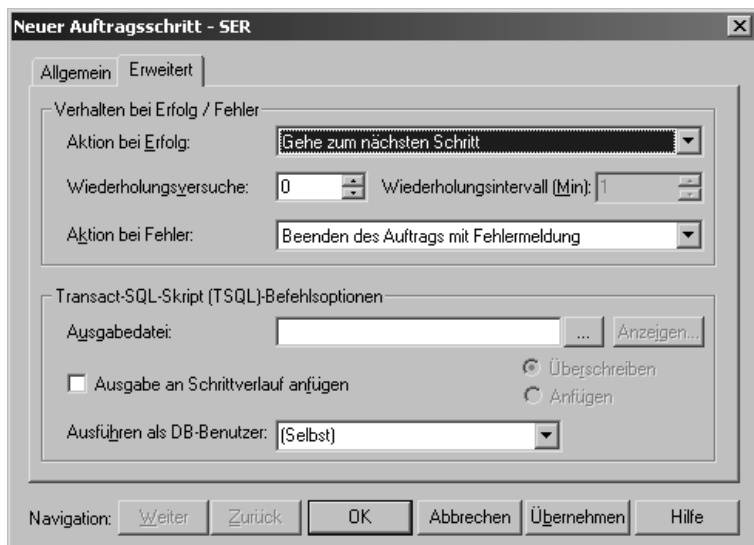


Abbildung 18.18:
Die Registerkarte
Erweitert des Dia-
logfelds Neuer Auf-
tragsschritt

Wenn Sie die Ergebnisse dieses Auftragsschrittes im Auftragsverlauf als Ganzes speichern möchten statt als einzelne Datei, schalten Sie das Kontrollkästchen AUSGABE AN SCHRITT-VERLAUF ANFÜGEN ein. Übernehmen Sie die anderen Standardwerte und klicken Sie auf OK, um diesen Auftragsschritt abzuschließen.

Fügen Sie nun einen weiteren Schritt hinzu, um den Sicherungsbefehl auszuführen. Ein gültiger Befehl sieht beispielsweise so aus:

```
BACKUP DATABASE master TO master_backup WITH INIT
```

Setzen Sie die Option *Aktion bei Erfolg* auf *Beenden des Auftrags mit Erfolgsmeldung*. Abbildung 18.19 zeigt das fertig ausgefüllte Dialogfeld für diesen Schritt.

Sie sollten auch die Dialogfelder für andere Auftragstypen untersuchen. Sehen Sie sich insbesondere auch die Auftragstypen *ActiveX-Skript* und *Betriebssystembefehl (CmdExec)* an. Das sind die drei Auftragstypen, die Sie am häufigsten benutzen werden. Beachten Sie, wie sich die Dialogfelder abhängig von der Art des ausgewählten Auftragstyps ändern.

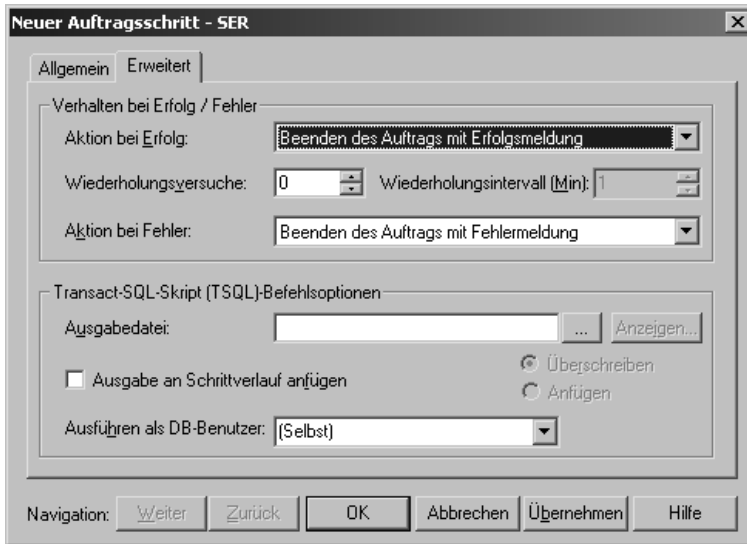


Abbildung 18.19:
Die Registerkarte
Erweitert für den
zweiten Schritt des
Sicherungsauftrages

Zeitplanung von Aufträgen

Als Nächstes müssen Sie den Zeitplan für den Auftrag definieren. Gehen Sie dazu auf die Registerkarte ZEITPLÄNE (siehe Abbildung 18.20) und klicken Sie dann auf die Schaltfläche NEUER ZEITPLAN, um den Zeitplan für diesen Auftrag einzurichten (siehe Abbildung 18.21).

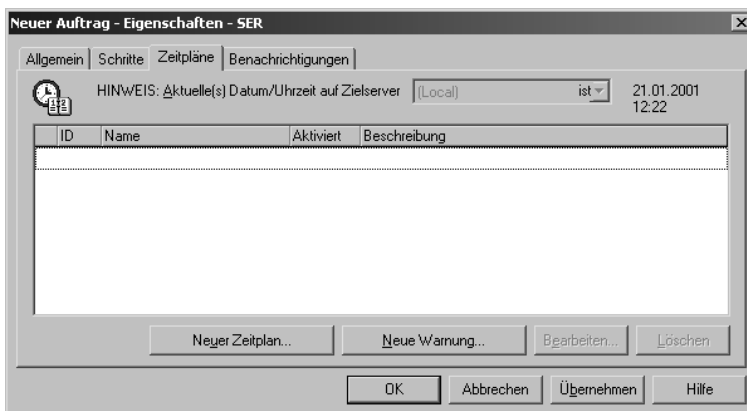


Abbildung 18.20:
Das Dialogfeld zum
Einrichten von Zeit-
plänen

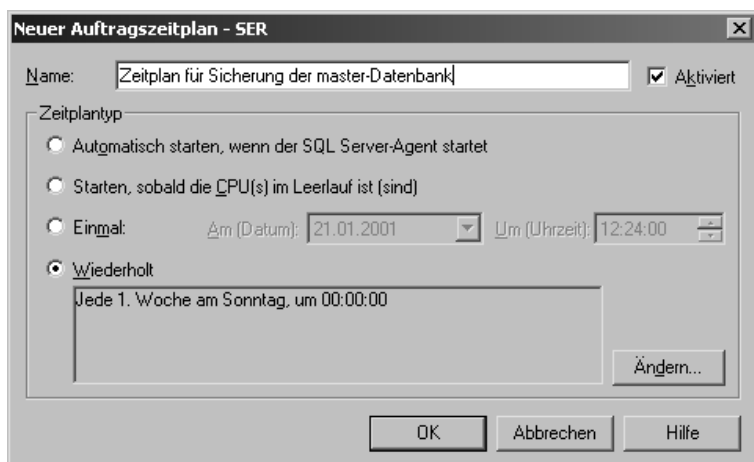


Abbildung 18.21:
Das Dialogfeld
Neuer Auftragszeit-
plan

Die Optionen sind größtenteils selbsterklärend. Sie können den Auftrag automatisch starten, wenn der SQL Server-Agent startet; Sie können die Auftragsausführung starten, wenn die CPU im Leerlauf arbeitet (den Leerlaufzustand haben Sie weiter oben in dieser Lektion definiert); oder Sie können den Auftrag einmalig ausführen lassen. Eine weitere Option besteht darin, in diesem Dialogfeld überhaupt nichts zu konfigurieren; damit startet der Auftrag nur auf Anforderung – wenn Sie ihn explizit anfordern.

Als Standard ist die Option *Wiederholt* eingestellt, bei der Sie einen festen Zeitplan für den Auftrag einrichten. Der Auftrag wird standardmäßig einmal jede erste Woche am Sonntag um Mitternacht ausgeführt. Klicken Sie auf die Schaltfläche *ÄNDERN*, um einen anderen Zeitplan festzulegen. Abbildung 18.22 zeigt dafür ein Beispiel.

Im Abschnitt *HÄUFIGKEIT* legen Sie fest, ob der Auftrag täglich, wöchentlich oder monatlich startet:

- Wenn Sie *Täglich* wählen, erscheint der Abschnitt *TÄGLICH* neben dem Abschnitt *HÄUFIGKEIT*. Damit können Sie festlegen, dass der Auftrag alle *x* Tage startet (täglich, an jedem zweiten Tag und so weiter).
- Bei der Option *Wöchentlich* erscheint der Abschnitt *WÖCHENTLICH*, in dem Sie festlegen können, dass der Auftrag alle *x* Wochen startet (mit dem Standardwert 1) und an welchen Wochentagen das geschehen soll. Dabei können Sie nach Belieben auch mehrere Wochentage markieren. Zum Beispiel lässt sich festlegen, dass ein Auftrag jede Woche von Montag bis Freitag ausgeführt werden soll.
- Wenn Sie die Option *Monatlich* wählen, können Sie festlegen, dass ein Auftrag zu einem bestimmten Tag des Monats (bzw. alle *x* Monate) oder am ersten, zweiten, dritten, vierten oder letzten Tag, Wochentag, Wochenendtag oder bestimmten Tag alle *x* Monate auszuführen ist. Diese Option ist also sehr flexibel.



Zeitplan für wiederkehrende Aufträge bearbeiten - SER

Auftragsname: (Neuer Auftrag)

Häufigkeit

☒ Täglich
☐ Wöchentlich
☐ Monatlich

Täglich: Alle 1 Tag(e)

Häufigkeit pro Tag

☒ Einmalig um: 06:00:00
☐ Jede: 1 Stunde(n) Startet um: 00:00:00 Endet um: 23:59:59

Dauer

Anfangsdatum: 21.01.2001 ☐ Enddatum: 21.01.2001 ☒ Kein Enddatum

OK Abbrechen Hilfe

Abbildung 18.22:
Das Dialogfeld Zeitplan für wiederkehrende Aufträge bearbeiten

Im Abschnitt HÄUFIGKEIT PRO TAG legen Sie fest, zu welcher Tageszeit der Auftrag starten soll. Mit der Option *Jede* können Sie auch die Ausführung alle x Stunden oder Minuten spezifizieren. Außerdem ist es möglich, die Start- und Endzeiten festzulegen.

Der Abschnitt DAUER erlaubt die Einstellung des Anfangsdatums (standardmäßig der aktuelle Tag) und des Enddatums (bzw. *Kein Enddatum*). Wenn Sie einen geplanten Auftrag nur für eine bestimmte Zeitdauer aktivieren und dann stoppen wollen, legen Sie das Enddatum fest. Wenn dieser Tag erreicht ist, wird der Auftrag nicht wieder gestartet.



Neuer Auftrag - Eigenschaften - SER

Allgemein Schritte Zeitpläne Benachrichtigungen

HINWEIS: Aktuelle(s) Datum/Uhrzeit auf Zielsystem [Local] ist 21.01.2001 13:03

ID	Name	Aktiviert	Beschreibung
(Neu)	Zeitplan master-Sicherung	Ja	Jeden 1. Tag, um 06:00:00

Neuer Zeitplan... Neue Warnung... Bearbeiten... Löschen

OK Abbrechen Übernehmen Hilfe

Abbildung 18.23:
Die fertig gestellte Registerkarte Zeitpläne

Im Beispiel (siehe Abbildung 18.22) wird der Auftrag täglich je einmal um 06:00 Uhr ausgeführt. Klicken Sie auf OK, um zum Dialogfeld NEUER AUFTRAGSZEITPLAN zurückzukehren. Geben Sie dem Zeitplan einen Namen (im Beispiel »Zeitplan master-Sicherung«) und klicken Sie schließlich auf OK, um zur Registerkarte ZEITPLÄNE zurückzukehren, welche jetzt Abbildung 18.23 entsprechen sollte.

Auftragsbenachrichtigungen

Auf der Registerkarte BENACHRICHTIGUNGEN (siehe Abbildung 18.24) können Sie festlegen, ob ein Operator per E-Mail, Pager oder (nur bei Windows NT/2000) mit einem NET SEND-Befehl zu benachrichtigen ist und ob das Ergebnis des Auftrags in das Windows 2000-Ereignisprotokoll geschrieben werden soll. Für jede Option können Sie angeben, ob die Benachrichtigung bei Auftragsfehler, bei erfolgreicher Auftragsausführung oder bei Auftragsbeendigung (d.h. immer) stattfinden soll.

Es lässt sich auch festlegen, ob der Auftrag gelöscht werden soll, wobei auch hier die drei genannten Optionen verfügbar sind. Konfigurieren Sie das Dialogfeld, wie in Abbildung 18.24 gezeigt. Die E-Mail-Einstellungen für diesen Auftrag legen Sie etwas später fest – nachdem Sie einen E-Mail-Operator hinzugefügt haben. Klicken Sie auf OK und Sie haben Ihren ersten Auftrag erfolgreich erstellt.

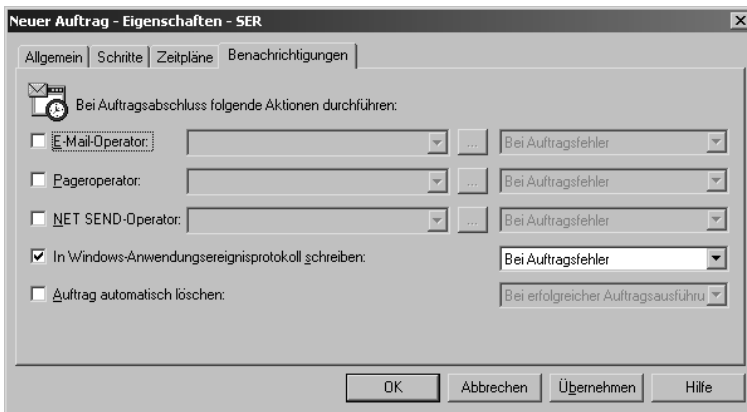


Abbildung 18.24:
Die fertig gestellte
Registerkarte
Benachrichtigungen

Auftragssystem-Optionen

Einige Optionen treffen auf alle Aufträge zu. Klicken Sie mit der rechten Maustaste auf Ihren SQL Server-Agent, wählen Sie EIGENSCHAFTEN. und gehen Sie auf die Registerkarte AUFTRAGSSYSTEM (siehe Abbildung 18.25)

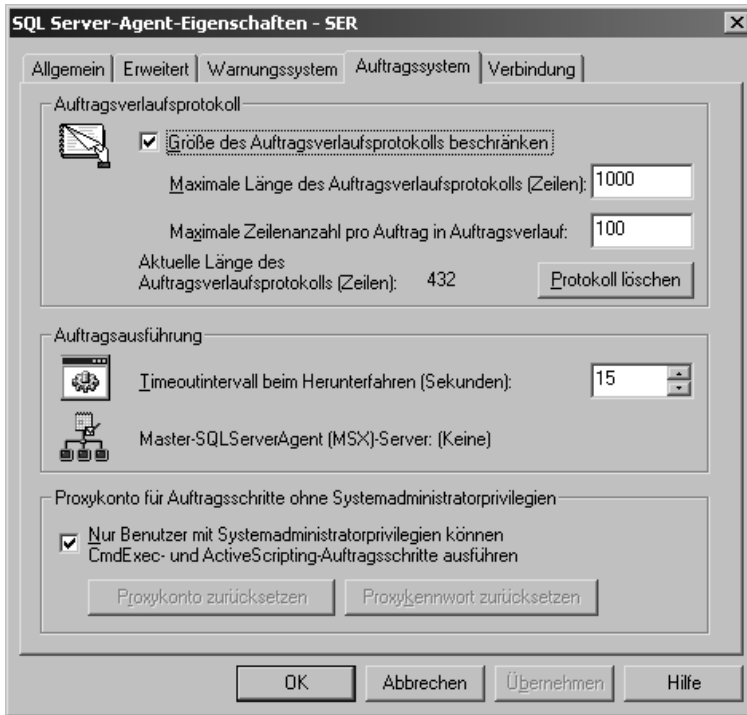


Abbildung 18.25:
Die Registerkarte
Auftragssystem

Hier können Sie die Größe des Auftragsverlaufsprotokolls für alle geplanten Aufträge Ihrer SQL Server-Installation beschränken – und zwar entweder über die maximale Länge des Auftragsverlaufsprotokolls (Standard ist 1000 Zeilen) oder über die maximale Zeilenanzahl pro Auftrag im Auftragsverlauf (Standardwert 100 Zeilen).

Wenn Sie diese Option deaktivieren, wird der Verlauf Ihrer Tasks bis in alle Ewigkeit aufgezeichnet. Der Auftragsverlauf steht in der Tabelle *sysjobhistory* der Datenbank *msdb*. Wenn Sie deshalb diese Parameter ändern oder die Größenbegrenzung ausschalten, müssen Sie höchstwahrscheinlich die Größe der Datenbank *msdb* erhöhen. Die Auftragsdefinitionen sind in den Systemtabellen *sysjobs*, *sysjobschedules*, *sysjobsteps* und *sysjobservers* der Datenbank *msdb* abgelegt.

Im Abschnitt **AUFTRAGSAUSFÜHRUNG** legen Sie die Zeit fest, die der SQL Server-Agent auf die Fertigstellung von Aufträgen wartet, nachdem Sie das Anhalten des Dienstes angefordert haben. Der Standardwert ist 15 Sekunden. Die andere Option in diesem Abschnitt gehört zu den Multiserver-Operationen, die in Kürze besprochen werden.

Schließlich ist das letzte Kontrollkästchen eingeschaltet, so dass nur Mitglieder der festen Serverrolle *sysadmin* das Recht erhalten, Aufträge vom Typ *ActiveXScripting* oder *CmdExec* auszuführen. Dieses Kontrollkästchen sollten Sie deshalb aktivieren, weil diese Auftrags-

typen den uneingeschränkten Zugriff auf den Server (oder sogar das Netzwerk) erlauben, als würde es sich um das Konto handeln, mit dem Sie SQL Server ausführen. Durch die angegebene Einschränkung legen Sie fest, dass nur der SQL Server-Administrator diese Art des Zugriffs hat. Es empfiehlt sich also, dieses Kontrollkästchen immer einzuschalten.

Klicken Sie auf OK, um die Auftragssystem-Optionen zu speichern.

Multiserveraufträge

Auf den letzten Seiten werden Sie verschiedene Optionen bemerkt haben, die sich auf Multiserver-Aufträge bezogen. Das ist ein erweitertes Merkmal und wird im Allgemeinen nicht benutzt, sofern nicht mehr als drei SQL Server im Netzwerk laufen. Für weitere Informationen zu Multiserveraufträgen sei daher auf die Onlinedokumentation oder die einschlägige Literatur verwiesen.

18.5 Warnungen und Operatoren

Warnungen erlauben es Ihnen, Ereignisse zu spezifizieren, die der SQL Server überwachen soll. Wenn diese Ereignisse eintreten, kann die Warnung je nach Anforderung eine E-Mail- oder Pagnachricht verschicken, auf Windows 2000-Systemen einen NET SEND-Befehl absetzen oder einen Auftrag starten. Einige Warnungen sind vorkonfiguriert, wenn Sie SQL Server installieren, andere Warnungen können Sie selbst hinzufügen.

Konfigurieren von Operatoren

Um einen Operator zu konfigurieren, klicken Sie mit der rechten Maustaste auf den Eintrag *Operatoren* unter dem SQL Server-Agent und wählen NEUER OPERATOR aus dem Kontextmenü. Daraufhin öffnet sich das in Abbildung 18.26 dargestellte Dialogfeld.

Geben Sie einen Namen für den Operator an und klicken Sie dann auf die Schaltfläche mit den drei Punkten neben dem E-Mail-Namen, um eine E-Mail-Adresse für Ihren Operator hinzuzufügen. Die gleichen Optionen treffen auf den Pager-E-Mail-Namen zu, weil er auf den E-Mail-Möglichkeiten von SQL Server 2000 basiert. Wenn Sie auf die Schaltfläche mit den drei Punkten neben dem Bearbeitungsfeld klicken, erscheint Ihr persönliches Adressbuch. Wenn es dort keine Einträge gibt (weil es zum Beispiel neu ist), klicken Sie auf die Schaltfläche NEU, um einen neuen Operator hinzuzufügen. Wählen Sie *Internet-Mail-Adresse* und komplettieren Sie alle gewünschten Optionen. Als Minimum müssen Sie einen Namen und eine E-Mail-Adresse eintragen.

Eigenschaften von neuem Operator - SER

Allgemein | Benachrichtigungen

 Name: ID : Neu

E-Mail-Name: ... Testen

Pager-E-Mail-Name: ... Testen

NET SEND-Adresse: Testen

Pager empfangsbereit am

☒ Montag		
☒ Dienstag		
☒ Mittwoch	Arbeitstag - Beginn	Arbeitstag - Ende
☒ Donnerstag		
☒ Freitag	08:00:00	18:00:00
☐ Samstag	08:00:00	18:00:00
☐ Sonntag	08:00:00	18:00:00

OK Abbrechen Übernehmen Hilfe

Abbildung 18.26:
Das Dialogfeld Eigenschaften von neuem Operator

Eigenschaften von neuem Operator - SER

Allgemein | Benachrichtigungen

 Name: ID : Neu

E-Mail-Name: ... Testen

Pager-E-Mail-Name: ... Testen

NET SEND-Adresse: Testen

Pager empfangsbereit am

☒ Montag		
☒ Dienstag		
☒ Mittwoch	Arbeitstag - Beginn	Arbeitstag - Ende
☒ Donnerstag		
☒ Freitag	08:00:00	18:00:00
☐ Samstag	08:00:00	18:00:00
☐ Sonntag	08:00:00	18:00:00

OK Abbrechen Übernehmen Hilfe

Abbildung 18.27:
Eine neue E-Mail-Adresse für einen Benutzer hinzufügen

Klicken Sie auf HINZUFÜGEN, um die E-Mail-Adresse Ihres Operators in das Dialogfeld EIGENSCHAFTEN VON NEUEM OPERATOR (sowie in das persönliche Adressbuch von SQL Server) zu übernehmen. Klicken Sie im Dialogfeld (siehe Abbildung 18.27) auf OK und Sie haben erfolgreich einen neuen Operator erstellt. Wenn Sie einen Pagereintrag erzeugen, können Sie auch den Zeitplan für den Operator definieren, zu dem dieser per Pager benachrichtigt wird. Die Registerkarte BENACHRICHTIGUNGEN brauchen Sie jetzt noch nicht anzuklicken, weil damit die Warnungen geprüft werden, die diesem Operator zugeordnet sind. Wie Sie Warnungen konfigurieren, zeigt der nächste Abschnitt.

Warnungen erstellen

Markieren Sie den Eintrag *Warnungen* unter dem Knoten *SQL Server-Agent*. Daraufhin erscheinen im Detailbereich mehrere Warnungen, die auf Ihrem Server vorkonfiguriert sind, um die schwerwiegendsten Fehler abzudecken, die in SQL Server auftreten können (siehe Abbildung 18.28). Außerdem zeigt die Abbildung Warnungen, die beim Einrichten der Replikation in Lektion 17 erstellt wurden.

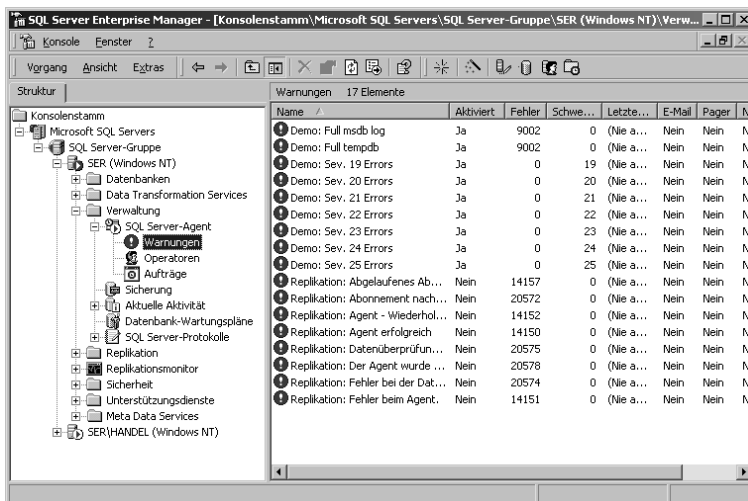


Abbildung 18.28:
Vorkonfigurierte
Warnungen von
SQL Server und die
bereits erstellten
Replikationswar-
nungen

Um eine neue Warnung zu erstellen, klicken Sie mit der rechten Maustaste auf den Eintrag *Warnungen* unter dem Knoten *SQL Server-Agent* und wählen *NEUE WARNUNG* aus dem Kontextmenü. Daraufhin öffnet sich das Dialogfeld *NEUE WARNUNG – EIGENSCHAFTEN* (siehe Abbildung 18.29).

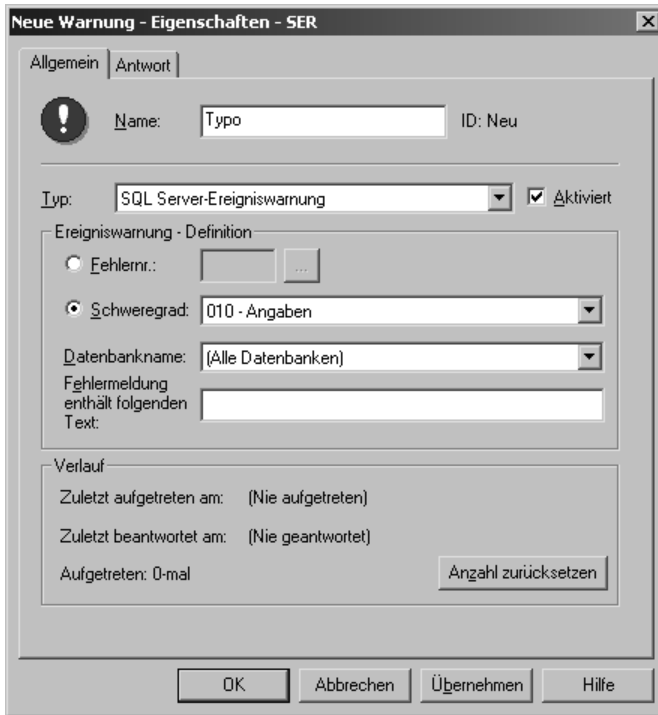


Abbildung 18.29:
Das Dialogfeld Neue Warnung – Eigenschaften

Im Beispiel von Abbildung 18.29 wurde bereits begonnen, eine Warnung mit dem Namen Typo zu erstellen. Wählen Sie im Abschnitt EREIGNISWARNUNG – DEFINITION die Option *Fehlernummer* und klicken Sie auf die Schaltfläche mit den drei Punkten, um nach einer Fehlermeldung, auf die diese Warnung achten soll, zu suchen. Um das Ganze nachvollziehbar zu gestalten, konfigurieren Sie Fehler 208 (ungültiger Objektname in einer Abfrage). Geben Sie 208 in das Bearbeitungsfeld FEHLERNUMMER ein und klicken Sie auf SUCHEN. Es erscheint das Dialogfeld SQL SERVER-MELDUNGEN VERWALTEN. Klicken Sie dort auf die Schaltfläche BEARBEITEN, um das Dialogfeld SQL SERVER-MELDUNG BEARBEITEN (siehe Abbildung 18.30) zu öffnen. Hier müssen Sie eine Standardoption für diese Fehlermeldung ändern.

Der SQL Server-Agent kann Nachrichten nur sehen, wenn diese in das Windows 2000-Ereignisprotokoll geschrieben werden (oder dieses Verhalten bei Windows 9x-Plattformen simulieren). Schalten Sie also das Kontrollkästchen IMMER IN WINDOWS-EREIGNISPROTOKOLL SCHREIBEN ein, wie es Abbildung 18.30 zeigt. Klicken Sie auf OK und dann nochmals auf OK, um die Auswahl dieses Fehlers abzuschließen. Ändern Sie den Datenbanknamen in *pubs*, damit Sie nicht mit dieser Fehlermeldung überhäuft werden, wenn andere Benutzer mit Ihrem SQL Server arbeiten. Die vollständig konfigurierte Warnung sollte wie die in Abbildung 18.31 aussehen.

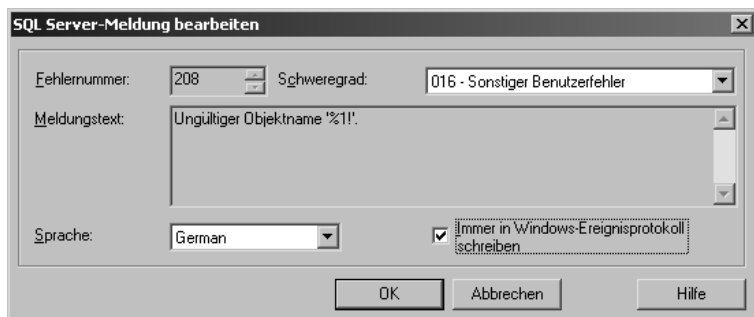


Abbildung 18.30:
Das Dialogfeld
SQL Server-Mel-
dung bearbeiten



Abbildung 18.31:
Die fertig konfigurierte War-
nung

Gehen Sie auf die Registerkarte ANTWORT (siehe Abbildung 18.32). Konfigurieren Sie die Optionen, wie es die Abbildung zeigt: Richard Waymire erhält eine Benachrichtigung, wenn diese Warnung auftritt. Beachten Sie, dass sich mit der Warnung auch ein Auftrag starten lässt. Schalten Sie dazu das Kontrollkästchen AUFTRAG AUSFÜHREN ein und wählen Sie einen Auftrag aus der Dropdown-Liste aus. Sie können von hier aus auch das Dialogfeld NEUER AUFTRAG – EIGENSCHAFTEN öffnen. Im Beispiel startet die Warnung den Auftrag zur Sicherung der Datenbank *master*, wenn der spezifizierte Fehler auftritt. Klicken Sie auf OK, wenn Sie die Antwortoptionen konfiguriert haben.

Neue Warnung - Eigenschaften - SER

Allgemein Antwort

☒ Auftrag ausführen: Sicherung der Datenbank master

Zu benachrichtigende Operatoren: Neuer Operator...

Operatormame	E-Mail	Pager	NET SEND
Richard Waymire	☒	☐	☐

Fehlermeldungstext einschließen in: ☒ E-Mail ☐ Pager ☒ NET SEND

Zu sendende zusätzliche Warnbenachrichtigung:

Antwortverzögerung: 1 Minuten 0 Sekunden

OK Abbrechen Übernehmen Hilfe

Abbildung 18.32:
Die Optionen für eine Warnung auf der Registerkarte Antwort

Testen der Warnung

Jetzt ist es an der Zeit, diese Warnung zu testen. Starten Sie den SQL Server Query Analyzer, wechseln zur Datenbank *pubs* und starten die folgende Abfrage:



```
Select * from nixda
```

Als Ergebnis sollten Sie die folgende Meldung erhalten:



```
Server: Nachr.-Nr. 208, Schweregrad 16, Status 1, Zeile 1  
Ungültiger Objektname 'nixda'.
```

Diese Ausgabe aktiviert Ihre Warnung, die daraufhin eine E-Mail schickt und den Auftrag zum Sichern der Datenbank *master* startet. Wechseln Sie zum Enterprise Manager, markieren Sie den Knoten *Warnungen* und sehen Sie sich im Detailbereich den Eintrag zur Warnung *Typo* an. In der Spalte *Anzahl* ist jetzt die Zahl 1 zu sehen (d.h., der Fehler ist

einmal aufgetreten und hat diese Warnung aktiviert). Sie können auch überprüfen, ob Sie eine E-Mail von SQL Server bekommen haben. Wenn Sie den Ordner *Aufträge* markieren, können Sie sich auch davon überzeugen, dass der Sicherungsauftrag ausgeführt wurde. Falls die geänderten Werte nicht zu sehen sein sollten, klicken Sie mit der rechten Maustaste in der Konsolenstruktur auf den Eintrag *Warnungen* bzw. *Aufträge* und wählen AKTUALISIEREN aus dem Kontextmenü.

Warnungen werden in der Systemtabelle *sysalerts* der Datenbank *msdb* gespeichert.

Warnungssystem-Optionen

Klicken Sie jetzt mit der rechten Maustaste auf *SQL Server-Agent*, wählen Sie EIGENSCHAFTEN und wechseln Sie im Dialogfeld SQL SERVER-AGENT-EIGENSCHAFTEN zur Registerkarte WARNUNGSSYSTEM (siehe Abbildung 18.33).

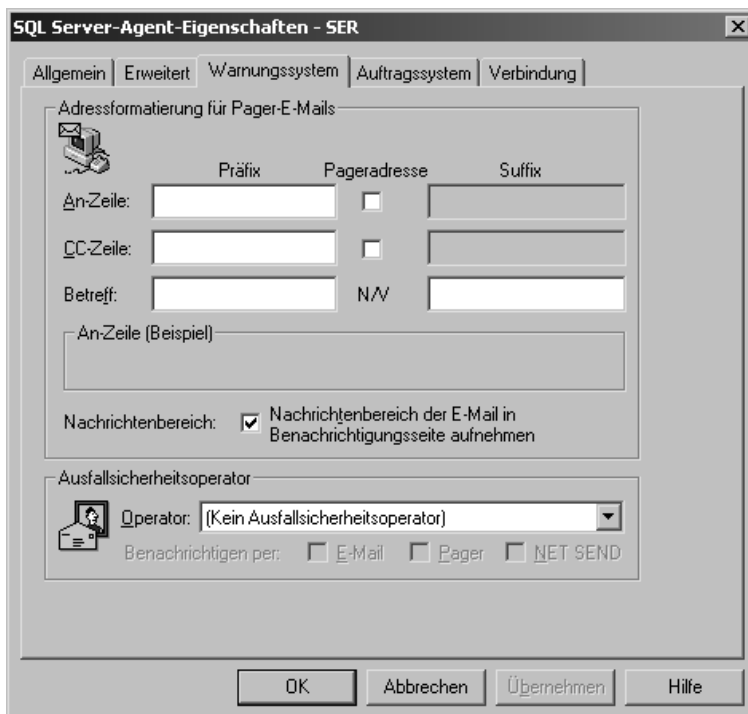


Abbildung 18.33:
Die Registerkarte
Warnungssystem

Die meisten dieser Optionen sind nur für Pager notwendig, zum Beispiel das Hinzufügen eines bestimmten Präfixes oder Suffixes für E-Mails. Entnehmen Sie bitte der Dokumentation Ihrer Pagersoftware, welche Werte hier einzutragen sind. Es lässt sich auch festlegen, dass der Nachrichtenbereich der E-Mail in die Benachrichtigungsseite aufzunehmen ist, die an den Operator gesendet wird (vorausgesetzt, dass der Operator einen Textpager hat).

Der Abschnitt für den Ausfallsicherheitsoperator bestimmt den zu verwendenden Ausfallsicherheitsoperator und ob dieser per E-Mail, Pager oder NET SEND zu benachrichtigen ist. Wenn mit einer Warnung etwas schief geht und keiner der normalerweise zu benachrichtigenden Operatoren erreichbar ist, können Sie hier einen Operator bestimmen, der die Nachricht über den Benachrichtigungsfehler erhält. Wählen Sie hierzu aus der Dropdown-Liste einen Operator aus. Sie können hier auch einen neuen Operator erstellen: Wenn Sie in der Dropdown-Liste den Eintrag *Neuer Ausfallsicherheitsoperator* wählen, gelangen Sie zum Dialogfeld EIGENSCHAFTEN VON NEUEM OPERATOR. Um keinen Ausfallsicherheitsoperator (Standardeinstellung) zu definieren, übernehmen Sie die Einstellung *Kein Ausfallsicherheitsoperator*.

18.6 Integration des Systemmonitors

Der Windows 2000-Systemmonitor wurde vollständig in SQL Server integriert – über Warnungen im SQL Server-Agent. Um diese Integration zu nutzen, gehen Sie zurück zum Dialogfeld NEUE WARNUNG – EIGENSCHAFTEN (klicken Sie dazu mit der rechten Maustaste auf *Warnungen* unter dem Knoten *SQL Server-Agent* und wählen Sie NEUE WARNUNG). Beachten Sie die Dropdown-Liste TYP auf der Registerkarte ALLGEMEIN. Aus dieser Dropdown-Liste können Sie den Eintrag *SQL Server-Leistungsstatuswarnung* wählen. Dadurch ist es möglich, einen SQL Server-Leistungsindikator (bzw. Zähler) auszuwählen und eine Warnung zu konfigurieren – genau wie Sie es mit SQL Server-Fehlermeldungen getan haben. Abbildung 18.34 zeigt eine Leistungsstatuswarnung für das Transaktionsprotokoll der Datenbank *pubs*. Wenn das Transaktions-Protokoll zu 80 Prozent (oder mehr) gefüllt ist, wird diese Warnung ausgelöst.

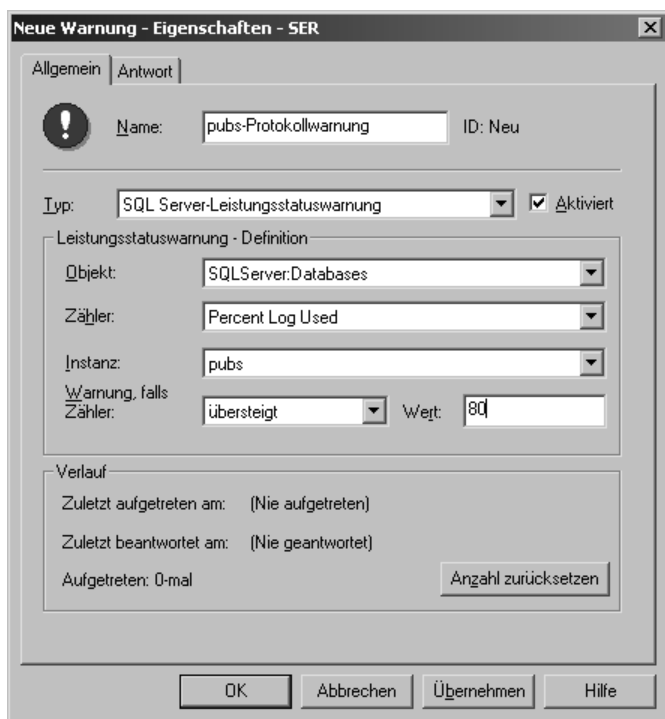


Abbildung 18.34:
Eine Warnung mit einem
Leistungsindikator des Sys-
temmonitors

18.7 Der Datenbank-Wartungsplanungs-Assistent

Der Datenbank-Wartungsplanungs-Assistent erstellt einen Auftrag (oder mehrere Aufträge), um automatisch die Integrität der Datenbank zu überprüfen, Indizes zu reorganisieren, nicht genutzten Speicherplatz in der Datenbank freizugeben sowie Datenbank- und Transaktionsprotokollsicherungen zu erstellen. Das Programm sqlmaint.exe, das eigentlich hinter den Kulissen arbeitet, haben Sie bereits in Lektion 3 kennen gelernt. Allerdings empfiehlt sich der Einsatz dieses Assistenten, sofern Sie nicht einen stark spezialisierten Plan für Ihre Datenbanken benötigen.

Um den Assistenten zu starten, markieren Sie einen Server im Enterprise Manager und wählen dann EXTRAS / ASSISTENTEN. Erweitern Sie im Dialogfeld ASSISTENTEN AUSWÄHLEN den Knoten *Verwaltung* und markieren Sie den Eintrag *Datenbank-Wartungsplanungs-Assistent*. Nachdem Sie auf OK geklickt haben, erscheint der Willkommen-Bildschirm des Assistenten. Klicken Sie auf WEITER, um zum nächsten Dialogfeld zu gelangen (siehe Abbildung 18.35). Hier wählen Sie eine oder mehrere Datenbanken aus, für die Sie eine regelmäßige Wartung durchführen möchten.

Wählen Sie lediglich die *pubs*-Datenbank aus, um die Dinge einfach zu halten. Durch Klicken auf WEITER kommen Sie zum Dialogfeld, das Abbildung 18.36 zeigt. Hier legen Sie fest, ob Sie die Daten- und Indexseiten neu organisieren, die Statistiken aktualisieren und nicht verwendeten Speicherplatz aus der Datenbank entfernen möchten. Welche Optionen Sie auswählen, hängt von der konkreten Datenbankumgebung ab. Abbildung 18.36 zeigt, dass im Beispiel die Option zum Reorganisieren der Indizes ausgewählt ist, was automatisch die Statistiken neu erstellt. Bei der Datenbank *pubs* ist es nicht notwendig, unbenutzten Speicherplatz freizugeben, aber für Ihre Benutzerdatenbanken können Sie die entsprechenden Einträge vornehmen.

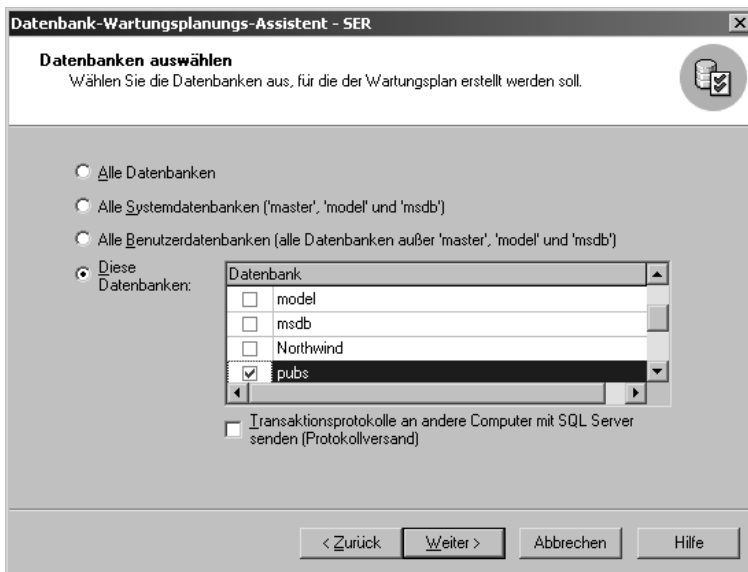


Abbildung 18.35:
Wählen Sie eine
oder mehrere Daten-
banken für die War-
tung aus.

Klicken Sie auf die Schaltfläche ÄNDERN neben dem Feld für den Zeitplan, um für die Datenbankwartung einen Zeitplan festzulegen, wie Sie es weiter vorn in dieser Lektion gelernt haben. Klicken Sie auf WEITER, wenn Sie die Parameter für die Datenoptimierung und den Zeitplan festgelegt haben. Damit gelangen Sie zum Dialogfeld DATENBANKINTEGRITÄTSPRÜFUNG (siehe Abbildung 18.37).



Schalten Sie nicht das Kontrollkästchen KLEINERE PROBLEME BEHEBEN ein. So verlockend diese Option klingt, sie versetzt Ihre Datenbank in den Einzelbenutzermodus, damit sich DBCC mit aktivierter Option REPAIR ausführen lässt. Das hat einen gravierenden Einfluss auf die Verfügbarkeit der Datenbank.

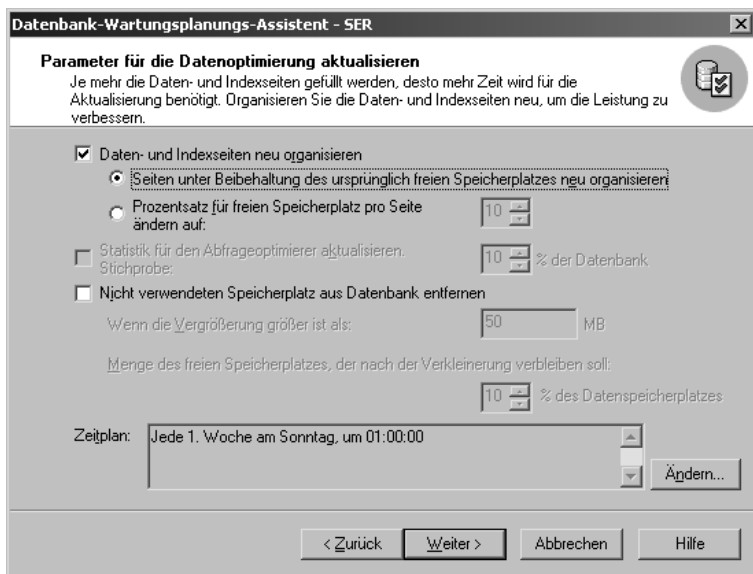


Abbildung 18.36:
Das Dialogfeld
Parameter für die
Datenoptimierung
aktualisieren

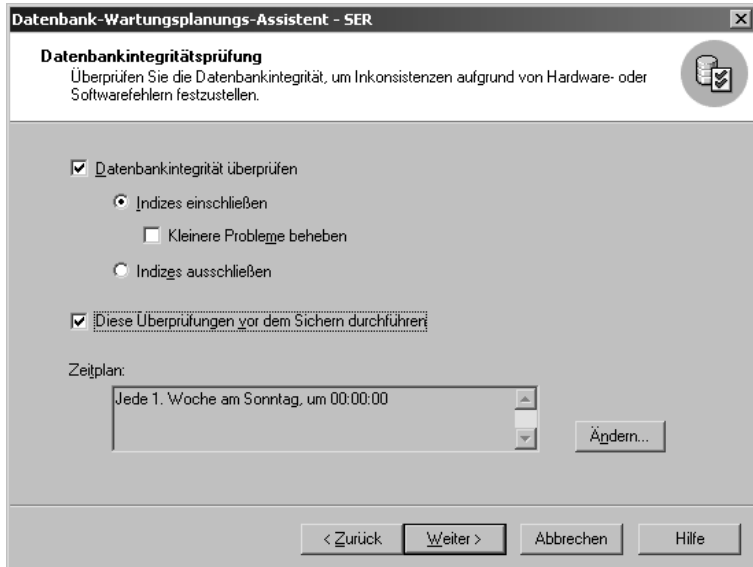


Abbildung 18.37:
Datenbankintegri-
tätsprüfung mit dem
Datenbank-War-
tungsplanungs-
Assistenten

Es ist hier wahrscheinlich am besten, die meisten Optionen einzuschalten, da sich damit eine vollständige Integritätsprüfung realisieren lässt und die Datenbanksicherung nur stattfindet, wenn keine Beschädigungen festgestellt werden. Legen Sie auch hier den Zeitplan für diese Integritätsprüfungen fest und klicken Sie dann auf WEITER. Die Optionen im

Dialogfeld SICHERUNGSPLAN FÜR DATENBANK FESTLEGEN (siehe Abbildung 18.38) sollten Ihnen mittlerweile vertraut sein; wählen Sie also die gewünschten Optionen aus und legen Sie wieder den Zeitplan fest. Klicken Sie dann auf WEITER.

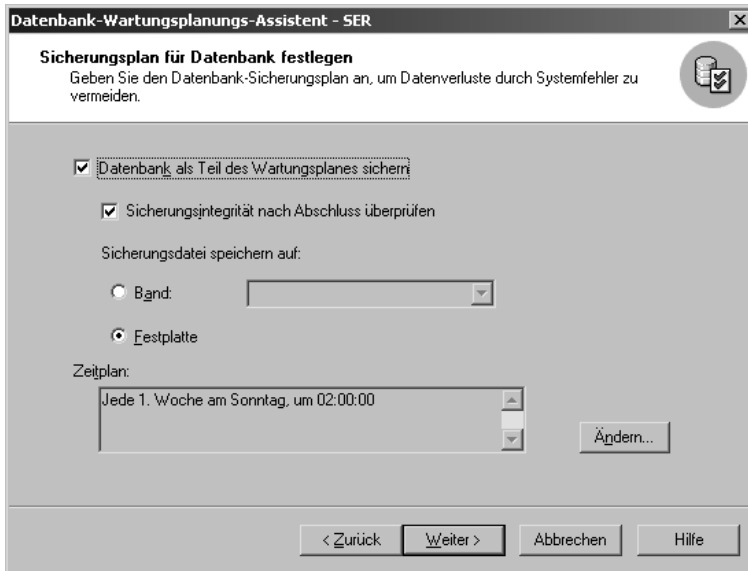


Abbildung 18.38:
Hier legen Sie den
Sicherungsplan für
die Datenbank fest.

Geben Sie an, wo Sie Ihre Sicherungen speichern wollen. Wenn Sie auf die Festplatte sichern, wählen Sie ein anderes Laufwerk als das, auf welchem Ihre Daten gespeichert sind. Es empfiehlt sich, für jede Datenbank ein Unterverzeichnis anzulegen (auf diesem Wege können Sie Ihre Sicherungen leichter finden). Außerdem sollten Sie überalterte Sicherungen automatisch löschen lassen. Abbildung 18.39 zeigt das Dialogfeld mit den entsprechenden Optionen.

Klicken Sie auf WEITER. Es erscheint das Dialogfeld SICHERUNGSPLAN FÜR TRANSAKTIONSprotokoll FESTLEGEN (siehe Abbildung 18.40). Wählen Sie auch hier die entsprechenden Optionen für Ihre Transaktionsprotokollsicherungen und klicken Sie auf WEITER.

Abbildung 18.41 zeigt das Dialogfeld FESTPLATTENVERZEICHNIS FÜR TRANSAKTIONSprotokollSICHERUNG ANGEBEN – es entspricht dem Dialogfeld für die Datenbanksicherung wie in Abbildung 18.39, außer dass es jetzt um das Transaktionsprotokoll geht. Legen Sie die passenden Optionen fest und klicken Sie dann auf WEITER.

Als Nächstes haben Sie die Möglichkeit, Berichte über die erfolgreiche oder fehlerhafte Ausführung Ihres Wartungsplanes zu generieren. Wie zu erwarten, können Sie natürlich auch einen E-Mail-Empfänger für den Wartungsplan konfigurieren (siehe Abbildung 18.42).

Datenbank-Wartungsplanungs-Assistent - SER

Festplattenverzeichnis für Sicherung angeben
Geben Sie das Verzeichnis an, in dem die Sicherungsdatei gespeichert werden soll.

Verzeichnis, in dem die Sicherungsdatei gespeichert werden soll:

☒ Standardsicherungsverzeichnis verwenden

☐ Dieses Verzeichnis verwenden: F:\Programme\Microsoft SQL Serve ...

☐ Unterverzeichnis für jede Datenbank erstellen

☒ Dateien entfernen, wenn älter als: 2 Monat(e)

Sicherungsdateierweiterung: BAK

< Zurück Weiter > Abbrechen Hilfe

Abbildung 18.39:
Hier legen Sie das Verzeichnis fest, in dem Sie die Sicherungen speichern möchten.

Datenbank-Wartungsplanungs-Assistent - SER

Sicherungsplan für Transaktionsprotokoll festlegen
Geben Sie den Sicherungsplan für das Transaktionsprotokoll an, um Fehler und Operatorfehler zu vermeiden.

☒ Transaktionsprotokoll als Teil des Wartungsplanes sichern

☒ Sicherungsintegrität nach Abschluss überprüfen

Sicherungsdatei speichern auf:

☐ Band:

☒ Festplatte

Zeitplan: Jede 1. Woche am Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, um 00:00:00

Ändern...

< Zurück Weiter > Abbrechen Hilfe

Abbildung 18.40:
Hier spezifizieren Sie den Sicherungsplan für das Transaktionsprotokoll.

Datenbank-Wartungsplanungs-Assistent - SER

Festplattenverzeichnis für Transaktionsprotokollsicherung angeben
Geben Sie das Verzeichnis an, in dem die Sicherungsdatei des Transaktionsprotokolls gespeichert werden soll.

Verzeichnis, in dem die Sicherungsdatei gespeichert werden soll:

☒ Standardsicherungsverzeichnis verwenden

☐ Dieses Verzeichnis verwenden: F:\Programme\Microsoft SQL Serv...

☐ Unterverzeichnis für jede Datenbank erstellen

☐ Dateien entfernen, wenn älter als: 4 Woche(n)

Sicherungsdateierweiterung: TRN

< Zurück Weiter > Abbrechen Hilfe

Abbildung 18.41:
Das Dialogfeld Festplattenverzeichnis für Transaktionsprotokollsicherung angeben

Datenbank-Wartungsplanungs-Assistent - SER

Erzeugte Berichte speichern
Geben Sie das Verzeichnis an, in dem die vom Wartungsplan erzeugten Berichte gespeichert werden sollen.

☒ Textberichtsdatei in folgendes Verzeichnis schreiben: F:\Programme\Microsoft SQL Serv...

☐ Textberichtsdateien löschen, die älter sind als: 4 Woche(n)

☒ E-Mail-Bericht an Operator senden: Richard Waymire

Neuer Operator...

< Zurück Weiter > Abbrechen Hilfe

Abbildung 18.42:
Berichtsoptionen für den Wartungsplan festlegen

Schließlich haben Sie die Möglichkeit, den Wartungsplanverlauf in den Systemtabellen der Datenbank *msdb* aufzeichnen zu lassen. Die Größe des Protokolls können Sie über die höchstzulässige Anzahl der Zeilen begrenzen (siehe Abbildung 18.43).

Abbildung 18.43:
Für den Wartungs-
plan lassen sich Ver-
laufsinfos aufnehmen.

Klicken Sie auf WEITER , um eine Zusammenfassung der gewählten Optionen anzuzeigen und dem Wartungsplan einen Namen zu geben. Wenn Sie diesen Schritt abgeschlossen haben, werden in Ihrem Namen mehrere geplante Aufträge konfiguriert. Diese Aufträge können Sie sich im Detailbereich des Enterprise Managers ansehen, wie Sie es in der heutigen Lektion bereits gesehen haben. Nachdem Sie die Konfiguration mit dem Datenbank-Wartungsplanungs-Assistenten fertig gestellt haben, laufen etwa 90 Prozent der erforderlichen Datenbankwartungsaufgaben ohne Ihr Mitwirken ab.

18.8 Zusammenfassung

Der SQL Server-Agent-Dienst ist zwar für den Einsatz von SQL Server nicht unbedingt erforderlich, er kann aber die Funktionalität Ihres Servers entscheidend erhöhen. Sie können ausgeklügelte Aufträge erstellen, um solche Funktionen wie Sicherungen und Datenbankintegritätsprüfungen zu automatisieren. Mit Warnungen können Sie auf Ereignisse Ihrer SQL Server-Datenbanken reagieren und es lassen sich sogar die Leistungsindikatoren des Windows 2000-Systemmonitors nutzen, um Warnungen auszulösen. Außerdem haben Sie die Möglichkeiten kennen gelernt, die SQL Server für die Benachrichtigung von Administratoren per E-Mail, Pager und NET SEND-Befehl bietet, wenn eine Warnung auftritt oder ein Auftrag ausgeführt wird.

Indem Sie die Integration der beschriebenen Merkmale in SQL Server nutzen und in wohlgedachten Wartungsplänen umsetzen, können Sie die Zuverlässigkeit und das Reaktionsverhalten Ihres Servers bei auftretenden Problemen entscheidend verbessern, die Ausfallzeiten des Servers verringern und letztendlich zur Zufriedenheit Ihrer Benutzer beitragen.

18.9 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Welcher Dienst ist für die Verwaltung von geplanten Aufträgen zuständig?*
- A** Der SQL Server-Agent-Dienst.
- F** *Welche Systemtabelle enthält die Warnungen, die Sie für SQL Server definiert haben?*
- A** Die sysalerts-Systemtabelle in der msdb-Datenbank.
- F** *Welcher Schweregrad kennzeichnet ein Serverproblem von SQL Server selbst?*
- A** Schweregrad 17 oder größer.
- F** *Was müssen Sie tun, um die Integration des Windows 2000-Systemmonitors in SQL Server 2000 zu nutzen?*
- A** Erstellen Sie eine so genannte SQL Server-Leistungsstatuswarnung auf Basis der Schwellenwerte der SQL Server-Leistungsindikatoren.

Kontrollfragen

1. Was heißt die Windows NT/2000-Alternative zum SQL Server-Agent-Dienst für die Zeitplanung von Aufgaben?
2. Auf welches Profil bezieht sich SQL Server, wenn er einen Mailprofilnamen verlangt?



3. Mit welcher Anweisung empfangen Sie eine per E-Mail übermittelte Abfrage und senden die Antwort in Form einer angehängten Textdatei an einen Benutzer?
4. Wie viele geplante Aufträge erstellen Sie, wenn Ihr Transaktionsprotokoll alle 3 Stunden täglich gesichert werden soll? Die erste Sicherung soll mit der Option `WITH INIT` um 05.00 Uhr durchgeführt werden, die restlichen Sicherungen sind mit der Option `WITH NOINIT` auszuführen.
5. Mit welchem Werkzeug oder welchem Assistenten richten Sie regelmäßige Sicherungen, Datenbankreorganisationen und Integritätsprüfungen ein?

Übungen

1. Wenn Sie die E-Mail-Integration in SQL Server noch nicht im Laufe dieses Kapitels vorgenommen haben, holen Sie sie jetzt in dieser Übung nach.
2. Konfigurieren Sie eine Systemmonitorwarnung und simulieren Sie dann die Auslösebedingungen. Überprüfen Sie, dass alles wie erwartet funktioniert.
3. Erstellen Sie einen geplanten Auftrag, der eine Windows 2000-Befehlsdatei ausführt. Experimentieren Sie mit den Erfolgs- und Fehleroptionen der Auftragsschritte.
4. Konfigurieren Sie einen Datenbank-Wartungsplanungs-Assistenten für jede Datenbank Ihres Systems.



Daten zwischen
Servern migrieren

Die heutige Lektion zeigt, wie Sie Daten von einem Standort zu einem anderen verschieben und dabei modifizieren können. Dabei sind in SQL Server 2000 prinzipiell die folgenden drei Verfahren möglich:

- Durch Sichern und Wiederherstellen verschiebt man alle Daten und anderen Objekte – wie zum Beispiel Tabellen – von einem Standort zu einem anderen. Dabei finden keine Änderungen an den Daten statt.
- Das Programm BCP (Bulk Copy Program) überträgt nur Daten von einem Standort zu einem anderen. Obwohl dieses Dienstprogramm keine Änderungen an den Daten vornimmt, können Sie es einsetzen, um bestimmte Spalten und Zeilen von einer Datenquelle zu sammeln und sie zu einem anderen Standort zu übertragen.
- Durch Masseneinfügen lassen sich Daten schnell von einer Datei in eine Datenbank übernehmen.

Außerdem verfügt SQL Server 2000 über ein Tool, das die Daten sowohl verschiebt als auch transformiert: die Data Transformation Services (DTS). Auf den ersten Blick ähnelt es dem alten Transfer Manager von SQL Server 6.x. Mit DTS lassen sich aber auch Daten und/oder Schemas verschieben. Außerdem bietet DTS die Fähigkeit, das Format der Daten von einem Datentyp in einen anderen zu ändern, die Daten neu zu strukturieren und verschiedenen Tabellen zuzuordnen, die Konsistenz der Daten sicherzustellen und die Daten auf Gültigkeit zu überprüfen. Darüber hinaus kann man logische Operationen einbinden, um die Daten zusammenzufassen, zu dekodieren, zu konvertieren und zu zerlegen.

In der Vergangenheit wurde diese Logik zumeist über ein anderes Werkzeug oder eine Programmiersprache wie etwa Delphi oder Visual Basic realisiert. SQL Server 2000 bietet diese Fähigkeit zur Programmierung über die DTS-Pakete. Da diese Pakete selbst vollständig modular aufgebaute Einheiten darstellen, können Sie sie auch in Ihren Zeitplandienst für automatisch ablaufende Datenübertragungen und -änderungen einbinden.

19.1 Daten verschieben mit Sichern und Wiederherstellen

Natürlich können Sie zum Übertragen von Datenbanken zwischen verschiedenen Servern die Datensicherungs- und Wiederherstellungsverfahren einsetzen; eine optimale Lösung ist dies jedoch nicht. In die Sicherung einer Datenbank sind nämlich auch sämtliche Systemtabellen einschließlich der Systemtabelle *sysusers* einbezogen, was zu Problemen führen kann, wenn Sie diese Datenbank später auf einem anderen Server wiederherstellen möchten. Falls der neue Server nämlich nicht genau dieselben Benutzernamen besitzt, sind die Zuordnungen des Sicherheitssystems ungültig.



Der Einsatz der Sicherung und Wiederherstellung zum Übertragen von Daten von einem SQL Server zu einem anderen ist keine empfohlene Methode für die Alltagspraxis. Falls Benutzer des ursprünglichen Servers nicht im neuen Server zu finden sind, können Sie auf Sicherheitsprobleme stoßen. Besser geeignet ist die Replikation oder DTS.

Die Sicherung und Wiederherstellung bietet sich für Situationen an, in denen Sie einen »Sicherungs-« oder »Standby-« Server bereitstellen möchten.

Zum Lieferumfang von SQL Server 2000 gehört die gespeicherte Systemprozedur `sp_change_users_login`, die das allergrößte Problem beseitigt, nämlich die ungünstigen Sicherheitszuordnungen beim Einsatz der Datensicherungs- und Wiederherstellungsverfahren (SQL Server 2000 validiert nur Benutzernamen). Nachdem die Wiederherstellung abgeschlossen ist, vergewissern Sie sich, dass die Benutzernamen korrekt an Datenbankbenutzer zugeordnet sind, und führen dann die folgende gespeicherte Systemprozedur aus:

```
sp_change_users_login {Auto_fix | Report | Update_One}
[, 'UserNamePattern' [, 'LoginName']]
```

Die Optionen *Auto_fix* | *Report* | *Update_One* zeigen an, welche Funktion ausgeführt werden soll. Die Optionen haben folgende Bedeutung:

- *Auto_fix* verknüpft Datenbankbenutzernamen mit Benutzernamen, geht aber von Annahmen aus, die unter Umständen nicht korrekt sind. Diese Option sollten Sie nur mit Vorsicht einsetzen.
- *Report* listet sämtliche nicht verknüpften Einträge auf. Wenn Sie hier keinen Parameter angeben, wird ein Bericht generiert.
- *Update_One* repariert eine einzige spezifizierte Verknüpfung, die in den folgenden Parametern genannt wird.
- *UserNamePattern* stellt einen »LIKE«-Vergleichsoperator wie etwa % für sämtliche Benutzer (falls Sie *Auto_fix* angegeben haben) dar. Der Standardwert ist NULL für *Report* (Bericht über sämtliche Benutzer). Ist *Update_One* angegeben, wird ein Bericht über den von Ihnen an dieser Stelle angegebenen Benutzernamen erzeugt.
- Der Parameter *LoginName* ist der Name eines SQL Server-Benutzernamens aus der Tabelle *sysxlogins* in der Datenbank *master*. Falls Sie *Update_One* angegeben haben, können Sie hier die Zuordnung festlegen, die Sie korrigieren lassen möchten. Andernfalls sollte dieser Parameter NULL sein.

Angenommen, Sie haben nach einer Wiederherstellung einen Benutzernamen Georg auf dem neuen Server vorliegen. Georg besitzt eine bestimmte SID oder GUID (SIDs werden für Benutzer mit Windows 2000-Authentifizierung verwendet, GUIDs für Benutzer mit SQL Server-Authentifizierung). Wenn Sie nun die Tabelle *sysusers* in der neu wiederhergestellten Datenbank abfragen, stellen Sie möglicherweise fest, dass Georg eine andere

GUID zugeordnet worden ist (SIDs werden von Windows 2000 erstellt und verwaltet und ändern sich nicht). Um Georg seine ursprüngliche GUID zurückzugeben, führen Sie nun die folgende Anweisung aus:

```
sp_change_users_login Update_One, 'Georg', 'Georg'
```

Dieses Beispiel ändert das Feld *GUID* in der Tabelle *sysusers* in der wiederhergestellten Datenbank so, dass es mit der GUID für Georg in der Tabelle *sysxlogins* in der Datenbank *masters* auf diesem Server übereinstimmt.

19.2 Daten mit BCP laden

Nachdem Sie nun die potentiellen Probleme kennen, die bei der Übertragung von Daten über die Sicherungs- und Wiederherstellungsverfahren auftreten können, wird es Zeit dafür, einen Blick auf das Programm BCP zum Massenkopieren zu werfen. BCP können Sie dazu benutzen, entweder Daten aus einer Datei nach SQL Server zu laden oder von SQL Server in eine Datei zu speichern. Dieses Tool bietet sich an, wenn Sie Daten aus SQL Server 2000 zu einer älteren Version von SQL Server oder in ein anderes Datenbanksystem wie etwa Oracle übertragen müssen. Außerdem können Sie Daten aus einem anderen Programm, einer anderen Datenbank oder einer älteren Anwendung importieren.



Vergessen Sie jedoch nicht, dass dieses Werkzeug nur Übertragungen in oder aus einer Betriebssystemdatei und nur die Übertragung von Daten beherrscht – zur Übertragung eines Datenbankschemas müssen Sie die Sicherungs- und Wiederherstellungsverfahren oder aber DTS einsetzen.

Außerdem können Sie Daten an eine vorhandene Tabelle anhängen, genau wie mit jedem anderen Programm, das Daten in eine Tabelle einfügt. Allerdings führt BCP diese Operation gewöhnlich schneller aus als ein entsprechendes Programm, das Sie selbst schreiben.

Es ist auch möglich, die Daten in eine Datei zu exportieren. Falls die Datei noch nicht existiert, legt BCP sie automatisch an. Andernfalls wird der Inhalt der Datei mit den heruntergeladenen Daten überschrieben.

Das Programm BCP verwendet die folgenden Standarddateierweiterungen:

- .bcp – Datendateien im systemeigenen BCP-Format
- .txt – ASCII-Textdateien
- .csv – Kommagetrennte Dateien
- .fmt – Formatdateien
- .err – Datenfehlerdateien

BCP-Berechtigungen

Um Daten von SQL Server in eine Datei zu kopieren, brauchen Sie `SELECT`-Berechtigungen für den Systemkatalog und natürlich für die Tabelle oder Sicht, von der Sie die Daten holen möchten. Da alle Benutzer, die auch einen Datenbankbenutzernamen haben, standardmäßig die Systemtabellen lesen können, sollte dies kein Problem darstellen. Darüber hinaus benötigen Sie die Betriebssystemberechtigungen zum Erstellen oder Ändern der Datei, in die Sie schreiben möchten. Natürlich kommen Dateiberechtigungen nur in einer Windows NT/2000-Umgebung ins Spiel, da Windows 9x keinerlei Dateiberechtigungen kennt. In einer Windows NT/2000-Umgebung benötigen Sie dagegen normalerweise die Berechtigungen *Ändern* oder *Vollzugriff* für das Verzeichnis sowie die Dateiberechtigungen, wenn Sie eine neue Datei anlegen. Für Partitionen mit FAT (File Allocation Table) und FAT32 gibt es keinerlei Sicherheitseinrichtungen, so dass auf Datei- und Verzeichnisebene keine speziellen Berechtigungen erforderlich sind.

Zum Importieren von Daten aus einer Datei nach SQL Server benötigen Sie die Berechtigung *Lesen* für die Datei (falls sich diese in einer NTFS-Partition befindet). Außerdem brauchen Sie `INSERT`-Berechtigungen für die SQL Server-Tabelle, in die Sie die Daten laden möchten.

BCP einsetzen

Sehen Sie sich die Syntax des Programms BCP an:



```
Bcp [[Datenbankname.]Besitzer.]Tabellenname | Sichtname | "Query"
{in | out | queryout | format } Datendatei
[/m maxerrors]
[/f formatfile]
[/e errfile]
[/F firstrow]
[/L lastrow]
[/b batchsize]
[/n] [/c] [/w] [/N]
[/V (60|65|70)][/q]
[/6][/q]
[/C code_page]
[/t field_term]
[/r row_term]
[/i inputfile]
[/o output_file]
[/a packet_size]
[/S server_name\Instanzname]
/U login_id
```

```
[/P password]
[/T] [/v] [/R] [/k] [/E]
[/h "hint" [,...n]]
```

Als Nächstes folgt ein Blick auf die zwingend erforderlichen Parameter (nicht diejenigen in eckigen oder geschweiften Klammern):

- *Tabellenname* ist der Name der Tabelle, die Sie exportieren oder importieren möchten. Dies kann eine globale temporäre Tabelle, eine Sicht oder eine Abfrage sein. Als *Datenbankname* gilt standardmäßig die Standarddatenbank des Benutzers, der den BCP-Befehl ausführt. Der *Besitzer* lautet standardmäßig auf den Datenbankbenutzernamen des angemeldeten Benutzers. Wenn Sie wollen, können Sie alle Parameter voll ausschreiben, wie etwa `pubs.dbo.authors` für die Tabelle *authors*, deren Besitzer der Benutzer *dbo* in der Datenbank *pubs* ist.
- *in / out / queryout / format* gibt an, ob die Daten nach SQL Server geladen, aus SQL Server in eine Datei geschrieben oder in Form einer Abfrage zurückgeliefert werden sollen.
- *Datendatei* ist der Name der Datei, aus der Sie Daten laden oder in die Sie Daten schreiben möchten. Hier müssen Sie den vollständigen Pfad- und Dateinamen angeben. Fehlt der Pfad, gilt wie bei jedem Windows 2000-Programm das lokale Verzeichnis, aus dem Sie BCP gestartet haben.



Obwohl der Dateiname erforderlich ist, erhalten Sie nicht unbedingt eine Fehlermeldung, wenn Sie BCP ohne Dateiname ausführen. Diesen Umstand können Sie sich zunutze machen: Falls Sie beispielsweise eine Formatdatei (wird weiter unten erläutert) erstellen möchten, aber die Daten noch nicht exportieren wollen, können Sie diesen Parameter freilassen und nur die Formatdatei erstellen. Folgen Sie beim Erstellen oder Referenzieren dieser Datei den weiter oben genannten Namenskonventionen.

- */U* spezifiziert den Benutzernamen, unter dem Sie sich anmelden möchten.

Die folgenden Parameter sind optional:

- */P* gibt das Kennwort für den Benutzernamen an, den Sie über den Parameter */U* spezifiziert haben. Falls Sie kein Kennwort angegeben haben, werden Sie danach gefragt. Es versteht sich von selbst, dass Sie ein Kennwort bereitstellen, wenn Sie BCP im Stapelbetrieb ausführen.
- */S* spezifiziert den Namen des Servers, mit dem Sie sich verbinden möchten. Falls Sie hier keinen Servernamen angeben, versucht BCP, sich mit einer lokalen Kopie von SQL Server zu verbinden. Geben Sie einen Servernamen an, stellt BCP die Verbindung zum Server über das Netzwerk her. In Bezug auf die Systemleistung ist es am besten, eine Datendatei ohne den Parameter */S* in den Server zu laden.

- */m maxerrors* kennzeichnet die maximale Anzahl der Fehler, die beim Importieren von Daten nach SQL Server (beispielsweise wegen schlecht formatierter Zeilen) erlaubt sind, bevor das Massenkopieren abgebrochen wird. Sinnvoll ist dies insbesondere zum Testen von großen Datenimportierungen. Wenn Sie beispielsweise falsche Eingaben gemacht haben oder auf andere Probleme stoßen, können Sie den Ladevorgang immer noch vorzeitig abbrechen, statt ewig zu warten und dann feststellen zu müssen, dass der Ladevorgang fehlgeschlagen ist. Ist dieser Parameter nicht angegeben, lautet der Standardwert 10 Fehler.
- */f formatfile* ist der Pfad- und Dateiname der Datei, in der die Formatbeschreibung enthalten ist. Daraus ergibt sich, wie die Zieldatei beim Extrahieren von Daten aus SQL Server zu formatieren ist, oder wie die Daten der Quelldatei beim Importieren nach SQL Server zu interpretieren sind. Wenn Sie bei der Ausgabe von Daten hier keinen Parameter angeben, fragt BCP die Formatinformationen ab und speichert sie am Ende als Datei mit dem Standardnamen *bcp.fmt*.
- */e errfile* ist der Pfad- und Dateiname der Datei, in die BCP Zeilen kopiert, die während des Ladevorgangs nach SQL Server nicht verarbeitet werden konnten. Damit haben Sie die Möglichkeit, jede nicht korrekt geladene Zeile zu untersuchen und die Probleme zu beheben. Anschließend können Sie die Fehlerdatei wiederum als Eingabedatei einsetzen und BCP erneut ausführen. Falls Sie keine Fehlerdatei angeben, wird auch keine derartige Datei erstellt und problematische Zeilen lassen sich im Nachhinein nicht mehr identifizieren. Die weiter oben schon genannten Namenskonventionen empfehlen für diese Datei die Erweiterung *.err*.
- */F firstrow* ist ein Parameter, mit dem Sie eine relative Zeilennummer spezifizieren können, bei der eine Ladeoperation beginnen soll. Falls Sie diesen Parameter nicht angeben, beginnt der Ladevorgang mit der Zeilennummer 1.
- */L lastrow* spezifiziert die letzte Zeile, die aus der Quelldatei beim Ladevorgang zu kopieren ist. Standardmäßig werden alle Zeilen bis zum Ende der Datei geladen.
- */b batchsize* gibt die Anzahl der einzufügenden Zeilen an, die als Stapel zu behandeln sind. In diesem Kontext behandelt SQL Server jeden Stapel als separate Transaktion. (Mehr über diese Option erfahren Sie in Kürze.) Falls Sie diesen Parameter nicht angeben, wird der gesamte Ladevorgang als einzelner Stapel behandelt.
- */n* legt fest, dass die Daten unter Verwendung der Datentypformate von SQL Server exportiert oder importiert werden sollen. Diese Option sollten Sie nur dann verwenden, wenn Sie Daten zwischen zwei Kopien von SQL Server übertragen, und auch nur dann, wenn es sich um die gleiche Plattform (Intel, Alpha usw.) handelt.
- */c* legt fest, dass bei Datenimporten und -exporten Zeichendatentypen anstelle der internen Datentypen von SQL Server verwendet werden sollen. Hierdurch wird ein tabstoppgetrenntes Format für die Spalten einschließlich eines Zeilenende- und Wagenrücklaufzeichens am Ende einer Zeile erstellt bzw. verwendet.

- */w* legt fest, dass die Daten unter Verwendung des Unicode-Formats übertragen werden.
- */N* legt fest, dass bei der Übertragung Unicode-Zeichen für Zeichendaten und das systemeigene Format für Nicht-Zeichendaten verwendet wird.
- */V (60|65|70)* gibt an, dass Sie Daten von einer Datenbank der SQL Server-Versionen 6.0, 6.5 bzw. 7.0 übertragen. Dieser Parameter ist nicht erforderlich, wenn Sie Daten von einem SQL Server 2000 zu einem anderen übertragen. Auch wenn Sie Daten im Zeichenmodus übertragen, brauchen Sie diesen Parameter nicht zu spezifizieren, weil keine Datentypumwandlungen vorzunehmen sind.
- */6* legt fest, dass bei der Übertragung die Datentypen von SQL Server 6.x verwendet werden. Dieser Parameter steht aus Gründen der Abwärtskompatibilität zur Verfügung.
- */q* legt fest, dass Bezeichner in Anführungszeichen (*quoted identifier*) verwendet werden. In diesem Fall müssen Sie sicherstellen, dass sämtliche Referenzen auf derartige Bezeichner in doppelte Anführungszeichen eingeschlossen sind.
- */C code_page* erlaubt die Festlegung einer Codeseite für die Datendatei. Dies ist nur für Zeichen im erweiterten Zeichensatz anwendbar.
- */E* legt fest, dass BCP Identitätsspalten mit Werten aus der Datendatei laden und nicht die *IDENTITY*-Eigenschaft der Identitätsspalte in der Tabelle verwenden soll. (Die *IDENTITY*-Eigenschaft haben Sie bereits in Lektion 14 kennen gelernt.) Andernfalls werden Identitätswerte auf normalem Weg zugewiesen.
- */t field_term* legt das standardmäßige Feldabschlusszeichen fest.
- */r row_term* spezifiziert den Wert, den BCP zur Bestimmung des Zeilenendes und des Beginns einer neuen Zeile verwenden soll.
- */i inputfile* legt eine Eingabedatei fest. Falls Sie nicht bei jedem Start von BCP all diese Parameter manuell eingeben möchten, können Sie sie in eine derartige Datei schreiben und anschließend BCP über diese Eingabedatei starten. Der Name einer Eingabedatei muss aus dem vollständigen Pfad- und Dateinamen bestehen.
- */o outputfile* spezifiziert den Namen der Datei, in der BCP Meldungen und andere Informationen ablegt. Auf diese Weise können Sie die Ausgaben von BCP speichern und später überprüfen. Dies ist insbesondere im Stapelbetrieb nützlich.
- */a packet_size* gibt die im Netzwerk verwendete Paketgröße an. Der Standardwert beträgt 4.096 Byte (4 KB), außer bei MS DOS-Arbeitsstationen, für die ein Wert von 512 Byte gilt. Diese Option lässt sich auch auf der Serverseite konfigurieren.
- */T* legt fest, dass BCP die Verbindung zu SQL Server über eine vertraute Verbindung herstellt.

- */v* gibt die aktuelle Versionsnummer der DB-Library aus.
- */R* spezifiziert, dass die Datums-/Zeitdaten sowie die Währungsangaben nicht ignoriert, sondern entsprechend der Ländereinstellungen des Clientcomputers übertragen werden.
- */k* legt fest, dass leere Spalten ihre ursprünglichen NULL-Werte behalten und keine Standardwerte zugewiesen bekommen.
- */h »hint«* ermöglicht Ihnen die Angabe von Hinweisen – beispielsweise Sortierung für Spalten oder CHECK-Einschränkungen. Weitere Informationen über dieses Thema finden Sie in der Onlinedokumentation von SQL Server.

Systemeigener Modus und Zeichenmodus

Wie Sie gerade erfahren haben, gibt es beim Übertragen von Daten zwei Verfahrensweisen: Den systemeigenen Modus und den Zeichenmodus. Im Zeichenmodus werden die Daten im Klartext übertragen, d.h. der Wert einer Ganzzahlspalte lässt sich als 12.500 lesen, während diese Zahl im systemeigenen Format in 4 Bytes codiert (0x000030D4 hexadecimal) dargestellt wird.

Der Zeichenmodus ist sehr flexibel. Den systemeigenen Modus dagegen können Sie nur dann verwenden, wenn Sie Daten zwischen SQL Servern übertragen; und auch hier vorzugsweise nur auf der gleichen Plattform unter Verwendung desselben Zeichensatzes. Der Zeichenmodus erlaubt es, Daten in nahezu jedes Programm zu importieren oder von dort zu exportieren. Dateien im Zeichenmodus sind darüber hinaus im Fehlerfall viel leichter zu reparieren.

Wenn Sie */c* angeben, fordert BCP Sie nicht mehr zu weiteren Eingaben auf, sondern geht davon aus, dass Ihre Daten als reiner ASCII-Text formatiert werden sollen. Wenn Sie */n* angeben, werden Sie ebenfalls nicht zur Eingabe weiterer Informationen aufgefordert. Andernfalls müssen Sie immer Angaben darüber machen, wie Ihre Datei formatiert werden soll.

Abbildung 19.1 zeigt ein Beispiel, bei dem mit BCP eine Tabelle mit einer einzigen Spalte vom Typ `char(10)` exportiert wurde.

Wenn Sie den Parameter */c* hinzufügen, fordert BCP nicht mehr dazu auf, die Formatparameter bereitzustellen und die Formatdatei zu erstellen (siehe Abbildung 19.2).

```

C:\> Eingabeaufforderung

D:\> bcp pubs..t1 out f:\test.txt
Kennwort:

Geben Sie den Speichertyp des Feldes Feld1 ein [char]:
Geben Sie die Präfixlänge des Feldes Feld1 ein [0]:
Geben Sie die Feldlänge für Feld1 ein [5]:
Geben Sie das Feldabschlusszeichen ein [none]:

Möchten Sie diese Informationen in einer Datei speichern? [Y/N]
Hostdateiname [bcp.fnt]:

Startet Kopiervorgang...

7 Zeilen kopiert.
Netzwerk-Paketgröße (Byte): 4096
Uhrzeit (ms.): gesamt 10

D:\> _

```

Abbildung 19.1:
Aufruf von BCP
zum Exportieren
einer einfachen
Tabelle

```

C:\> Eingabeaufforderung

D:\> bcp pubs..t1 out f:\test.txt /c
Kennwort:

Startet Kopiervorgang...

7 Zeilen kopiert.
Netzwerk-Paketgröße (Byte): 4096
Uhrzeit (ms.): gesamt 1

D:\>

```

Abbildung 19.2:
Exportieren einer
Tabelle im Zeichen-
format

Formatdateien

Mit Formatdateien spezifiziert man das Layout der Zieldatei oder dokumentiert das Layout für den Import. Wenn Sie beim Exportieren von Daten mit BCP nicht die Parameter */n* oder */c* angeben, können Sie eine Formatdatei erstellen lassen. Für das Importieren können Sie selbst eine Formatdatei erstellen.

Beispielsweise können Sie jede mögliche Option untersuchen, wenn Sie beim Exportieren mit BCP festlegen, dass Sie eine Formatdatei erstellen möchten. Dann erhalten Sie je nach Datentyp zusätzliche Fragen zum Format gestellt. Das folgende Beispiel zeigt die Fragen für einen Zeichendatentyp, wie er für die Tabelle *t1* in *pubs* definiert ist. Wenn Sie keine weiteren Angaben liefern und jeweils die <Eingabe>-Taste drücken, gelten die in eckigen Klammern angegebenen Werte:

Geben Sie den Speichertyp des Feldes Feld1 ein [char]:
 Geben Sie die Präfixlänge des Feldes Feld1 ein [0]:
 Geben Sie die Feldlänge für Feld Feld1 ein [5]:
 Geben Sie das Feldabschlusszeichen ein [none]:

Der Speichertyp beschreibt, wie die Daten in der Datei abgelegt werden sollen. Für Zeichendaten oder bei Verwendung eines zeichenorientierten Datentyps benutzen Sie den Datentyp `char`. Andernfalls übernehmen Sie einfach den Standardwert von SQL Server. Sie haben die Möglichkeit, jeden gültigen SQL Server-Datentyp zu benutzen.

Die Präfixlänge bestimmt, wie viel Platz für die Angabe der Länge dieses bestimmten Feldes reserviert werden soll. Für viele Typen können Sie hier einfach 0 eingeben bzw. übernehmen. Für Daten im `char`- oder `varchar`-Format ist ein Wert von 1 Byte möglich. Für `text`- oder `image`-Datentypen beträgt die Standardlänge 4 Byte.

Die Feldlänge gibt an, wie lang die Datenfelder beim Exportieren werden sollen. Übernehmen Sie hier am besten die von SQL Server vorgeschlagenen Standardwerte.

Die letzte Option betrifft das Feldabschlusszeichen. Fast jedes Einzelzeichen ist auch ein gültiges Feldabschlusszeichen. Allerdings sollten Sie ein Zeichen angeben, das nicht in den eigentlichen Daten auftaucht. An dieser Stelle können Sie jedes Zeichen einschließlich der folgenden Sonderzeichen festlegen:

- `\t` für das Tabulatorzeichen
- `\n` für Neue Zeile
- `\r` für den Wagenrücklauf
- `\\` für den Backslash
- `\0` für ein Nullwertzeichen (kein sichtbares Abschlusszeichen)

Diese Abschlusszeichen können Sie sowohl als Feldabschlusszeichen (siehe die Option `/t` in der Syntax des Befehls BCP weiter oben) oder als Zeilenabschlusszeichen (Option `/r`) verwenden.

Das genannte Beispiel erzeugt eine Formatdatei mit etwa folgendem Inhalt:

```
8.0
1
1      SQLCHAR      0      5      " "      1      Feld1
```

Die erste Zeile zeigt die Nummer der SQL Server-Version, mit der diese Formatdatei verbunden ist (bei SQL Server 7.0 erscheint hier 7.0, SQL Server gibt als Nachfolger die Versionsnummer 8.0 – und nicht 2000 – an). Die nächste Zeile gibt die Anzahl der Spalten (Felder) in dieser Formatdatei an. Anschließend folgt pro Spalte eine Ausgabzeile in der Datei.



Die erste Zahl in dieser Zeile stellt die relative Spaltennummer in der Datendatei dar. Der zweite Eintrag in der Zeile ist der Datentyp. Formatdateien von BCP enthalten keine Datentypen von SQL Server, sondern spezielle Datentypen für BCP-Dateien. Tabelle 19.1 gibt die gültigen Datentypen für BCP-Formatdateien an.

BCP-Datentyp	steht für den SQL Server-Datentyp
SQLBINARY	binary, image, timestamp, varbinary
SQLBIT	bit
SQLCHAR	char, sysname, text, varchar
SQLDATETIME	datetime
SQLDECIMAL	decimal
SQLFLT8	float
SQLMONEY	money
SQLNUMERIC	numeric
SQLFLT4	real
SQLDATETIME4	smalldatetime
SQLMONEY4	smallmoney
SQLTINYINT	tinyint

Tabelle 19.1: BCP- und SQL Server-Datentypen

Das nächste Feld gibt die Präfixlänge an, die weiter oben schon erläutert wurde. Danach folgt die tatsächliche Datenlänge. Der nächste Eintrag ist das Feldabschlusszeichen. Anschließend ist noch die Spaltensortierreihenfolge und der Serverspaltenname entsprechend der Tabellendefinition angegeben. Der letzte – im obigen Beispiel nicht angezeigte – Wert ist die Spaltensortierung (z.B. Latin1_General_CI_AS).

Stapelgrößen

Wenn Sie Daten mit BCP nach SQL Server laden, werden standardmäßig sämtliche Daten im Rahmen eines einzigen Stapels und einer einzigen Transaktion kopiert, was zur Folge hat, dass der gesamte Ladevorgang entweder als Ganzes erfolgreich ist oder scheitert. Auch Millionen von Zeilen werden auf diese Weise als ein einziger Datensatz in einer einzigen Transaktion kopiert, was sehr leicht die Transaktionsprotokolle Ihrer Datenbank »dichtmachen« kann. Aus diesem Grund erscheint es ratsam, ab und zu ein »Commit« auszulösen und einen neuen Stapel zu beginnen, was die Parallelität der Datenverarbeitung verbessert und Ihnen erlaubt, Operationen wie etwa das Abschneiden des Transaktionsprotokolls am Prüfpunkt vorzunehmen. Weitere Informationen über Transaktionen finden Sie in Lektion 12.

Schnelles BCP

BCP kann in zwei Modi betrieben werden: Normales und schnelles BCP. Auf der untersten Ebene führt BCP eigentlich nur Einfügeoperationen aus. Das bedeutet, dass jede Zeile unter Verwendung einer `INSERT`-Operation verarbeitet wird. Die Zeile wird in das Transaktionsprotokoll geschrieben. Anschließend werden die Daten verarbeitet und gegebenenfalls Indizes hinzugefügt. Das ist nicht gerade der schnellste Weg, um Daten in SQL Server einzufügen.

Wenn Sie jedoch sämtliche Indizes einer Tabelle löschen und die Option *Select into/bulkcopy* für eine Datenbank auf `True` setzen, wird dagegen ein schnelles BCP ausgeführt. In diesem Betriebsmodus werden keine `INSERT`-Operationen mehr in das Transaktionsprotokoll geschrieben. Da auch keine Indizes mehr zu verwalten sind, verlaufen diese Operationen außerordentlich schnell. Für die Erzwingung der Datentypen und Standardwerte wird jedoch nach wie vor auch während des schnellen Ladevorgangs gesorgt. Nicht erzwungen wird die Ausführung von Regeln, Einschränkungen und Triggern.

Führen Sie für eine schnelle Datenübertragung die folgenden Arbeitsschritte aus:

1. Löschen Sie die Indizes aller betroffenen Tabellen.
2. Schalten Sie die Option *Select into/bulkcopy* ein.
3. Laden Sie Ihre Daten mit BCP.
4. Überprüfen Sie die Datenintegrität, die sonst über Regeln, Einschränkungen und Trigger gewährleistet worden ist (lassen Sie beispielsweise Abfragen laufen).
5. Erstellen Sie Ihre Indizes neu.
6. Schalten Sie die Option *Select into/bulkcopy* wieder ab.
7. Sichern Sie Ihre Datenbank.

Am besten führen Sie jetzt einige BCP-Läufe aus, um sich mit dem praktischen Einsatz vertraut zu machen. Wenn Sie die Beispiele nachvollziehen möchten, müssen Sie die hier angegebenen Daten in Dateien auf Ihrem System kopieren.

In beiden Beispielen arbeiten Sie mit der folgenden Tabelle:

```
CREATE TABLE meinetabelle
(kund_id integer not null,
 kund_name char(30) null,
 stadt char(20) null,
 land char(1) not null DEFAULT 'D',
 plz char(10) null)
```

In dieser Tabelle sind Kundendaten enthalten, die Sie von einem anderen System erhalten haben. Die einzigen obligatorischen Felder sind *kund_id* und *bundesl*. Die zusätzlichen

Daten wie *stadt*, *kund_name* und *plz* sind optional und können aus diesem Grund NULL-Werte enthalten. Falls für *land* kein Wert eingegeben wird, lautet der Standardwert dieses Feldes auf D für Deutschland.

Im ersten Beispiel beginnen Sie mit dem Laden einer Tabelle aus einer Datendatei, die nicht mehr Spalten als die Tabelle selbst enthält.

Ihre kommagetrennte Datendatei hat folgenden Aufbau:

kund_id,stadt,land,postleitzahl

Hier zwei Beispielzeilen:

1,Berlin,D,10000
2,Augsburg,D,86199

Zum korrekten Laden dieser Daten benötigen Sie eine Formatdatei, die nachstehend aufgeführt ist (und die Sie für das Beispiel übernehmen können):

```
8.0
5
1  SQLCHAR  0  9  " ,"  1  kund_id
2  SQLCHAR  0  0  " "  0  kund_name
3  SQLCHAR  0  20  " ,"  3  stadt
4  SQLCHAR  0  2  " ,"  4  land
5  SQLCHAR  0  10  " \r\n"  5  plz
```

Beachten Sie, dass Sie die Serverspaltennummer für *kund_name* auf 0 ändern müssen, was BCP darüber informiert, dass die betreffenden Daten nicht in der Datei enthalten sind. Außerdem müssen Sie noch die Feldlänge dieser Spalte auf den Wert 0 setzen und angeben, dass kein Zeilenabschluss existiert.

Falls Sie jetzt den Ladevorgang mit BCP beginnen und weniger Spalten in der Datendatei als in der Tabelle haben, werden Sie etwas Ähnliches wie in Abbildung 19.3 sehen.

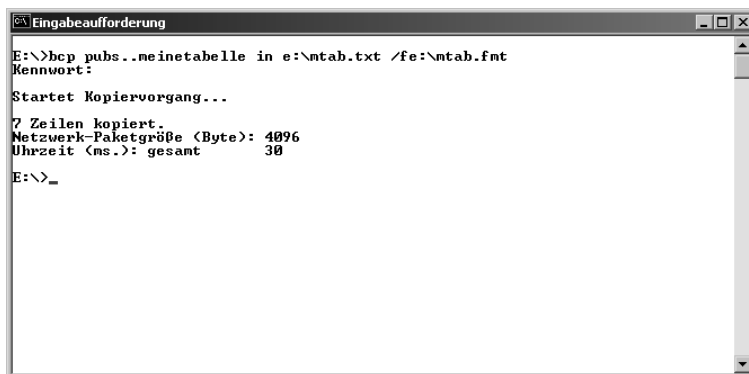


Abbildung 19.3:
Laden mit BCP

Führen Sie nun das zweite Beispiel aus. Jetzt besitzt Ihre Datei mehr Daten, als Sie tatsächlich laden möchten. Ihre Datei sollte in dem folgenden Format aufgebaut sein, wobei die Adresse 30 Zeichen lang ist:

```
kund_id,kund_name,kund_adr,stadt,land,postleitzahl
```

Betrachten Sie nun wiederum die zwei Beispielzeilen des Ergebnisses:

```
1,Gerd Gerdson,Bahnhofsweg 10,Berlin,D,10000
2,Rita Hagen,Vogelgasse 22,Augsburg,D,86199
```

Hier wird die Formatdatei die in der Datendatei tatsächlich vorhandene Spalte wiedergeben, die wiederum nicht in SQL Server existiert.

```
8.0
```

```
6
```

1	SQLCHAR	0	9	","	1	kund_id
2	SQLCHAR	0	30	","	2	kund_name
3	SQLCHAR	0	0	","	0	kund_adr
4	SQLCHAR	0	20	","	3	stadt
5	SQLCHAR	0	2	","	4	land
6	SQLCHAR	0	10	"\r\n"	5	plz

In diesem Beispiel wurde nun ein Feld für *kund_adr* aufgenommen, aber gleichzeitig durch Angabe der Serverspaltennummer 0 und der Datenlänge 0 darauf hingewiesen, dass es nicht in der Tabelle vorhanden ist. Beachten Sie, dass Sie hier trotzdem das Feldabschlusszeichen benötigen, damit BCP erkennen kann, wo ein Feld in Ihrer Datei endet.

Wie Sie selbst sehen können, ist BCP recht flexibel. Allerdings kann der Umgang mit den Formatdateien am Anfang etwas Einarbeitung erforderlich machen.

Der echte Vorteil des Massenkopierens und der Formatdateien besteht darin, dass Sie BCP als Befehlszeilenprogramm in Stapeldateien zur automatisierten Verarbeitung einbinden können.

19.3 Der Befehl BULK INSERT

Mit BULK INSERT sind Sie in der Lage, Datendateien wie ein OLE DB-Recordset zu lesen. SQL Server erhält auf diese Weise die Möglichkeit, das gesamte Recordset in einem einzigen Schritt oder, falls Sie eine vorher festgelegte Stapelgröße angeben, in mehreren Schritten in die Tabelle zu verschieben. Die Syntax des Befehls BULK INSERT lautet:



```
BULK INSERT [['Datenbankname'].['Besitzer'].['Tabellenname']
FROM Datendatei][WITH
([ BATCHSIZE [= batch_size]]
[[],] CHECK_CONSTRAINTS]
[[],] CODEPAGE [= 'ACP' | 'OEM' | 'RAW' | 'code_page']]
[[],] DATAFILETYPE [= {'char' | 'native' | 'widechar' | 'widenative'}]]
[[],] FIELDTERMINATOR [= 'field_terminator']]
[[],] FIRSTROW [= first_row]]
[[],] FORMATFILE [= 'format_file_path']]
[[],] KEEPIDENTITY]
[[],] KEEPNULLS]
[[],] KILOBYTES_PER_BATCH [= kilobytes_per_batch]]
[[],] LASTROW [= last_row]]
[[],] MAXERRORS [= max_errors]]
[[],] ORDER ({column [ASC | DESC]} [,...n]))
[[],] ROWS_PER_BATCH [= rows_per_batch]]
[[],] ROWTERMINATOR [= 'row_terminator']]
[[],] TABLOCK))]
```

Die Elemente in dieser Syntax haben folgende Bedeutung:

- *Datenbankname* ist der Name der Datenbank mit der Tabelle, in die Sie die Daten mit dem Masseneinfügen laden möchten.
- *Datendatei* bezeichnet den vollständigen Pfad und Dateinamen der Datei, die Sie importieren möchten.
- *CHECK_CONSTRAINTS* legt fest, dass Tabelleneinschränkungen während des Masseneinfügens geprüft werden sollen.
- *CODEPAGE* legt die in der Datendatei vorgefundene Codepage fest. Diese Option ist nur dann notwendig, wenn die Daten vom Zeichendatentyp sind und aus Zeichen des erweiterten Zeichensatzes bestehen.
- *DATAFILETYPE* gibt an, in welchem Format die Daten in der Datei gespeichert sind.
- *FIELDTERMINATOR* legt fest, welches Zeichen als Feldabschlusszeichen verwendet werden soll. Der Standardwert ist das Tabulatorzeichen (\t).
- *FIRSTROW* legt fest, mit welcher Zeile Sie den Masseneinfügevorgang beginnen lassen möchten. Standardmäßig ist die erste Zeile in der Datendatei eingestellt (d.h., FIRSTROW = 1).
- *FORMATFILE* legt einen vollständigen Pfad und Dateinamen einer Formatdatei fest, die in Verbindung mit dem Masseneinfügen verwendet werden soll. Diese arbeitet ähnlich wie die Formatdatei, die Sie im Zusammenhang mit dem BCP-Befehl erstellt haben.

- *KEEPIDENTITY* legt fest, dass *IDENTITY*-Werte beim Masseneinfügen ihre Werte beibehalten und keine neuen *IDENTITY*-Werte zugewiesen bekommen.
- *KEEPNULLS* legt fest, dass *NULL*-Werte beim Masseneinfügen ihre *NULL*-Werte beibehalten.
- *KILOBYTES_PER_BATCH* legt die Anzahl der Bytes fest, die auf einmal übertragen werden sollen. Einen Standardwert gibt es an dieser Stelle nicht, da der Befehl *BULK INSERT* immer versucht, die gesamten Daten in einem »Rutsch« einzuladen.
- *LASTROW* legt die letzte zu ladende Zeile der Daten fest. Standardmäßig ist dies die letzte Zeile in der Datendatei.
- *MAXERRORS* gibt die maximal zulässige Anzahl von Fehlern an, bevor *SQL Server* den *BULK INSERT*-Vorgang abbricht. Der Standardwert ist 100.
- *ORDER* ermöglicht die Festlegung der Sortierreihenfolge innerhalb der Datendatei. Falls die Datendatei sortiert ist und eine andere Sortierreihenfolge als die Daten in der Zieltabelle aufweist, kann die Angabe dieser Option diesen Vorgang erheblich beschleunigen.
- *ROWS_PER_BATCH* legt fest, wie viele Zeilen in jedem Stapel geladen werden sollen. Sie können diese Option verwenden, wenn *BATCHSIZE* nicht angegeben worden ist.
- *ROWTERMINATOR* legt das Ende einer Datenzeile fest. Der Standardwert ist das Zeichen Neue Zeile (*\n*).
- *TABLOCK* legt fest, dass für die Dauer des Masseneinfügens eine Tabellensperre ausgeübt wird. Dies kann Auswirkungen auf die Leistung des Masseneinfügens haben.

Für weitere Informationen über die Verwendung von *BULK INSERT* sei auf die *SQL Server*-Onlinedokumentation und die einschlägige Literatur verwiesen.

19.4 Verwenden der Data Transformation Services

Die Daten Ihres Unternehmens werden in aller Regel in den verschiedensten Formaten und an den verschiedensten Orten gespeichert sein. Mit *DTS* können Sie Daten aus heterogenen Quellen importieren und für heterogene Bestimmungsorte exportieren, was gänzlich über die Verwendung von *OLE DB*-Providern erledigt wird. Dabei haben Sie die Möglichkeit, sowohl die Daten als auch das Schema zu übertragen; und wenn Sie mit zwei *SQL Server 2000*-Computern arbeiten, können Sie auch Datenbankobjekte wie Indizes, Datentypen, Einschränkungen und Tabellen übertragen.

DTS legt verschiedene Schnittstellen offen, mit denen sich folgende Aufgaben durchführen lassen:

- Zeitplanung von DTS-Paketen zur automatischen Übertragung und Umwandlung von Daten.
- Erstellen angepasster DTS-Pakete, die über die COM-Schnittstelle in Produkte von Drittanbietern integriert werden können.
- Übertragen von Daten zu und aus folgenden Zielen bzw. Quellen:
 - ▶ Microsoft Access
 - ▶ Microsoft Excel
 - ▶ SQL Server
 - ▶ ODBC-kompatible Datenbanken wie Oracle, DB2 und Informix
 - ▶ ASCII-Textdateien (sowohl einfache Dateien als auch Dateien mit Trennzeichen)

Eine Datenumwandlung kann jede Operation sein, die für Ihre Daten vor der endgültigen Speicherung am Zielort vorgenommen wird. Hierzu kann die Berechnung neuer Werte, die Zusammenführung von Werten, die Aufteilung von Werten (wie etwa die Aufteilung eines Namensfeldes in den Vor- und Nachnamen) wie auch jede Art von Datenbereinigung gehören, die Sie im Rahmen der Übertragung vornehmen möchten.

Das Programm DTS gliedert sich in drei Hauptbestandteile:

- *Pakete*: Ein Paket umfasst sämtliche Schritte und Tätigkeiten, die zur Ausführung des Imports, Exports oder der Umwandlung erforderlich sind.
- *Import- und Export-Assistenten*: Mit diesen Assistenten können Sie interaktiv DTS-Pakete erstellen. Diese Werkzeuge stehen Ihnen außerdem über die Befehlszeile zur Verfügung, was Ihnen nicht nur die Möglichkeit gibt, sie in Skripts zu »verpacken«, sondern außerdem auch viele der in den Assistenten vorhandenen Dialoge zu umgehen.
- *DTS-Designer*: Der DTS-Designer stellt eine vollständige Desktopumgebung mit Toolboxen und Paletten dar, in der erfahrene Administratoren komplexe Tasks und Pakete grafisch erstellen können. Eine Beschreibung des DTS-Designers würde den Rahmen dieses Buches sprengen. Hier sei auf die Onlinedokumentation von SQL Server und die einschlägige Literatur verwiesen.

DTS-Pakete

Wie schon weiter oben erwähnt, besteht ein DTS-Paket aus einem Satz von einem oder mehreren Tasks, die in einer festgelegten Reihenfolge ausgeführt werden. Diese Pakete können automatisch über die Import- und Export-Assistenten, manuell über den Einsatz

einer Skriptsprache oder grafisch über den DTS-Designer erstellt werden. Nachdem Sie ein DTS-Paket erstellt haben, ist dieses vollkommen eigenständig und lässt sich über den Enterprise Manager, den Zeitplandienst oder über das Befehlszeilenprogramm `dtsrun` ausführen.

Ein DTS-Paket können Sie in den verschiedenen Formaten speichern, wobei jedes dieser Formate seine besonderen Vorteile besitzt.

Die Datenbank msdb des SQL Servers

Sie können Ihre DTS-Pakete in der Datenbank *msdb* speichern, was den Vorteil hat, dass dann andere SQL Server 2000 auf diese Pakete zugreifen können. Auf diesem Wege ist weniger Verwaltungsaufwand als bei Verwendung des Microsoft Repository im Spiel, was Ihnen dabei hilft, den Speicherplatzbedarf und den Zeitaufwand zu minimieren, der für den Zugriff und die Verwendung des Pakets erforderlich ist.

Microsoft Repository

Wenn Sie ein Paket im Repository speichern, können Sie es für andere Computer mit SQL Server 2000 zugänglich machen. Das Repository bietet die Möglichkeit, Metadaten anderen Anwendungen zur Verfügung zu stellen. Ein anderer wesentlicher Vorteil des Repository ist die Fähigkeit, die Herkunft oder den Verlauf der Umwandlungen, denen die Daten unterzogen worden sind, nachvollziehen zu können. Dies schließt Datenquellen, Zielorte und Änderungen ein, die an den Daten vorgenommen worden sind.

COM-strukturierte Speicherdatei

COM stellt eine Speicherarchitektur zur Verfügung, die aus Datenobjekten und Datenstreams besteht. Die Datenobjekte entsprechen dabei den Verzeichnissen in einem Dateisystem, während die Stream-Objekte den Dateien entsprechen. Einer der Hauptvorteile des COM-strukturierten Speichermodells besteht darin, dass die Stream-Objekte sehr einfach über die Dateiserver des Netzwerks oder über E-Mail weiterverteilt werden können. Jedes COM-kompatible Programm oder jede entsprechende Programmiersprache kann dann auf diese COM-Objekte zugreifen.



Da DTS aus einem Satz von COM-Komponenten und -Servern besteht, können Sie jede COM-kompatible Programmiersprache wie etwa Visual Basic, PerlScript oder JScript verwenden, um Ihre eigenen angepassten Transformationen, Pakete, Skriptobjekte und Datapump-Schnittstellen zu definieren.

Paketsicherheit

Auf alle Pakete lassen sich zwei Sicherheitsmaßnahmen anwenden: DTS-Besitzer und Kennwort sowie DTS-Operatorkennwort. Außerdem lässt sich ein DTS-Paket, das in einer COM-strukturierten Speicherdatei enthalten ist, verschlüsseln. Bei der Verschlüsselung eines Pakets werden alle seine Komponenten außer CreationDate, Description, ID, Name und VersionID verschlüsselt.

Ein Benutzer oder eine Anwendung mit dem Kennwort des DTS-Besitzers hat den vollständigen Zugriff auf alle Komponenten des Pakets. Wird ein Paket ohne ein Besitzerkennwort gesichert, dann ist es nicht verschlüsselt. Wenn Sie ein Paket mit einem Besitzerkennwort sichern, wird es standardmäßig gleichzeitig verschlüsselt.

Ein Benutzer oder eine Anwendung mit dem Kennwort des DTS-Operators kann das Paket ausführen, aber auf keine Komponente des Pakets direkt zugreifen.

Paketkomponenten

Ein DTS-Paket besteht aus mehreren Komponenten, die zusammen in ihrer Interaktion das Paket und die Workflows ausmachen: Taskobjekte, Schrittobjekte und Datapump. Die folgenden Abschnitte gehen näher auf diese Objekte ein, so dass Sie erkennen können, wie diese Komponenten miteinander in der Paketarchitektur zusammenwirken.

Taskobjekte

Pakete bestehen aus einem oder mehreren Tasks. Jeder Task definiert einen Prozess oder eine Aktion, der oder die ausgeführt werden soll. Tasks werden für die folgenden Aufgaben eingesetzt:

- Ausführen einer SQL-Anweisung oder eines Stapels
- Verschieben von Daten von einer OLE DB-Datenquelle in ein OLE DB-Datenziel
- Starten von externen Programmen, Befehlen oder Stapeldateien
- Sammeln der Ergebnisse von anderen DTS-Paketen
- Ausführen anderer DTS-Pakete
- Ausführen eines COM-kompatiblen Skripts (wie Visual Basic, JScript, PerlScript usw.)



Obwohl man einen Task mit mehr als einem Schritt verknüpfen kann, lässt sich ein Task nur in einer Instanz zu einem bestimmten Zeitpunkt ausführen. Wenn also sowohl Schritt 2 als auch Schritt 4 mit Task A verknüpft sind und Schritt 2 gerade läuft, kann Schritt 4 erst dann beginnen, wenn Task A in Schritt 2 abgeschlossen ist.

Schrittobjekte

Schrittobjekte steuern die Ausführung von Tasks, aus denen ein DTS-Paket besteht. Tasks, denen kein eigenes Schrittobjekt zugeordnet ist, werden niemals ausgeführt. Mit Schrittobjekten erstellen Sie Ihren Workflow.

Schritte lassen sich nach verschiedenen Methoden ausführen. Beispielsweise können Sie für einen Schritt festlegen, dass er nur nach Abschluss, nur nach erfolgreichem Abschluss oder nur bei fehlerhaftem Abschluss des vorhergehenden Schrittes ausgeführt wird. Die Art der Beziehung basiert auf *Rangfolgeeinschränkungen*. Dies hat zur Folge, dass der letztere Schritt eine Rangfolgeeinschränkung gegenüber dem vorhergehenden Schritt aufweist. Nachdem alle Rangfolgeeinschränkungen für einen Schritt erfüllt sind, kann der Schritt ausgeführt werden.

Nehmen wir als Beispiel ein Paket mit zwei Schritten an. Schritt 1 ist eine `CREATE TABLE`-Anweisung, Schritt 2 eine `CREATE INDEX`-Anweisung. Schritt 2 wird nur ausgelöst, nachdem Schritt 1 erfolgreich abgeschlossen wurde.

Da Taskobjekte von den Schrittobjekten getrennt sind, können Sie einem bestimmten Taskobjekt einen oder mehrere Schritte zuordnen.



Schrittobjekte können auch bedingt in Abhängigkeit vom Laufzeitzustand ausgeführt werden. Außerdem ist es zur Leistungsverbesserung möglich, Schritte gleichzeitig – *parallel* – auszuführen. Dazu erstellen Sie Schritte, denen keine Rangfolgeeinschränkungen zugewiesen werden.

Um Rangfolgeeinschränkungen zuzuweisen, können Sie die folgenden Werte von den vorhergehenden Schritten oder die Rückgabecodes, die von den Anweisungen einer Skriptsprache stammen, verwenden:

- **Bei Erfolg:** Schritt 2 wartet mit dem Start so lange, bis Schritt 1 erfolgreich verarbeitet worden ist.
- **Bei Fehler:** Erlaubt eine alternative Verzweigung zu Schritten, die nach einem Fehler ausgeführt werden sollen.
- **Bei Beendigung:** Schritt 2 startet nicht, bevor nicht Schritt 1 abgeschlossen worden ist; gleichgültig, ob erfolgreich oder fehlerhaft.

Durch den Einsatz von Skriptsprachen können Sie logische Verzweigungen der Art IF...THEN...ELSE in Ihren Workflows realisieren.

DTS-Verbindungsobjekte

DTS-Pakete, die sich mit Datenquellen verbinden, besitzen zwei Arten von Verbindungsobjekten:

- *Datenquellenverbindungen* spezifizieren die Datenquellen- und Zielsever, den Standort und das Format der Daten und notwendigen Kennwörter.
- *Dateiverbindungen* spezifizieren die Quell- und Zieldateien einschließlich des Standortes und des Formats der Daten.

Eine Verbindung wird hergestellt, wenn ein Taskobjekt, das diese Verbindung benötigt, in einem Schritt aufgerufen wird. Andernfalls befindet sich die Verbindung im Ruhezustand.

DTS-Datapump

Die Datapump stellt einen OLE DB-Serviceprovider dar, der die Importe, Exporte und die Data Transformation Services bearbeitet. Er lädt diese Dienste als prozessinterner COM-Server in den Ausführungsbereich von SQL Server 2000. Die Datapump wird in Verbindung mit einem oder mehreren Verbindungsobjekten eingesetzt. Weil die Datapump auf COM basiert, können Sie ihre Funktionalität mit jeder COM-kompatiblen Programmiersprache erweitern. Über diese Erweiterungen können Sie Ihre eigenen Programme mit komplexen Verarbeitungsschritten erstellen, die Sie dann implementieren und nach Bedarf wiederverwenden. Abbildung 19.4 zeigt den Informationsfluss durch die Datapump-Komponente.

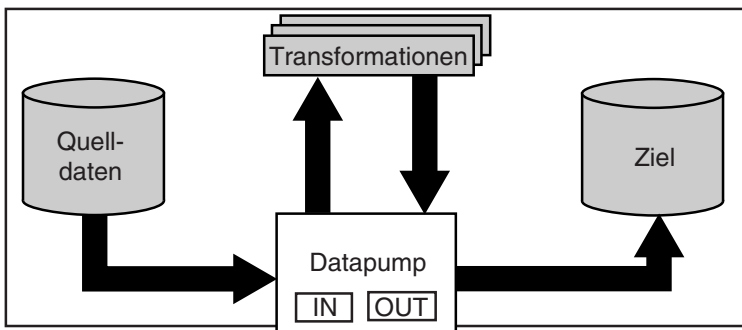


Abbildung 19.4:
Die DTS-Architektur

DTS-Objektübertragung

Die DTS-Objektübertragung stellt eine Aktualisierung des Transfer Manager Interface (TMI) von SQL Server 6.x dar. Mit der DTS-Objektübertragung können Sie Daten, Schemas und Objekte von SQL Server zu anderen SQL Server-Datenquellen übertragen. Dazu verwenden Sie die Import- und Export-Assistenten, die Sie weiter unten in der heutigen Lektion genauer kennen lernen.

Die DTS-Objektübertragung verwendet SQL-DMO anstelle der OLE DB-Funktionalität, um Verbindungen zu Datenquellen und Datenzielen herzustellen.

Daten mithilfe des DTS-Import/Export-Assistenten exportieren

Über die Verwendung der DTS-Assistenten können Sie interaktiv DTS-Pakete für das Importieren, Exportieren und die Umwandlung Ihrer Daten erstellen. DTS prüft dabei vorher die Zielobjekte auf Vorhandensein. Falls die Objekte bereits existieren, können Sie sie mit dem Assistenten über einen Zwischenschritt löschen und neu erstellen.

Als DTS-Benutzer können Sie die folgenden Operationen ausführen:

- Eine Tabelle kopieren
- Die Ergebnisse einer Abfrage kopieren
- Im Assistenten eine Abfrage mit dem Abfrage-Generator erstellen
- Die Verbindungseinstellungen für Datenquellen und -ziele festlegen
- Daten einschließlich der Spaltenüberschriften, Datentypen, Größen, Dezimalstellen, Genauigkeiten und der NULL-Zulässigkeit transformieren
- Ein COM-kompatibles Skript ausführen
- Pakete im Repository, in der Datenbank *msdb* oder in einer COM-strukturierten Speicherdatei ablegen
- Ein fertiges Paket an den Zeitplandienst zur Ausführung übergeben

Der DTS-Export-Assistent

Führen Sie zum Exportieren von Daten die folgenden Schritte aus:

1. Öffnen Sie den SQL Enterprise Manager und verbinden Sie sich mit Ihrem Server.
2. Wählen Sie im Menü EXTRAS den Befehl ASSISTENTEN.

3. Erweitern Sie im Dialogfeld ASSISTENTEN AUSWÄHLEN den Knoten *Data Transformation Services* und markieren Sie *DTS-Export-Assistent* (siehe Abbildung 19.5). Klicken Sie auf OK.

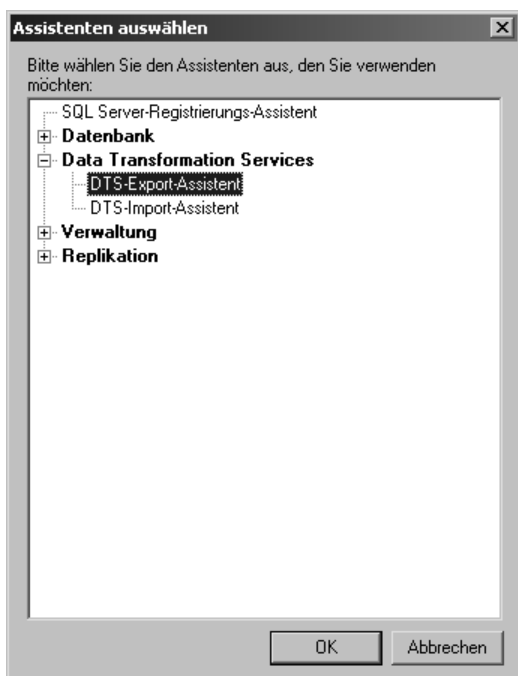


Abbildung 19.5:
Den DTS-Export-Assistenten auswählen

4. Klicken Sie im Willkommen-Bildschirm des DTS-Import/Export-Assistenten auf WEITER.
5. Im Dialogfeld DATENQUELLE WÄHLEN (siehe Abbildung 19.6) können Sie einen Exportserver wählen, die Anmeldeinformationen festlegen und die Exportdatenbank bestimmen. Nehmen Sie für das Beispiel die Datenbank *pubs* Ihres lokalen SQL Servers.
6. Über die Schaltfläche ERWEITERT gelangen Sie zu den erweiterten Verbindungseigenschaften, die Sie mit diesem Paket verbinden können. Kehren Sie zum Dialogfeld DATENQUELLE WÄHLEN zurück und klicken Sie auf WEITER.
7. Im Dialogfeld ZIEL WÄHLEN finden Sie den gleichen Satz von Eigenschaften wie im Dialogfeld DATENQUELLE WÄHLEN vor. Im Beispiel erstellen Sie nun eine neue Datenbank in Ihrem lokalen Server, in die Sie Ihre Daten exportieren können. Wählen Sie dazu im Kombinationsfeld DATENBANK den Eintrag *<neu>* aus. Daraufhin erscheint das Dialogfeld DATENBANK ERSTELLEN (siehe Abbildung 19.7)

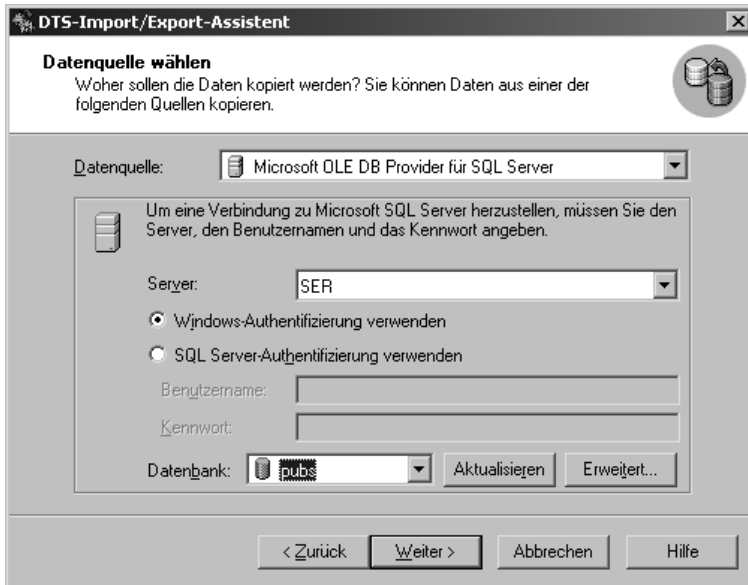


Abbildung 19.6:
Eine Datenquelle
wählen

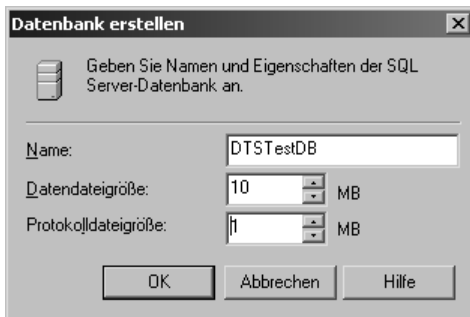


Abbildung 19.7:
Hier legen Sie die Eigenschaften für die neue Ziel-
datenbank fest.

8. Füllen Sie das Dialogfeld DATENBANK ERSTELLEN wie in Abbildung 19.7 aus. Der Name der Datenbank lautet DTSTestDB, die Dateigröße beträgt 10 MB und das Protokoll sollte 1 MB groß sein. Wenn Sie fertig sind, klicken Sie auf OK. Sie kehren damit zum Dialogfeld ZIEL WÄHLEN zurück (siehe Abbildung 19.8). Klicken Sie auf WEITER.
9. Im Dialogfeld TABELLE KOPIEREN ODER DATENBANK ABFRAGEN (siehe Abbildung 19.9) können Sie festlegen, ob Sie Tabellen kopieren, eine Abfrage ausführen oder Datenbankobjekte übertragen wollen. Klicken Sie auf WEITER.



Abbildung 19.8:
Das fertig gestellte
Dialogfeld Ziel wäh-
len

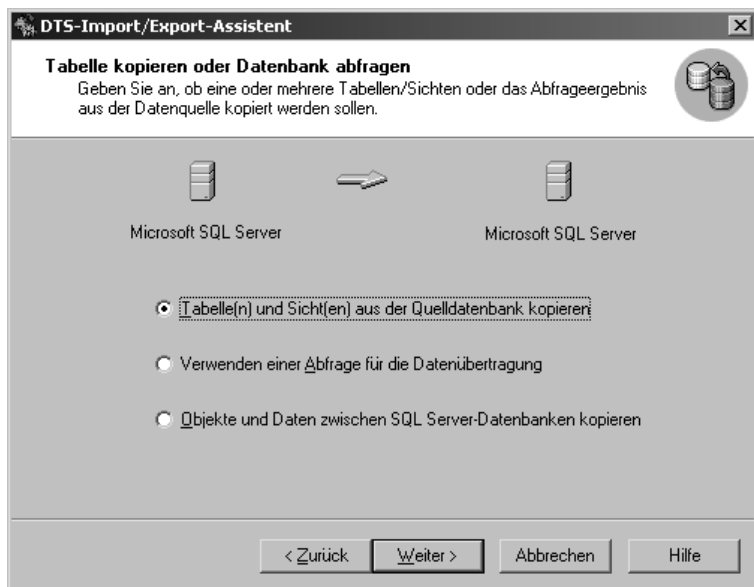


Abbildung 19.9:
Eine Tabellenkopie
oder eine Abfrage
auswählen

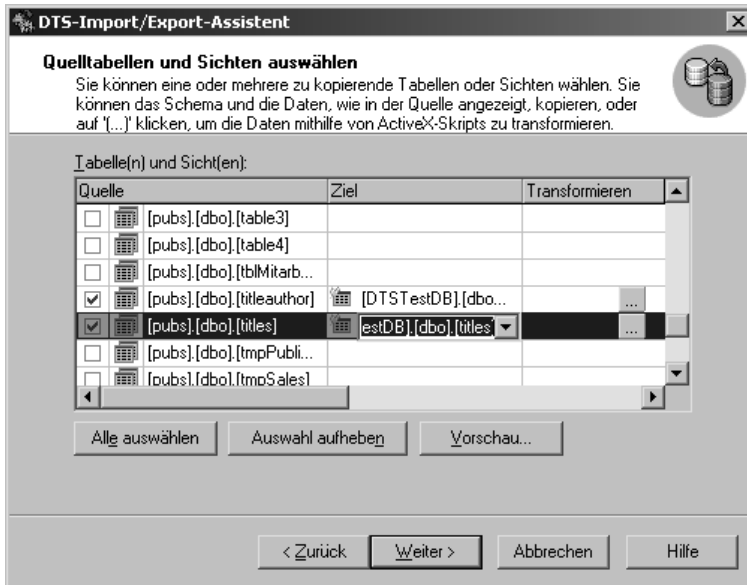


Abbildung 19.10:
Quelltabelle(n) und
Sichten auswählen

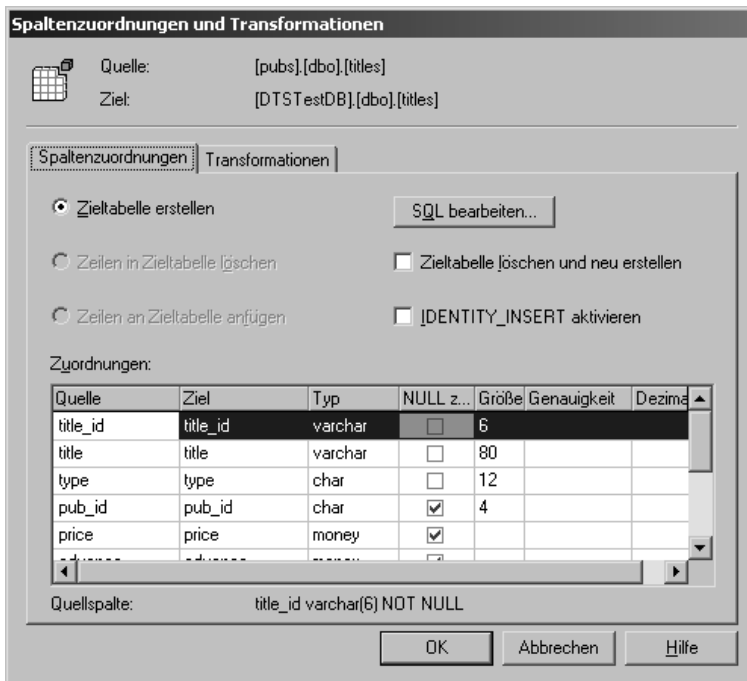


Abbildung 19.11:
Die Registerkarte
Spaltenzuordnungen
im Dialogfeld
Spaltenzuordnungen
und Transfor-
mationen

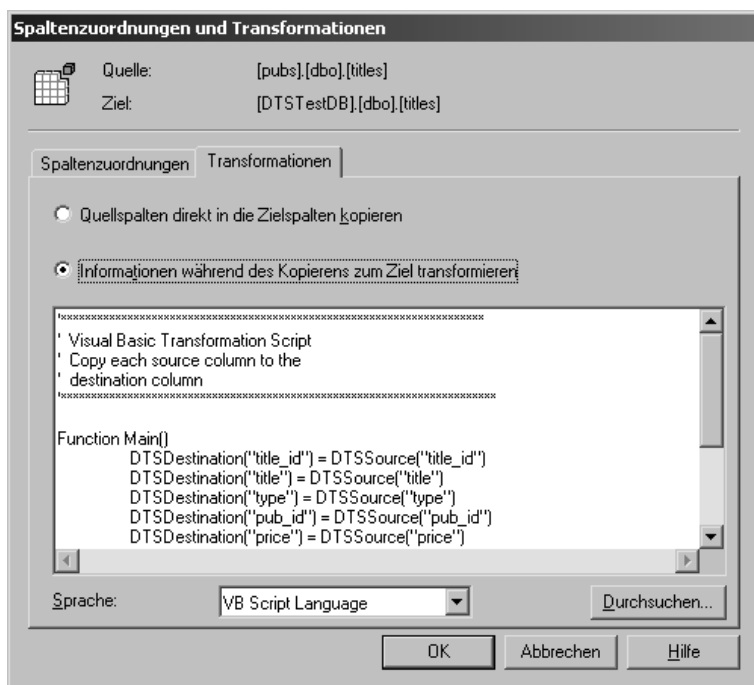


Abbildung 19.12:
Die Registerkarte
Transformationen
im Dialogfeld Spal-
tenzuordnungen
und Transformatio-
nen

10. Markieren Sie im Dialogfeld **QUELLTABELLEN UND SICHTEN AUSWÄHLEN** (siehe Abbildung 19.10) die Tabellen *authors*, *titles* und *titleauthor*. Verändern Sie nicht die Zieltabellen. Vielleicht haben Sie bemerkt, dass jede Tabelle über eine Schaltfläche in der Spalte **TRANSFORMIEREN (...)** verfügt. Wenn Sie auf diese Schaltfläche klicken, gelangen Sie zu Dialogfeldern mit erweiterten Transformationsoptionen, wie es die Abbildungen 19.11 und 19.12 zeigen.
11. Klicken Sie im Dialogfeld **QUELLTABELLEN UND SICHTEN AUSWÄHLEN** auf **WEITER**. Damit gelangen Sie zum Dialogfeld **Paket speichern, Ausführungstermin planen und replizieren** (siehe Abbildung 19.13). Legen Sie in diesem Dialogfeld fest, dass Sie das Paket sofort ausführen und in **SQL Server** speichern möchten.
12. Als Nächstes erscheint das Dialogfeld **DTS-PAKET SPEICHERN**. Hier spezifizieren Sie einen Paketnamen wie zum Beispiel *TestExport*. Außerdem legen Sie den Speicherort fest (siehe Abbildung 19.14).
13. Im Abschlussdialogfeld des **DTS-Import/Export-Assistenten** klicken Sie auf **FERTIG STELLEN**, um das Paket zu erstellen und zu speichern. Da Sie festgelegt haben, dass das Paket sofort auszuführen ist, erscheint ein Fortschrittsdialogfeld mit einer Zusammenfassung (siehe Abbildung 19.15), so dass Sie die Übertragung verfolgen können.

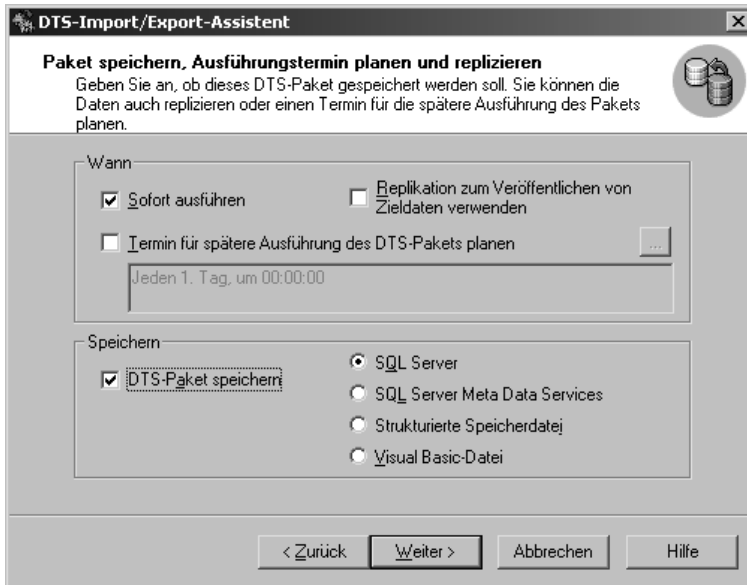


Abbildung 19.13:
Das Dialogfeld
Paket speichern,
Ausführungstermin
planen und replizie-
ren

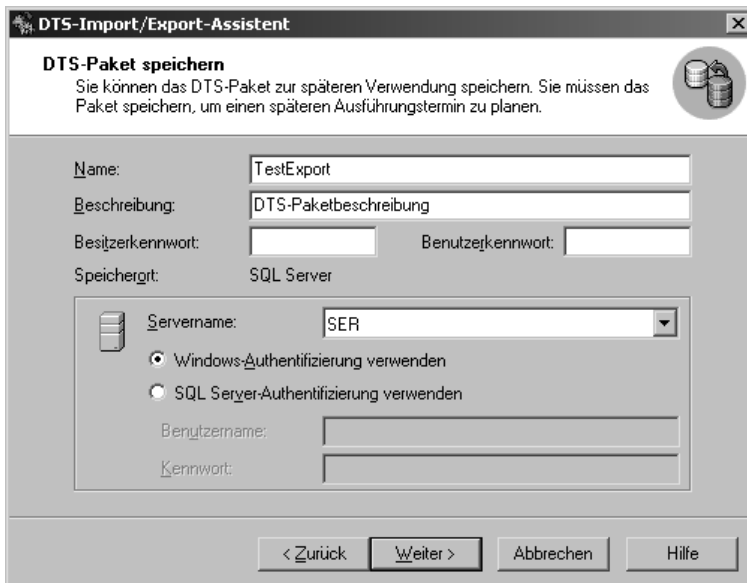


Abbildung 19.14:
Ein DTS-Paket spei-
chern

14. Klicken Sie im abschließenden Meldungsfeld auf OK und dann im Dialogfeld **PAKET AUSFÜHREN** auf **FERTIG**. Sehen Sie sich jetzt Ihre neue Datenbank und die übertragenen Tabellen an.

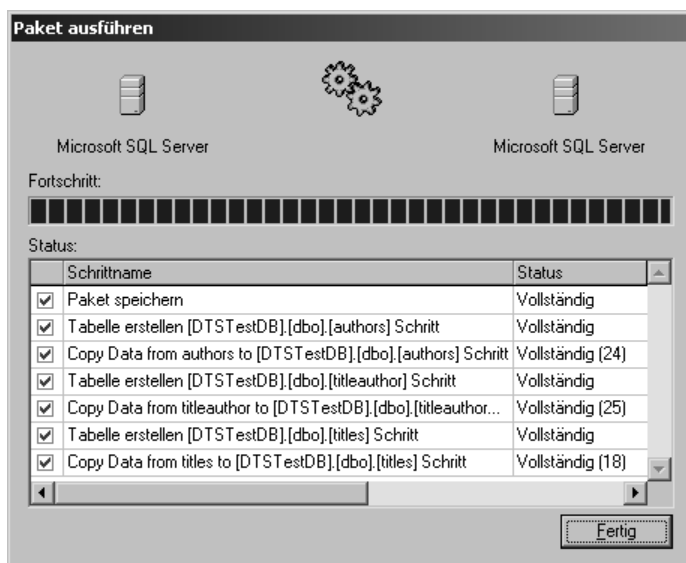


Abbildung 19.15:
Fortschrittsdialogfeld wäh-
rend der Datenübertragung

Übertragen von Datenbankobjekten

Die DTS-Objektübertragung können Sie dazu verwenden, über die DTS-Dialogfelder Datenbankobjekte wie gespeicherte Prozeduren und Sichten von einem SQL Server 2000-Computer zu einem anderen zu übertragen. Diese Art der Datenübertragung lässt sich auch über verschiedene Plattformen und Sortierreihenfolgen hinweg durchführen.

Zur Arbeit mit der Objektübertragung folgen Sie den Arbeitsschritten 1 bis 9 aus dem Abschnitt »Der DTS-Export-Assistent«. In Schritt 9 wählen Sie jetzt die Option *Objekte und Daten zwischen SQL Server-Datenbanken kopieren* (siehe dazu Abbildung 19.9). Dann können Sie die Standardeinstellung (alle Objekte übertragen) übernehmen oder Sie schalten das betreffende Kontrollkästchen aus und wählen dann einzelne Objekte zur Übertragung aus (zum Beispiel Tabellen, Sichten, gespeicherte Prozeduren und benutzerdefinierte Daten).

19.5 Zusammenfassung

In der heutigen Lektion haben Sie verschiedene Methoden zum Übertragen von Daten von einem Standort zu einem anderen kennen gelernt. SQL Server 2000 bietet drei Hauptmethoden zum Übertragen von Daten und eine Methode zum Übertragen und gleichzeitigen Umwandeln an.

Zum Übertragen sämtlicher Daten und Tabellenschemata von einem Ort zu einem anderen können Sie auch die Datensicherungs- und Wiederherstellungstechniken einsetzen – eine Lösung, die keinerlei Änderungen an den Daten vornimmt. Das Programm BCP (Bulk Copy Program) zum Massenkopieren setzen Sie ein, wenn Sie einfach nur Daten von einem Ort an einen anderen verschieben möchten. Dieses Werkzeug nimmt zwar keine Änderungen an Ihren Daten vor, erlaubt Ihnen jedoch die Auswahl bestimmter Spalten und Zeilen in einer Datenquelle und deren Übertragung in eine andere.

Das komfortabelste Tool ist der DTS-Manager von SQL Server 2000. Dieses Werkzeug ähnelt nur auf den ersten Blick dem alten Transfer Manager Interface von SQL Server 6.x. DTS kann dazu eingesetzt werden, Daten und/oder Schemas zu übertragen. DTS verfügt außerdem über die Fähigkeit zur Umwandlung (Änderung) Ihrer Daten. Mit DTS können Sie das Format Ihrer Daten von einem Datentyp in einen anderen ändern, die Daten neu strukturieren und anderen Standorten zuordnen sowie die Konsistenz und Gültigkeit der Daten sicherstellen. Darüber hinaus ist es möglich, mit logischen Anweisungen in Skripten die Daten zusammenzufassen, zu dekodieren, zu konvertieren und zu zerlegen.

19.6 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

F *Sollte ich zum Verschieben von Daten besser DTS oder die Replikation einsetzen?*

A Normalerweise setzen Sie die Replikation dann ein, wenn Sie die Daten auf anderen SQL Servern verfügbar machen möchten. Die Replikation kann regelmäßig ausgeführt werden und beinhaltet die entsprechende Funktionalität zur Gewährleistung der Konsistenz Ihrer Daten.

F *DTS sieht zwar interessant aus, aber brauche ich das in der Praxis wirklich?*

A In der Praxis besteht ein Hauptteil der Tätigkeiten oft aus reinen Datenaufklärungsarbeiten. Viele Unternehmen setzen Großrechnersysteme dafür ein, regelmäßig ASCII-Textdateien zu erzeugen, die dann in den SQL Server importiert werden müssen. Mit dem DTS-Programm haben Sie die Möglichkeit, nicht nur die

Daten zu importieren, sondern darüber hinaus auch notwendige Änderungen an den Datentypen, Aufteilungen von Spalten (etwa in Vor- und Nachname) und viele andere Umwandlungen dieser Art vornehmen zu können.

- F** *Sie haben kurz den DTS-Designer erwähnt. Ist seine Bedienung eigentlich schwer zu erlernen?*
- A** Das hängt ganz von Ihnen ab. Administratoren, die mit einer grafischen Programmierschnittstelle wie etwa Visual Basic oder Delphi vertraut sind, sollten problemlos damit zurechtkommen. Der Einsteiger ist vielleicht am Anfang mit der Vielfalt der Funktionen überfordert. Empfehlenswert ist, Ihre Pakete über die Assistenten zu sichern und sie danach im Designer zu betrachten. Das erleichtert Ihnen den Einstieg in den Gebrauch des Designers.

Kontrollfragen

1. Welche Berechtigungen sind zum Einsatz des Programms BCP erforderlich?
2. Welche Datenbankobjekte erzwingt und ignoriert BCP beim Übertragen von Daten in eine Tabelle?
3. Welche Voraussetzungen sind zu erfüllen, um ein schnelles BCP durchführen zu können?
4. Was können Sie beim Einsatz von DTS mit Ihren Daten alles anstellen?
5. Sie möchten eine Datenbank auf einem Server mit Intel Pentium II 400 MHz, der unter Windows 2000 läuft, auf eine DEC Alpha mit Windows NT 4.0 portieren. Außerdem möchten Sie noch die Sortierreihenfolge ändern. Wie können Sie diese Ziele erreichen?

Übungen

1. Setzen Sie den DTS-Import-Assistenten für folgende Aufgaben ein:
Erstellen Sie eine neue Datenbank namens *DTSNWind*.
Übertragen Sie von der *Northwind*-Datenbank sämtliche Objekte, außer den eigentlichen Daten.
2. Setzen Sie den Export-Assistenten dazu ein, sämtliche Daten von der *Northwind*-Datenbank in die neu erstellte Datenbank *DTSNWind* zu übertragen.



Leistungs-
überwachung

In der heutigen Lektion lernen Sie Möglichkeiten zur Konfiguration und Überwachung von SQL Server 2000 kennen – einschließlich Hardware, Windows 2000 (mit Windows 9x können Sie hier nicht viel ausrichten) und SQL Server selbst. Die Systemleistung von SQL Server lässt sich mit verschiedenen Tools überwachen. Dazu gehören der Windows-Systemmonitor, der SQL Server Profiler und der Indexoptimierungs-Assistent.



Eines muss von Anfang an klar sein: Die erste Voraussetzung für eine gute Leistung Ihrer Datenbank ist ein richtig normalisierter physischer Datenbankentwurf. Wenn der Datenbankentwurf dagegen schlecht umgesetzt ist (oder gar kein richtiges Konzept vorliegt), dann kann auch keine Einstellung an der Hard- und Software zu besseren Leistungen der Datenbank verhelfen.

20.1 Hardware auswählen und optimieren

Der erste Schritt besteht in der richtigen Auswahl der Hardware. Es wird oft gefragt, welches die wichtigste Komponente eines Systems mit SQL Server ist. Bei einem gut funktionierenden Computer müssen alle Komponenten optimal zusammenarbeiten. Wenn es aber um eine bestimmte Hardwarekomponente geht, die bestimmt werden soll, dann wäre an erster Stelle der Hauptspeicher zu nennen – SQL Server kann eigentlich gar nicht genug RAM (Random Access Memory) haben! Schauen Sie sich im Folgenden die einzelnen Hardwarekomponenten einmal genauer an.

Hauptspeicher

SQL Server speichert Daten, Indexseiten und Protokolldatensätze im RAM. In den Speicher werden außerdem auch kompilierte Abfragen und gespeicherte Prozeduren geladen. Dabei verwendet SQL Server automatisch den gesamten verfügbaren RAM. In diese Speichernutzung können Sie auch eingreifen – wie Sie dies tun, erfahren Sie später in der heutigen Lektion.

Eine andere Frage besteht darin, wie viel RAM Ihr Computer haben sollte – eine gute Ausgangsgröße für Windows 2000-Systeme mit SQL Server dürften etwa 256 MB RAM sein. Eine Investition in mehr Speicher zahlt sich aber immer aus! Kleine Implementierungen von SQL Server kommen unter Umständen mit 128 MB RAM aus. Bei den heutzutage niedrigen Preisen für Speicher sollten Sie allerdings mit so viel RAM wie möglich beginnen; eine Empfehlung, die übrigens auch für Computer mit Windows 9x gilt. SQL Server gibt sich zwar auch mit minimal 32 MB zufrieden, jedoch ist dies nicht die Größenordnung, bei der Sie die beste Leistung des Servers erhalten werden. Unter Windows NT 4.0 kann SQL Server einen Speicherraum von 2 GB RAM bzw. 3 GB mit der Windows NT 4.0 Server

Enterprise Edition nutzen. Mit AWE- (Address Windowing Extensions) Unterstützung kann SQL Server bis zu 8 GB RAM unter Windows 2000 Advanced Server und 64 GB RAM unter Windows 2000 Datacenter Server verwenden. Weitere Details zu AWE entnehmen Sie bitte der Onlinedokumentation von SQL Server.

Prozessor

Wie bei anderen Anwendungen schadet es auch bei SQL Server nicht, möglichst viel CPU-Leistung zu haben. Zwar ist SQL Server nicht unbedingt ein besonders CPU-intensives Programm, profitiert aber je nach Arbeitslast von einem leistungsfähigeren Prozessor. Sowohl SQL Server als auch Windows 2000 können Multiprozessorsysteme nutzen. SQL Server 2000 unterstützt in der Standard Edition bis zu vier Prozessoren, in der Enterprise Edition sogar 32 Prozessoren. Der SQL Server ist für eine Verteilung der Arbeitslast auf mehrere Prozessoren und Thread-Parallelisierung ausgelegt (d.h., SQL Server kann gleichzeitig auf mehreren CPUs laufen und dabei eine einzelne Abfrage in mehreren Threads parallel verarbeiten).

Netzwerk

Das Netzwerk, das oft in seiner Bedeutung unterschätzt wird, stellt einen kritischen Faktor für den Erfolg Ihrer SQL Server-Implementierung dar. Die erste Überlegung ist, welche Art von Netzwerkkarte Sie für Ihren Server brauchen: Sie sollten hier zu einer 32-Bit- oder 64-Bit-Busmastering-Karte greifen, die Ihnen nicht nur die beste Leistung hinsichtlich des Bruttodatendurchsatzes liefert, sondern außerdem auch die geringsten Belastungen für die CPU mit sich bringen wird. Eine weitere Erwägung schließt den Typ der Netzwerkkarte ein: Die meisten Netzwerke sind immer noch Ethernet-Netze mit 10 MB, wobei Varianten mit 100 MB wie auch ATM-, FDDI- und andere Netzwerktypen zunehmend populär werden. Vergessen Sie nicht, dass sämtliche Datenbankaktivitäten über das Netzwerk gesendet werden müssen! Bei einer Ethernet-Netzwerkkarte mit 10 MB können Sie von einem Durchsatz von 1,2 MB pro Sekunde ausgehen. Möchten Sie mit SQL Server mehr als diesen Datenumsatz verarbeiten, dann brauchen Sie eine zweite oder sogar eine dritte Netzwerkkarte.



Stellen Sie sicher, dass Sie jede Netzwerkkarte auch in einem physikalisch getrennten Netzwerk betreiben. Legen Sie mehrere Netzwerkkarten in dasselbe physikalische Netzwerk, werden Sie Fehler wegen doppelter Namen bekommen, was das Versagen ein oder mehrerer Karten zur Folge haben kann. Die Ursache liegt darin, dass SQL Server seine Anwesenheit im Netzwerk immer nur jeweils einer Netzwerkkarte bekannt gibt. Sobald die zweite Netzwerkkarte den Servernamen meldet, weist die erste Karte darauf, dass der Name schon

existiert. Da NetBIOS keinen hierarchischen Namensraum hat, müssen sämtliche Namen im Netzwerk eindeutig sein. Aus diesem Grund wird die zweite Netzwerkkarte nun Fehler über Namensprobleme zurückliefern und nicht mehr korrekt arbeiten. Legen Sie daher Ihre zweite Netzwerkkarte zur Vermeidung dieses Problems in ihr eigenes Subnetz. Wenn Sie DNS nur mit dem Active Directory einsetzen, spielen diese Betrachtungen keine große Rolle, aber die meisten Umgebungen verwenden noch eine Zeit lang sowohl NetBIOS als auch DNS.

Festplatten

Festplatten sind eine kritische Komponente jedes Servers, weil sich dort die Daten befinden. Wenn Sie auf Ihre Daten zugreifen, müssen diese normalerweise von der Festplatte geholt werden. Die Geschwindigkeit, mit der die Daten auf die Festplatte zurückgeschrieben werden, ist ebenfalls wichtig. Darüber hinaus ist auch die Bandbreite zu beachten. So weit, so gut. Zuerst müssen Sie sich darüber klar werden, welchen Festplattentyp Sie einsetzen wollen. Dabei sind mehrere kleine Platten besser geeignet als wenige große. SCSI (Small Computer System Interface) schneidet besser als IDE (Integrated Disc Electronic) ab, kostet aber mehr, obschon es das Geld wert ist. Außerdem kann es den Aufwand lohnen, in den Computer mehrere SCSI-Karten einzubauen und die Festplatten darauf zu verteilen. Stellen Sie auf jeden Fall sicher, dass alle Komponenten im SCSI-System dieselbe Geschwindigkeit besitzen (z.B. alle Fast oder Wide). Nicht zusammenpassende Geräte laufen nur noch mit der (niedrigen) Geschwindigkeit, die von allen Geräten eingehalten werden kann.

Eine weitere Frage, die oft gestellt wird, ist das empfohlene Layout der Festplatten. Die Systemdateien und das Transaktionsprotokoll sollten im Duplex-Betrieb laufen, womit gemeint ist, dass diese vorzugsweise auf verschiedenen SCSI-Controllern gespiegelt werden. Bei den Datendateien stehen zwei Optionen zur Verfügung, die jede für sich sinnvoll ist: Die teuerste Option besteht darin, Stripe-Festplatten einzurichten und diese Stripes zu spiegeln, was Sie über Hardware-Striping mit anschließender Spiegelung über Windows 2000 bewerkstelligen können. Auf diese Weise gibt es beim Ausfall einer Platte im System keine Leistungseinbrüche. Viele neuere Hardwaresysteme unterstützen jetzt systemseitig diesen Typ des Festplattenschutzes (als RAID 1+0 oder RAID 10 bezeichnet). Sie können aber auch die Festplatten unter Verwendung von RAID 5 einrichten, was die kostengünstigere Lösung darstellt. Allerdings schneidet in diesem Fall die Leistung schlechter ab, da Sie es bei einem Ausfall einer Ihrer Platten mit einem bedeutenden Leistungsverlust zu tun haben werden. Die Abbildungen 20.1 und 20.2 zeigen empfohlene Lösungen für mittleres und hohes Leistungsniveau.

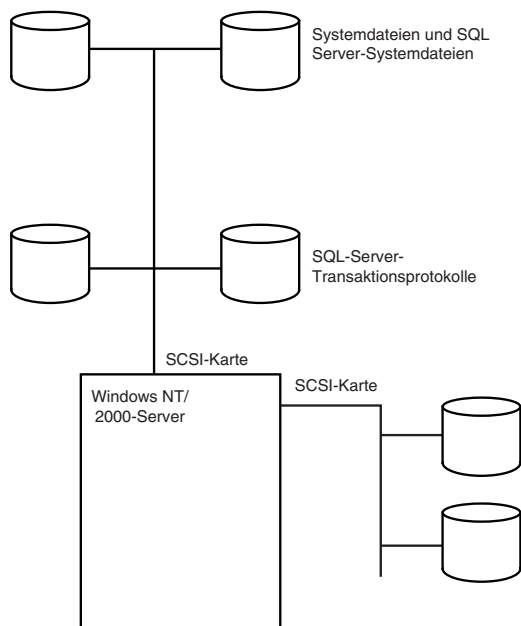


Abbildung 20.1:
Eine Serverkonfiguration mit hoher Fehlertoleranz und mittlerer Leistung

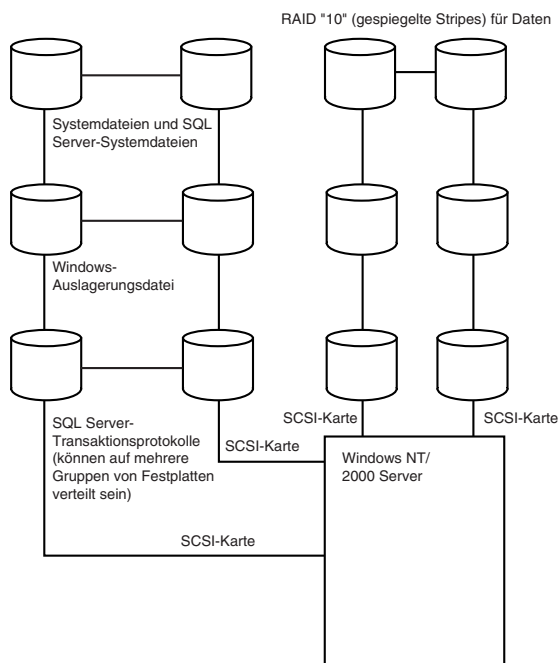


Abbildung 20.2:
Eine Serverkonfiguration mit hoher Fehlertoleranz und hoher Leistung

Beim Thema Festplatten gibt es eine ganze Menge zu berücksichtigen. Die meisten bekannten Server-Hersteller wie etwa Compaq, HP, Data General und IBM leisten hervorragende Arbeit darin, Systemlösungen zusammenzustellen, die alle diese kritischen Punkte abzudecken versuchen. Ihre Server-Systeme haben sich in der Praxis bewährt, sind außerordentlich gut gearbeitet und werden von einem Supportteam unterstützt, das sich mit der Optimierung und der Konfiguration von SQL Server auskennt und auch Systeme empfehlen kann, die Windows 2000 und SQL Server unterstützen.

20.2 Die Konfigurationsparameter von Windows 2000

Die meisten Konfigurationsoptionen für Windows 2000 werden standardmäßig schon vom Setupprogramm eingestellt. Die wichtigste Konfigurationsoption ist *Datendurchsatz für Netzwerkanwendungen maximieren* als Konfigurationswert der Datei- und Druckerfreigabe für Microsoft-Netzwerke (Ihr Server-Dienst). Um diesen Wert anzuzeigen, wählen Sie START / EINSTELLUNGEN / SYSTEMSTEUERUNG / NETZWERK- UND DFÜ-VERBINDUNGEN, klicken mit der rechten Maustaste auf Ihre Netzwerkkarte (wahrscheinlich mit LAN-VERBINDUNG bezeichnet), wählen EIGENSCHAFTEN, markieren den Eintrag *Datei- und Druckerfreigabe für Microsoft-Netzwerke* und klicken schließlich auf die Schaltfläche EIGENSCHAFTEN. Damit gelangen Sie zum Dialogfeld EIGENSCHAFTEN VON DATEI- UND DRUCKERFREIGABE FÜR MICROSOFT-NETZWERKE (siehe Abbildung 20.3). Das Setupprogramm von SQL Server stellt diesen Wert korrekt ein, und Sie sollten ihn auch nicht ändern.

Wie schon am 2. Tag über die Installation von SQL Server 2000 erwähnt, sollte SQL Server auf einem Mitgliedsserver (bzw. eigenständigem Server, was grundsätzlich das Gleiche ist) installiert werden. SQL Server arbeitet auch auf einem Domänencontroller, was aber keine empfehlenswerte Installation darstellt. Domänencontroller können zusätzliche Speicheranforderungen entwickeln, die dann mit den Speicheranforderungen von SQL Server in Konflikt geraten.

Damit kommen wir zu den zusätzlichen Diensten, die auf dem SQL Server-Computer laufen dürfen. Dass der Computer ein Domänencontroller ist, kommt vor allem daher, dass ein anderer Dienst auf dem Server ausgeführt wird: DHCP, WINS, DNS-Server oder andere Microsoft Serveranwendungen beeinträchtigen höchstwahrscheinlich die Leistung von SQL Server. Deshalb sollten diese Dienste nicht auf demselben Computer laufen, auf dem sich auch die Datenbank befindet.

Die Auslagerungsdatei von Windows 2000 ist ebenfalls ein Element, das man besser auf einer Festplatte unterbringt, die SQL Server nicht verwendet.

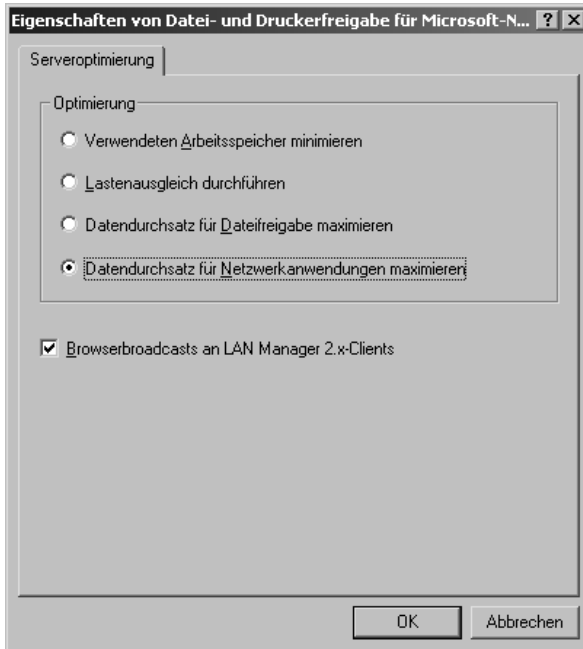


Abbildung 20.3:
Das Dialogfeld Eigenschaften von
Datei- und Druckerfreigabe für Mic-
rosoft-Netzwerke

Der Abschnitt »Festplatten« weiter oben hat RAID als Möglichkeit zum Schutz der Daten erwähnt. Eine hardwareseitig implementierte Form der Spiegelung bzw. RAID 5 anstelle eines Einsatzes der RAID-Fähigkeiten von Windows 2000 Server ist zu empfehlen, denn die meisten Hardwarelösungen sind leistungsfähiger als die eingebauten RAID-Fähigkeiten des Windows 2000-Servers.

20.3 Leistungskindikatoren des Windows 2000-Systemmonitors

Wir versuchen an dieser Stelle nicht, den Systemmonitor oder die Leistungskindikatoren von Windows 2000 erschöpfend zu behandeln. Hier sei auf die Dokumentation zu Windows 2000 und die einschlägige Literatur verwiesen. Dennoch sollten Sie einige Leistungskindikatoren kennen, da sie für SQL Server wichtig sind.

Alle hier beschriebenen Objekte und Leistungskindikatoren können Sie mit dem Systemmonitor von Windows 2000 überwachen (erreichbar über START / PROGRAMME / VERWALTUNG / SYSTEMMONITOR). Die folgenden vier Abschnitte erläutern die Leistungskindikatoren, die sich auf die oben behandelten Ressourcentypen beziehen.

Speicher

Bei einem in erster Linie auf SQL Server ausgerichteten Computer ist vor allem die Nutzung des Hauptspeichers zu verfolgen. Im Allgemeinen sollte man Seitenfehler auf einem Minimum halten. Seitenfehler treten auf, wenn Daten, die im Hauptspeicher erwartet werden, nicht vorhanden sind und von der Platte gelesen werden müssen. Mit dem Leistungsindikator *Seitenfehler/s* des Datenobjekts *Speicher* lassen sich Seitenfehler überwachen. Selbst wenn der Wert dieses Zählers nicht 0 sein wird, darf er auch nicht zu hoch liegen. Was aber ist ein hoher Wert? Salomonisch gesprochen ein Wert, der für Ihren Server nicht normal ist. Im Allgemeinen ist ein Wert nahe 0 anzustreben.

Im Datenobjekt *Speicher* sollte man auch den Leistungsindikator *Verfügbare Bytes* überwachen, dessen Wert nicht unter 4 MB fallen sollte. Andernfalls versucht Windows 2000, Arbeitsspeicher von Anwendungen (einschließlich SQL Server) abzugeben, um eine bestimmte Speichergröße verfügbar zu machen. Fällt dieser Wert häufig unter 4 MB, dann haben Sie SQL Server unter Umständen zuviel Speicher zugewiesen. Allerdings beugt SQL Server automatisch der Auslagerung vor, sofern Sie nicht die automatische Speicher-verwaltung abschalten.

Prozessor

Im Datenobjekt *Prozessor* überwacht man vor allem den Leistungsindikator *Prozessorzeit (%)*, der darüber Auskunft gibt, wie beschäftigt die CPU ist. Beachten Sie, dass es für jede CPU im System eine Instanznummer gibt.

Netzwerk

Zur Überwachung des Netzwerks können Sie entweder den Systemmonitor oder aber ein Netzwerküberwachungsprogramm wie etwa den Microsoft-Netzwerkmonitor verwenden. Im Netzwerk gibt es so viele Objekte und Datenquellen, die man überwachen kann, dass sie hier unmöglich alle genannt werden können. Erwähnt sei jedoch, dass Sie die Objekte auf der Grundlage der einzelnen Netzwerkprotokolle wie zum Beispiel NetBEUI, TCP, UDP oder IP verfolgen können. Es gibt außerdem einige für SQL Server spezifische Leistungsindikatoren, die Sie später in dieser Lektion kennen lernen.

Festplatte

Um die Festplattennutzung zu überwachen, müssen Sie gegebenenfalls die entsprechenden Leistungsindikatoren aktivieren. Dazu rufen Sie den Befehl `diskperf -y` von einer

Windows 2000-Eingabeaufforderung auf und starten dann den Server neu. In der Standardkonfiguration von Windows 2000 ist andernfalls das Datenobjekt *Logischer Datenträger* nicht im Systemmonitor aufgeführt.

Sie können aber auch das Objekt *Physikalischer Datenträger* wählen. Der Unterschied zwischen diesem und dem Objekt *Logischer Datenträger* besteht darin, dass die physikalischen Leistungsindikatoren die physischen Festplatten beobachten, während die logischen Indikatoren Informationen über die Laufwerke mit Laufwerksbuchstaben liefern. Wenn Sie also zwei logische Datenträger auf einer physischen Festplatte haben, können Sie mit dem Objekt *Physikalischer Datenträger* anzeigen, wie hoch die Aktivität auf der physischen Festplatte ist, während Sie mit dem Objekt *Logischer Datenträger* die Aktivitäten auf den einzelnen logischen Datenträgern verfolgen können.

Zeigt sich für eine bestimmte Festplatte ein zu hoher Aktivitätswert, können Sie die Daten mithilfe von Dateigruppen auf andere physische Festplatten im Computer verschieben bzw. verteilen.

20.4 SQL Server konfigurieren

Die Konfiguration von SQL Server 2000 ist eine relativ unkomplizierte Angelegenheit. Zwar gibt es immer noch manche Option, die sich abstimmen lässt, in der Regel brauchen Sie das aber gar nicht. Zumindest aber sollten Sie sich darüber informieren, welche Optionen verfügbar sind, so dass Sie bei Bedarf an der richtigen Stelle eingreifen können.



Dieser Abschnitt befasst sich ausschließlich mit den Windows 2000-basierten Versionen von SQL Server. Für die Personal Edition von SQL Server unter Windows 9x ist es bei den meisten Implementierungen reine Zeitverschwendung, sich mit der Feinabstimmung zu beschäftigen. Die automatische Optimierung übernimmt hier für Sie die wichtigsten verfügbaren Konfigurationseinstellungen.

Zur Untersuchung dieser Konfigurationsoptionen sollten Sie zunächst die Konfigurationsoption *Show Advanced Options* einschalten, damit Sie alle entsprechenden Parameter auch zu sehen bekommen. Dazu rufen Sie die gespeicherte Systemprozedur `sp_configure` wie folgt auf:



```
Exec sp_configure 'Show Advanced Options',1
GO
RECONFIGURE with Override
GO
Exec sp_configure
```

Das Ergebnis sieht etwa folgendermaßen aus:



name	mini	maximum	config_value	run_value
-				
affinity mask	0	2147483647	0	0
allow updates	0	1	0	0
awe enabled	0	1	0	0
c2 audit mode	0	1	0	0
cost threshold for parallelism	0	32767	5	5
cursor threshold	-1	2147483647	-1	-1
default full-text language	0	2147483647	1031	1031
default language	0	9999	1	1
fill factor (%)	0	100	0	0
index create memory (KB)	704	2147483647	0	0
lightweight pooling	0	1	0	0
locks	5000	2147483647	0	0
max degree of parallelism	0	32	0	0
max server memory (MB)	4	2147483647	2147483647	
2147483647				
max text repl size (B)	0	2147483647	65536	65536
max worker threads	32	32767	255	255
media retention	0	365	0	0
min memory per query (KB)	512	2147483647	1024	1024
min server memory (MB)	0	2147483647	0	0
nested triggers	0	1	1	1
network packet size (B)	512	65536	4096	4096
open objects	0	2147483647	0	0
priority boost	0	1	0	0
query governor cost limit	0	2147483647	0	0
query wait (s)	-1	2147483647	-1	-1
recovery interval (min)	0	32767	0	0
remote access	0	1	1	1
remote login timeout (s)	0	2147483647	20	20
remote proc trans	0	1	0	0
remote query timeout (s)	0	2147483647	600	600
scan for startup procs	0	1	1	1
set working set size	0	1	0	0
show advanced options	0	1	1	1
two digit year cutoff	1753	9999	2049	2049
user connections	0	32767	0	0
user options	0	32767	0	0

Trotz der ungeheuren Zunahme der Funktionalität verfügt SQL Server 2000 genau wie SQL Server 7.0 über weniger Optionen als die früheren Versionen. Wie eingangs erwähnt, brauchen Sie die große Mehrzahl dieser Optionen nicht selbst zu verwalten. Die meisten

von ihnen sind unter `config_value` und `run_value` auf 0 eingestellt – d.h. SQL Server übernimmt ihre Einstellung automatisch. Solange Sie keinen wirklich dringenden Grund für eine Änderung dieser automatisch verwalteten Optionen haben, sollten Sie sich auch nicht mit ihnen befassen. Beachten Sie, dass der Wert `run_value` momentan zugewiesen ist. Der Wert `config_value` kann sich davon unterscheiden, falls Sie eine Änderung angefordert, bisher aber SQL Server weder neu gestartet noch den Befehl `RECONFIGURE` ausgeführt haben.

Aus der Sicht der Systemleistung und der Optimierung gibt es nur wenige Konfigurationsparameter, die Sie vielleicht ändern würden. Um es noch einmal zu sagen: Außer für besonders große Implementierungen von SQL Server gibt es keine Notwendigkeit, diese Parameter zu verändern. Falls Sie jedoch einen SQL Server mit vier Prozessoren zu verwalten haben, kann es für Sie interessant sein, diese Konfigurationsoptionen zu berücksichtigen. Um Ihnen die Analyse dieser Optionen zu erleichtern, sind sie in dieser Lektion in vier Kategorien zusammengefasst: Prozessor, Speicher, Ein-/Ausgabe und Abfrage/Index.

Prozessoroptionen

Die erste Gruppe von Optionen betrifft die Nutzung des Prozessors durch SQL Server. Sie haben die Möglichkeit, die relative Priorität des SQL Server-Prozesses, die CPUs Ihres Computers, die der SQL Server verwendet, und die Anzahl der Betriebssystemthreads, die SQL Server auf Ihrem Computer nutzen kann, zu konfigurieren. Außerdem können Sie die maximale Anzahl der CPUs einstellen, die während der parallelen Bearbeitung von Abfragen benutzt werden sollen. Jede dieser Optionen ist in einem bestimmten logischen Zusammenhang zu betrachten, den Sie bei Änderungen an der Konfiguration Ihres SQL Servers berücksichtigen müssen. Anders ausgedrückt: Es ist vermutlich sinnvoller, vor einer Änderung des Kostenschwellwertes für Parallelität (wird zum Schluss untersucht) zuerst einmal die Option zur Höherstufung der Priorität zu ändern (wird zuerst untersucht).

Höherstufung der Priorität

Die Option *Höherstufung der Priorität* (priority boost) legt fest, ob die Threads von SQL Server auf Ihrem Computer mit normaler Priorität (Prioritätsebene 8 unter Windows 2000) oder hoher Priorität (Ebene 13 unter Windows 2000) laufen sollen. Wenn der Wert auf 0 steht, laufen sie mit normaler Priorität; steht der Wert dagegen auf 1, dann laufen sie mit hoher Priorität, wobei diese Einstellungen allerdings für einen Computer mit einem Prozessor gelten. Für Computer mit mehreren Prozessoren beträgt die normale Priorität für SQL Server 15 und die hohe Priorität 24. Nach der Änderung dieser Option müssen Sie den SQL Server-Dienst neu starten, damit der neue Wert wirksam wird.

Beobachten Sie Ihren Computer aufmerksam, da Sie durch die Änderung der Priorität gegebenenfalls andere Prozesse auf Ihrem SQL Server-Computer beeinträchtigen können.

Auf einem dedizierten SQL Server-Computer werden Sie jedoch nach Einstellung dieser Option eine Leistungsverbesserung erleben.

Affinitätsmaske

Die Option *Affinitätsmaske* (affinity mask) stellt eine Bitmap mit einer Repräsentation der CPUs dar, die SQL Server nutzt (auf einem Multiprozessorsystem). Um beispielsweise die Prozessoren 0, 1 und 3 einzustellen, müssen Sie das Bitmap-Muster auf 00001011 setzen (den Bitpositionen sind von rechts nach links die Prozessoren mit den Nummern 0, 1, 2 usw. zugeordnet). Zur Vereinfachung können Sie auch eine Dezimalzahl eingeben; in diesem Fall 11: $(8*1) + (4*0) + (2*1) + (1*1)$ mit Beginn ab dem ersten gesetzten Bit. SQL Server nutzt standardmäßig alle CPUs im System – diese Option ändern Sie nur dann, wenn Sie einen oder mehrere Prozessoren für Windows 2000 oder andere auf dem Server laufende Prozesse reservieren möchten. Allerdings ist es sinnvoll, SQL Server für die meiste Zeit alle Prozessoren im System zu überlassen.

Optionen für parallele Abfragen

In SQL Server 2000 gibt es zwei Optionen für parallele Abfragen. Erstens kann SQL Server Teile einer Abfrage parallel ausführen und dann die Verarbeitung zu einem einzigen Ergebnis vereinen. Diese Art der Parallelverarbeitung findet ständig in SQL Server statt.

Bei der zweiten Form einer parallel verarbeiteten Abfrage (die sich konfigurieren lässt) kann SQL Server einer einzelnen Abfrage mehrere Prozessoren zuweisen. Diese Art einer parallelen Abfrage steht nur auf Multiprozessorsystemen zur Verfügung und wird typischerweise dann ausgeführt, wenn Sie mehr Prozessoren als Benutzer haben.

Nehmen Sie dazu als Beispiel einmal an, dass es Samstag 21:00 Uhr ist. Im System befinden sich keine Benutzer mehr und Sie starten jetzt einen Auftrag, der einen riesigen Bericht generieren soll. In einem System mit vier Prozessoren hat SQL Server nun Gelegenheit, mit allen vier Prozessoren diesen Bericht so schnell wie möglich abzuarbeiten, was zu anderen Zeitpunkten nicht in dieser Form möglich wäre. Nachdem aber eine derartige Parallelverarbeitung gestartet ist, läuft sie mit allen konfigurierten Prozessoren bis zum Ende. Wenn sich nun ein Benutzer anmeldet, und eine andere Abfrage ausführen möchte, leidet die Leistung bei der Abfrage des neuen Benutzers aufgrund der bereits laufenden Parallelabfrage. Deshalb sollten Sie bei mehreren Prozessoren die Parallelität sehr sorgfältig konfigurieren.

Max Degree of Parallelism

Diese Option legt die Anzahl der für parallele Abfragen zur Verfügung stehenden Threads fest (zwischen 1 und 32). Der Wert 0 weist darauf hin, dass sämtliche konfigurierte Prozessoren (entsprechend der Option *affinity mask*) verwendet werden können, wobei auf einen Thread ein Prozessor kommt. Ein Wert von 1 schaltet die parallele Abfrage völlig ab. Alle

anderen Werte dagegen legen die maximale Anzahl der für die parallele Abfrage einzusetzenden CPUs fest. Noch einmal sei darauf hingewiesen, dass parallele Abfragen nur dann stattfinden, wenn Ihr SQL Server wenig belastet ist.

Auf Systemen mit einem Prozessor wird diese Option ignoriert. Änderungen der Option treten unverzüglich und ohne einen Neustart von SQL Server in Kraft.

Kostenschwellwert für Parallelität

Die Option *Kostenschwellwert für Parallelität* (cost threshold for parallelism) gibt die Anzahl der Sekunden an, die es dauert, bis für eine Abfrage, die nicht als parallele Abfrage ausgeführt wird, ein paralleler Ausführungsplan herangezogen wird. Der Standardwert beträgt 5 Sekunden, wobei die Option auf bis zu 32.767 Sekunden eingestellt werden kann. Anders ausgedrückt: Falls eine Abfrage ohne Parallelität mindestens 5 Sekunden benötigt, wird SQL Server die Möglichkeit zur parallelen Ausführung prüfen und gegebenenfalls verwenden.

Diese Option gilt nur für Multiprozessorsysteme, wenn die Optionen *affinity mask* und *max degree of parallelism* so eingestellt sind, dass parallele Abfragen erlaubt sind. Änderungen dieser Option treten unverzüglich ohne Neustart von SQL Server in Kraft.

Der SQL Server Enterprise Manager

Alle Konfigurationskategorien sind als Teil der Konfiguration Ihres SQL Servers auf einer entsprechenden Eigenschaftenseite im SQL Server Enterprise Manager untergebracht. Zum Zugriff auf diese Prozesseigenschaften Ihres Servers klicken Sie mit der rechten Maustaste auf den Server, wählen EIGENSCHAFTEN aus dem Kontextmenü und klicken dann auf die Registerkarte PROZESSOR (siehe Abbildung 20.4). Die Optionen in diesem Dialogfeld dürften nach den vorangegangenen Erläuterungen selbsterklärend sein.

Speicherkonfigurationsoptionen

Es gibt zwei Speicherkonfigurationsoptionen, die Sie näher untersuchen werden, nämlich die Optionen für den minimalen und maximalen Speicherverbrauch des Servers. Standardmäßig stellt SQL Server den minimalen Speicher auf 0 und den maximalen auf 2.147.483.647 Byte ein (2 GB). Diese Konfigurationseinstellung bedeutet, dass SQL Server die automatische Speicherverwaltung verwendet – falls er mehr Speicher benötigt, wird er sich entsprechend dazu auch mehr Speicher holen. Um jedoch Auslagerungen zu vermeiden, achtet SQL Server standardmäßig darauf, den Speicherverbrauch unterhalb des Minimums von 5 MB physikalischem Speicher zu halten. Auf einem System mit 128 MB bedeutet dies beispielsweise, dass SQL Server nicht mehr als 123 MB beansprucht (einschließlich Windows 2000 und anderer laufender Prozesse). Allerdings haben Sie auch die Möglichkeit, diese Option manuell einzustellen.

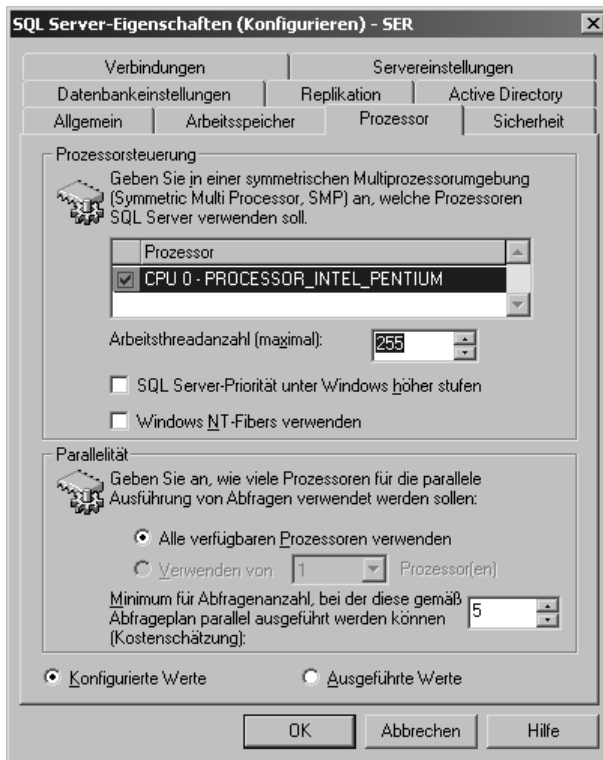


Abbildung 20.4:
Die Konfigurationsoptionen auf der
Registerkarte Prozessor

Normalerweise lassen Sie diese Einstellungen unverändert, denn SQL Server 2000 leistet eine hervorragende Arbeit in der dynamischen Reservierung und Freigabe des benötigten Speichers. Falls Sie jedoch andere speicherintensive Programme wie etwa Microsoft Exchange Server betreiben, müssen Sie gegebenenfalls die Menge des dem SQL Server zugewiesenen Hauptspeichers begrenzen. Ein dedizierter SQL Server ist also immer zu empfehlen.

Min Server Memory

Indem Sie *min server memory* einstellen, teilen Sie SQL Server mit, dass Sie niemals weniger als *n* MB Speicher verwenden. Beispielsweise bedeutet eine Konfiguration von *min server memory* von 50 (MB), dass SQL Server bei jedem Start 50 MB Hauptspeicher anfordern wird.

Wenn Sie *max server memory* ebenfalls auf 50 MB einstellen, setzen Sie damit einen »festen« Speicherbetrag fest, den SQL Server dann auch immer verwendet.

Max Server Memory

Durch die Einstellung von *max server memory* wird der Maximalbetrag des Speichers begrenzt, den SQL Server verwenden kann. Dieser Wert wird, wie schon erwähnt, in MB ausgedrückt.

Min Memory Per Query

Diese Option legt den Minimalbetrag an reserviertem Speicher fest, den jeder Benutzer auf Ihrem SQL Server für jede im Server ablaufende Abfrage erhalten kann. Der Standardwert beträgt 1.024 KB bzw. 1 MB, was bei den meisten SQL Server-Computern ein völlig ausreichender Wert ist. Betreiben Sie SQL Server 2000 als entscheidungsunterstützendes oder Datawarehouse-System, müssen Sie diesen Wert gegebenenfalls heraufsetzen. Die Leistung von Abfragen, die umfangreiche Sortiervorgänge oder Verknüpfungen durchführen, können Sie steigern, wenn Sie den Speicher zuweisen, bevor ihn diese Abfragen tatsächlich benötigen (statt erst zur Laufzeit freien Speicher suchen zu müssen).

SQL Server Enterprise Manager

Wenn Sie mit der rechten Maustaste auf Ihren Server klicken, Eigenschaften aus dem Kontextmenü wählen und auf die Registerkarte Arbeitsspeicher gehen, erscheint das Dialogfeld wie in Abbildung 20.5 gezeigt. Hier haben Sie die Möglichkeit, den Arbeitsspeicher von SQL Server selbst konfigurieren zu lassen, eine feste Speichergröße einzustellen oder die minimal bzw. maximal zu reservierenden Speichergrößen abzustimmen. Außerdem können Sie den Abfragearbeitsspeicher konfigurieren.



Mit der Option *Physischen Arbeitsspeicher für SQL Server reservieren* stellen Sie den Konfigurationsparameter *set working set size* ein. Diese Option bewirkt scheinbar keine Leistungssteigerung (schadet aber auch nicht), so dass sie hier nicht weiter untersucht wird.

Ein-/Ausgabeeoptionen

In SQL Server 2000 gibt es keinen Ein-/Ausgabeparameter, den Sie konfigurieren können. In SQL Server 7.0 und früheren Versionen konnte man den Parameter *max async I/O* einstellen, der aber in SQL Server 2000 nicht mehr vorhanden ist. SQL Server startet jetzt immer die benötigten Threads, um E/A-Operationen zum Betriebssystem auszulösen. Daher können Sie eine Leistungssteigerung im Bereich Festplatten und Ein-/Ausgabeeoperationen vor allem dadurch erreichen, dass Sie die Daten über mehrere Festplatten und mehrere Controller verteilen, wie es bereits weiter oben beschrieben wurde.

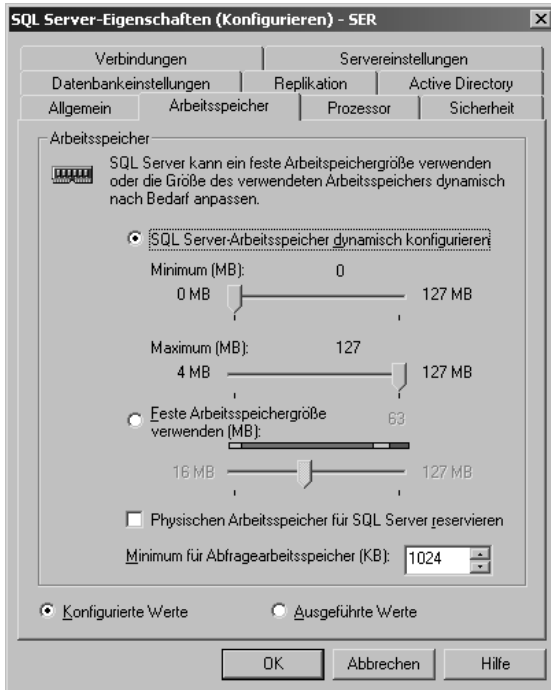


Abbildung 20.5:
Die Konfigurationsoptionen auf der
Registerkarte Arbeitsspeicher

Abfrage/Index-Optionen

Für die Optimierung oder Begrenzung von Abfragen und die Optimierung von Indizes stehen zwei Optionen zur Verfügung, zu denen der Füllfaktor gehört, den Sie bereits am 13. Tag kennen gelernt haben. Die andere Option heißt *Kostenlimitoption der Abfragekontrolle* (query governor cost limit).

Der Standardwert für die Abfragekontrolle ist 0, d.h. sämtliche Abfragen dürfen ohne Zeitbeschränkung ausgeführt werden. Wenn Sie jedoch diese Option auf einen Wert ungleich 0 setzen, verhindert SQL Server die Ausführung von Abfragen, deren geschätzte Ausführungszeit über dieser angegebenen Anzahl von Sekunden liegt. Seien Sie bei der Anwendung dieser Option vorsichtig, weil Sie mit ihr leicht andere Benutzer an der Ausführung von Abfragen hindern können, die diese für ihre alltägliche Arbeit benötigen. Andererseits können Sie durch eine sinnvolle Handhabung der Option natürlich auch übermäßig lang laufende Abfragen unterdrücken. Diese Option steht nicht über die Benutzeroberfläche des SQL Server Enterprise Managers zur Verfügung.

20.5 Überwachen der Leistung von SQL Server

Für die Überwachung der Leistung von SQL Server stehen mehrere Tools und Transact-SQL-Befehle zur Verfügung, zu denen die Leistungsindikatoren des Systemmonitors (die Sie in der gestrigen Lektion über Warnungen bereits kurz kennen gelernt haben), das Fenster AKTUELLE AKTIVITÄT des SQL Server Enterprise Managers, der SQL Server Profiler sowie bestimmte gespeicherte Systemprozeduren gehören. Die folgenden Abschnitte gehen näher auf diese Möglichkeiten ein.

SQL Server-Leistungsindikatoren im Systemmonitor

Bei Ihrer Tour durch die Überwachungstools von SQL Server halten Sie noch einmal beim Windows 2000-Systemmonitor an. Dieses Mal untersuchen wir die Leistungsindikatoren, die SQL Server freilegt. Analog zu den Konfigurationsoptionen versucht diese Lektion nicht, alle Indikatoren (d.h. mehr als 100) zu behandeln, sondern konzentriert sich auf die wichtigsten, die Sie unbedingt kennen sollten.

Speicher

Sehen Sie sich beispielsweise im Objekt *SQLServer:Buffer Manager* den Leistungsindikator *Buffer cache hit ratio* an, um zu bestimmen, wie effizient der Speicher auf Ihrem Server verwendet wird. Dieser Wert gibt wieder, wie häufig Daten von SQL Server im Speicher vorgefunden worden sind, und sollte höher als 90% und vorzugsweise nahe an 100% liegen. Starten Sie nun den Systemmonitor (in der Programmgruppe VERWALTUNG) und klicken Sie auf das Pluszeichen (+) in der Symbolleiste. Blättern Sie bis zum Objekt *SQLServer:Buffer Manager* (oder *MSSQL\$Instanzname:Buffer Manager* bei einer benannten Instanz). Hier finden Sie den Leistungsindikator *Buffer Cache Hit Ratio*, um den es jetzt geht.

Gegebenenfalls ist es sinnvoll, im Objekt *SQLServer:Buffer Manager* den Leistungsindikator *Database Pages* zu überwachen, um festzustellen, wie viel Speicher verwendet wird, um die Daten und Indizes aufzunehmen. Diese Informationen werden als Anzahl Seiten zurückgeliefert, so dass Sie sie mit 8 multiplizieren müssen, um die entsprechende Anzahl Kilobytes des verwendeten Speichers errechnen zu können.

Prozessor

Falls Sie die CPU-Zeit überwachen möchten, die SQL Server verbraucht, können Sie im Objekt *Prozess* den Indikator *Prozessorzeit (%)* für die Instanz *sqlservr* überwachen. Wenn Sie mehrere Instanzen von SQL Server ausführen, erscheinen diese alle hier (und sehen

leider fast identisch aus). Indem Sie die Instanz *sqlagent* überwachen, können Sie feststellen, wie viel Prozessorzeit der SQL Server-Agent-Dienst benötigt.

Netzwerk

Objekte von SQL Server für die Überwachung des Netzwerks gibt es nicht – Statistiken erhalten Sie über die Berichte der jeweiligen Netzwerkprotokolle.



Wenn Sie TCP/IP verwenden, müssen Sie den Dienst SNMP installieren, damit unter Windows NT 4.0 die TCP/IP-bezogenen Leistungsindikatoren erscheinen.

Festplatte

Für das *SQLServer*-Objekt gibt es eine ganze Reihe wichtiger E/A-Leistungsindikatoren. Beispielsweise überwachen die Indikatoren *Page Read/s* (Seitenlesevorgänge/Sekunde) und *Page Writes/s* (Seitenschreibvorgänge/Sekunde) im Objekt *SQLServer:Buffer Manager* die Leseaktivität (von Festplatte) bzw. die Schreibaktivitäten, die auf dem Server auftreten. Ein weiterer Indikator ist *Log Flushes/s* (Protokollleerungen/Sekunde) des Objekts *SQLServer:Databases*. Dieser Wert zeigt an, wie viele Protokoll Datensätze jede Sekunde auf die Platte geschrieben werden, und vermittelt Ihnen eine Vorstellung davon, wie viel Arbeit gerade erledigt wird. Mit diesem Leistungsindikator eng verbunden ist der Indikator *Transactions/s* (Transaktionen/Sekunde) im Objekt *SQLServer:Databases*, der anzeigt, wie viele aktive Transaktionen in einer Datenbank zu einem bestimmten Zeitpunkt vorhanden sind. Beachten Sie, dass diese letzten beiden Optionen datenbankspezifisch sind und deshalb zur Überwachung der Aktivitäten in einer bestimmten Datenbank keine Instanz ausgewählt werden muss.

Gespeicherte Systemprozeduren für die Überwachung der Leistung

Die Leistung von SQL Server lässt sich auch mit einigen gespeicherten Systemprozeduren überwachen. Dazu zählen vor allem *sp_who* und *sp_monitor*.



```
sp_who [[@login_name =] 'login']
```

In dieser Syntax bezeichnet *login* den Benutzernamen, den Sie überwachen möchten. Falls Sie den Benutzernamen nicht angeben, erhalten Sie Informationen zu allen Benutzernamen. Wenn Sie das Schlüsselwort *ACTIVE* anstelle eines Benutzernamens angeben, werden Informationen nur für Benutzer erstellt, die derzeit Abfragen ausführen. Systemverbindungen werden immer zurückgegeben, selbst wenn Sie *ACTIVE* spezifizieren.

Die Ausgabe von *sp_who* teilt Ihnen mit, wer verbunden ist, welche Computer die Benutzer einsetzen, mit welchen Datenbanken sie arbeiten und in welchem Befehlsstatus sie sich befinden. Der Aufruf

```
exec sp_who
```

liefert zum Beispiel folgendes Ergebnis:

spid	ecid	status	loginame	hostname	blk	dbname	cmd

1	0	background	sa		0	NULL	LAZY WRITER
2	0	sleeping	sa		0	NULL	LOG WRITER
3	0	background	sa		0	master	SIGNAL HANDLE
4	0	background	sa		0	NULL	LOCK MONITOR
5	0	background	sa		0	master	TASK MANAGER
6	0	sleeping	sa		0	NULL	CHECKPOINT SL
7	0	background	sa		0	master	TASK MANAGER
8	0	background	sa		0	master	TASK MANAGER
9	0	background	sa		0	master	TASK MANAGER
10	0	background	sa		0	master	TASK MANAGER
51	0	runnable	RHOME\rwaymi	RHOME	0	master	SELECT
52	0	sleeping	RHOME\rwaymi	RHOME	0	master	AWAITING COMM
53	0	sleeping	RHOME\rwaymi	RHOME	0	msdb	AWAITING COMM
54	0	sleeping	RHOME\SQLService	RHOME	0	msdb	AWAITING COMM
55	0	sleeping	RHOME\SQLService	RHOME	0	msdb	AWAITING COMM

(15 row(s) affected)

Die andere gespeicherte Systemprozedur, *sp_monitor*, liefert den Umfang der Systemaktivität seit dem letzten Aufruf dieser gespeicherten Systemprozedur zurück:

```
sp_monitor
```

Die gespeicherte Systemprozedur *sp_monitor* übernimmt keine Parameter.

Die folgende Ausgabe zeigt ein Beispiel, bei dem die Prozedur *sp_monitor* ausgeführt, dann verschiedene Aktivitäten vorgenommen wurden und schließlich erneut *sp_monitor* aufgerufen wurde:

last_run	current_run	seconds

2001-01-16 18:10:57.890	2001-01-16 18:11:59.327	4639578



cpu_busy	io_busy	idle	
-----	-----	-----	
25(22)-0%	0(0)-0%	69532(69467)-1%	
packets_received	packets_sent	packet_errors	
-----	-----	-----	
2390(1687)	2383(1941)	0(0)	
total_read	total_write	total_errors	connections
-----	-----	-----	-----
2021(1902)	579(-3410)	0(0)	38(27)

Wie die Ausgabe zeigt, erhalten Sie eine Menge nützlicher Informationen, beispielsweise den Auslastungsgrad des Prozessors oder die Aktivität der Festplatten. Damit Sie diese Informationen richtig nutzen können, müssen Sie die Prozedur jeweils vor und nach einer zu messenden Aktivität aufrufen.

20.6 Das Fenster Aktuelle Aktivität im SQL Server Enterprise Manager

Eine einfachere Möglichkeit zur Anzeige der Leistungsdaten von SQL Server besteht über das Fenster AKTUELLE AKTIVITÄT im SQL Server Enterprise Manager. Sie erreichen dieses Fenster, indem Sie den Ordner *Verwaltung* für Ihren SQL Server erweitern und anschließend den Knoten *Aktuelle Aktivität* erweitern. Markieren Sie in der Konsolenstruktur den Eintrag *Prozessinfo*. Das Fenster sollte nun etwa Abbildung 20.6 entsprechen. (Wenn Sie Symbole anstelle der einzelnen Einträge sehen sollten, wählen Sie *DETAILS* aus dem Menü *ANSICHT*.)

Dieses Fenster zeigt jede Verbindung in SQL Server an. Wenn Sie einen bestimmten Prozess hervorheben und dann mit der rechten Maustaste darauf klicken, können Sie diesen Prozess abbrechen (sofern Sie ein Mitglied der festen Serverrollen *sysadmin* oder *processadmin* sind). Sie können auch dem Benutzer eine Nachricht senden (über das Betriebssystem, nicht über das Abfragewerkzeug des Benutzers) oder über die Auswahl von *EIGENSCHAFTEN* aus dem Kontextmenü den letzten Befehl ermitteln, den der Benutzer ausgeführt hat (siehe dazu Abbildung 20.7).

Über die Auswahl von *Sperren / Prozess-ID* oder *Sperren / Objekte* in der Konsolenstruktur können Sie auch Details zu Sperren der jeweiligen Gruppen untersuchen.

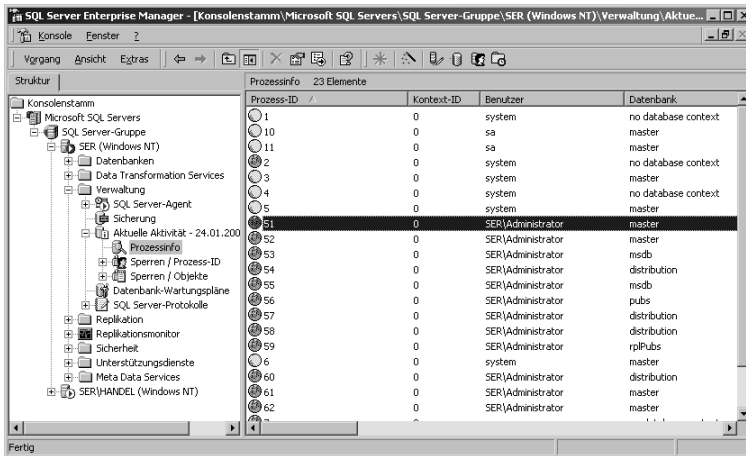


Abbildung 20.6:
Das Fenster Aktuelle Aktivität



Abbildung 20.7:
Im Dialogfeld Prozessdetails für einen Eintrag im Fenster Aktuelle Aktivität können Sie einen Prozess abbrechen oder eine Nachricht senden.

SQL Server Profiler

Mit dem SQL Server Profiler sind Sie in der Lage, sämtliche Aktivitäten auf Ihrem SQL Server zu überwachen. Es gibt eine Vielzahl von Ereignissen, die Sie mit dem SQL Server Profiler registrieren und aufzeichnen können. Der wohl nützlichste Teil des SQL Server Profilers ist jedoch dessen Fähigkeit, Transact-SQL-Befehle registrieren zu können, die auf Ihrem Computer ausgeführt werden. Hiermit erhalten Sie die Gelegenheit, Ablaufverfolgungen in Gang zu setzen oder auch einfach nur festzustellen, welche Tätigkeiten auf Ihrem Computer gerade ausgeführt werden. Eine derartige Ablaufverfolgung können Sie darüber hinaus auch als eine Eingabe für den Indexoptimierungs-Assistenten von SQL Server einsetzen, der Ihnen dabei hilft, den richtigen Index für Abfragen auszuwählen. Es ist praktisch, eine Ablaufverfolgung für die Dauer eines typischen Arbeitstages auszuführen und dann die Ablaufverfolgung dem Indexoptimierungs-Assistenten zu übergeben. Dieser analysiert die mit dem Profiler gewonnenen Eingabedaten und liefert als Ergebnis Empfehlungen für einen optimalen Satz von Indizes.

Einrichten und Starten einer Ablaufverfolgung

Der erste Schritt in der Verwendung dieses Werkzeugs besteht in der Einrichtung einer Ablaufverfolgung. Starten Sie den Profiler aus der Programmgruppe MICROSOFT SQL SERVER. Daraufhin erscheint das in Abbildung 20.8 dargestellte Fenster.

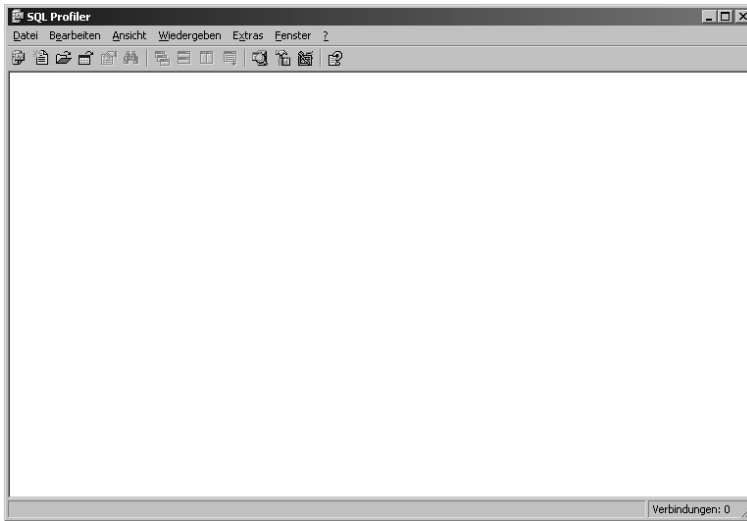


Abbildung 20.8:
Mit dem SQL Profiler richten Sie eine Ablaufverfolgung ein.

Um eine vorkonfigurierte Ablaufverfolgung zu starten, wählen Sie DATEI / NEU / ABLAUFVERFOLGUNG. Es erscheint die Aufforderung, die Verbindung zu einer Instanz von SQL Server herzustellen. Nachdem Sie verbunden sind, öffnet sich das Dialogfeld ABLAUFVERFOLGUNGSEIGENSCHAFTEN (siehe Abbildung 20.9). Zum Lieferumfang von SQL Server gehören mehrere vorkonfigurierte Ablaufverfolgungen in Form von Ablaufverfolgungsvorlagen. Standardmäßig lädt SQL Profiler die Vorlage *SQLProfilerStandard*.

Jede Vorlage definiert einen Standardsatz von Ereignissen (Gruppen von Informationen, die Sie anzeigen möchten) und Datenspalten (die einzelnen Teile der Informationen über jedes Ereignis). Übernehmen Sie für diese Ablaufverfolgung die Standardwerte und klicken Sie auf die Schaltfläche AUSFÜHREN, um eine Ablaufverfolgung aufzuzeichnen. Starten Sie nun den SQL Server Query Analyzer (beachten Sie das entsprechende Symbol für den Query Analyzer in der Symbolleiste) und geben Sie die folgende Abfrage ein:



```
Use pubs
Select * from authors
```

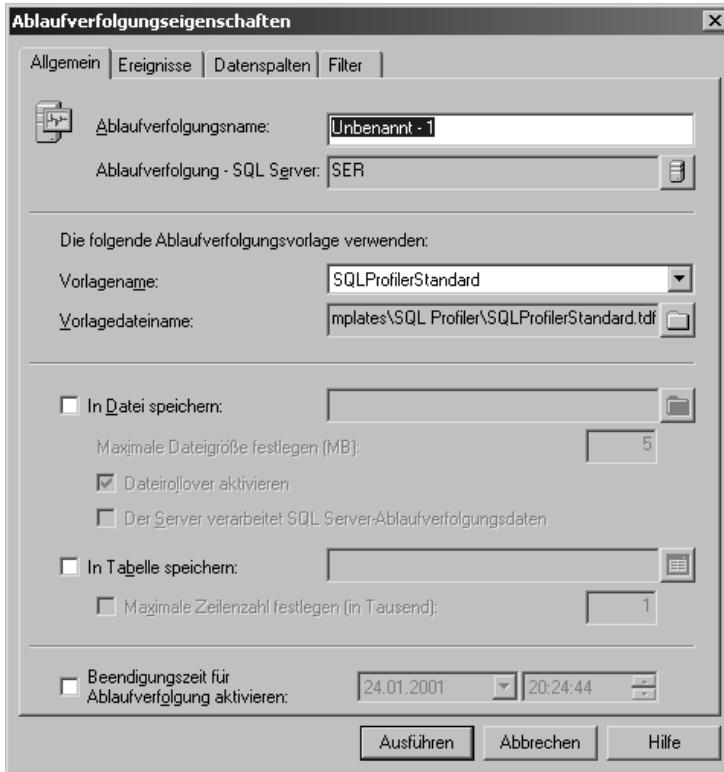


Abbildung 20.9:
Das Dialogfeld
Ablaufverfolgungseigenschaften von SQL
Profiler

Wechseln Sie jetzt wieder zum Profiler und schauen Sie sich die Transact-SQL-Anweisungen (neben anderen Informationen) an, die dieser registriert hat. Sie können erkennen, dass nicht nur die von Ihnen ausgeführte Abfrage (der Befehl USE mit der folgenden SELECT-Anweisung), sondern auch noch weitere Befehle registriert worden sind. Dies sind Befehle, die bei jedem Start des SQL Server Query Analyzers ausgeführt werden und mit denen die Umgebung konfiguriert wird, in der Sie mit der Ausführung von Abfragen beginnen. An dieser Stelle werden alle Befehle, auch solche, von deren Ablauf Sie normalerweise überhaupt nichts wissen, registriert.



SQL Server Profiler zeigt keine Kennwörter an, die als Parameter an gespeicherte Systemprozeduren übergeben worden sind.

Zum Stoppen der Ablaufverfolgung klicken Sie einfach nur auf die Stopp-Schaltfläche im Abfragefenster. Daraufhin hält die Ablaufverfolgung an und es werden keine Informationen mehr aufgezeichnet.

Passen Sie nun eine Ablaufverfolgung an Ihre Anforderungen an. Schließen Sie die Ablaufverfolgung, die Sie gerade betrachtet haben, und klicken Sie auf die Schaltfläche NEUE ABLAUFVERFOLGUNG in der Symbolleiste (die erste von links). Daraufhin erscheint das Dialogfeld entsprechend Abbildung 20.10 und Sie können mit der Eingabe der Eigenschaften für Ihre Ablaufverfolgung beginnen. Als Erstes müssen Sie einen Ablaufverfolgungsnamen eingeben (das Beispiel verwendet Ablauftest; normalerweise gibt man hier einen aussagekräftigeren Namen ein).

Abbildung 20.10:
Eine Ablaufverfolgung zum Testen einrichten

Sie können Ihre Ablaufverfolgungen auch in einer Datei oder in einer Tabelle von SQL Server aufzeichnen, statt sie lediglich in der Benutzeroberfläche anzeigen zu lassen. Wenn Sie das Kontrollkästchen IN DATEI SPEICHERN einschalten, werden Sie zur Eingabe eines Speicherortes für diese Datei aufgefordert, wobei als Standard der Ordner *Eigene Dateien* und der Dateiname *Ablaufverfolgungsname.trc* angegeben sind. Im Beispiel heißt die Datei also *Ablauftest.trc*.

Wenn Sie das Kontrollkästchen **IN TABELLE SPEICHERN** anklicken, werden Sie nach Servername, Datenbankname, Besitzernamen und Tabellennamen gefragt. Wählen Sie an dieser Stelle eine Datenbank auf Ihrem SQL Server und übernehmen Sie den vorgegebenen Tabellennamen [Ablaufverfolgung]. Beachten Sie, dass die Tabelle den gleichen Namen wie Ihre Ablaufverfolgung bekommen wird, so dass es sinnvoll ist, eine Ablaufverfolgung mit einem Namen zu erstellen, der einen gültigen Bezeichner im SQL Server darstellt. Abbildung 20.10 spiegelt alle beschriebenen Einstellungen wider.

Wenn Sie auf die Registerkarte **EREIGNISSE** gehen (siehe Abbildung 20.11), haben Sie die Möglichkeit, die Informationen auszuwählen, die Sie in Ihrer Ablaufverfolgung registrieren lassen möchten. Als Standard sind Ereignisse ausgewählt, wenn sich Benutzer verbinden (Audit Login), die Verbindung trennen (Audit Logout) und beim Start der Ablaufverfolgung verbunden sind (ExistingConnection). Außerdem können Sie Ereignisse für Remoteprozeduraufrufe (RPC) und Transact-SQL-Stapel (SQL) sehen. Die Stapelergebnisse sind Aufrufe an SQL Server, die keine RPC-Ereignisse darstellen.

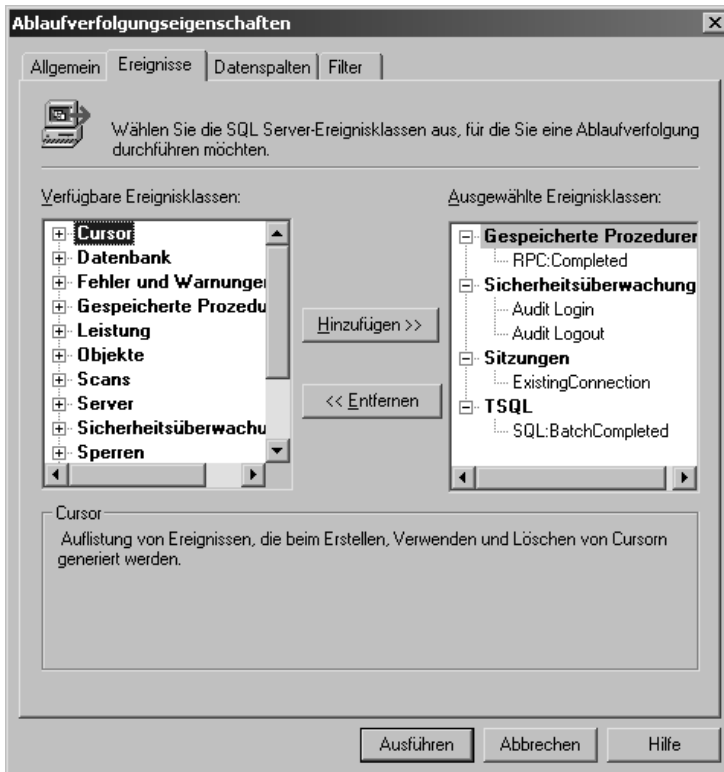


Abbildung 20.11:
Die Registerkarte
Ereignisse für eine
neue Ablaufverfolgung

Beachten Sie die umfangreiche Liste von verfügbaren Ereignisklassen auf der linken Seite des Dialogfelds. Auch wenn es relativ zeitaufwändig ist, sich mit diesen Ereignissen auseinander zu setzen, werden Sie schließlich feststellen, dass Sie nahezu alles mit Profiler anstellen können.

Klicken Sie nun auf die Registerkarte DATENSPALTEN, um die Daten auszuwählen, die in Ihrer Ablaufverfolgung angezeigt werden sollen (siehe Abbildung 20.12). Standardmäßig zeichnen Sie Ereignisse wie etwa den Benutzer auf, der eine Abfrage laufen lässt, welche Anwendung von diesem Benutzer eingesetzt wird, welchen SQL Server-Benutzernamen dieser Benutzer verwendet hat, verschiedene Systemleistungsdaten und natürlich noch den Text des Befehls, den der Benutzer ausgeführt hat. Außerdem können Sie Elemente mit ähnlichem Typ gruppieren. Falls Sie beispielsweise Ereignisse nach dem Benutzernamen der Anmeldung gruppieren möchten, markieren Sie diesen Eintrag in der Liste AUSGEWÄHLTE DATEN und klicken so lange auf die Schaltfläche AUFWÄRTS, bis *LoginName* unter dem Eintrag *Gruppen* steht. Im Bericht sind dann die Ereignisse nach Benutzernamen gruppiert und erscheinen nicht einfach fortlaufend. Mit der Gruppierung können Sie Ihre Berichte übersichtlicher gestalten.

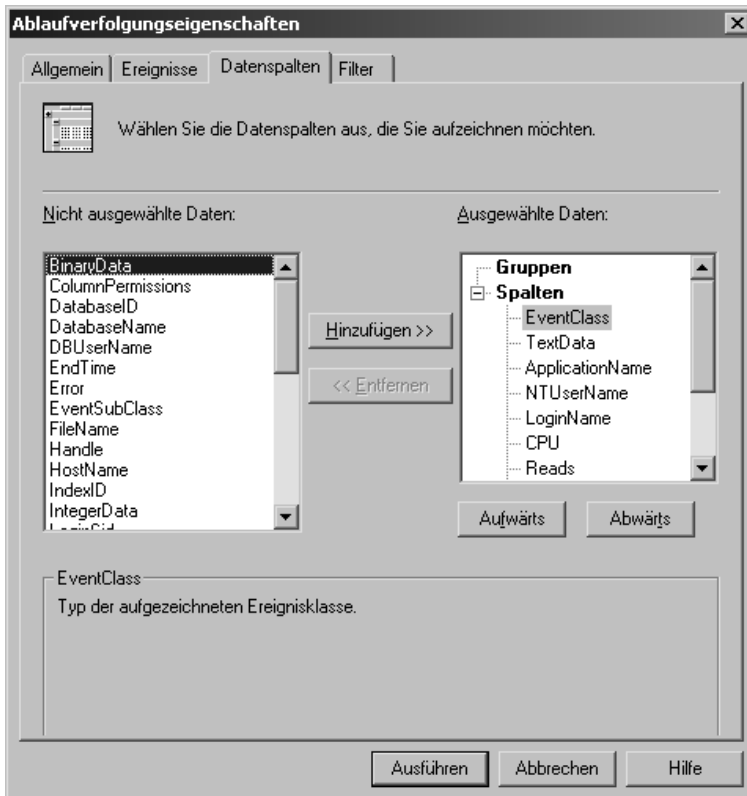


Abbildung 20.12:
Die Registerkarte
Datenspalten für
eine neue Ablaufver-
folgung

Zu guter Letzt können Sie Ihre Ablaufverfolgung auch noch filtern, so dass nur diejenigen Informationen gesammelt werden, die Sie auch wirklich wünschen. Beachten Sie in Abbildung 20.13, dass der Profiler seine Transact-SQL-Anweisungen standardmäßig ausschließt. Um die erweiterte gespeicherte Prozedur zu sehen, die das Dienstprogramm Profiler beim Starten aufruft, entfernen Sie diese Einschränkung, indem Sie den Eintrag SQL Profiler (unter Nicht wie) markieren und dann die <Entf>-Taste drücken.

Für jede Einschränkung, die Sie in Ihre Ablaufverfolgung aufnehmen möchten, markieren Sie die Option im Fenster KRITERIEN FÜR DIE EREIGNISABLAUFVERFOLGUNG und tippen dann in das jeweilige Textfeld das Element ein, das Sie ein- oder ausschließen möchten. Für jedes Ereigniskriterium wird ein entsprechendes Textfeld angezeigt.

Wenn Sie auf AUSFÜHREN klicken, startet die Ablaufverfolgung automatisch. Beachten Sie in Abbildung 20.14, dass SQL Server Profiler die gespeicherte Prozedur `sp_executesql` aufruft. Zwar ist die Verwendung einer grafischen Oberfläche erheblich einfacher, als selbst herauszufinden, welche erweiterte gespeicherte Prozedur aufzurufen ist, aber wenn Sie es wünschen, können Sie Aufrufe für diese Prozedur auch selbst einrichten.

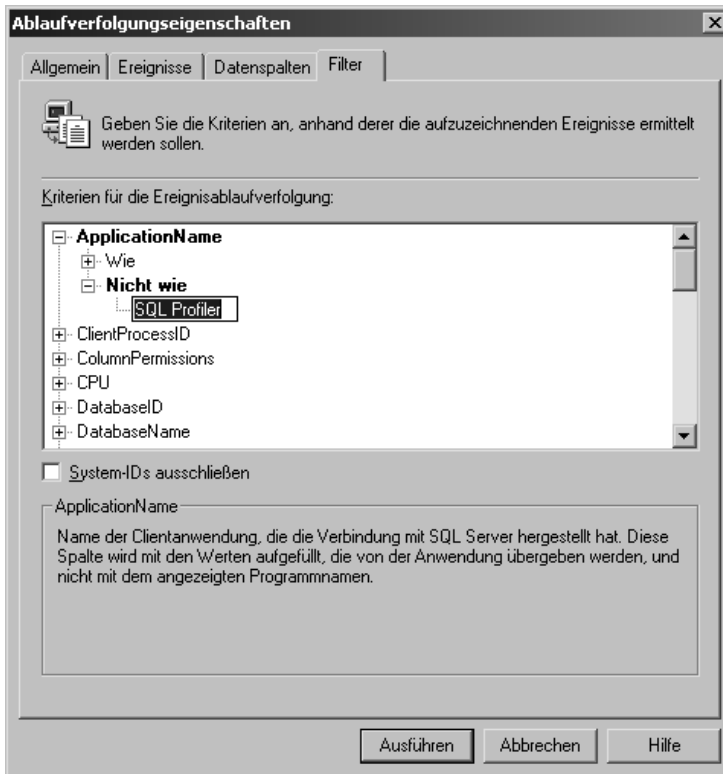
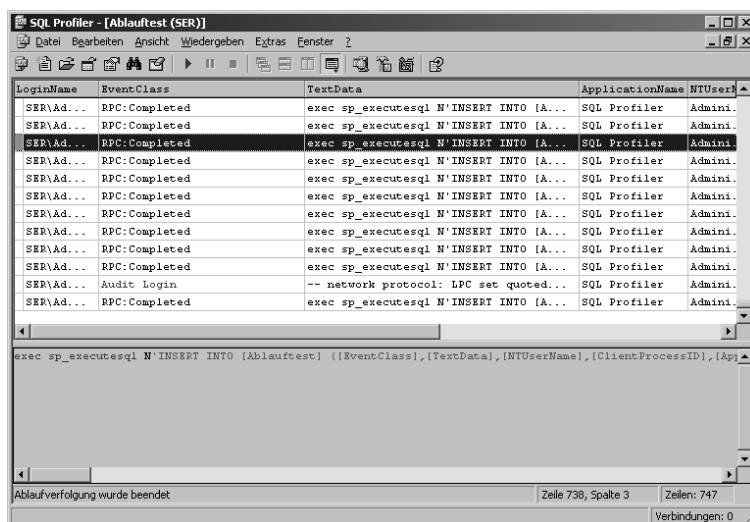


Abbildung 20.13:
Die Registerkarte
Filter für eine neue
Ablaufverfolgung



LoginName	EventClass	TextData	ApplicationName	NTUser
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...
SER\Ad...	Audit Login	-- network protocol: LPC set quoted...	SQL Profiler	Admini...
SER\Ad...	RPC:Completed	exec sp_executesql N'INSERT INTO [A...	SQL Profiler	Admini...

exec sp_executesql N'INSERT INTO [Ablauftest] ([EventClass],[TextData],[NTUserName],[ClientProcessID],[Ap...

Ablaufverfolgung wurde beendet

Zelle 738, Spalte 3 Zeilen: 747

Verbindungen: 0

Abbildung 20.14:
Eine aufgezeichne-
te Ablaufverfolgung

Es sei noch einmal darauf hingewiesen, dass diese Ablaufverfolgung einfach alle Transact-SQL-Anweisungen und RPC-Aufrufe registriert, die auf Ihrem SQL Server ausgeführt werden. Wenn Sie die Aufzeichnung nicht unterbrechen, läuft die Ablaufverfolgung unermüdlich weiter. Stoppen Sie aber jetzt die Ablaufverfolgung und überprüfen Sie, ob die Ablaufverfolgungsdatei angelegt worden ist. Wählen Sie dazu DATEI / ÖFFNEN / ABLAUFVERFOLGUNGSDATEI. Im Dialogfeld ÖFFNEN sollte die Datei *Ablauftest.trc* vorhanden sein. Wenn Sie diese Datei öffnen, können Sie sich davon überzeugen, dass alle Befehle aufgezeichnet worden sind.

Ablaufverfolgung automatisieren

Bis jetzt haben Sie den Grundstein für eine umfassende Ablaufverfolgung gelegt. Sie müssen jetzt nur noch alle Sicherheitsereignisse überwachen, alle Datenspalten hinzufügen und dann die Ablaufverfolgung ständig auf Ihrem Server laufen lassen. SQL Server 2000 kann sogar einen so genannten Dateirollover durchführen, d.h. eine neue Datei anlegen, wenn die bisherige Datei eine bestimmte Größe erreicht hat. Dadurch können Sie die Ablaufverfolgung ununterbrochen ausführen, ohne jeden einzelnen Verfolgungslauf sichern zu müssen. Wenn die aktuelle Ablaufverfolgungsdatei bis zur konfigurierten Größe gefüllt ist, wird eine neue Datei geöffnet, die bisherige geschlossen und Sie können die Datei bei Bedarf weiterverarbeiten.

Den Profiler-Lauf können Sie auch automatisch starten, wenn SQL Server startet. Prinzipiell kann das nach zwei Verfahren geschehen. Erstens können Sie alle Konfigurationsaufrufe in eine gespeicherte Prozedur einhüllen und dann diese gespeicherte Prozedur zur automatischen Ausführung markieren. Diese Methode ist tatsächlich recht einfach: die Ablaufverfolgung konfigurieren und ausführen. Dann wählen Sie aus dem Menü DATEI den Befehl SKRIPT FÜR ABLAUFVERFOLGUNG ERSTELLEN und SQL Server Profiler erstellt alle Aufrufe, um die Ablaufverfolgung zu erzeugen. Jetzt müssen Sie lediglich noch eine gespeicherte Prozedur um diese Aufrufe legen und markieren, dass diese Prozedur automatisch zusammen mit SQL Server startet.

Zweitens können Sie den Überwachungsmodus C2 mit der gespeicherten Prozedur `sp_configure` festlegen. Nachdem Sie SQL Server gestartet haben, läuft immer eine komplette Ablaufverfolgung ab. Weitere Informationen zur C2-Überwachung finden Sie in der SQL Server-Onlinedokumentation.

Systemleistung mit Ablaufverfolgung überwachen

Am Einfachsten lassen sich mit dem Profiler Leistungsprobleme ermitteln, indem Sie eine andere Ablaufverfolgungsvorlage wählen; hier bietet sich die Vorlage *SQLProfilerTuning* an. Gehen Sie dann auf die Registerkarte FILTER, um Einschränkungen für anzuzeigende bzw. nicht anzuzeigende Elemente festzulegen. Beispielsweise können Sie nur die Datenbank filtern, an der Sie speziell interessiert sind, und die Ergebnisse auf Abfragen einschränken, die länger als eine bestimmte Zeitspanne dauern (z.B. 1 Sekunde). Beachten Sie, dass die Werte in Millisekunden anzugeben sind. Abbildung 20.15 zeigt einen Filter, der die Profiler-Ergebnisse auf Abfragen einschränkt, die mindestens 1 Sekunde in der Datenbank *pubs* laufen (der in Abbildung 20.15 nicht sichtbare Knoten über *Wie* heißt *DatabaseName*).

Das war's auch schon! Starten Sie jetzt die Ablaufverfolgung und beobachten Sie die Abfragen, die sich in Ihrer Umgebung anhäufen und den eingerichteten Filterkriterien entsprechen. Vergessen Sie nicht, den Dateistandort festzulegen, wo die Ablaufverfolgung aufgezeichnet werden soll.

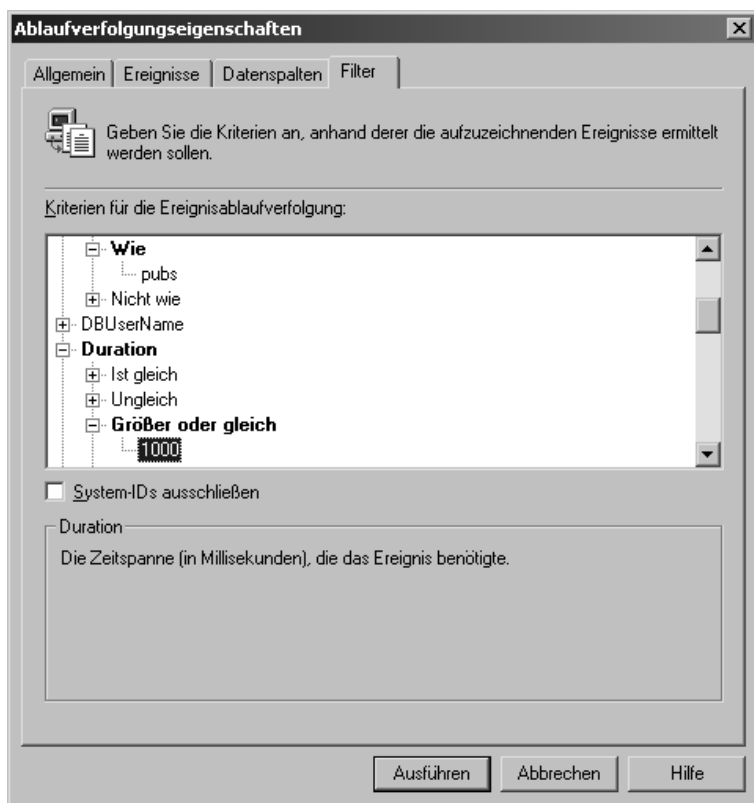


Abbildung 20.15:
Filtern im Hinblick
auf Leistungsproble-
me (der nicht sicht-
bare Knoten über
Wie heißt Database-
Name)

20.7 Der Indexoptimierungs-Assistent

Die im letzten Abschnitt erläuterte Ablaufverfolgung führt fast zwangsläufig zum Indexoptimierungs-Assistenten. Dieser Assistent kann eine gespeicherte Ablaufverfolgungsdatei lesen und dann Änderungen an Ihren Indizes auf dem Server vorschlagen, um die Leistung der während der Ablaufverfolgung registrierten Abfragen zu verbessern. Den Assistenten setzen Sie wie folgt ein:

1. Wählen Sie EXTRAS / INDEXOPTIMIERUNGS-ASSISTENT im Menü des SQL Server Profilers. Lesen Sie den Einführungsbildschirm und klicken Sie dann auf WEITER, um zum Dialogfeld SERVER UND DATENBANK AUSWÄHLEN (siehe Abbildung 20.16) zu gelangen, nachdem Sie sich bei SQL Server angemeldet haben.

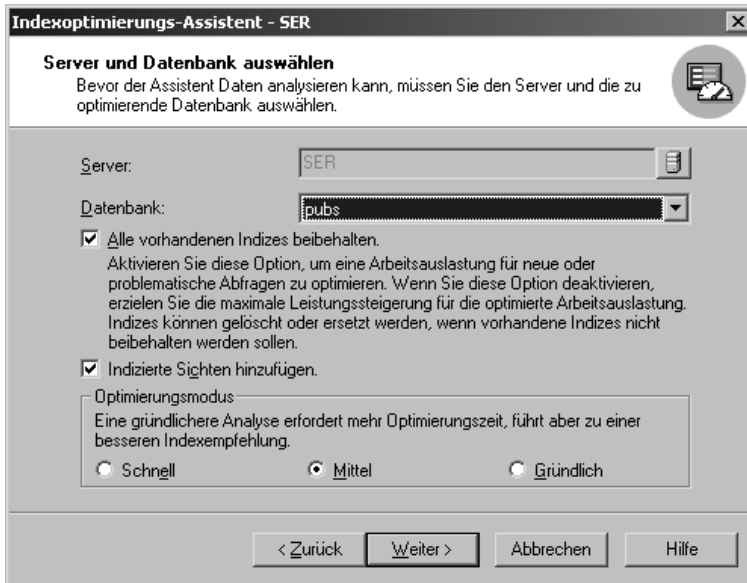


Abbildung 20.16:
Das Dialogfeld Ser-
ver und Datenbank
auswählen

2. Falls Sie einen anderen Server überwachen möchten, wählen Sie diesen aus. Anschließend legen Sie eine Datenbank fest. Der Indexoptimierungs-Assistent kann Indexempfehlungen immer nur für jeweils eine Datenbank geben. In der Standardeinstellung werden alle vorhandenen Indizes beibehalten und die Gründlichkeit der Analyse ist auf *Mittel* eingestellt. Allerdings empfiehlt es sich, die Option *Gründlich* für den Optimierungsmodus zu wählen, damit der Assistent keine Indizes vorschlägt, die leistungsmäßig schlechter als die bisherigen Indizes sind.
3. Klicken Sie auf WEITER. Sie werden nun nach der zu analysierenden Arbeitsauslastung gefragt (siehe Abbildung 20.17). Falls Sie bereits eine Ablaufverfolgungsdatei vorliegen haben, können Sie entweder diese Datei auswählen oder auf eine Ablaufverfolgungstabelle verweisen.
4. Geben Sie eine vorhandene Ablaufverfolgungsdatei an (z.B. *Ablauftest.trc* wie in Abbildung 20.17) oder wählen Sie die SQL Server-Tabelle mit Ihren gespeicherten Ablaufverfolgungsinformationen aus. Klicken Sie auf die Schaltfläche ERWEITERTE OPTIONEN, um das in Abbildung 20.18 dargestellte Dialogfeld zu öffnen.

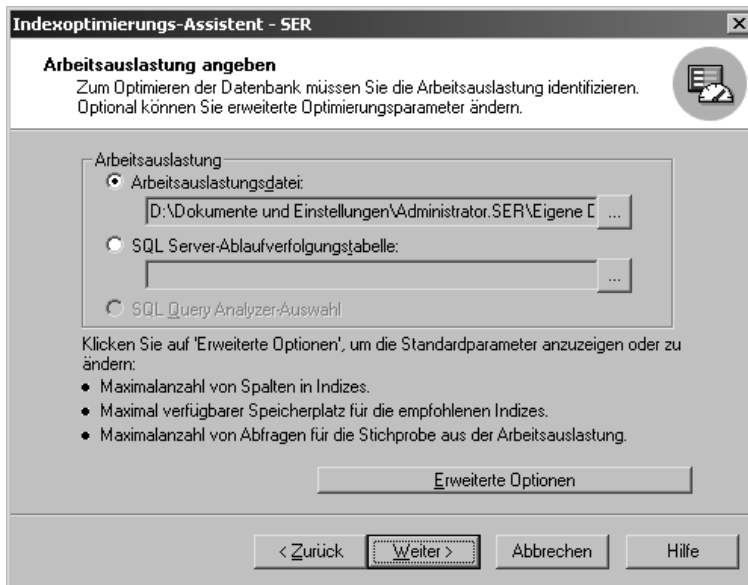


Abbildung 20.17:
Hier geben Sie an,
wo der Assistent eine
Arbeitsauslastung
finden kann.

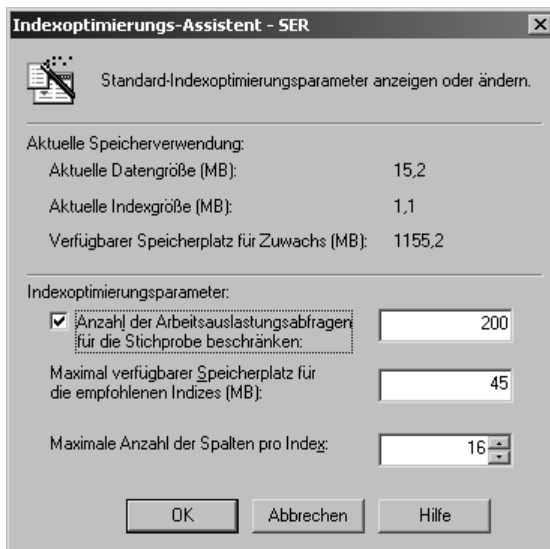


Abbildung 20.18:
Erweiterte Konfigurationsoptionen für
den Indexoptimierungs-Assistenten

5. Klicken Sie zur Übernahme Ihrer Änderungen auf OK und dann auf WEITER, um zum Dialogfeld ZU OPTIMIERENDE TABELLEN AUSWÄHLEN zu gelangen (siehe Abbildung 20.19). Standardmäßig versucht der Assistent nicht, Tabellen zu optimieren. Wählen Sie also alle Tabellen in Ihrer Datenbank aus. Klicken Sie dann auf WEITER.



Abbildung 20.19:
Klicken Sie auf **Alle Tabellen auswählen**, da standardmäßig keine Tabellen einbezogen werden.

Der Indexoptimierungs-Assistent führt nun eine Analyse der Arbeitsauslastung aus, die Sie ihm angegeben haben. Das kann je nach Größe der Datei eine ganze Weile dauern. Schließlich zeigt der Assistent die empfohlenen Indizes im Dialogfeld INDEXEMPFEHLUNGEN an (siehe Abbildung 20.20).

6. Da die Ablaufverfolgung *Ablauftest* nur zwei simple Abfragen enthält, empfiehlt der Indexoptimierungs-Assistent keine Änderungen. Empfehlungen zeigen Sie an, indem Sie auf die Schaltfläche ANALYSE klicken und sich den Bericht für die aktuelle Konfiguration ansehen; beachten Sie, dass er hier identisch zum Bericht für die empfohlene Konfiguration ist.
7. Klicken Sie auf WEITER. Wenn der Assistent Änderungen empfohlen hat, können Sie nun wählen, ob Sie die empfohlenen Änderungen nach einem gegebenen SQL Server-Agent-Zeitplan ausführen oder die Änderungen als Transact-SQL-Skript speichern möchten (siehe Abbildung 20.21). Wenn Sie die Änderungen speichern, können Sie sie später bei Bedarf überprüfen oder ausführen.
8. Da der Assistent im Beispiel keine Empfehlungen geben kann, erscheint auch nicht das in Abbildung 20.21 gezeigte Dialogfeld und Sie gelangen direkt zum Abschlussdialogfeld des Assistenten. Andernfalls hat der Assistent Änderungen vorgeschlagen. Um die Empfehlungen sofort auszuführen, klicken Sie auf WEITER und dann auf FERTIG STELLEN.

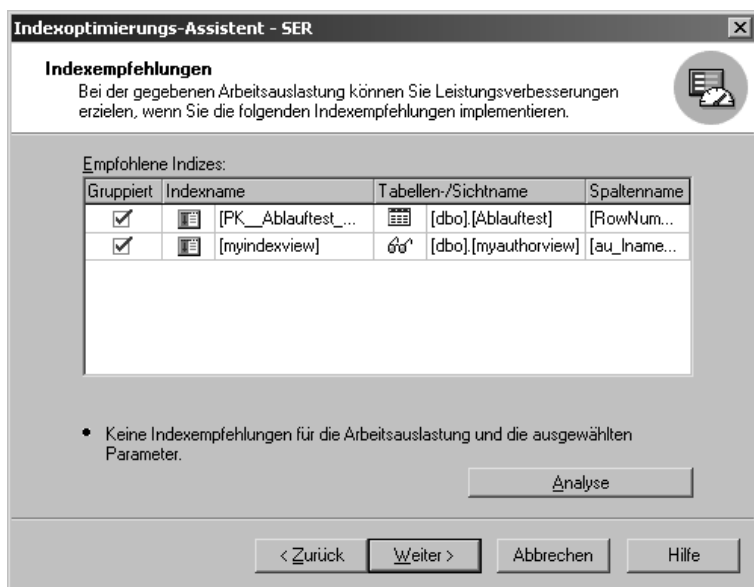


Abbildung 20.20:
Das Dialogfeld Indexempfehlungen des Indexoptimierungs-Assistenten

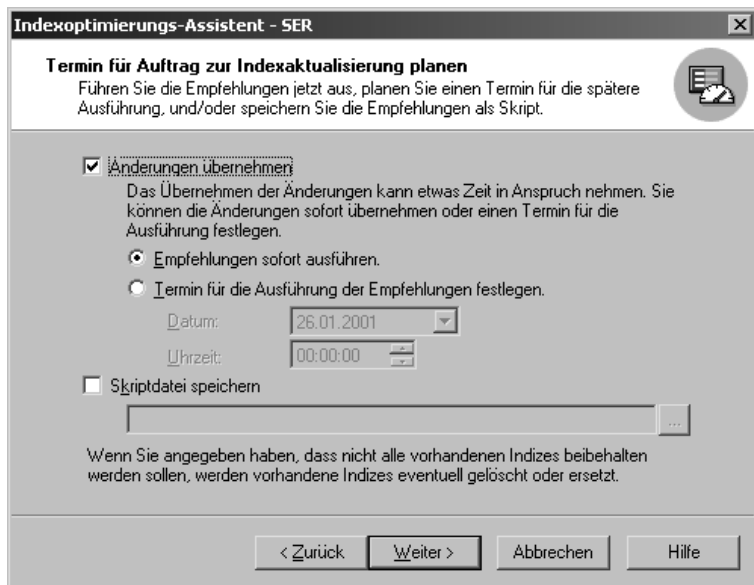


Abbildung 20.21:
Das Dialogfeld Termin für Auftrag zur Indexaktualisierung planen

Das war es auch schon! Die Entwickler von SQL Server haben tatsächlich einen ungeheuer komplexen Auftrag (wie die Auswahl der auf dem Server benötigten Indizes) in eine Angelegenheit von wenigen Mausklicks verwandelt!

20.8 Zusammenfassung

Sie haben in der heutigen Lektion gelernt, wie Sie Windows 2000 und SQL Server auf optimale Leistung einstellen. Außerdem haben Sie erfahren, wie Sie SQL Server über den Systemmonitor von Windows 2000 überwachen und verschiedene Tools für die Optimierung einsetzen, die im Lieferumfang von SQL Server enthalten sind. Hierzu gehören der SQL Server Profiler, der Indexoptimierungs-Assistent und das Fenster AKTUELLE AKTIVITÄT im SQL Server Enterprise Manager.

20.9 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Wo bekomme ich einen raschen Überblick über die Aktivitäten von SQL Server?*
A Im Fenster Aktuelle Aktivität des SQL Server Enterprise Managers.
- F** *Welche Art von Netzwerkkarte sollte ich in meinem Server installieren?*
A Eine 32- oder 64-Bit-Busmastering-Netzwerkkarte.
- F** *Welche Art RAID-Array eignet sich am besten für Transaktionsprotokolle?*
A Ein gespiegelter oder sogar geduplexer Festplattensatz.
- F** *Mit welchem Werkzeug kann ich die richtigen Indizes in meiner Datenbank auswählen?*
A Mit dem Indexoptimierungs-Assistenten.

Kontrollfragen

1. Welche Komponente des Systems verdient aus der Sicht der Auswirkungen auf die Gesamtleistung die stärkste Beachtung bei der Optimierung der Leistung?
2. Warum sollten die Transaktionsprotokolle auf anderen Festplatten als die Auslagerungsdatei von Windows 2000 untergebracht werden?
3. Welchen Leistungsindikator von Windows 2000 würden Sie überwachen, wenn Sie den Speicher von SQL Server manuell konfiguriert haben und nun befürchten, dass Sie zuviel Speicher zugewiesen haben?
4. Welchen Parameter in SQL Server würden Sie zur Anpassung der von SQL Server benutzen Prozessoren ändern?
5. Welchen SQL Server-Leistungsindikator würden Sie beobachten, um zu ermitteln, wie viel Arbeitslast von SQL Server erledigt wird?

Übungen

Die Leistungsoptimierung beinhaltet normalerweise die Überwachung eines Systems mit vielen Benutzern. Falls Sie eine solche Systemumgebung nicht zur Verfügung haben, können Sie folgende Schritte zur Simulation einer derartigen Umgebung unternehmen:

1. Starten Sie den SQL Server Profiler und legen Sie eine Ablaufverfolgung an. Vergewissern Sie sich, dass Sie die Ablaufverfolgungsinformationen in eine Datei sichern. Starten Sie dann mehrere Sitzungen des SQL Server Query Analyzers und führen Sie über jede Verbindung zu SQL Server gespeicherte Prozeduren und Abfragen aus.
2. Starten Sie den SQL Enterprise Manager und überwachen Sie die Aktivitäten über das Fenster AKTUELLE AKTIVITÄT. Beobachten Sie, welche Informationen Sie erhalten und probieren Sie die verschiedenen Optionen wie etwa das Abbrechen eines Prozesses aus. Beobachten Sie das Verhalten der Clientanwendungen (in diesem Fall des Query Analyzers).
3. Nachdem Sie eine Reihe von Abfragen ausgeführt und per Ablaufverfolgung aufgezeichnet haben, stoppen Sie die Ablaufverfolgung und führen den Indexoptimierungs-Assistenten aus. Speichern Sie die Empfehlungen des Assistenten in einem Skript und untersuchen Sie, was der Assistent tatsächlich für Sie tun kann.



XML und SQL Server 2000

In der gestrigen Lektion haben Sie die Tools kennen gelernt, mit denen sich die Stabilität und Leistung von Windows und SQL Server überwachen lässt. Die heutige Lektion beschäftigt sich mit einer neuen Technologie in SQL Server 2000: XML. Sieht man sich aktuelle Computerzeitschriften an, kann der Eindruck entstehen, dass man XML die gleiche euphorische Bedeutung beimisst wie noch vor wenigen Jahren dem Internet. Und in gewisser Weise stimmt das auch. Mit XML lassen sich Daten in einer selbstbeschreibenden Art behandeln. Diese Fähigkeit zielt letztlich darauf ab, den E-Commerce zu vereinfachen, einschließlich der Business-zu-Business (B2B) und Business-zu-Consumer (B2C) Märkte. In das SQL Server 2000-Modul ist eine direkte XML-Unterstützung integriert.

21.1 Was ist XML?



XML steht für *Extensible Markup Language* – etwa: erweiterbare Beschreibungssprache. Sicherlich kennen Sie bereits die Seitenbeschreibungssprache HTML (Hypertext Markup Language), mit der sich Webseiten im Internet anzeigen lassen. HTML basiert auf einem festgelegten Satz von Tags (wie zum Beispiel `<P>` und `</P>`). Ein großer Teil der Leistung von HTML besteht darin, dass es leicht zu unterstützen und universell verfügbar ist, das stellt aber gleichzeitig seine größte Beschränkung dar.

Es gab zahlreiche Ansätze, HTML zu erweitern, beispielsweise DHTML (Dynamic HTML) von Microsoft. Allerdings lässt sich der Datenaustausch mit diesen anderen Beschreibungssprachen nicht zufriedenstellend lösen. XML ist ein Versuch, mehrere Fliegen mit einer Klappe zu erschlagen. Die Ziele von XML lauten:



Dieses Kapitel bringt Auszüge aus der W3C-Spezifikation von XML. Die vollständige XML-Spezifikation finden Sie unter <http://www.w3.org/TR/REC-xml>.

- XML soll sich unkompliziert über das Internet einsetzen lassen.
- XML soll eine breite Vielfalt von Anwendungen unterstützen.
- XML wird mit SGML kompatibel sein.
- Es ist einfach, Programme zu schreiben, die XML-Dokumente verarbeiten.
- Die Anzahl der optionalen Features in XML ist auf einem unbedingt notwendigen Minimum zu halten – im Idealfall Null.
- XML-Dokumente sollen im Klartext lesbar und verständlich sein.
- Der XML-Entwurf soll schnell vorbereitet sein.

- Der Entwurf von XML wird formell und knapp sein.
- XML-Dokumente werden leicht zu erstellen sein.
- Die Kürze spielt in der XML-Beschreibung eine untergeordnete Rolle.

Es ist also ein ehrgeiziges Projekt. Unsere Konzentration auf XML besteht darin, seine Fähigkeiten zum Datenaustausch zu verwenden. Mit XML lässt sich der Datenaustausch unterstützen, weil man eigene Tags definieren kann.

Nehmen wir an, dass Sie einen Kundennamen, eine Adresse und eine Telefonnummer haben. In SQL Server kann man das als Tabelle wie folgt definieren:

```
Create table customer
(customer_id int not null,
lastname nvarchar(50) not null,
firstname nvarchar(50) not null,
address nvarchar(500) not null,
City nvarchar(50) not null,
State nvarchar(30) not null,
Postal_code nvarchar(10) null)
```

Das Gleiche lässt sich in XML so schreiben:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Customer>
<Customer_Id></Customer_Id>
<lastname></lastname>
<firstname></firstname>
<address></address>
<city></city>
<state></state>
<postal_code></postal_code>
</Customer>
```



Als erster Unterschied sollte Ihnen auffallen, dass das XML-Dokument keine Auskunft über den Datentyp einer Spalte gibt. Diese Informationen sind in die Schemadefinition von SQL Server integriert, aber nicht unbedingt in XML. Diese Definitionen lassen sich über eine so genannte *Document Type Definition* (DTD) erhalten. DTDs speichern die Schemainformationen, die den Inhalt des XML-Dokuments definieren.



Stylesheets werden ebenfalls referenziert. Ein *Stylesheet* gibt die Format- und Anzeigeeinformationen an, die für die Formatierung der XML-Dokumente zu verwenden sind. Die *Extensible Stylesheet Language* (XSL) ist die Formatierungssprache für die XML-Dokumente.

Wenn Sie sich allerdings an die Definitionen halten, ist der Datenaustausch problemlos möglich. Wenn man Daten in ein Dokument im XML-Format unterbringt, kann man die Daten mit jedem anderen System behandeln, das XML unterstützt. Es spielt keine Rolle, ob die Daten in einer SQL Server-Datenbank, einem Excel-Tabellenblatt oder in einer gewöhnlichen Textdatei enthalten sind. Wichtig ist vor allem, dass man die Daten nicht mehr in der gleichen Weise speichern muss, sondern nur, dass man die Daten zwischen Systemen mit diesem allgemeinen Format überträgt.

Für einen tiefergehenden Einstieg in XML sei auf entsprechende Literatur verwiesen. In diesem Buch geht es nur um einige Details, die für das Verständnis dieser Lektion erforderlich sind.

SQL Server 2000 legt XML in verschiedener Weise frei. Auf SQL Server kann man über Internet Information Server (IIS) zugreifen. Man kann Abfragen direkt als URLs absenden (in herkömmlichen Strings der Art *http://www.microsoft.com/sql*), man kann SQL Server-Ergebnismengen in vorformatierten Dokumenten verarbeiten und man kann Daten direkt von SQL Server über die *SELECT*-Anweisung auswählen. Man kann sogar XML-formatierte Dokumente über Transact-SQL öffnen.

21.2 Auf SQL Server via IIS zugreifen

Der erste Weg, nach dem man auf SQL Server mittels XML zugreifen kann, besteht über den Webserver von Microsoft – IIS. Man kann eine Datenbank mit einer speziellen Erweiterung offen legen, die das SQL Server-Team für IIS geschrieben hat. Auf diese Weise kann man Abfragen an SQL Server via XML ausführen und natürlich die Ergebnisse direkt auf einer Website anzeigen. Noch besser: die via XML zurückgegebenen Daten sind unter Verwendung Ihres XML-Schemas vorformatiert und selbstbeschreibend für jeden, der das gleiche XML-Schema versteht.

Um den Zugriff via IIS zu erlauben, ist zuerst das virtuelle Verzeichnis für SQL Server zu installieren.

Virtuelle Verzeichnisse und SQL Server

Wenn Sie SQL Server einrichten wollen, um direkten Zugriff vom Internet zu unterstützen, müssen Sie einige Konfigurationsschritte durchlaufen. Zuerst richten Sie IIS ein und legen ein virtuelles Verzeichnis an. Haben Sie eine Standardinstallation von Windows 2000 durchgeführt, ist IIS bereits installiert und läuft auf Ihrem Computer. Andernfalls öffnen Sie das Programm *SOFTWARE* in der *SYSTEMSTEUERUNG*, wählen *WINDOWS-KOMPONENTEN HINZUFÜGEN/ENTFERNEN* und fügen die Komponente *Internet-Informationdienste (IIS)* hinzu.

Nachdem IIS verfügbar ist, fahren Sie mit der Einrichtung des virtuellen Verzeichnisses fort.

IIS und das virtuelle Verzeichnis einrichten

Wählen Sie **START / PROGRAMME / MICROSOFT SQL SERVER / XML-UNTERSTÜTZUNG VON SQL IN IIS KONFIGURIEREN**, um das Dienstprogramm **IIS VIRTUAL DIRECTORY MANAGEMENT FOR SQL SERVER** zu starten (siehe Abbildung 21.1).

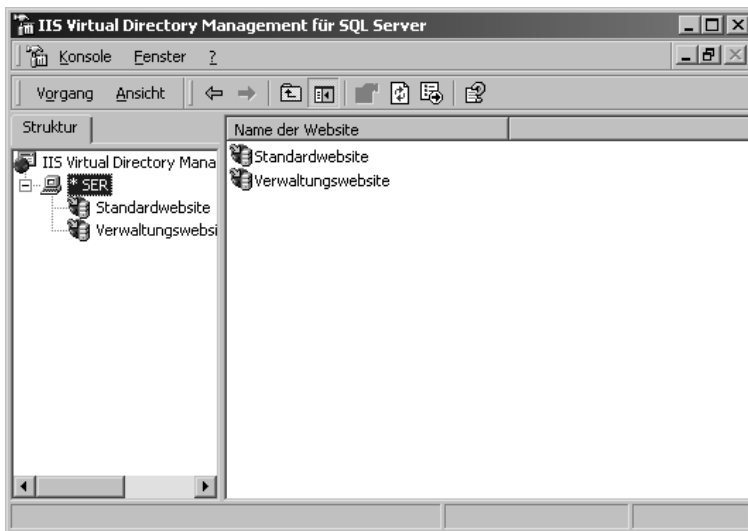


Abbildung 21.1:
Die virtuelle Ver-
zeichnisverwaltung
von IIS für SQL
Server

Erweitern Sie den Knoten Ihres Servers, markieren Sie den Knoten *Standardwebsite* und wählen Sie aus dem Menü **VORGANG** den Befehl **NEU** und danach **VIRTUELLES VERZEICHNIS**. Daraufhin befinden Sie sich im Dialogfeld **EIGENSCHAFTEN VON NEUES VIRTUELLES VERZEICHNIS**. Jetzt können Sie ein virtuelles Stammverzeichnis erstellen. Für das Beispiel im Buch erzeugen Sie ein virtuelles Verzeichnis, das die Datenbank *pubs* widerspiegelt. Legen Sie also den Namen des virtuellen Verzeichnisses mit *pubs* fest. Außerdem sollten Sie ein Verzeichnis unter Ihrem Verzeichnis *inetpub\wwwroot* anlegen. Abbildung 21.2 zeigt die Beispieleinstellungen.

Gehen Sie jetzt auf die Registerkarte **SICHERHEIT** (siehe Abbildung 21.3) und konfigurieren Sie die Sicherheitseinstellungen für das virtuelle Verzeichnis. Auf dieser Registerkarte sind mehrere wichtige Entscheidungen zu treffen, wählen Sie also mit Bedacht.



Abbildung 21.2:
Die Registerkarte Allgemein
mit den Festlegungen für das
neue virtuelle Verzeichnis

Die Standardauswahl ist *Immer anmelden als*. Wenn Sie Ihren SQL Server im gemischten Sicherheitsmodus eingerichtet haben, können Sie in diesem Dialogfeld die Standardsicherheitsinformationen konfigurieren. Wie bereits mehrfach im Buch erwähnt, sollten Sie die Standardsicherheit nur ausnahmsweise verwenden; die Windows-Authentifizierung ist wesentlich besser geeignet. Wenn Sie im Abschnitt ANMELDEINFORMATIONEN den Kontotyp auf *Windows* setzen, wird standardmäßig das IIS-Dienstkonto (*IUSER_Computername*) verwendet. Diese Einstellung können Sie überschreiben und ein beliebiges Konto eintragen.

Der Nachteil dieser Methoden besteht darin, dass Sie nie wissen, wer sich hier angemeldet hat, wenn er es erst einmal bis zu SQL Server geschafft hat – jeder meldet sich als der Benutzer an, den Sie hier konfiguriert haben, und jeder hat die gleichen Sicherheitsberechtigungen. Überdenken Sie diese Option gründlich, bevor Sie hier ein Konto einrichten.

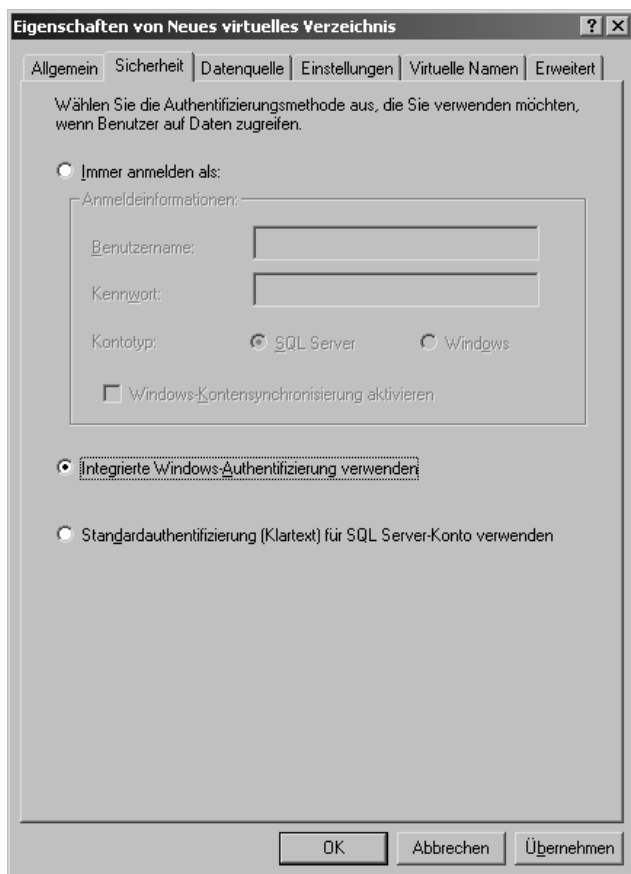


Abbildung 21.3:
Die Registerkarte Sicherheit des
neuen virtuellen Verzeichnisses

Eine bessere Alternative ist die Option *Integrierte Windows-Authentifizierung verwenden*. Wenn Sie diese Einstellung wählen, kann SQL Server die Sicherheitsinformationen des Endbenutzers empfangen – *kann* deshalb, weil normalerweise die Windows NT/2000-Sicherheitsinformationen eines Benutzers nicht an den nächsten Computer weitergereicht werden, falls sich SQL Server auf einem anderen Computer als IIS befindet. Wenn jedoch die Windows 2000- und Kerberos-Sicherheitsdelegierung richtig festgelegt sind, werden die Sicherheitsinformationen des Endbenutzers an SQL Server weitergegeben. Weitere Informationen zu Kerberos und zur Delegierung von Sicherheitskonten entnehmen Sie bitte der SQL Server-Onlinedokumentation.



Kerberos ist das mit Windows 2000 neu eingeführte Sicherheitsprotokoll. Es ist wesentlich sicherer als vorherige Implementierungen und etwas komplexer. Allerdings bringt es auch die Fähigkeit mit, die bereits erwähnte Delegierung zu realisieren.

Schließlich können Sie die Option *Standardauthentifizierung (Klartext)* für *SQL Server-Konto verwenden* wählen. Schon die Bezeichnung lässt vermuten, dass es sich hier nicht um die beste Wahl handelt. Der Endbenutzer muss seine Sicherheitsinformationen (einen normalen SQL Server-Benutzernamen) eingeben und diese Informationen werden an SQL Server weitergeleitet. In der Standardeinstellung gehen sie auch in dieser Form ohne jegliche Verschlüsselung über das Netzwerk – zumindest zwischen Benutzer und IIS.

Legen Sie also für das Beispiel die Option *Integrierte Windows-Authentifizierung verwenden* fest und wechseln Sie dann zur Registerkarte DATENQUELLE (siehe Abbildung 21.4).

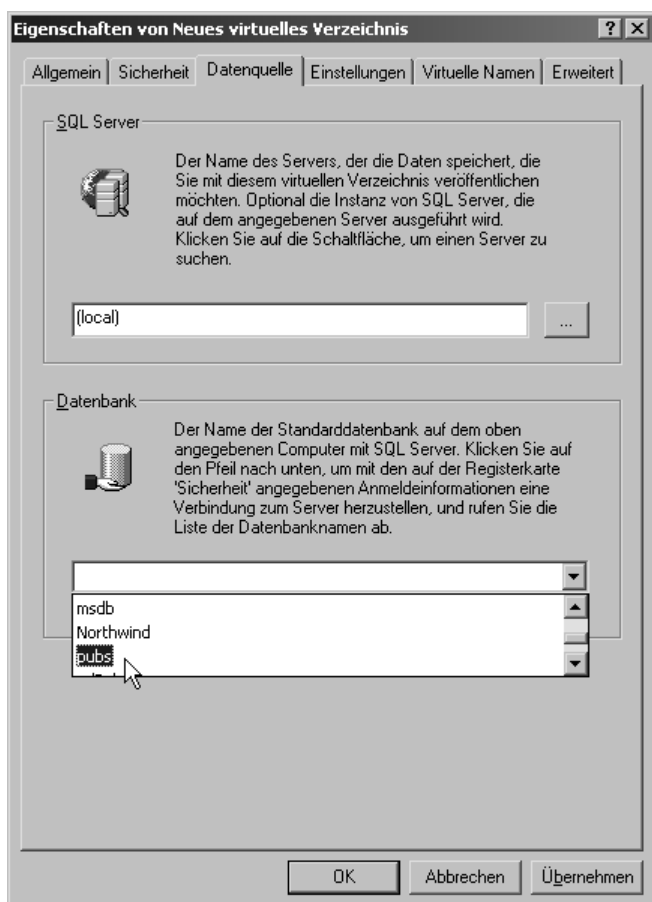


Abbildung 21.4:
Eine Datenquelle auswählen

Hier wählen Sie den Server aus, zu dem Sie die Verbindung herstellen möchten. Die Standardeinstellung *(local)* meint eine Standardinstanz einer lokalen Kopie von SQL Server. Sie können hier den Namen Ihres Servers eintippen oder aus einer Liste der im Netzwerk

verfügbaren SQL Server auswählen. Klicken Sie dann auf die Dropdown-Liste der Datenbanken, wie es Abbildung 21.4 zeigt, um die zu verwendende Datenbank auszuwählen – für das Beispiel soll es wieder die Datenbank *pubs* sein.

Als Nächstes gehen Sie auf die Registerkarte EINSTELLUNGEN (siehe Abbildung 21.5). Auch hier sind mehrere kritische Entscheidungen zu treffen.

In der Voreinstellung ist nur die Option *Vorlagenabfragen zulassen* aktiviert. Es handelt sich hier sozusagen um eine relativ »gefahrlose« Option. Wie Abbildung 21.5 zeigt, sind für das Beispiel alle Optionen eingeschaltet, damit Sie sich von der Funktionsweise ein Bild machen können. Die Optionen haben folgende Bedeutung:

- *URL-Abfragen zulassen* ermöglicht den Benutzern, die auszuführenden Transact-SQL-Anweisungen direkt im URL – bei der Angabe der Internetadresse im Internet Explorer – zu spezifizieren. Diese Option birgt ein gewisses Risiko in sich, weil es absolut keine Einschränkungen gibt, welche Arten von Transact-SQL-Anweisungen abgesetzt werden können. Die Abfragen geben Sie in der Form

`http://SER/pubs?SQL=select++from+authors+for+XML+AUTO`

- ein, wobei Sie Ihren Servernamen für *SER* einsetzen. Beachten Sie, dass der Ausdruck ?SQL= von der Groß-/Kleinschreibung abhängig ist.



Für diese URL-Abfragen ist *jede* Art von Transact-SQL-Anweisung gültig. Nachdem Sie alles ordnungsgemäß eingerichtet haben, starten Sie einen Versuch: Was passiert wohl, wenn Sie `CREATE DATABASE randomtest` ausführen? Sofern Sie über die Rechte für SQL Server verfügen, können Sie die Datenbank erstellen. Vielleicht sollten Sie sich doch noch einmal mit den betreffenden Optionen auf der Registerkarte SICHERHEIT auseinandersetzen.

- *Vorlagenabfragen zulassen* ist die – relativ betrachtet – sicherste Option, die Sie auf dieser Registerkarte aktivieren können, weil Benutzer nur Abfragen ausführen können, die Sie in Vorlagendateien vorkonfiguriert haben. Ein Beispielaufruf hat die Form `http://SER/pubs/myquery.xml`.
- *XPath zulassen* ermöglicht den Benutzern, Abfragen mit der Abfragesprache XPath zu senden, die SQL Server-Sichten verwenden. Zwischen den XML-Abfragen und der SQL Server-Datenbank findet eine Schemazuordnung statt.
- *POST zulassen* erlaubt es, ein HTML-Formular zu erstellen und es dann vom Kunden ausfüllen zu lassen. Letztendlich liefern die Felder des Formulars die Parameter für die Komplettierung einer Transact-SQL-Abfrage. Die Größe von POST-Abfragen können Sie beschränken, so dass übermäßig große Abfragen nicht zu SQL Server übertragen werden.

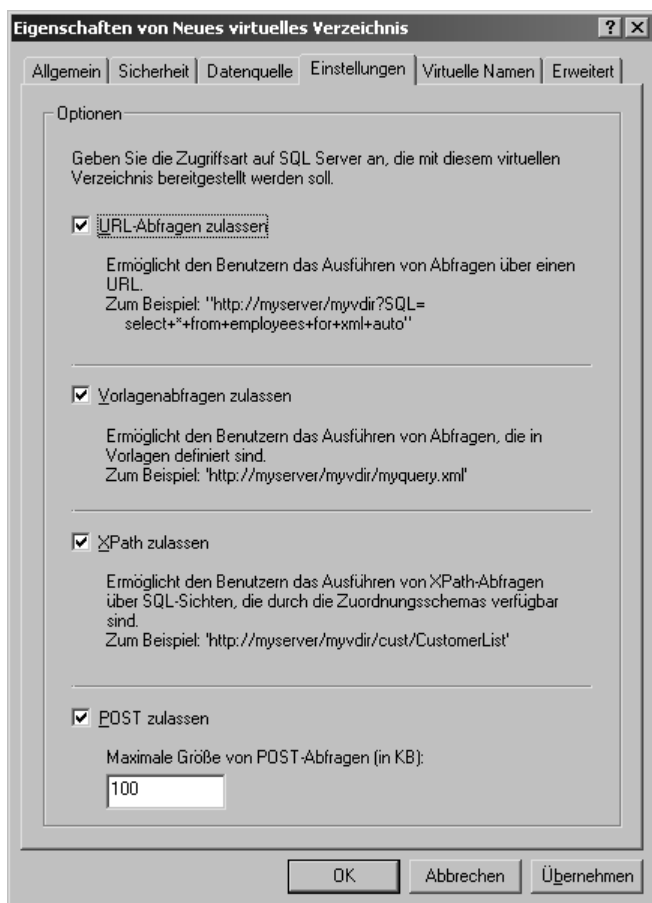


Abbildung 21.5:
Die erlaubten Zugriffsarten
auswählen

Schalten Sie auf der Registerkarte EINSTELLUNGEN alle Kontrollkästchen ein, damit Sie die Optionen in Kürze ausprobieren können.

Auf der Registerkarte VIRTUELLE NAMEN (siehe Abbildung 21.6) können Sie optional Neuuzuordnungen vornehmen, damit sich die Benutzer auch bei komplizierteren Verzeichnisstrukturen in IIS nicht mit umständlichen Pfadnamen plagen müssen. Außerdem ist es möglich, einen virtuellen Namen als Alias anzugeben, so dass Sie Dateien bei Bedarf verschieben können, ohne dass die Benutzer neue URLs lernen müssen.

Schließlich können Sie auf der Registerkarte ERWEITERT (siehe Abbildung 21.7) weitere Optionen für Ihr virtuelles Verzeichnis festlegen. Diese Einstellungen sollten Sie – wenn überhaupt notwendig – erst ändern, nachdem Sie tiefer in das gesamte Thema eingedrungen sind. Für das Beispiel können Sie einfach die Voreinstellungen übernehmen.

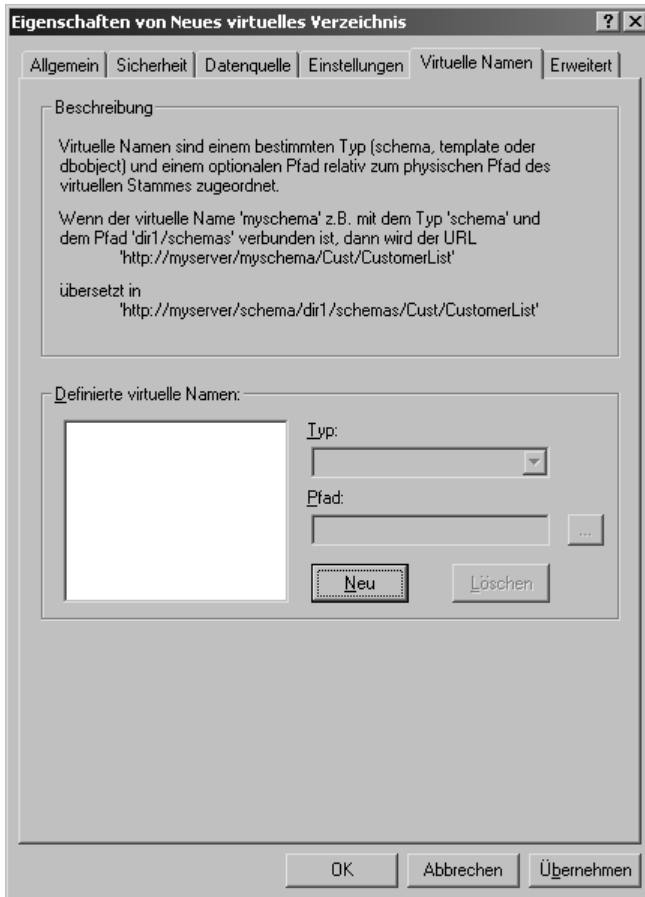


Abbildung 21.6:
Die Registerkarte Virtuelle
Namen

Klicken Sie auf OK, um das virtuelle Verzeichnis zu erstellen. Gratulation! Damit haben Sie Ihr virtuelles SQL Server-Verzeichnis in IIS eingerichtet, so dass der Zugriff per XML auf SQL Server möglich ist.

SQL Server über URLs ansprechen

Jetzt kommen wir zu den eigentlich interessanten Dingen: Nachdem Sie diese Website eingerichtet haben, können Sie SQL Server direkt über das Internet verwenden. SQL-Anweisungen lassen sich als Teil des in den Webbrowser eingegebenen URLs ausführen oder Sie können Abfragen mithilfe von Vorlagendateien vordefinieren. Vorher müssen Sie sich aber noch mit einigen Sonderzeichen auseinandersetzen, um die Beschränkungen der URL-Schnittstelle zu umgehen.

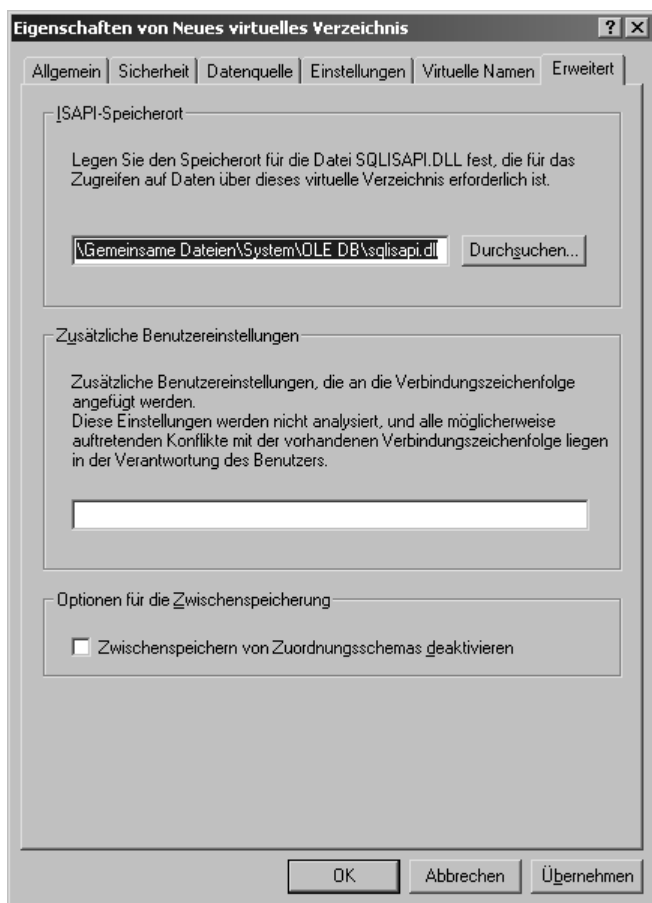


Abbildung 21.7:
Erweiterte Konfiguration des
virtuellen Verzeichnisses

Sonderzeichen in einem URL

Einige Zeichen haben in einem URL oder in einem XML-Dokument eine besondere Bedeutung. Damit ein URL gültig ist, müssen Sie diese Zeichen in der entsprechenden Codierung (siehe Tabelle 21.1) angeben.

Wert	Bedeutung
%20	Leerzeichen (« »)
%23	Lesezeichen (#)
%25	Sonderzeichen (%)

Tabelle 21.1: Sonderzeichen in einem URL

Wert	Bedeutung
%26	Trennzeichen zwischen den Parametern in einem URL (&)
%2F	Verzeichnistrennzeichen (/)
%3F	Trennt die Abfrage vom URL (?)

Tabelle 21.1: Sonderzeichen in einem URL (Forts.)

Abfragen ausführen

Um eine Abfrage auszuführen, geben Sie einfach den URL mit Ihrem Servernamen ein, dann das virtuelle Verzeichnis, das Sie weiter oben in dieser Lektion eingerichtet haben, und anschließend die Abfrage (ein ?sql= und die eigentliche SQL-Abfrage). Schließlich endet der URL mit &root=ROOT (abhängig von der Groß-/Kleinschreibung). Geben Sie zum Beispiel den folgenden URL in Ihren Webbrowser ein, wobei Sie den Servernamen entsprechend ersetzen:



`http://ser/pubs?sql=Select+au_id,+au_lname,+au_fname+from+authors&root=ROOT`

Leider erhalten Sie dabei den in Abbildung 21.8 dargestellten Fehler.

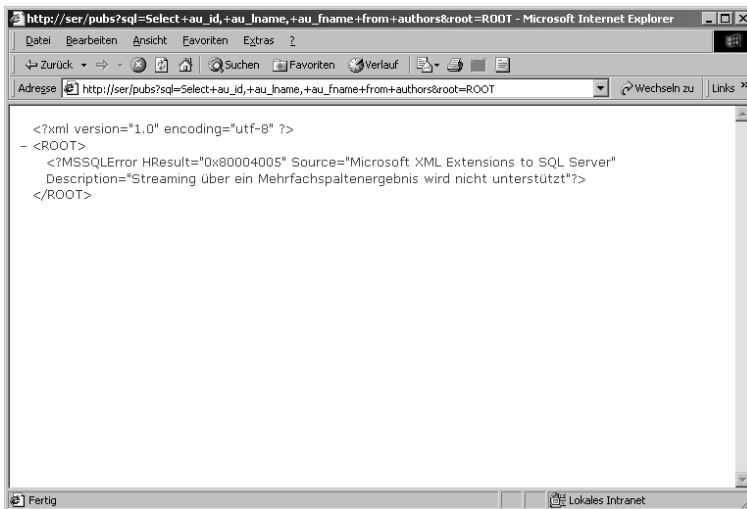


Abbildung 21.8:
XML-Fehler, wenn
die FOR XML-
Klausel nicht spezi-
fiziert .

Dieser Fehler kommt vom Webbrowser, der XML zurück erwartet und keine richtige SQL Server-Ergebnismenge. Deshalb müssen Sie SQL Server anweisen, XML zurückzugeben.

Wie das genau funktioniert, erfahren Sie in Kürze, für das Beispiel hängen Sie einfach `for xml raw` an die Abfrage an:



`http://ser/pubs?sql=Select+au_id,+au_lname,+au_fname+from+authors+for+xml+raw&root=ROOT`

Abbildung 21.9 zeigt das Ergebnis dieser Abfrage.

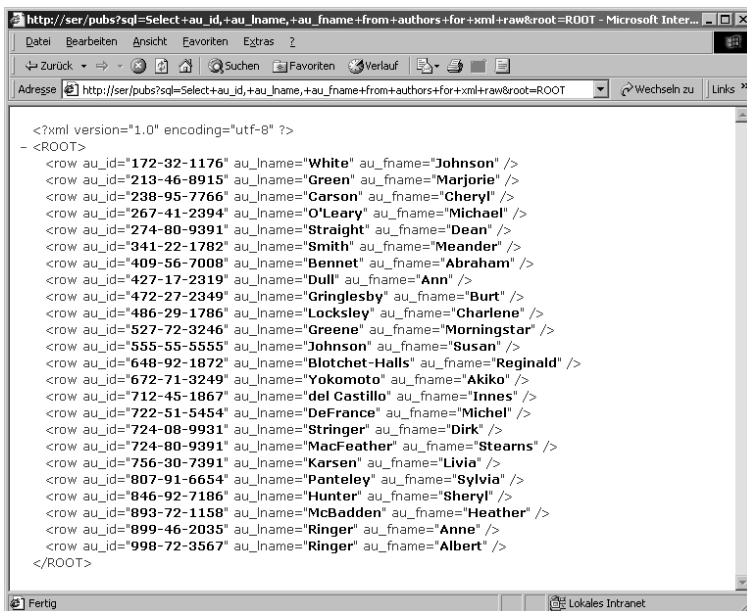


Abbildung 21.9:
Die erste Abfrage,
die XML an das
Internet zurückgibt.

Grundsätzlich können Sie jetzt jede gewünschte Abfrage an die Datenbank *pubs* abschicken. Probieren Sie die folgende Anweisung aus, um sich von den weitreichenden Möglichkeiten zu überzeugen:



`http://ser/pubs?sql=create+database+SQLXML&root=ROOT`

Dieses Mal erhalten Sie folgende Meldung zurück:



```
<?xml version="1.0" encoding="utf-8" ?>
<ROOT />
```

Sehen Sie sich nun Abbildung 21.10 (und Ihren eigenen Enterprise Manager) an. Da Sie für das virtuelle Verzeichnis die Windows-Authentifizierung zur Anmeldung bei SQL Server festgelegt haben, kann der jeweilige Benutzer alles unternehmen, was für das Verzeichnis zugelassen ist. Wenn Sie also (wie es für das Beispiel der Fall ist) ein Mitglied der Rolle *sysadmin* sind, haben Sie unbeschränkten Zugriff. Wie Abbildung 21.10 zeigt, wurde durch die obige URL-Abfrage eine Datenbank namens *SQLXML* erstellt. Wenn Sie also eine Datenbank erstellen können – stellen Sie einfach einmal vor, was alles noch möglich wäre!

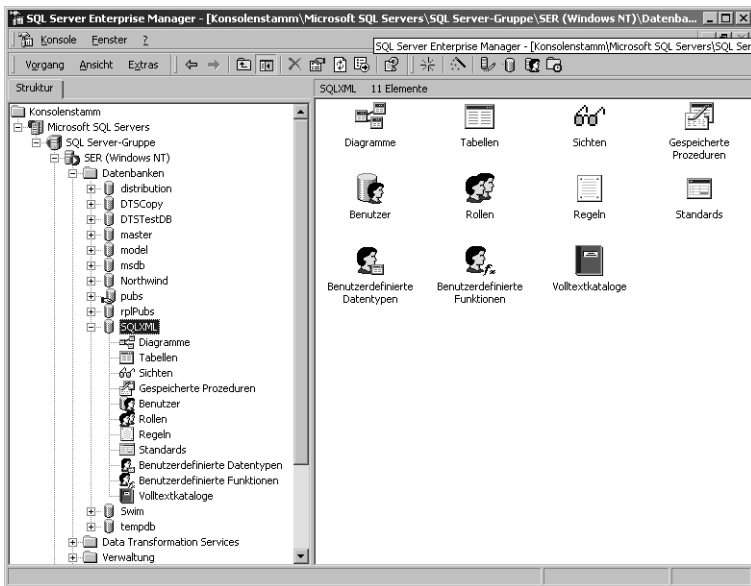


Abbildung 21.10:
Der Enterprise
Manager mit der
neu erstellten
Datenbank



Sind Sie immer noch nicht beunruhigt? Dann probieren Sie folgenden Befehl aus:

```
http://ser/pubs?sql=drop+database+SQLXML&root=ROOT
```

Jetzt sollten bei Ihnen aber die Warnleuchten angehen! Was kann man hier tun? Stellen Sie sicher, dass nur autorisierte Benutzer mit den richtigen Sicherheitskonten über diese Schnittstelle zugreifen können.

Gespeicherte Prozeduren ausführen

Solange Sie Transact-SQL-Abfragen abschicken können, lassen sich auch Aufrufe von gespeicherten Prozeduren realisieren. Das ist nach zwei Mechanismen möglich:

- Mit dem herkömmlichen Execute-Befehl, wie zum Beispiel:

Exec sp_helpdb

- Mit der ODBC-Aufrufsyntax, wie zum Beispiel:

{call+sp_helpdb}

Versuchen Sie nun, eine gespeicherte Prozedur zu erstellen und dann aufzurufen. Mit den folgenden Anweisungen erzeugen Sie erst einmal zwei gespeicherte Prozeduren, die Sie später aufrufen können:



```
Use Pubs
Go
Create proc testxml1
As
Select au_lname, au_fname from authors for xml auto
Return
Go
Create proc testxml2 (@lastname nvarchar(255))
As
Select au_lname, au_fname from authors
where au_lname = @lastname for xml auto
Return
GO
```

Rufen Sie als Erstes die gespeicherte Prozedur testxml1 auf. Diese Prozedur übernimmt keine Parameter:

<http://ser/pubs?sql=exec+testxml1&root=ROOT>

Abbildung 21.11 zeigt das Ergebnis.

Als Nächstes rufen Sie die gespeicherte Prozedur testxml2 auf, die als Parameter den Nachnamen (lastname) übernimmt. Wählen Sie einen Namen aus, den die Abfrage gemäß Abbildung 21.11 aufgelistet hat (im Beispiel Johnson White). Der URL sieht damit folgendermaßen aus:

<http://ser/pubs?sql=exec+testxml2+'White'&root=ROOT>

Damit erhalten Sie nur die Zeile für Johnson White zurück. Den Parameter können Sie auch nach dem Namen referenzieren:

<http://ser/pubs?sql=exec+testxml2+@lastname=+'White'&root=ROOT>

Beide Anweisungen liefern das gleiche Ergebnis zurück, wie es Abbildung 21.12 zeigt.

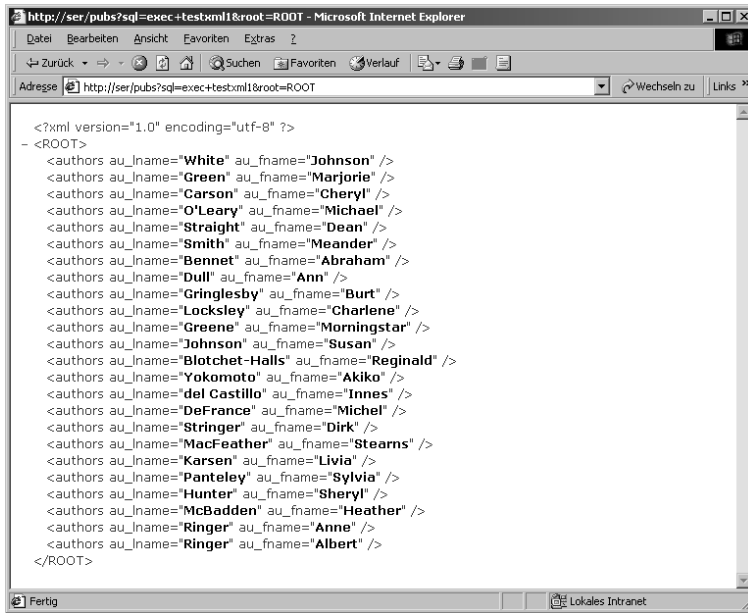


Abbildung 21.11:
Ergebnisse einer
gespeicherten Proze-
dur (keine Parame-
ter)

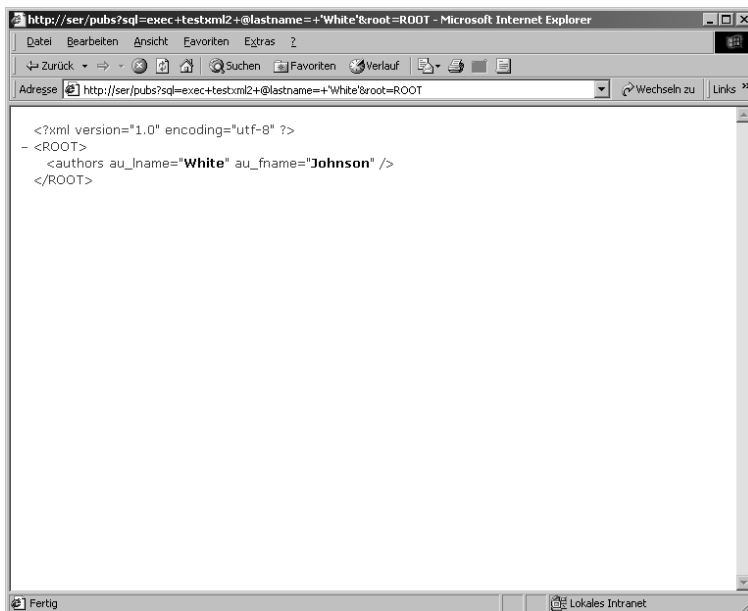


Abbildung 21.12:
Aufrufen einer
gespeicherten Proze-
dur mit einem Para-
meter

Wie man sieht, ist das Absenden von Abfragen über einen URL recht einfach und leistungsfähig – und zwar so leistungsfähig, dass man in der Tat die in einer Abfrage zulässigen Transact-SQL-Anweisungen beschränken muss. Für diese Aufgabe bieten sich Vorlagendateien an.

Vorlagendateien

Mit Vorlagendateien lassen sich die Möglichkeiten des Benutzers im Hinblick auf die Ausführung von Transact-SQL-Anweisungen einschränken. Man kann festlegen, dass die Benutzer nur die SQL-Anweisungen ausführen können, die in der Vorlagendatei codiert sind, oder man schränkt die Benutzer auf eine SQL-Anweisung mit der Übergabe von Parametern (einschließlich Aufrufen von gespeicherten Prozeduren) ein.

Die Funktionsweise dieser Einschränkungen lässt sich am besten an einigen Beispielen erläutern. Erstellen Sie zunächst mit dem Windows-Explorer ein Verzeichnis unter Ihrem virtuellen *pubs*-Verzeichnis. In der Beispielkonfiguration ist das der Verzeichnispfad *d:\inetpub\wwwroot\pubs\template*. Als Nächstes legen Sie eine einfache Vorlagendatei an und speichern Sie – im Beispiel unter dem Namen *authors.xml* – in diesem Ordner. Ihre erste Vorlagendatei hat folgendes Aussehen:

```
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql">
  <sql:query>
    SELECT au_lname, au_fname
    FROM authors
    FOR XML AUTO
  </sql:query>
</ROOT>
```

Jetzt müssen Sie das virtuelle Vorlagenverzeichnis im IIS Virtual Directory Management-Dienstprogramm aktivieren. Klicken Sie dazu mit der rechten Maustaste auf das virtuelle Verzeichnis *pubs*, wählen Sie EIGENSCHAFTEN und gehen Sie auf die Registerkarte VIRTUELLE NAMEN. Klicken Sie auf NEU, um das in Abbildung 21.13 gezeigte Dialogfeld zu öffnen. Füllen Sie dieses Dialogfeld entsprechend Abbildung 21.13 aus und beachten Sie dabei, dass Sie das Verzeichnis *template* verwenden, das Sie eben unter dem virtuellen Verzeichnis *pubs* angelegt haben.

Klicken Sie auf SPEICHERN und dann auf OK, um die Änderungen zu übernehmen. Starten Sie dann den Internet Explorer, um Ihre Website zu betrachten (im Beispiel *http://ser/pubs/template/authors.xml*). Abbildung 21.14 zeigt diese Site.

Wie sieht es nun mit der Übergabe von Parametern aus? Das folgende Beispiel zeigt eine andere Vorlagendatei, die jetzt einen Aufruf der weiter oben angelegten gespeicherten Prozedur *testxml2* enthält. Dieser Prozedur ist ein Parameter zu übergeben.

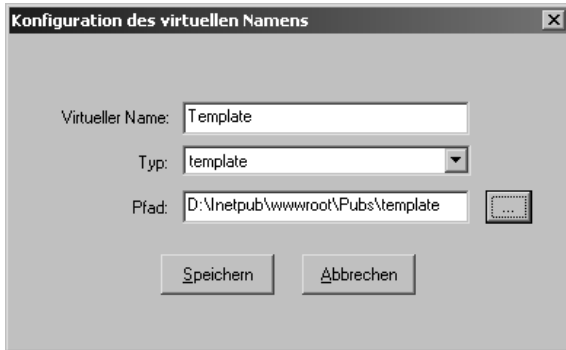


Abbildung 21.13:
Eine neue virtuelle Konfiguration erstellen

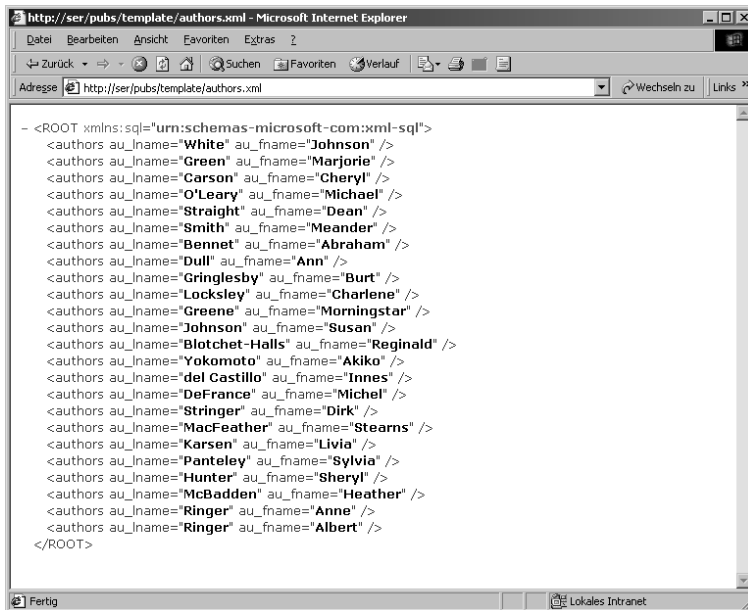


Abbildung 21.14:
Ergebnisse von der
XML-Vorlagendatei

Die folgende XML-Datei (im Beispiel *sp.xml* genannt) übernimmt den Parameter, den der Benutzer angegeben hat, oder legt den Standardwert *White* für den Nachnamen fest:

```
<ROOT xmlns:sql='urn:schemas-microsoft-com:xml-sql'>
  <sql:header>
    <sql:param name='LastName'>White</sql:param>
  </sql:header>
  <sql:query>
    exec testxml2 @LastName
  </sql:query>
</ROOT>
```

Diese neue XML-Vorlagendatei (*sp.xml*) lässt sich nach zwei Methoden aufrufen. Erstens können Sie

`http://ser/pubs/template/sp.xml`

ausführen und erhalten die Ergebnisse für den Standardwert (White) zurück. Bei der zweiten Methode übergeben Sie den Parameter als Teil des URL. Probieren Sie beispielsweise den Aufruf

`http://ser/pubs/template/sp.xml?LastName=Bennet`

aus. Abbildung 21.15 zeigt die Ergebnisse für den Autor mit dem Nachnamen Bennet.

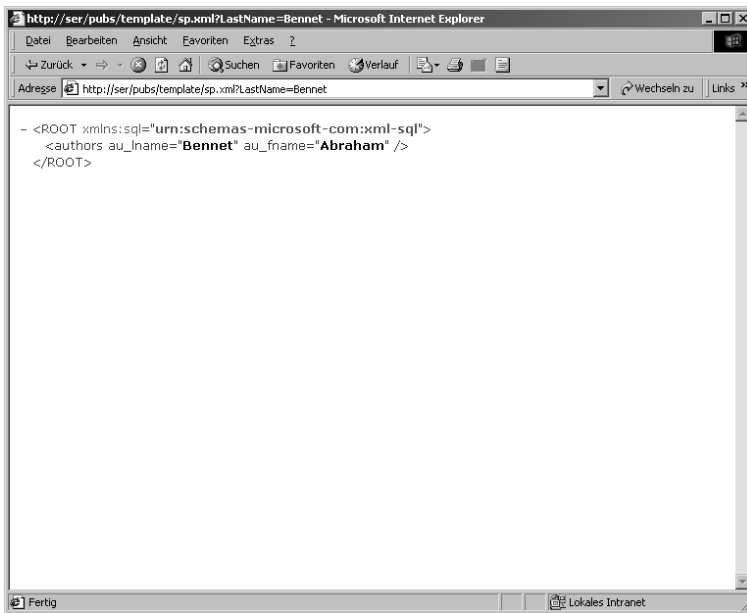


Abbildung 21.15:
Ergebnisse für den
Aufruf der XML-
Vorlagendatei mit
einem Parameter

Wie diese Beispiele zeigen, ist der Aufruf von Vorlagendateien sehr einfach (und erfordert wesentlich weniger Tipparbeit). Vor allem aber ist diese Lösung bereits vom Konzept her sicherer, weil Sie die zulässigen Transact-SQL-Anweisungen von vornherein spezifizieren können.

Sowohl XPATH als auch POST – d.h. die beiden anderen Möglichkeiten, die der Abschnitt zu den Optionen beschrieben hat – gehen über das Ziel dieses Buches hinaus. Deshalb sei an dieser Stelle auf die Onlinedokumentation von SQL Server verwiesen.

21.3 XML in SQL Server lesen und schreiben

Mittlerweile wissen Sie, wie Sie Daten per XML über Ihren Webbrowser aus SQL Server erhalten können. Jetzt untersuchen wir die andere Richtung. Das Datenbankmodul von SQL Server ist erweitert worden und unterstützt jetzt XML direkt von Transact-SQL. Diese Fähigkeiten haben Sie weiter vorn in der heutigen Lektion schon bei Abfragen benutzt. Wenn Sie eine `FOR XML`-Klausel an die `SELECT`-Anweisungen anhängen, weisen Sie das SQL Server-Modul an, die Ergebnisse als XML formatiert zurückzugeben. Dabei lässt sich auch genau festlegen, wie Sie diese Ergebnisse formatiert haben möchten. Mit diesen Optionen beschäftigen sich die folgenden Abschnitte.

Gültige `SELECT`-Anweisungen

Leider gibt es einige Einschränkungen hinsichtlich der Arten von `SELECT`-Anweisungen, für die Sie eine `FOR XML`-Klausel angeben können:

- Innerhalb einer Transact-SQL-Unterabfrage (eine `SELECT`-Anweisung in einer anderen `SELECT`-Anweisung) kann man kein `FOR XML` angeben.
- Die `SELECT`-Anweisung muss eine echte Ergebnismenge zurückgeben und Optionen, die keine »normalen« relationalen Rowsets liefern, werden nicht unterstützt. Dazu gehört beispielsweise die `COMPUTE BY`-Klausel.
- Aggregationen wie `GROUP BY` werden nicht unterstützt, wenn Sie `FOR XML AUTO` spezifizieren.
- In einer `CREATE VIEW`-Anweisung ist kein `FOR XML` möglich.
- Im Auswahlteil einer `INSERT SELECT`-Anweisung ist kein `FOR XML` möglich.
- Die Auswahlanweisung kann keine Spalte per `CONVERT` oder `CAST` in die Datentypen `image` oder `text` konvertieren.
- Einige gültige SQL Server-Spaltennamen sind für XML ungültig und werden bei der Rückgabe der Ergebnisse übersetzt. Einzelheiten hierzu entnehmen Sie bitte der Onlinedokumentation. Wenn Sie sich aber an die normalen Buchstaben halten, liegen Sie auf der richtigen Seite.

Grundsätzlich lassen sich diese Einschränkungen auf zwei Regeln komprimieren:

- Die Daten müssen SQL Server verlassen.
- Es muss sich um eine geeignete relationale Ergebnismenge handeln.

Die FOR XML-Klausel

Bei den meisten gültigen Transact-SQL-Anweisungen können Sie die Klausel `FOR XML` anhängen. Die Syntax sieht wie folgt aus:



`FOR XML mode [, XMLDATA][, ELEMENTS][, BINARY Base64]`

Als XML-Modus (`mode`) sind die Werte `RAW`, `AUTO` und `EXPLICIT` möglich.

FOR XML RAW

Die Option `FOR XML RAW` wandelt jede Zeile der Ergebnismenge in ein XML-Dokument mit dem Kennzeichner `row` um. Jede Spalte ist mit einem Attributnamen des Transact-SQL-Spaltennamens versehen. Alle Zeilen werden als geschlossene Datenmenge an den Client zurückgegeben. Abbildung 21.16 zeigt den Query Analyzer bei Ausführung der folgenden Abfrage:

```
select au_lname from authors FOR XML RAW
```

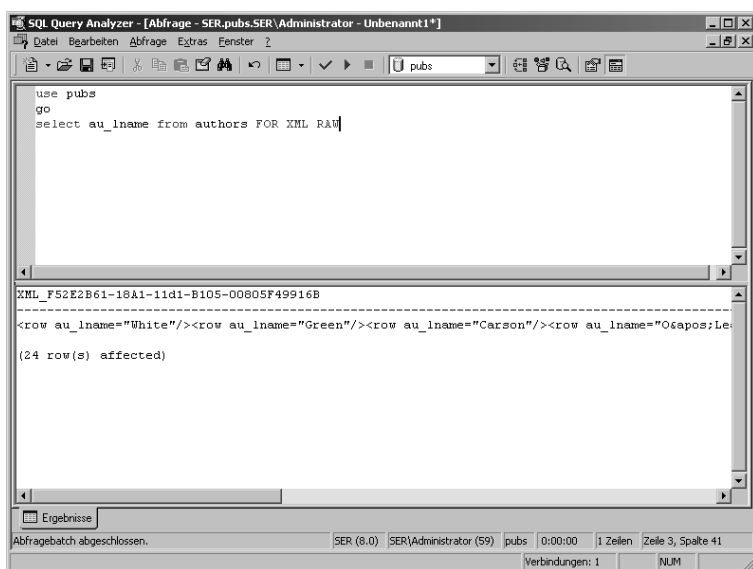


Abbildung 21.16:
Ergebnisse der
`SELECT...FOR
XML RAW`-Anwei-
sung

Führen Sie nun die gleiche Anweisung aus, aber fügen Sie die Option `XMLDATA` an. Diese Option gibt zusammen mit den Daten zusätzliche Informationen zurück, die das Schema beschreiben (man kann sich das als Informationen zum Erstellen einer Tabelle vorstellen).

```
select au_lname from authors FOR XML RAW, XMLDATA
```

Abbildung 21.17 zeigt das Ergebnis dieser Abfrage im Query Analyzer.

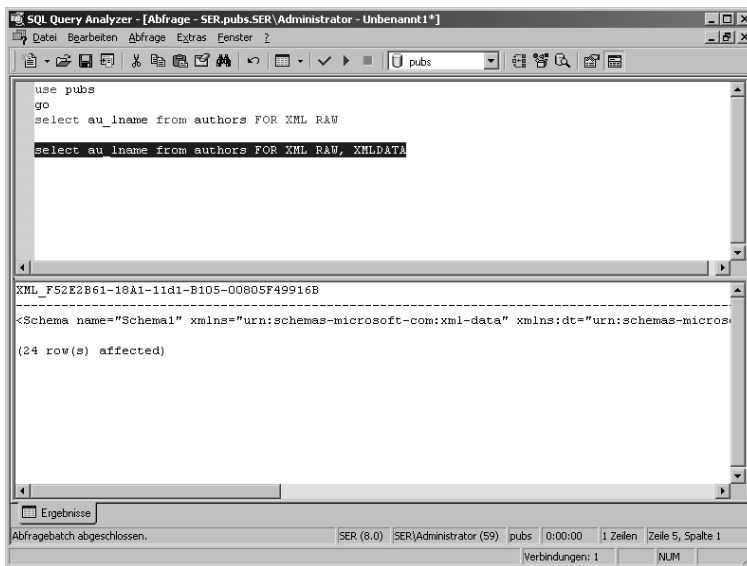


Abbildung 21.17:
Ergebnisse der
SELECT...FOR
XML RAW, XML-
DATA-Anweisung



Wenn Sie die Klausel `FOR XML RAW` mit der Option `XMLDATA` angeben, werden die Ergebnisse möglicherweise abgeschnitten. Um die vollständigen Ergebnisse zu sehen, wählen Sie im Query Analyzer **EXTRAS / OPTIONEN**, gehen auf die Registerkarte **ERGEBNISSE** und ändern den Wert im Feld **MAXIMALE ZEICHENANZAHL PRO SPALTE** auf etwa 1000.

FOR XML AUTO

Die Option `AUTO` ist der Option `RAW` ähnlich, liefert aber mehr Informationen über die zurückgegebenen Daten. Außerdem gibt der `AUTO`-Modus jede Tabelle aus der Abfrage als geschachteltes XML-Element zurück. Am besten ist es, wenn Sie sich die Ergebnisse einer Abfrage mit dieser Option am praktischen Beispiel ansehen. Führen Sie dazu die gleiche Abfrage wie bisher aus, dieses Mal aber mit der Option `AUTO`:

```
select au_lname from authors FOR XML AUTO
```

Abbildung 21.18 zeigt die Ergebnisse.

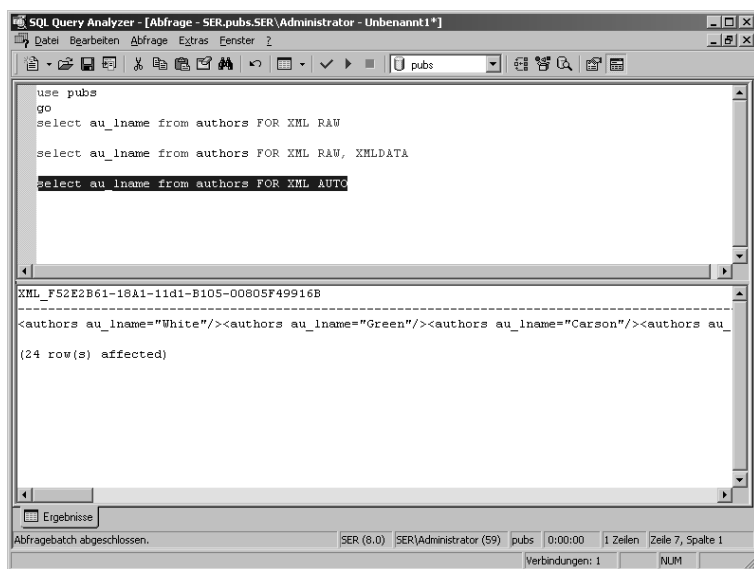


Abbildung 21.18:
Ergebnisse von der
Anweisung
SELECT...FOR
XML AUTO

Den Unterschied zeigt aber erst eine Abfrage, die sich über mehrere Tabellen erstreckt. Nehmen Sie als Beispiel die folgende Anweisung:



```
select au_lname, title, royaltyper
from authors inner join titleauthor on authors.au_id =
titleauthor.au_id
inner join titles on titleauthor.title_id = titles.title_id
FOR XML AUTO
```

Abbildung 21.19 gibt die Ergebnisse dieser Anweisung an.

Die Ergebnisse sind qualifiziert – d.h. die Elemente enthalten den Namen der Tabelle, von der das Ergebnis stammt. Außerdem sind die Ergebnisse verschachtelt und enthalten die Spaltennamen sowie die Daten. Da Query Analyzer die Ergebnisse überhaupt nicht formatiert, gibt Abbildung 21.20 eine im Editor überarbeitete Version an. Hier zeigt sich deutlich, dass die Ergebnisse hierarchisch aufgebaut sind und welche Aufgabe die Option AUTO erfüllt.

Bei der Option FOR XML AUTO können Sie die zusätzliche Option ELEMENTS spezifizieren:



```
select au_lname, title, royaltyper
from authors inner join titleauthor on authors.au_id =
titleauthor.au_id
inner join titles on titleauthor.title_id = titles.title_id
FOR XML AUTO, ELEMENTS
```

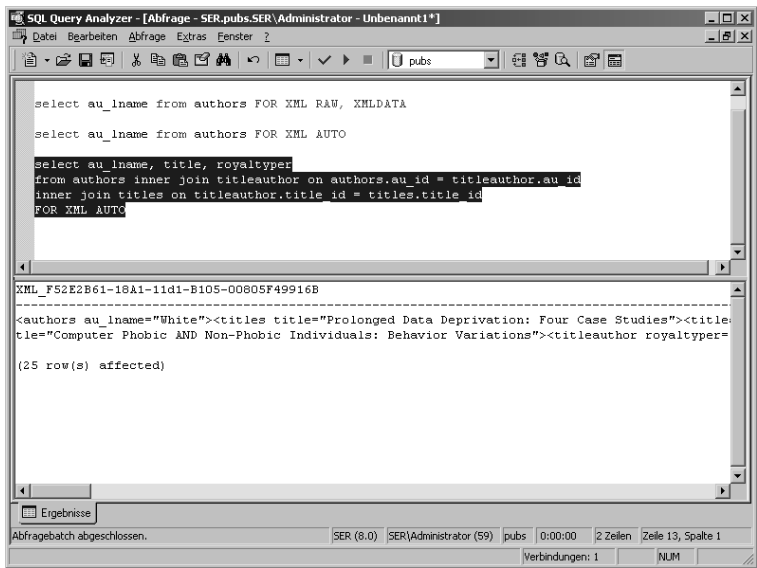


Abbildung 21.19:
Ergebnisse aus der
Verknüpfung mehrerer
Tabellen für die
FOR XML AUTO-
Anweisung

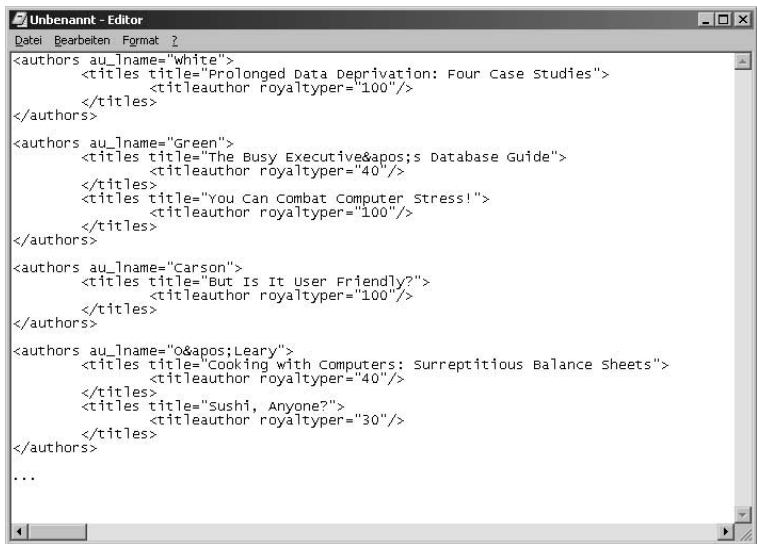


Abbildung 21.20:
Die gleichen Ergeb-
nisse im Editor
angezeigt und for-
mattiert

Wenn Sie diese Option angeben, wird jeder Spaltentitel als separates XML-Element dargestellt. Abbildung 21.21 zeigt, wie sich diese kleine Änderung in den Ergebnissen bemerkbar macht.

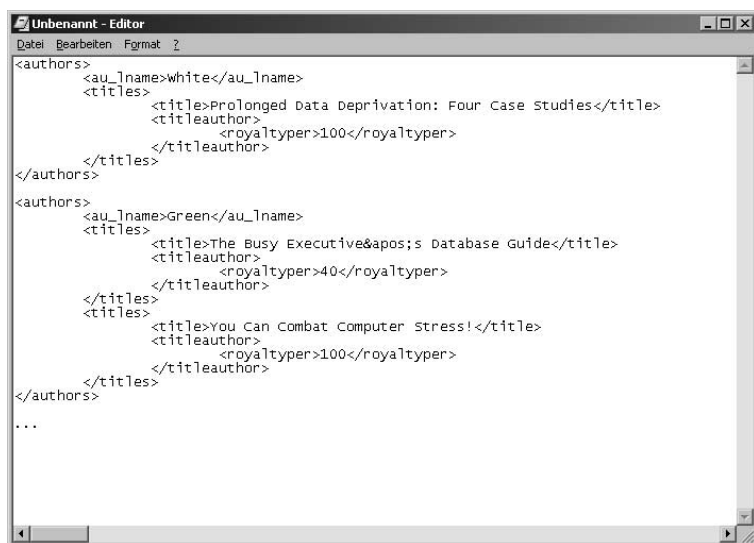


Abbildung 21.21:
Die gleiche Abfrage
mit der zusätzlichen
Option ELE-
MENTS

FOR XML EXPLICIT

Schließlich können Sie die Option `FOR XML EXPLICIT` angeben und damit genau festlegen, in welcher Weise das aus der Abfrage zurückgegebene XML formatiert ist. Diese Option ist zwar die leistungsfähigste, birgt aber auch viele Stolpersteine. Sie müssen SQL Server genau mitteilen, wie die XML-Ergebnisse konfiguriert sein sollen. Außerdem sind Sie als Verfasser der Abfrage dafür verantwortlich, dass die zurückgegebenen Ergebnisse gültiges XML darstellen.

Prinzipiell fällt das in den Aufgabenbereich der »richtigen« Programmierer. Die in diesem Buch vermittelten Kenntnisse reichen aber bei Weitem nicht aus, dass Sie sich in diese Arena wagen können. Auch wenn Sie weitere Informationen in der Onlinedokumentation von SQL Server finden, sind eine Menge Erfahrung und viel Experimentierfreude gefragt, bis Sie die gewünschten Ergebnisse in Perfektion realisieren können.

Die Option XMLDATA

Wie Sie beim Modus `FOR XML RAW` gesehen haben, erhalten Sie zusätzlich zum eigentlichen Rowset noch eine Menge Zusatzinformationen zurück, wenn Sie die Option `XMLDATA` angeben. Diese Option liefert Schemainformationen, die über die normalerweise von XML zurückgegebenen Informationen hinausgehen. Microsoft empfiehlt, dass Sie auf diese Option verzichten, weil sie Ihren Server bremst und es sich bei den meisten zurückkommenden XML-Daten ohnehin um einfache Textzeichenfolgen handelt.

21.4 Zusammenfassung

Die heutige Lektion hat Sie mit der XML-Unterstützung sowohl über Internet Information Server als auch direkt in Transact-SQL bekannt gemacht. Beide Optionen haben ihre Berechtigung. Sie können Benutzern erlauben, Abfragen als Teil ihrer URL-Zeichenfolgen zu formulieren, oder Sie wählen die sichere Option und erlauben den Benutzern die Abfrage Ihrer Datenbank ausschließlich über XML-Vorlagendateien.

In SQL Server 2000 sind weitere komplexere Features der XML-Unterstützung verfügbar und das XML-Team plant die Freigabe zusätzlicher Merkmale nach der Freigabe von SQL Server 2000. Es empfiehlt sich, regelmäßig unter <http://www.microsoft.com/sql> nach Aktualisierungen für die XML-Fähigkeiten von SQL Server 2000 nachzusehen.

21.5 Workshop

Die Kontrollfragen im Workshop sollen Ihnen helfen, die neu erworbenen Kenntnisse zu den behandelten Themen zu festigen. Die Übungen geben Ihnen die Möglichkeit, praktische Erfahrungen mit dem gelernten Stoff zu sammeln. Die Antworten finden Sie in Anhang A. Lösen Sie alle Aufgaben und versuchen Sie, die Antworten zu verstehen, bevor Sie zur nächsten Lektion weitergehen.

Fragen & Antworten

- F** *Mit welchem Tool richte ich die XML-Abfragefähigkeiten für SQL Server 2000 ein?*
- A** Mit dem IIS Virtual Directory Management-Dienstprogramm.
- F** *Was erledigt die FOR XML-Klausel für mich?*
- A** Sie generiert automatisch XML-Tags, die für Ihre Daten geeignet sind.
- F** *Kann ich automatisch Abfragen über einen URL an SQL Server schicken?*
- A** Nein. Zuerst müssen Sie die Unterstützung für diese Arten von Abfragen im IIS Virtual Directory Management-Dienstprogramm aktivieren.

Kontrollfragen

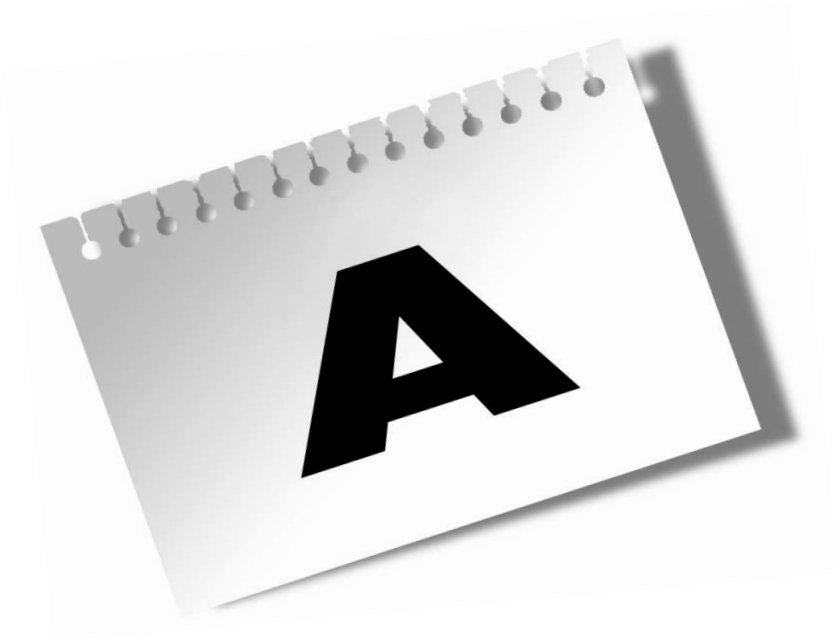
1. Ist die folgende URL-Abfrage gültig?

`http://ser/pubs?SQL=select city from authors for XML AUTO&root=ROOT`

2. Welche Arten von Anweisungen sind über URL-Abfragen erlaubt?
3. Wie unterscheidet sich die `FOR XML RAW`-Option von `FOR XML AUTO`?

Übungen

1. Schreiben Sie XML-Vorlagendateien, um gespeicherte Prozeduren und Abfragen aufzurufen, wobei Sie Parameter spezifizieren. Das ist höchstwahrscheinlich die Methode, nach der Sie XML den Zugriff auf SQL Server erlauben.
2. Fügen Sie ein neues virtuelles Verzeichnis in das Dienstprogramm für virtuelle Verzeichnisse hinzu und zeigen Sie auf eine Systemdatenbank, wie zum Beispiel *master*. Probieren Sie Abfragen bezüglich der `INFORMATION_SCHEMA`-Ansichten aus und stellen Sie sich dann vor, wie Sie sie verwenden können, um Berichte über Ihr Datenbank-schema auf einer Webseite zu verwalten.



Antworten



A.1 Antworten für Tag 1

Antworten zu den Kontrollfragen

1. Die Tabelle
2. Tabellen, Spalten, Datentypen, gespeicherte Prozeduren, Trigger, Regeln, Schlüssel, Einschränkungen, Standardwerte und Indizes
3. Üblicherweise der SQL Server-Administrator

Lösung zu den Übungen

Die hier angegebene Lösung ist eine Möglichkeit, um Objekte zu erstellen und sie in Tabellen und Spalten zu organisieren. Tag 9 geht näher auf die verschiedenen Datentypen ein.

Normalisieren Ihres Datenbankentwurfs

Beim Normalisieren der Datenbank gruppiert man verwandte Objekte in Tabellen.

Betrachten Sie beispielsweise die Datenbank des Gebrauchtwarenhandels wie die Tabellenblätter einer Tabellenkalkulation. Die Spalten wären Fahrgestellnummer, Hersteller, Modell, Farbe, Kilometerstand, Foto, Klimaanlage, Motorengröße, Getriebe, Allradantrieb, Leistungsdrosselungen, andere Merkmale, Preis, Listenpreis, geforderter Preis, Kaufdatum, Verkaufsdatum, Anzahlung, Anzahlungsdatum, Kundenname, Kundenadresse, Kundentelefonnummer, Verkäufername, Verkäuferadresse, Verkäufer-Provision und Verkäuferbezeichner.

Für jedes Auto müssen Sie insgesamt 25 Spalten speichern. Für ein kleines Geschäft mag die Tabellenkalkulation vollkommen ausreichend sein, bei größeren Unternehmen ist sie dagegen überfordert. Sie brauchen einen eindeutigen Bezeichner, um diese Tabellen miteinander zu verknüpfen. Fahrzeuge haben natürlich einen einzigartigen Bezeichner in ihrer Fahrgestellnummer. Sie können Kunden und Verkaufspersonal unverwechselbare Nummern zuweisen oder bestehende Bezeichner (wie Sozialversicherungsnummer oder Nachname + Adresse) verwenden. In diesem Fall bieten sich zugewiesene Kennnummern (IDs) für das Verkaufspersonal und die Sozialversicherungsnummer des Kunden als Bezeichner an.

Sie können sehen, dass Informationen über Fahrzeuge, Verkaufspersonal, Kosten und Verkäufe sowie Kunden verwandte Daten sind. Diese Variablen können dementsprechend in mehrere Tabellen eingehen:



- **Fahrzeug** – Fahrgestellnummer, Hersteller, Modell, Farbe, Kilometerstand, Foto, Klimaanlage, Motorgröße, Getriebe, Allradantrieb, Leistungsdrosselungen, andere Merkmale
- **Verkaufspersonal** – Verkäufername, Verkäuferadressen, Verkäufer-Provision, Verkäuferbezeichner
- **Verkaufsdaten** – Fahrgestellnummer, Kosten, Listenpreis, geforderter Preis, Kaufdatum, Verkaufsdatum, Anzahlung, Anzahlungsdatum, Kundenbezeichner, Verkäuferbezeichner
- **Kunde** – Kundenname, Kundenadresse, Kundentelefon, Kundenbezeichner, Kommentare

Fertig stellen des Datenbankentwurfs

Um den Entwurf fertig zu stellen, wählen Sie Spaltennamen, Datentyp und Länge jeder Spalte (wo es angebracht ist). Teilen Sie auch Namen in Vor- und Nachnamen auf, weil Sie sie möglicherweise später alphabetisch ordnen wollen.

Der Detailentwurf einer Datenbank wird auch als *Systemkatalog* bezeichnet, weil er die Variablen und Tabellen in der Datenbank definiert. Der Systemkatalog für die als Beispiel angegebene Datenbank *Fahrzeughandel* kann beispielsweise wie folgt gegliedert sein:

Fahrzeug: Diese Tabelle enthält Fahrzeuginformationen und hat die folgenden Variablen:

- *Fahrgestellnummer* (Primärschlüssel) ist die Identifizierungsnummer des Fahrzeugs. Dieses Zeichenfeld hat eine variable Länge von 30 Zeichen (auch für Fahrzeuge, die vom Standard abweichen usw.).
- *Hersteller* gibt den Hersteller des Autos an. Gültige Werte sind z.B. Ford oder Nissan. Dieses Zeichenfeld hat eine variable Länge von 30 Zeichen.
- *Modell* ist das Modell des Autos. Gültige Werte sind z.B. Sharan oder Escort. Dieses Zeichenfeld hat eine variable Länge von 30 Zeichen.
- *Farbe* ist die Farbe des Autos. Dieses Zeichenfeld hat eine variable Länge von 20 Zeichen.
- *Kilometerstand* ist der Kilometerstand des Autos. Dieses numerische Feld ist 7 Zeichen lang (maximal 9 999 999 – den Kilometerstand messen Sie nicht auf ein Zehntel Kilometer genau).
- *Foto* ist ein Foto des Autos als JPEG-Bild mit einer Auflösung von 640 * 480 * 256, damit es sich leicht im Internet veröffentlichen lässt. Es handelt sich um ein image-Feld.
- *KA* gibt an, ob eine Klimaanlage vorhanden ist. Gültige Antworten sind J oder N. Dieses Zeichenfeld ist ein Zeichen lang.



- *Motorgrösse* zeigt an, wie viele Zylinder der Motor hat. Gültige Werte sind hier 3, 4, 6, 8, 10 und 12. Das Feld hat den Datentyp `tinyint` (kleine Ganzzahl – Bereich von 0 bis 255).
- *Getriebe* ist der Getriebetyp, wobei gültige Werte »Standard« oder »Automatik« sind. Dieses variable Zeichenfeld hat eine maximale Länge von 9 Zeichen.
- *ARA* gibt an, ob das Fahrzeug einen Allradantrieb hat. Gültige Werte sind J oder N. Dieses Zeichenfeld ist ein Zeichen lang.
- *LD* gibt an, ob das Fahrzeug in der Leistung gedrosselt ist. Gültige Werte sind J oder N. Dieses Zeichenfeld ist ein Zeichen lang.
- *Kommentar* wird verwendet, um Bemerkungen zum Fahrzeug anzugeben. Das Feld ist eine Textvariable.

tblVerkaufspersonal: Diese Tabelle enthält Informationen über das Verkaufspersonal:

- *Verk_Bezeichner* (Primärschlüssel) ist der Bezeichner des Verkäufers. Jedem Mitglied des Verkaufspersonals wird ein Bezeichner zugewiesen, um beispielsweise Verkäufe zurückverfolgen zu können. Das Feld hat den Datentyp `smallint` (kleine Ganzzahl – Bereich von -32.768 bis 32.767).
- *Vorname* ist der Vorname des Verkäufers. Dieses variable Zeichenfeld hat maximal 30 Zeichen.
- *Nachname* ist der Nachname des Verkäufers. Dieses variable Zeichenfeld hat maximal 30 Zeichen.
- *Addr* ist die Adresse des Verkäufers. Dieses variable Zeichenfeld hat maximal 30 Zeichen.
- *Stadt* ist die Stadt, in der der Verkäufer wohnt. Dieses variable Zeichenfeld hat maximal 30 Zeichen.
- *Staat* ist der Bundesstaat, in dem der Verkäufer wohnt. Dieses Zeichenfeld hat 2 Zeichen.
- *PLZ* ist die Postleitzahl für die Stadt des Verkäufers. Es ist ein Zeichenfeld mit 5 Zeichen.
- *Provision* ist der Grundprozentwert des Verkäufers für seine Provision. Dies kann später verwendet werden, um Provisionen bezogen auf die Gewinne an verkauften Autos zu berechnen. Dies ist eine `tinyint`-Variable.



tblVerkaufe: Diese Tabelle zeichnet Informationen über Einkauf und Verkauf des Fahrzeugs auf.

- *Rechnung* (Primärschlüssel) ist die Rechnungsnummer des Originaleinkaufs des Autos. Dies ist ein `varchar`-Feld (variables Zeichenfeld) mit einer Maximallänge von 20 Zeichen.
- *Fahrgestellnummer* (Fremdschlüssel) ist das gleiche Feld wie das erste Feld in der Fahrzeugtabelle und wird verwendet, um einen Schlüssel für diese Tabelle zu erzeugen.
- *Kundenbezeichner* (Fremdschlüssel) gibt es auch in der Kundentabelle. Dieses Feld ist ein Verweis auf die Kundentabelle.
- *Verk_Bezeichner* (Fremdschlüssel) gibt es auch in der Tabelle *Verkaufspersonal*. Dies ist ein Verweis auf die Verkaufspersonaltabelle.
- *Preis* ist der tatsächliche Preis des Fahrzeugs. Dieses `smallmoney`-Feld kann bis zu 214.000 DM/EURO usw. handhaben).
- *Liste* ist der Listenpreis für das Auto. Dieses Feld hat den Datentyp `smallmoney`.
- *Forderung* ist der geforderte Preis. Dieser kann sich aufgrund eines Ausverkaufs, einer Anzeige oder durch einen Verkaufsanreiz ändern. Dieses Feld hat den Datentyp `smallmoney`.
- *Eingangsdatum* ist das Datum, an dem der Händler das Fahrzeug gekauft hat. Dieses Feld hat den Datentyp `smalldatetime` (hält Datumsangaben bis zu Intervallen von einer Minute fest).
- *Ausgangsdatum* ist das Datum, an dem das Fahrzeug einem Kunden verkauft worden ist. Dieses Feld hat den Datentyp `smalldatetime`.
- *Anzahlung* ist die Höhe einer Anzahlung (falls eine erfolgt ist), die ein Kunde für ein Auto getätigt hat. Dieses Feld hat den Datentyp `smallmoney`.
- *Anzahlungsdatum* gibt an, wann die Anzahlung erfolgt ist. Dieses Feld hat den Datentyp `smalldatetime`.

tblKunden: Diese Tabelle behält die Übersicht über die Kunden.

- *Kundenbezeichner* (Primärschlüssel) enthält die Sozialversicherungsnummer des Kunden, die auch die Doppelfunktion als Bezeichner für den Kunden hat. Es ist ein Zeichenfeld für 9 Ziffern.
- *Vorname* ist der Vorname des Kunden. Es ist ein variables Zeichenfeld mit maximal 30 Zeichen.
- *Nachname* ist der Nachname des Kunden. Es ist ein variables Zeichenfeld mit maximal 30 Zeichen.



- *Adresse* ist die Adresse des Kunden. Es ist ein variables Zeichenfeld mit maximal 30 Zeichen.
- *Stadt* ist die Stadt, in der der Kunde wohnt. Es ist ein variables Zeichenfeld mit maximal 30 Zeichen.
- *Staat* ist der Bundesstaat, in dem der Kunde wohnt. Es ist ein Zeichenfeld mit 2 Zeichen.
- *PLZ* ist die Postleitzahl des Kunden. Es ist ein Zeichenfeld mit 5 Zeichen.
- *Telefon* ist die Telefonnummer des Kunden. Es ist ein Zeichenfeld für 10 Ziffern.
- *Kommentar* ist ein Textfeld, in das Bemerkungen zum Kunden eingegeben werden können.

Abbildung A.1 zeigt, wie die Tabellen und deren Beziehungen aussehen können.

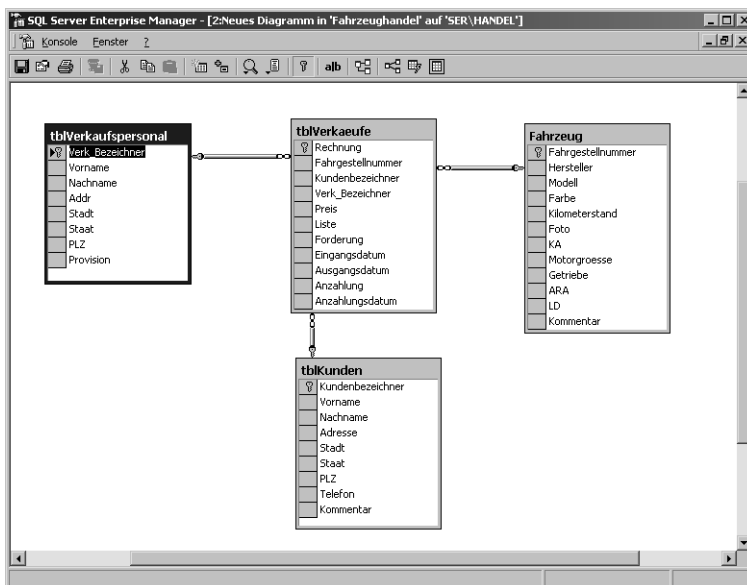


Abbildung A.1:
Beziehungen
zwischen den
Tabellen

Zum Lieferumfang von SQL Server gehören zwei Standarddatenbanken, mit denen Sie sich in das Produkt SQL Server einarbeiten können. Leser, die SQL Server bereits in der Vergangenheit eingesetzt haben, werden die Datenbank *pubs* kennen. Diese Datenbank ist eine Zusammenstellung von Tabellen, die Bücher, Verleger, Autoren, Buchhandlungen und Verkäufe verfolgt. Abbildung A.2 gibt eine Beispielansicht der Tabellen in der *pubs*-Datenbank und deren Beziehungen wieder.

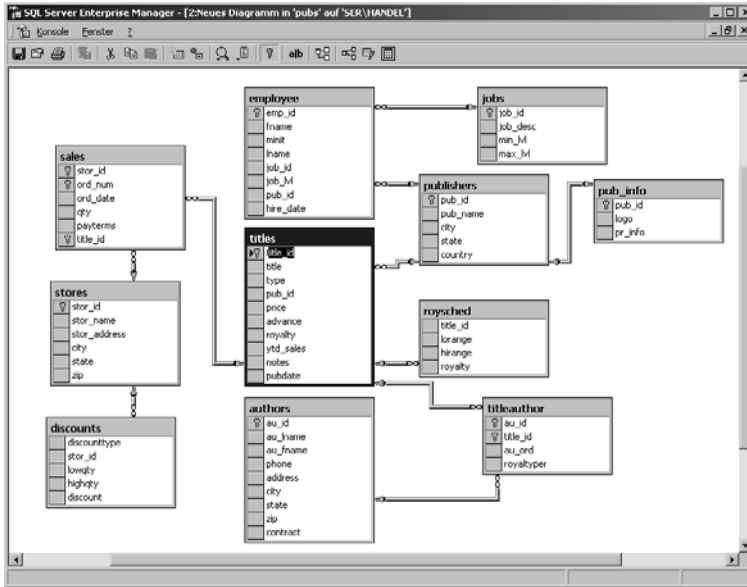


Abbildung A.2:
Datenbankdia-
gramm für die Bei-
spielsdatenbank pubs

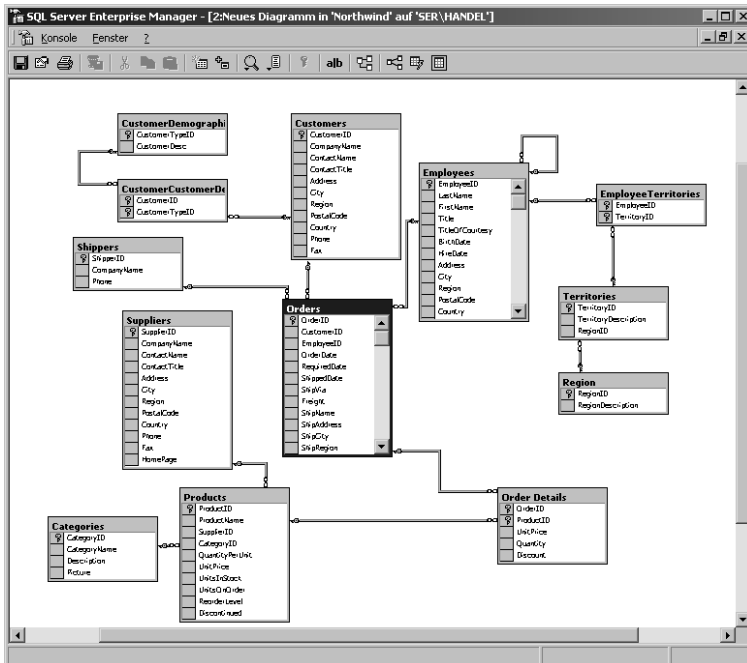


Abbildung A.3:
Datenbankdia-
gramm für die Bei-
spielsdatenbank Nor-
thwind



Leser, die mit der Datenbank *Northwind* von Microsoft Access vertraut sind, finden eine SQL Server-Version dieser Datenbank als Beispiel, das sie zur Einarbeitung nutzen können. Northwinds Traders ist eine fiktive Firma, die exotische Nahrungsmittel aus allen Teilen der Welt kauft und verkauft. Abbildung A.3 zeigt ein Beispiel-Layout der *Northwind*-Datenbank.

A.2 Antworten für Tag 2

Antworten zu den Kontrollfragen

1. Die binäre Sortierreihenfolge gibt Daten in ASCII-Reihenfolge zurück.
2. Nur Unicode-Datenspalten in SQL Server 2000.
3. Die MDACs (Microsoft Data Access Components, Microsoft-Datenzugriffskomponenten).
4. Sie sollten ein spezielles Benutzerkonto einrichten, das Mitglied der Administratorengruppe auf dem lokalen Rechner ist.
5. Verwenden Sie den SQL Server-Dienst-Manager, den SQL Server Enterprise Manager oder Dienstprogramme von Windows 2000. Sie können auch den Befehl NET START MSSQLServer ausführen.
6. Das Dateisystem NTFS. Es bietet die größte Sicherheit und Zuverlässigkeit.
7. Wählen Sie während einer benutzerdefinierten Installation unter *Serverkomponenten* den Eintrag *Volltextindexierung*.

A.3 Antworten für Tag 3

Antworten zu den Kontrollfragen

1. Konfigurieren Sie die Registrierungsinformation so, dass sie die Anmeldeinformationen standardmäßig nicht speichern, und fordern Sie für jede Verbindung eine separate Anmeldung.
2. Im Dienstprogramm SQL Server-Clientkonfiguration.
3. Im Dienstprogramm SQL Server-Netzwerkkonfiguration.



4. Systemmonitor, SQL Server Profiler und SQL Server Query Analyzer.
5. *SQLDiag.exe*.

A.4 Antworten für Tag 4

Antworten zu den Kontrollfragen

1.

```
CREATE DATABASE Accounting
ON
(NAME = Accounting_Data,
FILENAME = 'F:\Programme\Microsoft SQL Server\mssql
\data\Accounting_Data.mdf',
SIZE = 20,
FILEGROWTH = 2,
MAXSIZE = 40)
LOG ON
(NAME = Accounting_Log,
FILENAME = 'F:\Programme\Microsoft SQL Server\mssql
\data\Accounting_Log.ldf',
SIZE=5,
FILEGROWTH = 1,
MAXSIZE=10)
```

2. Dies fügt eine neue Datendatei namens *TestData2* mit einem Dateinamen *TestData2.ndf* im Standardverzeichnis hinzu. Die Anfangsgröße ist 10 Mbyte. Die Standarddateivergrößerung 10% wird verwendet und die Datei hat keine Maximalgröße.
3. Stellen Sie sicher, dass Sie sich in der Datenbank *master* befinden, wenn Sie die folgende Abfrage ausführen:

```
DROP DATABASE Von, Ron, Don, Christina
```

Lösungen zu den Übungen

1.

```
CREATE DATABASE Frog
ON
(NAME = FrogData1,
```



```
FILENAME = 'F:\Programme\Microsoft SQL Server\mssql
\data\FrogData1.mdf',
SIZE = 3,
FILEGROWTH = 2,
MAXSIZE = 20),
(NAME = FrogData2,
FILENAME = 'F:\Programme\Microsoft SQL Server\mssql
\data\FrogData2.ndf',
SIZE = 3,
FILEGROWTH = 2,
```

```
LOG ON
(NAME = FrogLog1,
FILENAME = 'F:\Programme\Microsoft SQL Server\mssql
\data\FrogLog1.ldf',
MAXSIZE = 5),
(NAME = FrogLog2,
FILENAME = 'F:\Programme\Microsoft SQL Server\mssql
\data\FrogLog2.ldf',
MAXSIZE = 5)
GO
```

2.

```
ALTER DATABASE Frog
ADD FILE
(NAME = FrogData3,
FILENAME = 'F:\Programme\Microsoft SQL Server\mssql
\data\FrogData3.ndf')
GO
```

3.

```
DBCC SHRINKDATABASE (Frog, 80)
GO
```

4.

```
USE Frog
GO
DBCC SHRINKFILE (FrogLog2,EMPTYFILE)
GO
```

5.

```
ALTER DATABASE Frog
REMOVE FILE FrogLog2
GO
```



6.

```
USE Master
GO
EXEC sp_dboption Frog, 'Single User', True
GO
EXEC sp_renamedb 'Frog', 'TreeFrog'
GO
EXEC sp_dboption TreeFrog, 'Single User', False
GO
```

7.

```
USE MASTER
GO
DROP DATABASE TreeFrog
GO
```

A.5 Antworten für Tag 5

Antworten zu den Kontrollfragen

1. Durch Löschen der Gruppe *BUILTIN\Administrators* im Enterprise Manager oder durch Ausführen der gespeicherten Systemprozedur *sp_revokelogin*. Dies würde jedoch die Rechte der Administratoren zum Anmelden an SQL Server insgesamt entziehen. Sie könnten auch die gespeicherte Systemprozedur *sp_dropsrvrolemember* ausführen, um die Mitgliedschaft dieser Administratoren in der festen Serverrolle *sysadmin* zu entfernen.
2. Einem *dbo*-Benutzer in der Datenbank können Sie einen Benutzer hinzufügen. Es ist jedoch besser, den jeweiligen Benutzer zur Datenbank hinzuzufügen und seinen Benutzernamen zu einem Mitglied der festen Datenbankrolle *db_owner* zu machen.
3. Wenn der einzelne Benutzer bestimmte Berechtigungen benötigt, die sich von den Berechtigungen anderer Mitglieder in den Gruppen, in denen er Mitglied ist (oder sein müsste), unterscheiden. Dieses Szenario dürfte nur selten vorkommen.



Lösungen zu den Übungen

1. Die Aufgabe lässt sich mit den gespeicherten Prozeduren `sp_addlogin` und `sp_adduser` oder mit dem SQL Server Enterprise Manager realisieren. Der Code sieht folgendermaßen aus:

```
Exec sp_addlogin 'george'  
Exec sp_addlogin 'Henry'  
Exec sp_addlogin 'Ida'  
Exec sp_addlogin 'John'  
Use pubs  
Exec Sp_adduser 'George'  
Exec sp_adduser 'Henry'  
Exec sp_adduser 'Ida'  
Exec sp_adduser 'John'
```

2. Entfernen Sie den Benutzer John aus der Datenbank und führen Sie dann `sp_changedbowner` aus, um John zum *dbo* zu machen:

```
Use pubs  
Exec sp_changedbowner 'John','TRUE'  
-- Ooh!! Eine Fehlermeldung erhalten!  
exec sp_dropuser 'John'  
exec sp_changedbowner 'John','TRUE'
```

A.6 Antworten für Tag 6

Antworten zu den Kontrollfragen

1. GRANT SELECT ON MeineTabelle TO MARY.
2. Paul braucht SELECT-Berechtigungen für *Joe.JoeSicht* und SELECT-Berechtigungen für *Mary.MaryTabelle*.
3. Joe kann keine SELECT-Operationen bezüglich *MEINETABELLE* durchführen. DENY überschreibt alle anderen Berechtigungen.
4. Joe kann SELECT-Operationen bezüglich *MEINETABELLE* durchführen.
5. Lassen Sie alle Objekte durch den *dbo* oder mit *dbo* als Besitzer erstellen.
6. GRANT SELECT ON MEINETABELLE TO PUBLIC.
7. Alle Anweisungsberechtigungen außer CREATE DATABASE.



A.7 Antworten für Tag 7

Antworten zu den Kontrollfragen

1. RAID 1 hat (allgemein) eine bessere Leistung als RAID 5 für Schreibvorgänge, somit ist es eine bessere Option, um Transaktionsprotokolle aufzunehmen.
2. Ja. Netzwerkstandorte stehen für Sicherungsmedien zur Verfügung.
3. Wenn die Datenbankoption *Truncate Log On Checkpoint* aktiv ist oder *Select Into/Bulkcopy* gesetzt und keine vollständige Sicherung der Datenbank durchgeführt wurde, können Sie keine Sicherung des Transaktionsprotokolls vornehmen.
4. Nach Möglichkeit an einer außerhalb der Firma befindlichen Stelle (beispielsweise im Haus eines Mitarbeiters oder bei einer speziellen Dienstleistungsfirma).
5. Nein, ein Prüfpunkt ist nur eine Kopie der Datenseiten vom Speicher auf die Festplatte. Er hat keine Auswirkung auf die Sicherungen des Transaktionsprotokolls.

Lösungen zu den Übungen

1. Um die Datensicherungsmedien zu erstellen, verwenden Sie den SQL Server Enterprise Manager oder geben folgende Anweisungen ein:

```
exec sp_addumpdevice 'disk', 'master_backup',  
    'F:\Programme\Microsoft SQL Server\mssql\backup\master_backup.dat'  
exec sp_addumpdevice 'disk', 'msdb_backup',  
    'F:\Programme\Microsoft SQL Server\mssql\backup\msdb_backup.dat'  
exec sp_addumpdevice 'disk', 'model_backup',  
    'F:\Programme\Microsoft SQL Server\mssql\backup\model_backup.dat'  
exec sp_addumpdevice 'disk', 'pubs_backup',  
    'F:\Programme\Microsoft SQL Server\mssql\backup\pubs_backup.dat'  
exec sp_addumpdevice 'disk', 'northwind_backup',  
    'F:\Programme\Microsoft SQL Server\mssql\backup\northwind_backup.dat'
```

Zur Sicherung der Datenbanken nehmen Sie den SQL Server Enterprise Manager oder führen die folgende Anweisungen aus:

```
BACKUP DATABASE MASTER TO master_backup WITH INIT  
BACKUP DATABASE MSDB TO msdb_backup WITH INIT  
BACKUP DATABASE MODEL TO model_backup WITH INIT  
BACKUP DATABASE PUBS TO pubs_backup WITH INIT  
BACKUP DATABASE NORTHWIND TO northwind_backup WITH INIT
```



Um die Datenbankoptionen anzusehen, verwenden Sie SQL Server Enterprise Manager oder führen die folgende Anweisung aus:

```
exec sp_helpdb
```

2. Beachten Sie, dass NOINIT Sie nur daran hindert, Ihre vorhergehenden vollständigen Datenbanksicherungen auf demselben Sicherungsmedium zu löschen. Sie könnten ebenso leicht neue Sicherungsmedien erstellen lassen.

```
BACKUP DATABASE Pubs TO pubs_backup WITH NOINIT, DIFFERENTIAL
BACKUP DATABASE NORTHWIND TO northwind_backup WITH NOINIT,
    DIFFERENTIAL
```

- 3.

```
exec sp_addumpdevice 'disk', 'pubs_log_backup',
    'F:\Programme\Microsoft SQL Server
    \mssql\backup\pubs_log_backup.dat'
use pubs
go
-- Die Option truncate log on checkpoint abschalten:
exec sp_dboption 'pubs', 'trunc. Log on chkpt.', FALSE
go
-- Eine vollständige Sicherung ist erforderlich, nachdem Sie
-- truncate Log on checkpoint abgeschaltet haben:
BACKUP DATABASE Pubs TO pubs_backup WITH INIT
-- Schließlich das Transaktionsprotokoll sichern:
BACKUP LOG Pubs TO pubs_log_backup WITH INIT
```

A.8 Antworten für Tag 8

Antworten zu den Kontrollfragen

1. Alle Systemdatenbanken außer *master* sowie *pubs* und *northwind* werden neu initialisiert. Sie müssen alle Benutzerdatenbanken wiederherstellen oder anhängen.
2. Eine Liste der Dateien für jede Datensicherung, die Sie in einer Datensicherungsdatei oder einem Band untersuchen.
3. Ja, eine parallel verteilte Datensicherung kann mit weniger Bandlaufwerken wiederhergestellt werden, als zur Erstellung der Sicherung verwendet wurden.
4. Die Option *Wiederherstellungsintervall*. Sie wird automatisch auf Standardwerte gesetzt, die Sie wahrscheinlich auch nicht zu ändern brauchen.



5. Stellen Sie die letzte vollständige Sicherung der Datenbank wieder her und wenden Sie dann die Transaktionsprotokolle an, wobei Sie den Parameter STOPAT angeben, um bis zu einem bestimmten Zeitpunkt wiederherzustellen.

A.9 Antworten für Tag 9

Antworten zu den Kontrollfragen

1. Wenn maximale Geschwindigkeit im Vordergrund steht und Speicherplatz kein Thema ist.
2. Wenn der Speicherplatz im Vordergrund steht und Geschwindigkeit kein Thema ist.
3. Legen Sie die Daten in einer Datei außerhalb der Datenbank ab und speichern Sie den Dateinamen in der Tabelle. Sie können auch Ihren Text und die Bilder in einer separaten Tabelle speichern. Gegebenenfalls lässt sich ein text-Datentyp in mehrere kleinere char-Felder aufteilen.
4. Geben Sie immer NULL oder NOT NULL an. Verwenden Sie ROWGUID für global eindeutige Bezeichner, die Eigenschaft IDENTITY zum automatischen Erhöhen von Werten und Unicode für internationale Datenbanken.
5.
 - a. Für einen fünfstelligen Postleitzahlcode: char(5), wenn es 10 Ziffern geben kann wie in 40317-2291, dann varchar(10).
 - b. smalldatetime.
 - c. Um nur das Jahr zu speichern, tinyint; sonst nehmen Sie smalldatetime.
 - d. Nimmt man für die Fahrgestellnummer bis zu 20 Zeichen an, char(20).
 - e. char(10).
 - f. varchar(50).
 - g. smalldatetime.
 - h. numeric(,6). Wenn Sie für Währungsdaten den Typ money verwenden, müssen Sie beachten, dass er nur bis zu 4 Stellen nach dem Dezimaltrennzeichen unterstützt.
6. Die Maximalgröße ist $10 + 15 + 15 + 255 + 3$ (3 Byte Steuerungszeichen für die Spalte varchar) = 298 Byte pro Zeile.



7. Ja. Beachten Sie, dass das Feld *fname* zu einem `varchar`-Feld verändert wird und die Text- und Bild-Spalten einen 16-Byte-Zeiger auf die Datenseiten haben, in denen ihre Daten gespeichert werden. Die maximale Zeilengröße beträgt 582 Byte plus alle erforderlichen Steuerzeichen für die Verwaltung von Feldern variabler Länge.

Lösungen zu den Übungen

1.

```
sp_addtype plz, 'char(5)', 'NOT NULL'
sp_addtype Telefon_nummer, 'char(14)'
sp_addtype Artikel_id, 'int', 'NOT NULL'
sp_addtype fax, 'char(14)'
sp_addtype email, 'varchar(50)'
```

2.

```
CREATE TABLE Artikel(
bezeichner store_id,
Name varchar(30),
addr1 varchar(50),
addr2 varchar(50),
stadt varchar(30),
bundesland char(2) NOT NULL,
PLZ zip_code,
besitzer varchar(30),
kontaktinfo varchar(30),
fax fax,
email email
)
```

3.

```
CREATE TABLE sales (
Bezeichner store_id,
verk_datum datetime,
umsatz money,
gewinn money,
einlage money
)
```




A.10 Antworten für Tag 10

Antworten zu den Kontrollfragen

1.
 - a. Diese Abfrage gibt alle Spalten aus der Tabelle *authors* zurück, in denen der Nachname des Autors mit dem Buchstaben M beginnt.
 - b. Diese Abfrage gibt alle Zeilen aus der Tabelle *employee* für die Spalten *emp_id*, *fname* und *lname* zurück. Die Spaltenüberschriften erhalten die Aliasnamen *EmployeeID*, *LastName* und *FirstName*.
 - c. Diese Abfrage rundet den Dollarbetrag mit einer einzigen Ziffer nach dem Dezimaltrennzeichen auf 7725,5.
 - d. Diese Abfrage gibt eine Spalte mit dem Alias *Name* und die Spalte *emp_id* mit dem Alias *EmployeeID* aus der Tabelle *employee* zurück. Die Spalte *Name* ist aus dem Nachnamen, einem Komma (,) und dem ersten Buchstaben des Vornamens zusammengesetzt.
2. Ja, in den meisten Fällen lässt sich eine Unterabfrage als Verknüpfung und umgekehrt implementieren.
3. Falsch. ROLLUP und CUBE sind vorgesehen, um Zusammenfassungsinformationen zu liefern.
4. Ja. Sie müssen nur das Symbol ## vor den Tabellennamen setzen, um die Tabelle zu erstellen.

Lösungen zu den Übungen

1.

```
SELECT title_id, title, price
FROM titles
WHERE (pub_id = '0877'
OR title LIKE '%computer%')
And price is not NULL
(gibt 14 Zeilen zurück)
```
2.

```
SELECT * FROM titles
WHERE price IS NOT NULL
(gibt 16 Zeilen zurück)
```



3. SELECT title, price

```
FROM titles
ORDER BY price DESC
(gibt 18 Zeilen zurück)
```

4. SELECT AVG(ytd_sales) FROM titles

5. SELECT title_id, count(title_id)

```
FROM titleauthor
GROUP BY title_id
HAVING count(title_id) > 1
```

6. SELECT stor_id, qty

```
FROM sales
ORDER BY stor_id
COMPUTE SUM(qty) BY stor_id
```

7. ANSI-Syntax

```
SELECT authors.au_fname, authors.au_lname, titles.title
FROM authors
INNER JOIN titleauthor ON authors.au_id = titleauthor.au_id
INNER JOIN titles ON titleauthor.title_id = titles.title_id
ORDER BY authors.au_lname
```

SQL Server-Syntax

```
SELECT authors.au_fname, authors.au_lname, titles.title
FROM authors, titles, titleauthor
WHERE authors.au_id = titleauthor.au_id
AND titleauthor.title_id = titles.title_id
ORDER BY authors.au_lname
```

8. SELECT * FROM authors

```
WHERE authors.state IN
(SELECT state from stores)
```

9. SELECT *

```
INTO #tmpEmployees
FROM employee
GO
SELECT * FROM #tmpEmployees
GO
```



A.11 Antworten für Tag 11

Antworten zu den Kontrollfragen

1. Ein DELETE ohne WHERE löscht jede Zeile aus einer Tabelle. Wenn Sie wirklich alle Zeilen aus einer Tabelle entfernen möchten (ohne die Tabelle selbst zu löschen), ist die Anweisung TRUNCATE TABLE effizienter.
2. Falsch. Identitätswerte werden nie bereitgestellt, sondern durch SQL Server berechnet. Für Spalten, die einen Standardwert haben oder NULL zulassen, brauchen Sie keinen Wert bereitzustellen, können es aber durchaus tun.
3. Durch Verknüpfungen in einer DELETE- oder UPDATE-Anweisung können Sie auf Werte in einer anderen Tabelle zugreifen, um damit die zu ändernden Zeilen zu bestimmen. Die zweite Tabelle dient lediglich als Nachschlagetabelle und unterliegt keinen Änderungen. Mit diesen Operationen zur Datenänderung lässt sich jeweils nur eine Tabelle auf einmal ändern.

Lösungen zu den Übungen

1. Lösung:

```
select title_id, title, pub_id, price
into #cook_books
from tmpTitles
where type = 'mod_cook'
```

2. Lösung:

```
insert into #cook_books
select title_id, title, pub_id, price
from tmpTitles
where type = 'trad_cook'
```

3. Lösung:

```
update #cook_books
set price = price * 1.2
```

4. Lösung:

```
update #cook_books
set price = price * 0.9
where pub_id =
(select pub_id from publishers
where pub_name = 'Binnet & Hardley' )
```



5. Lösung:

```
delete #cook_books
where price < $10
```

6. Lösung:

```
delete #cook_books
where title_id in
  (select title_id from tmpTitles
   where ytd_sales > 10000)
```

A.12 Antworten für Tag 12

Antworten zu den Kontrollfragen

1. CREATE DATABASE, CREATE TABLE, CREATE INDEX

2. BEGIN TRAN

```
UPDATE authors SET au_lname = 'Johnson' WHERE au_lname = 'Smith'
INSERT publishers VALUES ('9991','SAMS','Indianapolis','IN','USA')
SELECT * FROM publishers (HOLDLOCK)
COMMIT TRAN
```

Die Anweisung UPDATE belegt die Tabelle *authors* mit zwei exklusiven Zeilensperren sowie zwei beabsichtigten exklusiven Seitensperren (für die gelöschten und die eingefügten Zeilen). Die INSERT-Anweisung nimmt eine exklusive Zeilensperre in einer Datenseite in der Tabelle *publishers* vor (Seite 99 in der folgenden Tabelle). Schließlich bewirkt die SELECT-Anweisung mehrere Schlüsselbereichssperren in der Tabelle *publisher* (wegen der Option HOLDLOCK). Die gespeicherte Prozedur *sp_lock* liefert folgende Ergebnisse:

spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
8	2	0	0	DB		S	GRANT
8	5	0	0	DB		S	GRANT
8	2	0	0	EXT	1:80	X	GRANT
8	5	117575457	1	PAG	1:87	IX	GRANT
8	5	117575457	2	PAG	1:122	IX	GRANT
8	5	197575742	1	PAG	1:99	IX	GRANT
8	5	117575457	0	TAB		IX	GRANT
8	5	197575742	1	KEY	(ffffffffffffff)	IS-S	GRANT
8	5	117575457	2	KEY	(d59632855aa8)	X	GRANT
8	5	197575742	0	TAB		IX	GRANT



8	5	117575457	2	KEY	(39747cc5fc43)	X	GRANT
8	5	117575457	1	KEY	(02c094e89f8a)	X	GRANT
8	5	197575742	1	KEY	(030431363232)	IS-S	GRANT
8	5	197575742	1	KEY	(030130373336)	IS-S	GRANT
8	5	197575742	1	KEY	(040131373536)	IS-S	GRANT
8	5	197575742	1	KEY	(090a31333839)	IS-S	GRANT
8	5	197575742	1	KEY	(070f30383737)	IS-S	GRANT
8	5	197575742	1	KEY	(000039393939)	IS-S	GRANT
8	5	197575742	1	KEY	(000139393938)	IS-S	GRANT
8	5	197575742	1	KEY	(0c0b39393532)	IS-S	GRANT
8	5	197575742	1	KEY	(090839393031)	IS-S	GRANT
8	5	197575742	1	KEY	(000839393931)	X	GRANT

(22 row(s) affected)

3. Durch Setzen der Konfigurationseinstellung `REMOTE_PROC_TRANSACTIONS`. Sie können diese Einstellung entweder mit der gespeicherten Prozedur `sp_configure` oder im Konfigurationsdialogfeld des Enterprise Managers festlegen. Vorher müssen Sie Remote-server oder Verbindungsserver einrichten, zu denen Sie die Verbindung herstellen möchten.
4. Mit den Anweisungen `ROLLBACK TRAN` oder `ROLLBACK WORK`.
5. Ja. Das `ROLLBACK TRAN` im Trigger bricht sowohl die Transaktion als auch die Stapelverarbeitung ab. Die Anweisung `SELECT * FROM AUTHORS` befindet sich im zweiten Stapel und wird deshalb ausgeführt. Beachten Sie jedoch, dass die Anweisung `COMMIT TRAN` scheitert.

Server: Nachr.-Nr. 3902, Schweregrad 16, Status 1

Die `COMMIT TRANSACTION`-Anforderung hat keine entsprechende `BEGIN TRANSACTION`.

Lösung zur Übung

BEGIN TRAN

```

INSERT publishers VALUES ('9993','WAYCOOL PUBLISHERS',
    'Indianapolis', 'IN', 'USA')
INSERT authors VALUES ('111-11-1111','Jackson','Ann','425 999-9000',
    'PO Box 1193','Snoqualmie','WA', '98065', 1)
INSERT authors VALUES ('111-22-1111','Greene','Bob','425 999-9000',
    '1204 Sycamore Lane','Boulder City','NV', '89005', 1)
INSERT TITLES VALUES ('BU1403','Surfen im Netz in drei leichten Schritten',
    'business', '9993', $19.95, $3000.00, NULL, NULL, NULL,'8/1/1998')
INSERT TITLEAUTHOR VALUES ('111-11-1111','BU1403', 1, 50)
INSERT TITLEAUTHOR VALUES ('111-22-1111','BU1403', 1, 50)
COMMIT TRAN

```



A.13 Antworten für Tag 13

Antworten zu den Kontrollfragen

1. Indem Sie einen Optimierungshinweis mit einer Indexzahl von 0 angeben.
2. Mindestens 120% der Größe der Tabelle, mit der Sie den Index erstellen.
3. Aktivieren Sie `SHOWPLAN_TEXT` oder den grafischen Ausführungsplan des SQL Server Query Analyzers.

Lösungen zu den Übungen

1. `set showplan_text on`

```
go
Select * from pubs..sales where title_id = 'BU1032'
Go
Set showplan_text off
Go
Set statistics io on
Go
Select * from pubs..sales where title_id = 'BU1032'
Go
```

(Denken Sie daran, dass Sie nicht in einem einzigen Schritt eine Abfrage ausführen und den Ausführungsplan sehen können.)

2. `Select * from pubs..sales (index=1) where title_id = 'BU1032'`

`Go`

3. Die Lösungen können unterschiedlich aussehen. Eine mögliche Lösung lautet:

```
Create table myindextab2
(col1 int not null,
 col2 char(5) not null,
 col3 varchar(50) null)
go
create unique clustered index myclindex on myindextab2 (col1)
    with fillfactor = 100
go
create index mynonclindex1 on myindextab2 (col1,col2)
    with fillfactor = 25
go
```



```
create index mynonclindex2 on myindextab2 (col3)
  with fillfactor = 75
go
dbcc dbreindex ('myindextab2', ' ', 50)
go
```

4. Die Lösung dieser Aufgabe mit der grafischen Benutzeroberfläche des Enterprise Managers dürfte keine Probleme darstellen, wenn Sie Übung 3 erfolgreich absolviert haben.

A.14 Antworten für Tag 14

Antworten zu den Kontrollfragen

1. Ja.
2. Nein. Um eine Regel zu löschen, müssen Sie vorher eine eventuell bestehende Bindung an eine Spalte oder einen benutzerdefinierten Datentyp aufheben.
3. Ja, wenn die Anzahl der Dezimalstellen auf 0 gesetzt ist.
4. Nein, Fremdschlüssel können nur Tabellen in derselben Datenbank referenzieren.
5. Nein, sie erzeugt einen eindeutigen Index, der nicht deaktiviert werden kann.
6. Nicht direkt. Sie müssen sie durch Steuerung der Einschränkungen verwalten.

A.15 Antworten für Tag 15

Antworten zu den Kontrollfragen

1. Datenintegrität, referenzielle Integrität und Geschäftsregeln.
2. So viele, wie Sie wollen. Es kann unterschiedliche Trigger für INSERT-, UPDATE- und DELETE-Anweisungen geben.
3. Falsch. Sichten sind nicht schneller als gespeicherte Prozeduren. Alle anderen Kriterien treffen jedoch auf Sichten zu.
4. Falsch. Mit einer Sicht können Sie nur eine einzige zugrunde liegende Tabelle aktualisieren.



5. Richtig. Dies ist einer der Vorzüge der Erstellung einer gespeicherten Prozedur.
6. Falsch. Sie müssen die Berechtigung für die zugrunde liegenden Tabellen haben, um eine Sicht auszuführen. Es ist jedoch interessant, dass Sie eine Sicht auf Basistabellen erzeugen können, auf die Sie keinen Zugriff haben.
7. Falsch. Obwohl diese Dinge die meisten Trigger unnötig gemacht haben, gibt es dennoch viele wirksame Anwendungen für Trigger – beispielsweise kaskadierende Aktualisierungen und Löschvorgänge.
8. Nur Trigger werden gelöscht.
9. Falsch. Die Option SCHEMABINDING soll gerade das verhindern.
10. 10.Richtig. Das sind genau die Einsatzgebiete, für die sie vorgesehen sind.

Lösungen zu den Übungen

1. CREATE VIEW myView AS select au_lname, au_fname, t.title from

```
authors inner join titleauthor ta on authors.au_id =  
    ta.au_id inner join titles t on ta.title_id = t.title_id
```

2. Create trigger mytrigger on titles for insert as

```
if(select count(*) from inserted I inner join publishers p on  
    i.pub_id = p.pub_id) <> 1 raiserror(50001, 10, 1)
```

3. CREATE PROC myproc AS

```
SELECT s.stor_name, t.title FROM  
stores s, titles t, sales sa  
WHERE t.title_id = sa.title_id  
AND s.stor_id = sa.stor_id  
ORDER BY s.stor_name
```

4. Eine mögliche Lösung lautet:

```
CREATE FUNCTION dbo.fnAvgPriceBooksSold(@MyDate datetime)  
RETURNS money  
AS  
BEGIN  
    DECLARE @RetVal money  
    SET @RetVal = (SELECT CONVERT(money, AVG(s.qty * t.price))  
    FROM sales s, titles t  
    WHERE s.title_id = t.title_id  
    AND s.ord_date <= @MyDate)  
    RETURN @RetVal  
END
```




Das Ergebnis können Sie mit den beiden folgenden Abfragen testen:

```
SELECT dbo.fnAvgPriceBooksSold('10/10/94')
SELECT dbo.fnAvgPriceBooksSold('10/10/93')
```

A.16 Antworten für Tag 16

Antworten zu den Kontrollfragen

1. Der Protokolllese-Agent.
2. Der Verteilungs-Agent verschiebt replizierte Transaktionen von der Verteilungsdatenbank zu den Abonnenten.
3. Das Szenario mit zentralem Abonnenten.
4. Der Merge-Agent befindet sich auf dem Verteilungsserver, wenn Sie Pushreplikation verwenden. Bei Pullabonnements befindet sich der Merge-Agent auf jedem Abonnenten.

Lösung zur Übung

Es gibt verschiedene Lösungen. Angesichts der Fakten empfiehlt sich eine Hybridinstallation. Ein Ansatz besteht in der Implementierung eines zentralen Abonnentenszenarios im Depot in Wichita, Kansas. Die anderen vier Standorte, die über die Vereinigten Staaten verteilt sind, würden Daten an den zentralen Standort publizieren.

Die Wichita-Daten könnten dann in einem Szenario mit publizierenden Abonnenten mit einem der britischen Standorte verwendet werden. Die in die Wichita-Datenbank eingehenden Daten könnten über die langsamen und teuren Telefonleitungen an eine einzige Datenbank auf den britischen Inseln geschickt werden. Diese britische Datenbank könnte dann die Daten erneut an die anderen administrativen Standorte publizieren, die sich in Großbritannien befinden.



A.17 Antworten für Tag 17

Keine Kontrollfragen – keine Lösungen.

A.18 Antworten für Tag 18

Antworten zu den Kontrollfragen

1. Der AT-Befehl und der Windows 2000-Zeitplandienst.
2. Auf den Namen des Mailprofils des Windows 2000-Benutzerkontos, unter dem der SQL Server- oder SQL Server-Agent-Dienst ausgeführt wird.
3. Exec sp_processmail.
4. Zwei geplante Tasks (einen für den Befehl mit der Klausel `WITH INIT`, den anderen für den Befehl ohne diese Klausel, der alle drei Stunden läuft).
5. Der Datenbank-Wartungsplanungs-Assistent.

A.19 Antworten für Tag 19

Antworten zu den Kontrollfragen

1. Sie müssen `SELECT`-Berechtigungen in der Quelldatenbank sowohl für die Daten- als auch die Systemtabellen haben. In der Zieldatenbank müssen Sie der `dbo` sein.
2. BCP erzwingt immer Standardwerte und Datentypen, es ignoriert immer Regeln, Einschränkungen und Trigger.
3. Sie müssen die Datenbankoption `select into/bulk copy` eingeschaltet haben und alle Indizes in der Tabelle löschen.
4. Sie können Daten direkt übertragen, in andere Datentypen umwandeln, zusammenfassen, miteinander verketten, in Komponentenspalten aufteilen und viele andere Umwandlungen durchführen.
5. Sie können hier nicht das Verfahren der Datensicherung und Wiederherstellung verwenden, weil sich die Sortierreihenfolge und die Prozessorarchitektur geändert haben.



Sie sollten SQL Server 2000 auf dem neuen Alpha-Rechner installieren und dann mit der DTS-Objektübertragung alle Objekte, Benutzer und Berechtigungen von dem Intel-Rechner auf den Alpha-Rechner übertragen.

A.20 Antworten für Tag 20

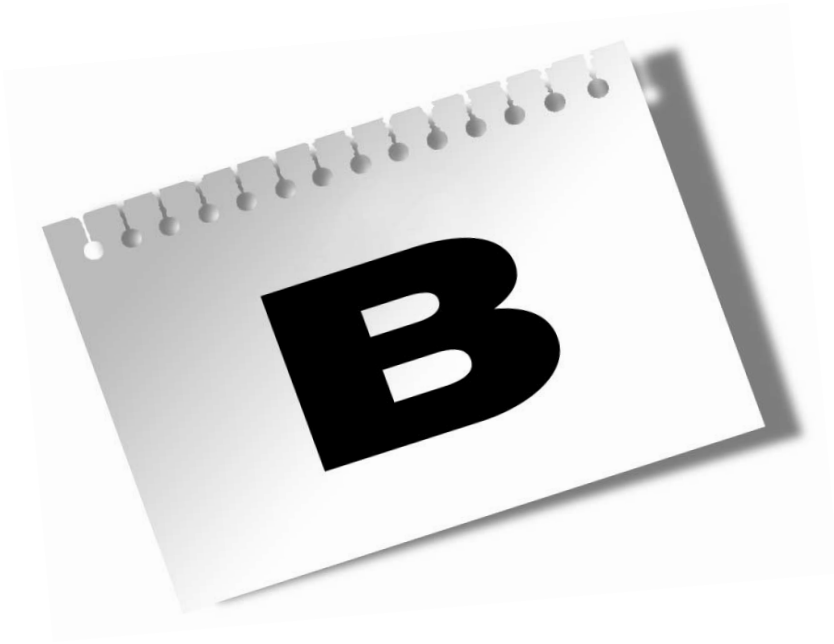
Antworten zu den Kontrollfragen

1. Die physikalische Implementierung des logischen Datenbankentwurfs. Keine Einstellung der Hardware- oder Software-Optimierung gleicht einen schlechten Entwurf aus.
2. Damit die Plattenaktivität des Protokolls die höchste Priorität haben und den Gesamtdurchsatz von SQL Server beschleunigen kann.
3. Überwachen Sie im Objekt Speicher den Leistungsindikator Seiten/s, um nach einem hohen Wert für Auslagerungsaktivitäten zu forschen.
4. Die Konfiguration der Affinitätsmaske mit der gespeicherten Systemprozedur `sp_configure`.
5. Im Objekt `SQLServer:Databases` gibt der Indikator *Log Flushes/sec* den besten Näherungswert für die Anzahl der in SQL Server durchgeführten Änderungen. Die Aktivität können Sie auch mit dem Leistungsindikator *Prozessorzeit (%)* des Objekts *Prozess* messen, wobei Sie als Instanz `sqlservr` wählen. In der Tat gibt es keinen »besten Leistungsindikator«; die Überwachung lässt sich nicht auf einen einzigen Indikator reduzieren.

A.21 Antworten für Tag 21

Antworten zu den Kontrollfragen

1. In einem URL sind keine Leerzeichen erlaubt. Der korrekte URL lautet:
`http://ser/pubs?SQL=select+city+from+authors+for+XML+AUTO&root=ROOT.`
2. Anweisungen aller Art – legen Sie deshalb die Sicherheitseinstellungen sorgfältig fest.
3. Die Option `RAW` liefert in der Standardeinstellung weniger detaillierte XML-Tags.



Die CD zum Buch



Das Verzeichnis \EBOOKS enthält sowohl das vorliegende Buch komplett als HTML-Datei sowie als Bonus aus der gleichen Reihe den Titel *SQL in 21 Tagen* (ISBN 3-8272-2020-3). Die Texte können Sie mit jedem Browser anzeigen und rufen sie am besten mit der Datei »inhalt.htm« im Verzeichnis des entsprechenden Buches auf.

Mit dem Buch *SQL in 21 Tagen* können Sie Ihr Wissen zur Sprache SQL vertiefen. Das Buch geht nicht nur auf den SQL-Dialekt von SQL Server, sondern auch auf die SQL-Dialekte anderer Datenbankprodukte ein. Es bietet Ihnen alle Informationen, damit Sie die Sprache SQL umfassend verstehen und nutzen können – gleichgültig, ob Sie Programmierer, Administrator oder Datenbankentwickler sind. Der leichtverständliche Lehrband stellt SQL ausführlich vor und zeigt Ihnen, wie man Abfragen konstruiert und optimiert, mit gespeicherten Prozeduren und Triggern die Leistung einer Datenbank maximiert und die grundlegenden Konzepte und Merkmale von SQL meistert.

Für das Buch *SQL in 21 Tagen* sind unter anderem folgende Lernziele genannt:

- Einmaleins für effektives Datenbankdesign
- Funktionen und Klauseln für das Abrufen von Daten
- Erzeugen von Tabellen und komplexen Abfragen
- Erkennen und Meistern von häufigen Fehlern beim Einsatz von SQL
- Feinabstimmung von Datenbanken mit gespeicherten Prozeduren und Triggern

Wir wünschen viel Spaß und Erfolg beim Lesen und Lernen!

Stichwortverzeichnis

Symbols

! 416
!= 416
" 416
369
\$ 369
() 416
+ 407
= 416
@ 369
@@IDENTITY 592
@@trancount 487
- 369

A

Abfragen
-, Arbeitsspeicher 843
-, gespeicherte Prozeduren 633
-, korrelierte Unterabfrage 449
-, Kostenlimitoption 844
-, min memory per query-Option 843
-, Optimierung 844
-, parallele 840
-, SELECT-Anweisung 398
-, Sortieren 427
-, Unterabfrage 447, 472
-, URL-Abfragen 873
-, Vorlagen-Abfragen 873
-, WHERE 416
Abfrageoptimierer 537
-, Abfragehinweise 545
-, indizierte Sichten 557
-, NOT in Suchkriterien 420
Abfragesprachen, XPath 873
Abhängige Objekte 265
Abhängigkeiten
-, Berechtigungen 250
-, Besitzketten 265
Ablaufverfolgung
-, einrichten 850
-, filtern 855
Ablaufverfolgungsflags 140
Ablaufverfolgungsvorlagen 850
Abonnements
-, anonyme 681
-, Pull 681
-, Push 680
Abonnementsserver,
 Wiederherstellung 747
Abonnenten, Identitätswechsel 710
Abrufintervall 104
ABS 405
Abwärtskompatibilität
-, DB-Library 64
-, Sortierung 71
-, sysprotects 235
ACID-Eigenschaften 484
ACOS 405
ActiveX-Skripts 752
Address Windowing Extensions
 (AWE) 830
Administratorenkonto 67
Adressformatierung für Pager 787
ADSP 74
Änderungen, kaskadierende 648
Äußere Verknüpfung 444
Affinity mask 840
AFTER 644
Agentverlauf 726
Aggregatfunktionen 429
Aktionen, kaskadierende
 referenzielle 575
Aktualisieren
-, Daten 474
-, sofortiges 673
-, Suchtabellen 476
-, über eine Warteschlange 672
-, Warteschlange 673
Aktualisierungs-Assistent 87, 131
Aktualisierungssperren 508
Aliasnamen 215
-, Spaltenüberschriften 401

- , Tabellen 447
- , Verknüpfungen 447
- ALL 249
- Alt+F1 124
- ALTER DATABASE 165
- , Wiederherstellungsmodell 293
- ALTER TABLE, Einschränkungen
 - löschen 607
- ALTER TRIGGER 646
- ALTER VIEW 631
- Anfangssnapshot 682
- Anmelden, Rechte entziehen 204
- anonyme Abonnements 681
- ANSI
 - , Datentypen 383
 - , DEFAULT 580
 - , Einschränkungen 596
 - , Konvertierung in OEM 108
 - , SQL-92 398
 - , Standardwerte 580
- ANSI_NULLS 483
- Anweisungen
 - , ALTER DATABASE 165
 - , BACKUP DATABASE 241, 308
 - , BACKUP LOG 241, 249, 314
 - , bedingt ausführen 499
 - , BEGIN DISTRIBUTED TRANSACTION 495
 - , CASCADE 258
 - , CHECKPOINT 494
 - , CREATE DATABASE 249
 - , CREATE DEFAULT 580
 - , CREATE INDEX 531
 - , CREATE TABLE 388
 - , CREATE TRIGGER 645
 - , DBCC CHECKALLOC 305
 - , DBCC CHECKCATALOG 304
 - , DBCC CHECKDDB 301
 - , DBCC CHECKTABLE 305
 - , DBCC FREEPROCCACHE 238
 - , DBCC SHRINKDATABASE 165
 - , DBCC SHRINKFILE 165
 - , DELETE 470
 - , DENY 251, 258
 - , DISK INIT 240
 - , DISK MIRROR 240
 - , DISK REFIT 240
 - , DISK REINIT 240
 - , DROP TABLE 390
 - , GO 482
 - , GRANT 251, 256
 - , IF EXISTS 500
 - , IF...ELSE 499
 - , KILL 239
 - , PRINT 498
 - , RECONFIGURE 238
 - , RESTORE 341
 - , RESTORE DATABASE 241
 - , RESTORE LOG 241, 347
 - , REVOKE 251, 257
 - , SELECT 398
 - , SELECT INTO 450
 - , SHUTDOWN 238
 - , WHILE 506
- Anweisungsberechtigungen 248f.
- , BACKUP DATABASE 250
- , BACKUP LOG 250
- , DENY 251
- , entfernen 251
- , erteilen 251
- , GRANT 251
- , REVOKE 251
- , verweigern 251
- , zuweisen 251
- Anwendungsprogrammierschnittstelle (API) 32
- Anwendungsrollen 227
- API 32
- AppleTalk 49, 74
- Arbeitsspeicher
 - , Abfragen 843
 - , Größe festlegen 842
- Artikel 677
- AS 402
- ASC 428
- ASCII 407
- ASIN 405
- Assistenten
 - , Datenbank-Diagrammerstellung 610
 - , Datenbank-Wartungsplanung 142
 - , DTS-Export 819

- , DTS-Import/Export 819
- , Indexoptimierung 858
- , Publikationserstellung 713
- , Publizierungs- und
Verteilungskonfiguration 705
- , SQL Server-Aktualisierung 131
- , SQL Server-Registrierung 124
- , Transformieren publizierter
Daten 694
- , Volltextindizierung 566
- ATAN 405
- ATN2 405
- Aufbewahrungsdauer 312, 735
- , Abonnent wiederherstellen 747
- , Replikationsverlauf 735
- Aufbewahrungsfrist 287
- Auflösung 633
- Aufträge 771
 - , Auftragssystem 778
 - , Benachrichtigungen 778
 - , E-Mail 778
 - , Ergebnisse speichern 774
 - , erstellen 772
 - , Häufigkeit 776
 - , Multiserver 780
 - , NET SEND 778
 - , Pager 778
 - , universelle Optionen 778
 - , Zeitplan 775
- Auftragsverlauf, sysjobhistory 779
- Ausfallsicherheitsoperator 787
- Ausführungsmodus 117
- Ausführungspläne 543
- Ausgangswert, IDENTITY 591
- Authentifizierung
 - , Windows 50, 67, 189
 - , Windows 9x 190
- AUTO, FOR XML 887
- Automatische Transaktionen 490
- Automatische Wiederherstellung 331
- Autonomie 667
- AVG 430
- AWE 830

B

- Backup 282
- BACKUP DATABASE 241, 250, 308
- BACKUP LOG 241, 249f., 314

- backupfile 340
- backupmediafamily 340
- backupmediaset 340
- backupset 340
- Bandbezeichnungen 289
- Bandgeschwindigkeit 287
- Bandsicherungen,
 - Aufbewahrungsdauer 312
- Banyan VINES 49, 74
- Batch Siehe Stapel
- Batchtrennzeichen 135
- B-Baum 525
 - , Blattseiten 525
 - , Datenseiten 526
 - , doppelt verkettete Liste 526
 - , Seitenteilung 535
 - , Stammseite 525
- BCP (Bulk Copy Programm) 146, 800
 - , Berechtigungen 801
 - , Dateierweiterungen 800
 - , Feldabschlusszeichen 807
 - , Formatdateien 806, 808
 - , schnelles 809
 - , Stapelgrößen 808
 - , Syntax 801
 - , systemeigener Modus 805
 - , Zeichenmodus 805
- beabsichtigte Sperren 509
- Bedingungen
 - , CASE 501
 - , testen 501
- BEGIN DISTRIBUTED
TRANSACTION 495
- BEGIN TRAN 485
- BEGIN TRANSACTION 485f.
- BEGIN...END 498
- Beispieldatenbanken 81
 - , instpubs.sql 479
 - , neu installieren 479
 - , Northwind 83
 - , pubs 82
- Benachrichtigungen, Aufträge 778
- benannte Instanzen 85
 - , installieren 86
- benannte Parameter 639
- Benutzer, einrichten 198
- Benutzerberechtigungen 248

- Benutzerdefinierte Datentypen 371
 - , Domänenintegrität 576
 - , erstellen 578
 - , löschen 579
- Benutzernamen
 - , aktuellen ermitteln 598
 - , guest 213
 - , hinzufügen 207
- Berechnete Spalten 371
- Berechneter Datentyp 386
- Berechtigungen 191, 234
 - , Abhängigkeiten 250
 - , Anweisung 248
 - , anzeigen 237
 - , BCP 801
 - , benutzerdefinierte Funktionen 265
 - , CREATE DATABASE 165, 249
 - , CREATE TABLE 250
 - , DEFAULT 250
 - , FAT 801
 - , FAT32 801
 - , FUNCTION 250
 - , gespeicherte Prozeduren 264
 - , INSERT 464, 801
 - , NTFS 801
 - , Objekt 248
 - , Objektbesitzer 236
 - , PROCEDURE 250
 - , RULE 250
 - , Sichten 259, 264
 - , Strategien 268
 - , Typen 235
 - , VIEW 250
 - , Vorrang 236
- Berechtigungsebenen 236
- Berechtigungstypen 236
- Besitzketten 265
 - , Einzelbesitzer 266
 - , unterbrochene 266
- Beständigkeit 484
- Betriebssysteme, Windows 9x 49
- BETWEEN 417
- Bezeichner
 - , in Anführungszeichen verwenden 163
 - , Regeln 369
- bigint 371
- binary large Object (BLOB) 377
- binary(n) 371
- bit 371
- BLOB, Indexstatistiken 548
- Blockierungssituation 508
- Blocksperrern 508
- Blöcke 498
- Bulk Copy Programm Siehe BCP
- BULK INSERT 811
- BULK_LOGGED 293
- bulkadmin 219
- C**
- C2 857
- CASCADE 258
- CASE 501
- , COALESCE 504
- , NULLIF 505
- CEILING 406
- CHAR 407
- char(n) 371
- CHARINDEX 407
- CHECK, Beispiele 600
- Checkpoint 494
- Cleanup, Replikation 735
- Client/Server
 - , Anwendungen 40
 - , Modell 40
- Clientkonfiguration
 - , Alias 107
 - , DB-Bibliotheksoptionen 108
 - , Netzwerkbibliotheken 109
 - , Protokollverschlüsselung 107
- Clientkonnektivität 63
- CNVT 512
- COALESCE 412, 504
- Codebeispiele 64
- COL_LENGTH 412
- COL_NAME 412
- COLLATE 155
- COM, DTS 814
- COMMIT 331, 486
- COMMIT TRAN 485
- COMMIT TRANSACTION 485f.
- COMMIT WORK 486

COMPUTE 433
 Computerverwaltung 198
 Concat NULL Yields NULL 164
 console 289
 CONSTRAINT 596
 CONVERT 414
 COS 405
 cost threshold for parallelism-
 Option 841
 COT 406
 COUNT 430
 CREATE DATABASE 151, 249
 –, Datenbank anhängen 335
 –, FOR ATTACH 335
 CREATE DEFAULT 580
 CREATE INDEX 531
 CREATE PROCEDURE 634
 –, WITH RECOMPILE 641
 CREATE RULE 587
 –, Beispiel 589
 CREATE TABLE 250, 388
 CREATE TRIGGER 645
 CREATE VIEW 624
 –, Optionen 624
 CROSS JOIN 442
 CUBE 436
 Cubes 76
 CURRENT_TIMESTAMP 598
 CURRENT_USER 598
 Cursor, schließen 164
 Cursor close on commit 164

D

Data Transformation Services
 (DTS) 813
 DataBase Consistency Checker 301
 DATALENGTH 412
 Datapump 818
 Datawarehouse 843
 Datawarehousing 76
 DATEADD 410
 DATEDIFF 410
 Dateien
 –, Fehlerprotokoll 140
 –, Wiederherstellung 351
 Dateigruppen 150, 175
 –, benutzerdefinierte 175
 –, entfernen 178
 –, PRIMARY 175
 –, Wiederherstellung 351
 Datei-ID 172
 Dateisysteme 51, 280
 –, verschlüsselndes 51
 Dateiverbindungen 818
 Daten
 –, aktualisieren 474
 –, einfügen 461
 –, löschen 470
 –, manipulieren 403
 –, schützen 276
 DATENAME 411
 Datenbankadministrator 216
 Datenbankbenutzernamen 191
 Datenbank-Diagrammerstellungs-
 Assistent 610
 Datenbanken
 –, anhängen 335
 –, Berechtigungen 235
 –, Besitzer ändern 216
 –, distribution 286
 –, Dumps 282
 –, entwerfen 34
 –, erstellen 150
 –, erweitern 167
 –, fehlerverdächtig 358
 –, Größe verändern 165
 –, Integritätsprüfung 302
 –, Konsistenz 241, 301
 –, löschen 180
 –, master 81, 294
 –, model 81f., 150
 –, msdb 81f.
 –, Northwind 81, 83
 –, Objekte 33
 –, pubs 81f.
 –, relationale 32
 –, Rollen 221
 –, Systemkatalog 895
 –, tempdb 81f.
 –, trennen 335
 –, umbenennen 174

- , verkleinern 171
- , Zugriff 210
- Datenbankentwurf 37
- Datenbankintegritätsprüfung 789
- Datenbankkonsistenz 241
- Datenbankmodul 30
- Datenbankobjektbesitzer 236, 246
- Datenbankoptionen
 - , ANSI NULL ist Standard 162
 - , Auswahl / Massenkopieren 292
 - , Automatisch schließen 162
 - , Automatisch verkleinern 162
 - , Bezeichner in Anführungszeichen verwenden 163
 - , Concat null yields null 164
 - , Cursor close on commit 164
 - , Default to local cursor 164
 - , Einzelbenutzermodus 162
 - , Erkennung von zerrissenen Seiten 162
 - , festlegen 161
 - , Kompatibilitätsebene 163
 - , Merge publish 165
 - , Mitglieder von 'db_owner', 'dbcreator' oder 'sysadmin' 162
 - , offline 165
 - , Protokoll bei Prüfpunkt abschneiden 292
 - , Published 165
 - , rekursive Trigger 162
 - , schreibgeschützt 162
 - , Select Into/Bulkcopy 292
 - , Statistiken automatisch aktualisieren 162
 - , Statistiken automatisch erstellen 163
 - , Subscribed 165
 - , Truncate Log on Checkpoint 292
- Datenbankrollen 221
 - , benutzerdefinierte 222
 - , db_accessadmin 222, 242
 - , db_backupoperator 243
 - , db_datareader 222, 244
 - , db_datawriter 222, 244
 - , db_ddladmin 222, 243
 - , db_denydatareader 222, 244
 - , db_denydatawriter 222, 244
 - , db_dumpoperator 222
 - , db_owner 222, 241
 - , db_securityadmin 222, 243
 - , feste 240
 - , Mitgliedschaft 225
 - , öffentliche 248
 - , Public 248
- Datenbanksicherungen, differenzielle 311
- Datenbank-Wartungsplanungs-Assistent 142, 788
- Datenbankzugriff, entziehen 211
- Datendateien, Erweiterungen 154
- Datendefinitionssprache 241
- Datenkonvergenz 668
- Datenquellenverbindungen 818
- Datensatzherkunft 684
- Datenseiten, Sperren 508
- Datensicherung
 - , differenzielle 282
 - , Funktionsweise 281
 - , Konsistenz 281
- Datenträgerspiegelung 276
- Datentypen 370f.
 - , arithmetische Operatoren 404
 - , automatisch generiert 371
 - , benutzerdefinierte 371, 384
 - , berechnete Spalten 371, 386
 - , besondere 371, 377
 - , bigint 374
 - , Binärzeichenfolgen 371, 373
 - , binary(n) 373
 - , bit 377
 - , BLOB (binary large object) 377
 - , char(n) 372
 - , datetime 380
 - , Datum und Zeit 371, 380
 - , decimal 375
 - , Domänenintegrität 576
 - , Erstellen benutzerdefinierter 577
 - , erstellen mit Enterprise Manager 385
 - , Familien 379
 - , float 375
 - , ganze Zahlen 371, 374
 - , genaue numerische 376

- , genaue numerische Werte 371
- , hinzufügen benutzerdefinierter 385
- , IDENTITY 376
- , image 378
- , int 374
- , löschen benutzerdefinierter 385
- , money 382
- , numeric 375
- , Parameter von gespeicherten
Prozeduren 639
- , Rangfolge 404
- , Rangfolge der Operatoren 404
- , real 375
- , RowGUID 377f.
- , rowversion 382
- , smalldatetime 380
- , smallint 374
- , smallmoney 382
- , sp_addtype 577
- , sql_variant 379
- , Synonyme 371, 383
- , table 380
- , timestamp 382
- , tinyint 374
- , ungefähre numerische 375
- , ungefähre numerische Werte 371
- , Unicode 371, 384
- , uniqueidentifier 378
- , varbinary(n) 373
- , varchar(n) 372
- , Währungsdaten 382
- , Währungswerte 371
- , Zeichenfolgen 371f.
- , Zeichenfolgen fester Länge 372
- Datentypfamilien 379
- DATEPART 411
- datepart 410
- datetime 371
- Datum und Zeit 371
- , datepart 410
- , Ordinalwert 380
- Datumseinheit 410
- Datumsfunktionen 410
- DAY 411
- db_accessadmin 222, 242
- db_backupoperator 243
- db_datareader 222, 244
- db_datawriter 222, 244
- db_ddladmin 222, 243
- db_denydatareader 222, 244
- db_denydatawriter 222, 244
- db_dumpoperator 222
- DB_ID 412
- DB_NAME 412
- db_owner 222, 241
- db_securityadmin 222, 243
- DBA 216
- DB-Bibliotheksoptionen 108
- DBCC 241, 301
- , Ergebnisse in Tabellenform 303
- DBCC CHECKALLOC 305
- DBCC CHECKCATALOG 304
- DBCC CHECKDB 301
- DBCC CHECKTABLE 305
- DBCC DBREINDEX 534, 554
- DBCC FREEPROCCACHE 238
- DBCC INDEXDEFRAG 534, 556
- DBCC LOG 494
- DBCC SHOWCONTIG 535, 552
- DBCC SHRINKDATABASE 165, 171
- DBCC SHRINKFILE 165, 172
- , Datei-ID ermitteln 172
- DBCC UPDATEUSAGE 551
- DBCC USEROPTIONS 517
- dbcreator 219, 239
- dbo 212, 236, 244
- DBOO 246
- DDL 241
- Deadlock 508, 514
- decimal 371
- DEFAULT 250, 466, 580
- , Beispiele 599
- , Einschränkungen 598
- Default to local cursor 164
- DEGREES 406
- Deinstallation 89
- Deklarative Integrität 574
- Deklarative referenzielle Integrität 256
- Delegierung von
Sicherheitskonten 871

- DELETE 470
- , Suchtabellen 471
- , WHERE 471
- DELETE-Anweisung 647
- DENY 251, 258
- DESC 428
- DHTML (Dynamic HTML) 866
- Dienste
- , Microsoft Search 565
- , SQL Server-Agent 755
- Dienstkonten 66
- Dienst-Manager 101
- , Abrufintervall 104
- , Optionen 104
- , Verbinden 104
- Dienstprogramme
- , console 289
- , Rebuild Master 141, 361
- DIFFERENCE 407
- Dirty Read 510
- DISK INIT 240
- DISK MIRROR 240
- DISK REFTT 240
- DISK REINIT 240
- diskadmin 219, 240
- DISTINCT 425
- distribution 286, 295
- Document Type Definition (DTD) 867
- Domänenintegrität 575
- , benutzerdefinierte Datentypen 576
- DRI (deklarative referenzielle Integrität) 256
- DROP DEFAULT 583
- DROP PROCEDURE 638
- DROP RULE 588
- DROP TABLE 390
- DROP VIEW 632
- DROP_EXISTING 536
- DTD (Document Type Definition) 867
- DTS
- , Assistenten 819
- , COM 814
- , Datapump 818
- , Dateiverbindungen 818
- , Datenquellenverbindungen 818
- , Pakete 814
- , Paketkomponenten 816
- , Paketsicherheit 816
- , Rangfolgeeinschränkungen 817
- , Schnittstellen 814
- , Schrittoobjekte 817
- , Taskobjekte 816
- , Tasks 814, 816
- , Verbindungsobjekte 818
- , Workflows 816
- DTS-Designer 814
- DTS-Export-Assistent 819
- DTS-Import/Export-Assistenten 819
- DTS-Pakete, Replikation 694
- dump device 298
- Dumps 282
- Duplex (Festplatten) 832
- Duplexing 276
- Durchsuchenmodus 625
- E**
- E/A-Operationen
- , logische 544
- , optimieren 544
- , physische 544
- , Scatter-Gather 50
- Eigenschaften, IDENTITY 679
- Eigenschaftsblätter 128
- Einfügen
- , INSERT 461
- , INSERT VALUES 463
- , mit gespeicherten Prozeduren 468
- , SELECT 466
- , Standardwerte 466
- Einschränkungen
- , abschalten 615
- , ALTER TABLE 596
- , ANSI 596
- , CHECK 599
- , CREATE TABLE 596
- , DEFAULT 598
- , Enterprise Manager 608
- , Fremdschlüssel 603
- , kaskadierende 606
- , löschen 607

- , NOCHECK 614
- , NOCHECK ALL 615
- , Primärschlüssel 601
- , Spaltenebene 597
- , Tabellenebene 597
- , UNIQUE 602
- , zurückweisen 614
- Einzelbenutzermodus 139f., 162
- Einzelbesitzerketten 266
- ELEMENTS 888
- ELSE 499
- E-Mail 755
- , Adressformatierung für Pager 787
- , Aufträge 778
- , Replikation 735
- , senden 769
- Endlosschleifen 507
- English Query 49, 76
- Enterprise Manager 124
- , Anweisungsberechtigungen 253
- , benutzerdefinierte Datentypen 578
- , Benutzernamen hinzufügen 207
- , Datentypen erstellen 385
- , Eigenschaftsblätter 128
- , Einschränkungen 608
- , Fenster Aktuelle Aktivität 848
- , Index verwalten 559
- , Konsolenstruktur 127
- , Objektberechtigungen 260
- , Prozessor konfigurieren 841
- , Regeln 590
- , Replikation einrichten 705
- , Replikationsmonitor 712
- , Sicherungen 317
- , Sicherungsmedien 300
- , Speicher konfigurieren 843
- , Standards 584
- , Tabellen erstellen 389
- , Tabellen löschen 390
- , Tabelleneigenschaften 462
- , Volltextindex erstellen 566
- , Wiederherstellung 352
- Entitätsintegrität 576
- Entwicklungstools 64
- Ereignisprotokolle 88
- Ergebnisse, Sortieren 427
- Erkennung von zerrissenen Seiten 162
- Errorlog 88
- Erweiterungen
- , BCP-Dateien 800
- , Datendateien 154
- , Protokolldateien 154
- Eskalationsschwelle 511
- EXEC PROCEDURE 641
- Exit 136
- Exklusive Sperre 485, 508
- EXP 406
- expire_date 312
- EXPLICIT 890
- Explizite Transaktionen 485
- , Regeln 488
- Extensible Markup Language (XML) 866
- Extensible Stylesheet Language (XSL) 867
- F**
- Failovercluster 55
- Familien, Datentypen 379
- FAT 51, 280
- , Berechtigungen 801
- FAT16 280
- FAT32 51, 280
- , Berechtigungen 801
- Fehlerbehebung, Replikation 745
- Fehlermeldungen, Schweregrad 759
- Fehlerprotokoll 88
- Fehlerprotokolldatei 140
- Fehlerverdächtig 358
- Feldabschlusszeichen 807
- Festplatten 276
- , Duplexing 276
- , Leistungsindikatoren 846
- , Optimierung 832
- , Spiegeln 276
- , Striping 276
- , Systemmonitor 846
- fillfaktor 533
- Firewall 694
- float 371
- FLOOR 406
- FOR ATTACH 335

- FOR LOAD 290
- FOR XML 886
- , ELEMENTS 888
- , EXPLICIT 890
- , XMLDATA 890
- FOR...AFTER 646
- Formatdateien
- , BCP 806
- , Beispiel 810
- , Datentypen 808
- Fremdschlüsseinschränkungen 603
- , ON DELETE 604
- , ON UPDATE 604
- FTP, Snapshotordner 695
- Füllfaktor 533
- , pad_index 535
- FULL 293
- FULL OUTER JOIN 444
- FUNCTION 250
- Funktionen
- , ABS 405
- , ACOS/ASIN 405
- , Aggregate 429
- , ASCII 407
- , ATAN 405
- , ATN2 405
- , AVG 430
- , benutzerdefinierte 654
- , Berechtigungen 265
- , CEILING 406
- , CHAR 407
- , CHARINDEX 407
- , COALESCE 412, 504
- , COL_LENGTH 412
- , COL_NAME 412
- , CONVERT 414
- , COS 405
- , COT 406
- , COUNT 430
- , CUBE 436
- , Cursor 655
- , DATALENGTH 412
- , DATEADD 410
- , DATEDIFF 410
- , DATENAME 411
- , DATEPART 411
- , Datum 410
- , Datum und Zeit 410
- , DAY 411
- , DB_ID 412
- , DB_NAME 412
- , DEGREES 406
- , DIFFERENCE 407
- , EXP 406
- , FLOOR 406
- , GETANSINULL 412
- , GETDATE 411
- , HOST_ID 412
- , HOST_NAME 412
- , IDENT_INCR 412, 592
- , IDENT_SEED 413, 592
- , INDEX_COL 413
- , Inline 655
- , ISDATE 413
- , ISNULL 413
- , ISNUMERIC 413
- , LEFT 408
- , LOG 406
- , LOWER 408
- , LTRIM 408
- , mathematische 405
- , MAX 430
- , mit mehreren Anweisungen 658
- , MONTH 411
- , NCHAR 408
- , nicht deterministische 655
- , NILADIC 598
- , NULLIF 413, 505
- , OBJECT_ID 413, 551
- , OBJECT_NAME 413
- , PATINDEX 408
- , PI 406
- , POWER 406
- , QUOTENAME 408
- , RADIANS 406
- , RAND 406
- , REPLACE 408
- , REPLICATE 408
- , RETURNS 654
- , REVERSE 408
- , RIGHT 408
- , ROLLUP 436

- , ROUND 406
- , RTRIM 408
- , SCOPE_IDENTITY 593
- , SIGN 406
- , SIN 405
- , Skalar- 655
- , SOUNDEX 408
- , SPACE 408
- , SQAURE 406
- , SQRT 406
- , STATS_DATE 413, 550
- , STDEV 430
- , STDEVP 430
- , STR 408
- , STUFF 408
- , SUBSTRING 409
- , SUM 430
- , SUSER_ID 413
- , SUSER_NAME 413
- , System 412
- , Tabellenrückgabe 655
- , TAN 406
- , TOP 430
- , UNICODE 409
- , UPPER 409
- , USER_ID 413
- , USER_NAME 413
- , VAR 430
- , VARP 430
- , YEAR 411
- , Zeichenfolge 407

G

- Gemeinsame Sperre 507
- Gemischter Modus 192
- Geplante Tabellenaktualisierung 680
- Geschäftsregeln
 - , Gespeicherte Prozeduren 633
 - , kapseln 650
- Gespeicherte Prozeduren,
 - Berechtigungen 264
- gespeicherte Prozeduren Siehe auch Prozeduren
- GETANSINULL 412
- GETDATE 411
- Gleichheitsverknüpfungen 440

- GLOBAL 164
- GO 482
- GRANT 251, 256, 512
- Groß-/Kleinschreibung
 - , Benutzernamen 83
 - , Installation 71
 - , Sortierung 70
 - , URL-Abfragen 873
- GROUP BY 431
- GROUP BY ALL 432
- Gruppen
 - , einrichten 198
 - , gespeicherte Prozeduren 637
- Gruppiertes Index 525
- Gültigkeitsbereiche,
 - Identitätswerte 593
- guest 213
- GUID 799
- , erzeugen mit NEWID 679

H

- Hardware-Kompression 290
- Hauptspeicher 830
- HAVING 431
- Heap 525, 536
- Herkunft 684
- HOLDLOCK 510
- HOST_ID 412
- HOST_NAME 412
- HTML (Hypertext Markup Language) 866

I

- IANA (Internet Assigned Number Authority) 73
- IDENT_INCR 412, 592
- IDENT_SEED 413
- Identitätsspalten
 - , @IDENTITY 592
 - , IDENTITYCOL 592
 - , manuelles Einfügen 594
- Identitätswechsel, Abonnenten 710
- Identitätswerte
 - , Gültigkeitsbereich 593
 - , letzter Wert 594
 - , SCOPE_IDENTITY 594

- IDENTITY 376, 590, 679
- , @@IDENTITY 592
- , Ausgangswert 591
- , identity_insert 594
- , mögliche Datentypen 591
- , Schrittweite 591
- , UNIQUEIDENTIFIER 595
- IDENTITYCOL 592
- IF EXISTS 500
- IF...ELSE 499
- IGNORE_DUP_KEY 537
- IIS 868
- , Anmeldeinformationen 870
- , virtuelles Verzeichnis 869
- IIS (Internet Information Server) 695
- image 371
- Implizite Transaktionen 490
- IN 418
- Index
- , absteigend 530
- , aufsteigend 530
- , Ausführungsplan 543
- , Auswahl 563
- , B-Baum 525
- , Blattebene 529
- , DBCC DBREINDEX 554
- , DBCC SOWCONTIG 552
- , DBCC UPDATEUSAGE 551
- , E/A-Operationen 544
- , eindeutig 529
- , einspaltig 530
- , Enterprise Manager 559
- , fillfactor 533
- , gruppiert 527, 564
- , gruppierter 525
- , Heap 525
- , löschen 536
- , mehrdeutig 529
- , mehrspaltig 530
- , nicht gruppiert 529, 565
- , Nutzung 538
- , Nutzung überprüfen 539
- , Optionen 527, 533
- , pad_index 535
- , Seitenteilung 535
- , SET SHOWPLAN_ALL ON 539
- , SET SHOWPLAN_TEXT ON 539
- , SET STATISTICS IO ON 544
- , Sichten 556
- , Statistiken abschalten 549
- , statistischer Blob 548
- , Struktur 525
- , Volltextindex 565
- , Zeilenlokator 567
- , zusammengesetzter 530
- INDEX_COL 413
- Index-ID, Integritätsprüfung 306
- Indexoptimierungs-Assistent 858
- Indexseiten, Sperren 508
- Indizes, neu organisieren 789
- Indizierte Sichten,
 - Einschränkungen 558
- INFORMATION_SCHEMA 463
- Inlinefunktionen 655
- , Tabellenrückgabe 658
- INNER JOIN 440
- INSERT 461
- , Berechtigungen 464
- , DEFAULT 466
- , gespeicherte Prozeduren 468
- , Standardwerte 466
- INSERT INTO 463
- , sp_spaceused 469
- Installation
- , benannte Instanzen 86
- , benutzerdefinierte 60
- , Clientkonnektivität 63
- , Codebeispiele 64
- , Entwicklungstools 64
- , Komponenten auswählen 61
- , MDAC (Microsoft Data Access Components) 59
- , Onlinedokumentation 63
- , Ordner 78
- , Problembehebung 87
- , Serverkomponenten 62
- , Standarddatenbanken 81
- , Versionsumstellung 86
- , Verwaltungstools 62
- , Windows-Registrierung 80
- Instanzen
- , benannte 84
- , Standard- 84
- instcat.sql 497

INSTEAD OF 644, 653
 instpubs.sql 479
 int 371
 Integrität
 –, deklarative 574
 –, prozedurale 574
 Integritätsprüfung 302
 –, Index-ID 306
 Intel, Itanium 47
 Internet
 –, Replikation 695
 –, Snapshotordner 695
 Internet Assigned Number Authority
 (IANA) 73
 Internet-E-Mail 763
 Internet-Mail 765
 IS NOT NULL 421
 IS NULL 421
 ISDATE 413
 ISNULL 413
 ISNUMERIC 413
 Isolation 484
 Isolationsstufen 517
 –, DBCC USEROPTIONS 517
 –, REPEATABLE READ 510
 –, SET TRANSACTION ISOLATION
 LEVEL 517
 ISP (Internet Service Provider) 766
 ISQL 133
 ISQLW.exe 112
 Itanium 47

K

Kana 384
 Kartesisches Produkt 442
 Katastrophenplan 288
 Kennwörter 192
 Kennwortrichtlinien 196
 Kerberos 871
 KILL 239
 Klammern, Rangfolge der
 Operatoren 405
 Kompatibilität, Sperren 513
 Kompatibilitätsebene 163
 Konfiguration
 –, erweiterte Optionen 837

–, minimal konfigurierter Modus 139
 –, priority boost 839
 –, Show Advanced Options 837
 –, Wiederherstellungsintervall 333
 Konfigurationsoption 511
 Konfigurieren von SQL Mail 768
 Konflikthanzeige 63
 Konfliktlöser 673
 Konsistenz 484
 Konsistenzprüfungen 286
 Konsolenstruktur 127
 Konventionen 25
 Konvertierung, ANSI in OEM 108
 Kostenlimitoption 844
 Kostenschwellwert für Parallelität 841

L

LDF 154
 Leerzeichen 876
 LEFT 408
 LEFT OUTER JOIN 444
 Leistungsindikatoren 741, 835, 845
 –, Festplatte 846
 –, sqlagent (Instanz) 845
 –, sqlservr (Instanz) 845
 Leistungsstatuswarnung 787
 Lesesperren 507
 Lesezeichen 876
 LIKE 420
 Listen, NOT 419
 Literale, SELECT-Anweisung 402
 Lizenzierung 51
 –, Personal Edition 52
 –, Pro-Arbeitsplatz 51
 –, Pro-Prozessor 51
 LOCKS 511
 Löschen
 –, kaskadierendes 648
 –, Suchtabellen 471
 LOG 406
 Lookup-Tabellen Siehe Suchtabellen
 LOWER 408
 LTRIM 408

M

Macintosh 74
makepipe 137
Manuelle Wiederherstellung 334
Markierung, Wiederherstellung 347
master 81, 293
–, Verlust 359
–, Wiederherstellung 359
MAX 430
max degree of parallelism 840
max server memory 843
MDAC (Microsoft Data Access Components) 59, 74
MDF 154
media retention 312
Medienfamilien 285, 340
Mediensatz 284
Meldungen
–, benutzerdefinierte 498
–, PRINT 498
Merge publish 165
Merge-Agent 682
Mergereplikation 165, 668
–, Funktionsweise 682
–, ROWGUIDCOL 728
–, uniqueidentifizier 728
Metadaten, Durchsuchenmodus 625
Microsoft Access 497, 696
Microsoft Data Access Components 74
Microsoft Excel 497
Microsoft Exchange Server 763
Microsoft Management Konsole 74
Microsoft Outlook 762
Microsoft Proxy Server
–, Internet 695
–, Replikation 695
Microsoft Repository 815
Microsoft Search 565
min memory per query-Option 843
min server memory 842
Minimal konfigurierter Modus 139
MIS (Managementinformations-system) 671
Mitglieder von 'db_owner', 'dbcreator' oder 'sysadmin' 162
MMC 74

model 82, 150, 293, 295
Modus
–, systemeigener 805
–, Zeichenmodus 805
money 371
MONTH 411
MOVE 345
msdb 82, 293f., 340
–, Wartungsplanverlauf 793
MSDTC (Microsoft Distributed Transaction Coordinator) 74, 495
MSReplication_queue 674
Multiprotokoll 50, 73, 190
Multiserver 780

N

Named Pipes 49, 72, 190
–, makepipe 137
–, readpipe 137
–, überprüfen 137
Natürliche Verknüpfungen 441
NCHAR 408
nchar 371
NET SEND 735
–, Aufträge 778
NET STOP 359
Netzwerk
–, Datendurchsatz 834
–, Optimierung 831
–, Systemmonitor 846
–, überwachen 836
Netzwerkbibliothek 72, 109, 190
–, AppleTalk 74
–, Banyan VINES 74
–, Multiprotokoll 73
–, Named Pipes 72
–, NWLink IPX/SPX 74
–, Standard 105
–, TCP/IP-Sockets 73
Netzwerkconfiguration 109
–, Winsockproxy 112
Neuverleger 690
NEWID, GUID 679
Nicht deterministische Funktionen 655
Nicht protokollierte Operationen 291

- NILADIC 598
- NOCHECK 614
- NOCHECK ALL 615
- NOLOCK 510
- NORECOVERY 346
- Normalformen 39
- Normalisieren 38
- Northwind 83
- NOT 419
 - , Abfrageoptimierer 420
- NOT FOR REPLICATION 679
- NOT NULL 372
- Novell NetWare 192
- ntext 371
- NTFS 50f.
 - , Berechtigungen 801
- NULLIF 413, 505
- NULL-Werte 465
 - , ändern in 0 505
- NULL-Zulässigkeit 372
- numeric 371
- nvarchar 371
- NWLink IPX/SPX 74
- O**
- OBJECT_ID 413, 551
- OBJECT_NAME 413
- Objektberechtigungen 248, 255
 - , zuweisen 256
- Objektbesitzer 236
- Objekte
 - , abhängige 265
 - , globale temporäre 369
 - , löschen 579
 - , relationaler Datenbanken 33
 - , temporäre 369
- ODBC 108
- ODBCPing 133, 137
- OEM 108
- Offline 165
- OLAP 49, 76
 - , Replikation 666
- OLE DB, DTS 818
- ON DELETE 604
- ON UPDATE 604
- Online Analytical Processing 49
- Onlinedokumentation 63, 95
- Onlinereorganisation 556
- Operationen, nicht protokollierte 291, 378
- Operatoren 780
 - , arithmetische 404
 - , Konfigurieren 780
 - , NOT 419
 - , Rangfolge 404
 - , Rangfolge der Datentypen 404
 - , Replikation 735
 - , ROLLUP 436
 - , UNION 452
 - , Vergleich 416
- Optimieren, E/A-Operationen 544
- Optimierung
 - , Abfragen 844
 - , Ablaufverfolgung 850
 - , Affinitätsmaske 840
 - , cost threshold for parallelism 841
 - , Datendurchsatz 834
 - , DHCP 834
 - , diskperf -y 836
 - , Duplexbetrieb 832
 - , Enterprise Manager 841, 843
 - , Fenster Aktuelle Aktivität 848
 - , Festplatten 832
 - , Hauptspeicher 830, 836
 - , Höherstufung der Priorität 839
 - , Indexoptimierungs-Assistent 858
 - , Kostenlimitoption der Abfragekontrolle 844
 - , Kostenschwellwert für Parallelität 841
 - , Logischer Datenträger 836
 - , max degree of parallelism 840
 - , max server memory 843
 - , min memory per query, Option 843
 - , min server memory 842
 - , Netzwerk 831
 - , parallele Abfrageverarbeitung 840
 - , physikalischer Datenträger 837
 - , physischen Arbeitsspeicher für SQL Server reservieren 843
 - , Prozessor 831, 836
 - , Prozessorzeit (%) 836

- , query governor cost limit 844
- , Seitenfehler 836
- , set working set size 843
- , sp_configure 837
- , Speicher 841
- , SQL Server Profiler 849
- , Systemmonitor 835, 845
- , Verfügbare Bytes 836
- , Windows 2000 834
- , WINS 834
- Optionen
 - , Dienst-Manager 104
 - , Sicherung bei Abschluss
 - überprüfen 339
- Oracle 696
- ORDER BY 427
- Ordinalwert 380
- OS/2 31
- OSQL 133
- osql 398
 - , Exit 136
 - , Reset 136
- OUTER JOIN 444
- OUTPUT 639

P

- pad_index 535
- Pager
 - , Adressformatierung 787
 - , Aufträge 778
 - , E-Mails 787
 - , Präfix 787
 - , Suffix 787
- PAGLOCK 510
- Pakete, DTS 814
- Parameter
 - , benannte 639
 - , Datentypen 639
- Paritätsinformationen 279
- Partitionen
 - , horizontale 677
 - , vertikale 677
- PATINDEX 408
- Personal Edition, Lizenzierung 52
- PI 406

- POP3 (Post Office Protocol 3) 766
- POWER 406
- Primärschlüsseleinschränkungen 601
- PRIMARY 175
- PRINT 498
- Priorität, SQL Server 839
- priority boost 839
- Pro-Arbeitsplatz-Lizenzierung 51
- Problembehebung, Installation 87
- PROCEDURE 250
- processadmin 219, 239
- Projektionen 628
- Pro-Prozessor-Lizenzierung 51
- Protokolldateien
 - , Erweiterungen 154
 - , Installationsprobleme 88
- Protokolllese-Agent 682
- Protokollsequenznummern 338
- Protokollverschlüsselung 107
- Prozedurale Integrität 574
- Prozeduren
 - , Abfragestruktur 633
 - , Abhängigkeiten 637
 - , Auflösung 633
 - , automatisch ausführen 642
 - , Cache 633
 - , Code anzeigen 634
 - , CREATE PROCEDURE 634
 - , DROP PROCEDURE 638
 - , erstellen 634
 - , EXEC PROCEDURE 641
 - , Geschäftsregeln 633
 - , gespeicherte 632
 - , Gruppen 637
 - , löschen 638
 - , Neukompilierung erzwingen 642
 - , OUTPUT 639
 - , Parameter 638
 - , Regeln beim Erstellen 634
 - , remote gespeicherte 469, 643
 - , RETURN 640
 - , sp_procoption 643
 - , sp_recompile 642
 - , Text der Prozedurdefinition 636
 - , Trigger 644

- , verzögerte Namensauflösung 634
- , WITH ENCRYPTION 643
- , WITH RECOMPILE 641
- Prozessor
 - , Optimierung 831
 - , Systemmonitor 845
 - , überwachen 836
- Prozessorzeit (%) 836
- Prüfpunkte 283, 333, 494
 - , automatische 494
 - , manuelle 494
- public 218, 248
- Publikationserstellungs-Assistent 713
- Publikationsserver,
 - Wiederherstellung 746
- Published 165
- Publizierungs- und Verteilungskonfigurations-Assistent 705
- pubs 82
 - , neu installieren 479
- Pullabonnements 681
 - , anonyme Abonnements 681
 - , erstellen 720
- Pushabonnements 680

Q

- Query Analyzer 112
 - , Abfragen 116
 - , Ausführungsmodus 117
 - , Ausführungspläne 543
 - , Hilfe 123
 - , Optionen 120
 - , Serverablaufverfolgung anzeigen 117
 - , Symbolleisten 116
- query governor cost limit 844
- QUOTED_IDENTIFIER 483
- QUOTENAME 408

R

- RADIANS 406
- RAID 175, 278
 - , hardwarebasiert 280
 - , softwarebasiert 280
- RAM Siehe Hauptspeicher
- RAND 406

- Rangfolge
 - , arithmetische Operatoren 404
 - , Klammern 405
- Rangfolgeneinschränkungen 817
- RAW 886
- READ COMMITTED 517
- READ UNCOMMITTED 517
- READPAST 510
- readpipe 137
- READUNCOMMITTED 510
- real 371
- Rebuild Master 141, 361
- rebuildm.exe 141, 361
- RECONFIGURE 238
- RECOVERY INTERVAL 494
- Redundant Array of Inexpensive Disks 278
- Referenzielle Integrität 575, 648, 652
- Regeln 586
 - , binden 588
 - , Bindung aufheben 588
 - , löschen 588
 - , sp_bindrule 588
 - , sql_variant-Vergleiche 379
- Registrieren, Server 124
- Registrierung, wiederherstellen 142
- Registrierungsschlüssel 331
- Rekursive Trigger 162
- Relationen 32
- remote gespeicherte Prozeduren 469
- REMOTE_PROC_TRANSACTIONS 489
- Remoteserver 496
- Remoteverteiler 689
- Reorganisation, Online 556
- REPEATABLE READ 510, 517
- REPLACE 345, 408
- REPLICATE 408
- Replikation 286, 666
 - , 2PC 676
 - , Abonnement erstellen 719
 - , Abonnent 677
 - , Abonnententreiber 696
 - , Agent 682, 731, 744
 - , Agentverlauf 726

- , Aktualisieren über eine Warteschlange 672
- , Anfangssnapshot 682
- , Anfangssynchronisation 726
- , anonyme Abonnements 681
- , Artikel 677
- , Aufträge 744
- , benachrichtigen 735
- , Cleanup 735
- , DTS-Pakete 694
- , E-Mail 735
- , Empfehlungen 695
- , Fehlerbehebung 745
- , Firewall 694
- , geplante Tabellenaktualisierung 680
- , Herkunft 684
- , heterogene Umgebung 696
- , IDENTITY 679
- , IIS (Internet Information Server) 695
- , installieren 704
- , Internet 695
- , Konfliktlöser 673
- , Merge 668
- , Merge-Agent 682
- , Microsoft Access 696
- , Microsoft Proxy Server 695
- , Monitor 731
- , NET SEND 735
- , Neuverleger 690
- , Nicht SQL Server-Abonnent 697
- , OLAP 666
- , Operatoren 735
- , Oracle 696
- , Partitionen 677
- , Protokolllese-Agent 682
- , Publikation 680
- , Publikation erstellen 712
- , publizierender Abonnent 690
- , Pullabonnements 681
- , Pushabonnements 680
- , Server einrichten 703
- , Sicherheit 702
- , Skripts 739
- , Snapshot 670
- , Snapshot-Agent 682
- , Snapshotordner 695
- , sofortiges Aktualisieren 673
- , Synchronisation 725
- , Systemmonitor 741
- , Szenarios 689
- , Testen 726
- , Transaktion 671
- , Transformieren publizierter Daten 694
- , Typen 668
- , Überlegungen 728
- , Überprüfen 726
- , Überwachen 730
- , Verlauf 730
- , Verleger 677
- , Verlegereigenschaften 736
- , Verteiler 677
- , Verteilereigenschaften 736
- , Verteilungs-Agent 682
- , Verwalten 730
- , Verwaltung 744
- , Voraussetzungen 702
- , Warnungen 730, 744
- , Warteschlangenlese-Agent 682
- , Wiederherstellung 746
- , zentraler Abonnent 691
- , zentraler Verleger 689
- , zentraler Verleger mit Remoteverteiler 689
- Replikationsleistungsverlauf 736
- Replikationsmonitor, Enterprise Manager 712
- Replikationsszenarios 689
- Replikationsverlauf, Aufbewahrungsdauer 735
- Repository 815
- Reservierungssystem 693
- Reset 136
- Ressource Kit 754
- RESTORE 341
- RESTORE DATABASE 241
- RESTORE FILELISTONLY 337f.
- RESTORE HEADERONLY 337f.
- RESTORE LABELONLY 337
- RESTORE LOG 241, 347
- restorefile 340

restorefilegroup 340
 restorehistory 340
 retain_days 312
 RETURN 640
 RETURNS 654
 REVERSE 408
 REVOKE 251, 257
 RID (Row Identifier) 508
 RIGHT 408
 RIGHT OUTER JOIN 444
 Rollback 331
 ROLLBACK TRAN 486
 ROLLBACK TRANSACTION 487
 ROLLBACK WORK 486
 Rollen 217
 –, Mitglieder hinzufügen 223
 –, Mitglieder löschen 224
 –, öffentliche 218, 248
 –, public 218
 –, Server herunterfahren 218
 –, serverweite 218
 Rollforward 331
 ROLLUP 436
 ROUND 406
 RowGUID 377
 ROWGUIDCOL 378
 –, Mergereplikation 728
 ROWLOCK 510
 rowversion 371, 382
 RTRIM 408
 Rückgängigdatei 341, 343
 RULE 250

S

sa 83, 240
 SAVE TRAN 486
 SAVE TRANSACTION 487
 savepoint 486
 Scandichte 553
 Scatter-Gather-E/A 50
 SCHEMABINDING 558
 Schleifen
 –, Endlos 507
 –, WHILE 506
 Schlüsselwörter
 –, ALL 249

–, AS 402
 –, char 373
 –, GLOBAL 164
 –, IDENTITYCOL 592
 –, varchar 373
 Schnittstellen, DTS 814
 Schreiben, synchrones 331
 Schreibgeschützt 162
 Schritte, parallele Ausführung 817
 Schrittobjekte 817
 –, Rangfolgeeinschränkungen 817
 Schrittweite, IDENTITY 591
 Schweregrad 759
 SCOPE_IDENTITY 593f.
 –, Gültigkeitsbereiche 593
 SEARCHABLE 563
 Security Support Provider Interface
 (SSPI) 205
 securityadmin 219, 238
 Seitendichte 553
 Seitenteilung 535
 Selbstverknüpfungen 446
 SELECT 398
 –, BETWEEN 417
 –, COMPUTE 433
 –, CUBE 437
 –, Datumsfunktionen 411
 –, DISTINCT 425
 –, einfügen 466
 –, GROUP BY 431
 –, HAVING 431
 –, IN 418
 –, korrelierte Unterabfrage 449
 –, LIKE 420
 –, NULL-Werte 422
 –, ORDER BY 427
 –, ROLLUP 437
 –, Selbstverknüpfungen 446
 –, Sortieren 427
 –, Unterabfragen 447
 –, WHERE 416
 SELECT * FROM-Anweisung 399
 SELECT INTO 450
 –, Trigger 653
 SELECT INTO/BULK COPY, Erstellen
 permanenter Tabellen 451

- Select into/bulkcopy, BCP 809
- Senden, E-Mail 769
- SEQUENCE 31
- SERIALIZABLE 510, 517
- Server
 - , herunterfahren 218
 - , registrieren 124
 - , remote 496
- serveradmin 218, 238
- Serverbenutzernamen, verwalten 219
- Serverkonfiguration
 - , Aufbewahrungsdauer 312
 - , media retention 312
- Serveroptimierung 834
- Serverprozeduren,
 - sp_addsrvrolemember 219
- Serverrolle 218
 - , Berechtigungen 237
 - , bulkadmin 219
 - , dbcreator 219, 239
 - , diskadmin 219, 240
 - , processadmin 219, 239
 - , securityadmin 219, 238
 - , serveradmin 218, 238
 - , setupadmin 219, 238
 - , sysadmin 218, 237
- SESSION_USER 598
- SET IMPLICIT_TRANSACTIONS
 - ON 490
- SET
 - REMOTE_PROC_TRANSACTION
 - S 496
- SET SHOWPLAN_ALL ON 539
- SET SHOWPLAN_TEXT ON 539
- SET STATISTICS IO ON 544
- SET TRANSACTION ISOLATION
 - LEVEL 517
- set working set size 843
- setupadmin 219, 238
- Show Advanced Options 837
- SHUTDOWN 238, 494
- Sicherheit
 - , Delegierung 871
 - , gemischter Modus 192
 - , integrierte 50
 - , Kerberos 871
 - , Modell 188
 - , Skripts 270
 - , syspermissions 235
 - , sysprotects 235
 - , virtuelles Verzeichnis 869
- Sicherheitsmodi 191
- Sicherung
 - , Aufbewahrung 289
 - , Bandgeschwindigkeit 287
 - , parallele 284
 - , parallele verteilte 307
 - , Replikation 286
 - , Systemdatenbanken 286
 - , Wiederherstellungsmodelle 290
- Sicherungskopien, Aufbewahrung 287
- Sicherungsmedien 281
 - , erstellen 296
 - , verwenden 306
- Sicherungsmediensätze 340
- Sicherungspunkte 486
- Sicherungssatz 284, 337
 - , Überprüfung 339
- Sichten 622
 - , ändern 631
 - , ALTER VIEW 631
 - , berechnete Spalte 630
 - , Berechtigungen 259, 264
 - , Daten ändern 631
 - , DROP VIEW 632
 - , Durchsuchenmodus 625
 - , erstellen 624
 - , indizierte 556
 - , INFORMATION_SCHEMA 463
 - , Informationen 627
 - , löschen 632
 - , Metadaten 625
 - , Projektionen 628
 - , Regeln und Restriktionen 626
 - , SCHEMABINDING 558
 - , sp_helptext 627
 - , Typen 628, 630
 - , Verknüpfungen 629
 - , Vorteile 623
- SID 799
- SIGN 406
- SIMPLE 293

- SIN 405
- Skalarfunktionen 655
- Skripts 483
- , Sicherheit 270
- , Transact-SQL 122
- smalldatetime 371
- smallint 371, 591
- smallmoney 371
- SMTP (Simple Mail Transfer Protocol) 766
- Snapshot, Agentverlauf 726
- Snapshot-Agent 682
- Snapshotordner 695
- Snapshotreplikation 670
- , aktualisierende Abonnenten 674
- , Verteilungsdatenbank 677
- Sonderzeichen, URL 876
- Sortieren
- , Abfrageergebnisse 427
- , absteigend 428
- , ASC 428
- , aufsteigend 428
- , DESC 428
- , SELECT 427
- Sortierung 372
- , Abwärtskompatibilität 71
- Sortierungskennzeichner 70
- Sortierungsname 152
- SOUNDEX 408
- sp_addalias 215
- sp_addapprole 228
- sp_addgroup 210
- sp_addlinkedserver 497
- sp_addlinkedsrvlogin 497
- sp_addlogin 194
- sp_addrole 223
- sp_addrolemember 223
- sp_addsrvrolemember 219
- sp_addtype 385, 577
- sp_addumpdevice 240, 297
- sp_adduser 210
- sp_attach_db 336
- sp_autostats 549
- sp_bindefault 581
- sp_bindrule 588
- sp_change_users_login 799
- sp_changedbowner 216
- sp_changegroup 210
- sp_changeobjectowner 242, 247
- sp_configure 218, 238, 837
- , fillfactor 533
- sp_dbfixedrolepermission 240
- sp_dboption 163, 494
- sp_defaultdb 196
- sp_defaultlanguage 196
- sp_denylogin 202
- sp_depends 250, 637
- sp_detach_db 336
- sp_diskdefault 240
- sp_dropalias 216
- sp_dropapprole 228
- sp_dropdevice 240, 298
- sp_dropgroup 210
- sp_droplogin 196
- sp_droprolemember 224
- sp_dropsrvrolemember 219
- sp_droptype 385, 578
- sp_dropuser 210
- sp_fulltext_catalog 242
- sp_fulltext_column 242
- sp_fulltext_database 242
- sp_fulltext_service 238
- sp_fulltext_table 242
- sp_grantdbaccess 210
- sp_grantlogin 202
- sp_helpdb 158
- , Datei-ID ermitteln 172
- sp_helpdevice 299
- sp_helpfilegroup 176
- sp_helpindex 541
- sp_helplogins 196
- sp_helprole 225
- sp_helprolemember 225
- sp_helptext 627, 636, 654
- sp_helpuser 212
- sp_lock 511
- sp_monitor 846
- sp_password 195
- sp_processinfo 511
- sp_processmail 770
- sp_procoption 643
- sp_readerrorlog 239

- sp_recompile 241, 642
- sp_rename 242
- sp_renamedb 174
- sp_revokedbaccess 211
- sp_revokelogin 202, 204
- sp_spaceused 468
- sp_srvrolepermission 237
- sp_tableoption 242
- sp_unbinddefault 582
- sp_unbindrule 588
- sp_who 846
- SPACE 408
- Spalten
 - , Aliasnamen 401
 - , berechnete, in einer Sicht 630
 - , berechneter Ausdruck 386
 - , BLOB (binary large object) 377
 - , Datentypen 370
 - , DEFAULT 580
 - , Domänenintegrität 575
 - , Formatdatei 807
 - , Identitätsspalte 591
 - , Index 563
 - , INSERT 461
 - , NULL-Zulässigkeit 372
 - , Regel erstellen 589
 - , Sortierung 372
 - , Statistiken 550
 - , Tabellen 368
 - , Überschriften in SELECT-Anweisungen 401
 - , UNIQUEIDENTIFIER als Identitätswert 595
 - , VALUES 463
- Spaltenstatistiken 550
- Speicher
 - , AWE 830
 - , Optimierung 841
- Speicherplatz 162
- Sperren 507
 - , Aktualisierung 508
 - , beabsichtigte 508f.
 - , Block 508
 - , Datenseiten 508
 - , Eskalationsschwelle 511
 - , exklusive 485, 508
 - , Fenster Aktuelle Aktivität 513
 - , gemeinsame 485, 507
 - , HOLDLOCK 510
 - , Indexseiten 508
 - , Kompatibilität 513
 - , Lese 507
 - , NOLOCK 510
 - , PAGLOCK 510
 - , READPAST 510
 - , READUNCOMMITTED 510
 - , REPEATABLEREAD 510
 - , ROWLOCK 510
 - , SERIALIZABLE 510
 - , Status 512
 - , Tabelle 509
- , TABLOCK 510
- , TABLOCKX 510
- , Transaktionen 485
- , Überwachung 511
- , UPDLOCK 510
- , Verhalten steuern 509
- , Zeilen 508
- Sperrenausweitung 511
- Sperrentypen 508
- Sperrhinweise 509
 - , Parameter 510
- Spermodus 512
- SQL 31
 - , Dialekte 31
 - , Transact-SQL 31
- SQL Mail 761
 - , Integration 761
 - , Konfigurieren 768
- SQL Query Analyzer Siehe Query Analyzer
- SQL Server 31
 - , Anmeldung 112
 - , Automatisierung 752
 - , Diagnosewerkzeug 139
 - , Leistungsindikatoren 741
 - , Starten als Anwendung 360
 - , virtuelles Verzeichnis 868
 - , XML-Unterstützung konfigurieren 869
- SQL Server 2000 30
 - , Desktop Engine 48

- , Developer Edition 47
- , Enterprise Edition 47
- , Enterprise Evaluation Edition 48
- , Hardware 48
- , Installation 46
- , Internetpublikation 694
- , Lizenzierung 51
- , MSDE 48
- , Personal Edition 47
- , Produktfamilie 46
- , Standard Edition 46
- , Voraussetzungen 48
- SQL Server 6.0 31
- SQL Server 6.5 31
- , Aktualisierungs-Assistent 87
- , deinstallieren 87
- , Versionsumstellung 86
- SQL Server Profiler 849
- , Ablaufverfolgung 850
- , Ablaufverfolgung einrichten 852
- , Ablaufverfolgungsvorlagen 850
- , automatisch ausführen 857
- , filtern 855
- , Indexoptimierungs-Assistent 858
- SQL Server Query Analyzer Siehe Query Analyzer
- SQL Server-Agent 755
- , Komponenten 755
- , Konfiguration 756
- , Warnungssystem 786
- SQL Server-Agent-Dienst 294
- , msdb-Datenbank sichern 294
- SQL Server-Clientkonfiguration 105
- SQL Server-Dienst 193
- SQL Server-Profiler 131
- SQL Server-Registrierungs-Assistent 124
- sql_variant 371, 379
- , Datentypfamilien 379
- , Vergleiche 379
- SQL-92 398
- sqlagent 845
- SQLDiag.exe 145
- SQL-DMO (Database Management Objects) 697
- SQLMaint.exe 142
- SQLServr.exe 139
- sqlstp.log 88
- SORT 406
- SQUARE 406
- Standardinstanzen 84
- Standards
- , binden 581
- , Enterprise Manager 584
- , Regeln 581
- , syscomments 581
- , Systemtabellen 581
- Standardsicherheit 192
- Standardwerte 466
- , ANSI 580
- , löschen 583
- STANDBY 346
- Stapel 135
- , ANSI_NULLS 483
- , QUOTED_IDENTIFIER 483
- Stapelverarbeitung 482
- STATISTICS_NORECOMPUTE 53
- 7
- Statistik
- , Abfrageoptimierer 537
- , abschalten 549
- , automatisch aktualisieren 162
- , automatisch erstellen 163
- , neu berechnen 537
- , Scandichte 553
- , Seitendichte 553
- , STATISTICS_NORECOMPUTE 5
- 37
- , STATS_DATE 550
- STATS_DATE 413, 550
- STDEV 430
- STDEVP 430
- Steuerungsstrukturen 498
- , BEGIN...END 498
- , IF EXISTS 500
- STR 408
- Striping 276
- , Paritätsinformationen 279
- STUFF 408
- Stylesheets 867
- , XSL 867
- Subscribed 165

- SUBSTRING 409
- Suchtabellen
 - , Aktualisieren 476
 - , DELETE 471
 - , Löschen 471
 - , UPDATE 476
- SUM 430
- Superaggregate 436
- SUSER_ID 413
- SUSER_NAME 413
- Sybase 31
- Symbole 416
 - , ! 416
 - , != 416
 - , # 369
 - , \$ 369
 - , () 416
 - , + 407
 - , = 416
 - , @ 369
 - , _ 369
- Synchrones Schreiben 331
- Synonyme 371
- sysadmin 218, 237
- sysalerts 786
- syscomments 581
- sysdevices 296
- sysindexes
 - , korrigieren 551
 - , Nummer des Index 546
- sysindexes-Tabelle 525
- sysjobs 779
- sysjobschedules 779
- sysjobservers 779
- sysjobsteps 779
- sysmembers 225
- sysobjects 581
- syspermissions 235
- sysprocedures-Tabelle 633
- sysprotects 235
- SYSTEM_USER 598
- Systemadministrator 83
- Systemdatenbank 60, 293
 - , Wiederherstellung 362
- Systemfunktionen 412
- Systemkatalog 296, 895
- Systemmonitor 741, 787, 845
 - , diskperf -y 836
 - , Festplatte 846
 - , Leistungsindikatoren 835, 845
 - , Leistungsstatuswarnung 787
 - , Logischer Datenträger 836
 - , Netzwerk 846
 - , Physikalischer Datenträger 837
 - , Prozessor 836, 845
 - , Prozessorzeit (%) 836
 - , Replikation 741
 - , Seitenfehler/s 836
 - , Speicher 836
 - , Verfügbare Bytes 836
- Systemprozeduren 632
 - , Abhängigkeiten 250
 - , sp_addalias 215
 - , sp_addapprole 228
 - , sp_addgroup 210
 - , sp_addlinkedserver 497
 - , sp_addlinkedsrvlogin 497
 - , sp_addrole 223
 - , sp_addrolemember 223
 - , sp_addtype 385, 577
 - , sp_addumpdevice 240, 297
 - , sp_adduser 210
 - , sp_attach_db 336
 - , sp_bindrule 588
 - , sp_change_users_login 799
 - , sp_changedbowner 216
 - , sp_changegroup 210
 - , sp_changeobjectowner 242, 247
 - , sp_configure 218, 238, 837
 - , sp_dbfixedrolepermission 240
 - , sp_dboption 163, 494
 - , sp_defaultdb 196
 - , sp_defaultlanguage 196
 - , sp_denylogin 202
 - , sp_depends 250
 - , sp_detach_db 336
 - , sp_diskdefault 240
 - , sp_dropalias 216
 - , sp_dropapprole 228
 - , sp_dropdevice 240, 298
 - , sp_dropgroup 210
 - , sp_droplogin 196

- , sp_droprolemember 224
- , sp_dropsrvrolemember 219
- , sp_droptype 385, 578
- , sp_dropuser 210
- , sp_fulltext_catalog 242
- , sp_fulltext_column 242
- , sp_fulltext_database 242
- , sp_fulltext_service 238
- , sp_fulltext_table 242
- , sp_grantaccess 210
- , sp_grantdbaccess 210
- , sp_grantlogin 202
- , sp_helpdb 158
- , sp_helpdevice 299
- , sp_helpfilegroup 176
- , sp_helplogins 196
- , sp_helprole 225
- , sp_helprolemember 225
- , sp_helpuser 212
- , sp_lock 511
- , sp_monitor 846
- , sp_password 195
- , sp_processinfo 511
- , sp_processmail 770
- , sp_readerrorlog 239
- , sp_recompile 241
- , sp_rename 242
- , sp_renamedb 174
- , sp_revokedbaccess 211
- , sp_revokelogin 202
- , sp_spaceused 468
- , sp_srvrolepermission 237
- , sp_tableoption 242
- , sp_who 846
- , syspermissions 235
- , xp_sendmail 769
- Systemtabellen
- , Auftragsverlauf 779
- , Standards 581
- , sysalerts 786
- , syscomments 581
- , sysjobs 779
- , sysjobschedules 779
- , sysjobsservers 779
- , sysjobsteps 779
- , sysmembers 225

- , sysobjects 581
- , sysusers 225, 798
- , Wartungsplanverlauf 793
- sysusers 225
- , Daten verschieben 798

T

- Tabellen 368
- , Bezeichner 369
- , CREATE TABLE 388
- , Datentypen 370
- , deleted 645
- , DROP TABLE 390
- , Eigenschaften 462
- , geplante Aktualisierung 680
- , Heap 525
- , inserted 645
- , leeren 473
- , löschen 390
- , NULL-Zulässigkeit 372
- , SELECT INTO/BULK COPY 451
- , Sortierung 372
- , Spalten 368
- , Spaltenlänge 369
- , temporäre 451
- , Verknüpfung 439
- , Zeilenlänge 368
- Tabellenalias 447
- Tabellenscan 524, 538
- Tabellensperren 509
- table 371, 380
- TABLERESULTS 303
- tableresults 553
- TABLOCK 510
- TABLOCKX 510
- TAN 406
- Taskobjekte 816
- Tasks, DTS 814
- Tastenkombinationen, Alt+F1 124
- TCP/IP 190
- TCP/IP-Sockets 73
- tempdb 82, 293, 295
- text 371
- Threads, SQL Server 839
- timestamp 371
- tinyint 371

- TOP 430
- Transact-SQL 31
 - , ANSI-Standard 398
 - , BULK INSERT 811
 - , Skripts 122
 - , SQL-92 398
 - , Steuerungsstrukturen 498
- Transaktionen 283
 - , ACID-Eigenschaften 484
 - , automatische 490
 - , BEGIN DISTRIBUTED TRANSACTION 495
 - , CHECKPOINT 494
 - , Deadlocks 514
 - , explizite 485
 - , implizite 490
 - , Isolationsstufen 510, 517
 - , MSDTC-Dienst 495
 - , Prüfpunkte 494
 - , Regeln 488
 - , remote gespeicherte Prozeduren 489
 - , serialisierbare 508
 - , SET REMOTE_PROC_TRANSACTIONS 496
 - , Sperren 485
 - , verschachteln 487
 - , verteilte 495, 676
- Transaktionsprotokoll 150
 - , Leistungsstatuswarnung 787
 - , sichern 282
 - , Wiederherstellung 346
- Transaktionsreplikation 671
- Trennzeichen 877
- Trigger 644
 - , AFTER 644
 - , ALTER TRIGGER 646
 - , CREATE TRIGGER 645
 - , DELETE 647
 - , deleted 645
 - , erstellen 645
 - , FOR...AFTER 646
 - , Geschäftsregeln kapseln 650
 - , INSERT 646
 - , inserted 645
 - , INSTEAD OF 644, 653

- , kaskadierende 648
- , referenzielle Integrität 648, 652
- , Regeln 646
- , SELECT INTO 653
- , sp_helptext 654
- , TRUNCATE TABLE 647
- , UPDATE 646
- , verschlüsseln 646
- TRUNCATE TABLE 473
- , Trigger 647

U

- UDF (benutzerdefinierte Funktionen)
 - Siehe Funktionen
- Überwachen, Sperren 511
- Überwachung, C2 857
- Unbeschränkte Verknüpfung 442
- UNICODE 409
- Unicode 69, 371
 - , Datentypen 384
 - , Kana-Zeichensatz 384
- Unicode-Gebietsschema-ID 338
- UNION-Operator 452
- UNIQUE 602
 - uniqueidentifizier 378
- , Mergereplikation 728
- , Vergleichsoperatoren 379
- Unteilbarkeit 484
- Unterabfrage 447
 - , Arten 448
 - , DELETE 472
 - , korrelierte 449
- UPDATE 474
 - , Suchtabellen 476
 - , WHERE 476
- UPDATE STATISTICS 549
- UPDLOCK 510
- UPPER 409
- URL
 - , Abfragen 873
 - , Sonderzeichen 876
- USER 598
- USER_ID 413
- USER_NAME 413

V

VALUES 463
 –, DEFAULT 466
 VAR 430
 varbinary(n) 371
 varchar(n) 371
 Variablen
 –, globale 487
 –, lokale 369
 VARP 430
 Verbinden, Dienst-Manager 104
 Verbindungen, vertraute 197
 Verbindungsserver 497
 Vergleiche
 –, Datentypfamilien 379
 –, sql_variant 379
 –, uniqueidentifier 379
 Verknüpfungen 439
 –, äußere 444
 –, Aliasnamen 447
 –, CROSS JOIN 442
 –, INNER JOIN 440
 –, Kartesisches Produkt 442
 –, natürliche 441
 –, OUTER JOIN 444
 –, Selbst 446
 –, Sichten 629
 –, unbeschränkte 442
 Verlaufstabellen 340
 Verleger
 –, Eigenschaften 736
 –, zentraler 689
 –, zentraler mit Remoteverteiler 689
 Verschachteln, Transaktionen 487
 Versionumstellung 86
 Verteiler
 –, Aufbewahrungsdauer 735
 –, Cleanup 735
 –, Eigenschaften 736
 –, verwalten 735
 Verteilte Transaktionen 495, 676
 Verteilungs-Agent 682
 Verteilungsdatenbank 677
 –, Größe 735
 –, Sicherung 735
 Verteilungsmethoden 668

Verteilungsserver
 –, einrichten 705
 –, installieren 704
 –, wählen 705
 –, Wiederherstellung 747
 Vertraute Verbindungen 197
 Verwalten, Verteiler 735
 Verwaltungstools 62
 Verzeichnistrennzeichen 877
 Verzögerte Namensauflösung 634
 VIEW 250
 Visual Studio, Prozeduren
 debuggen 64
 Vollständige Wiederherstellung 340
 Volltextindex 565
 –, Zeilenlokator 567
 Volltextindizierungs-Assistent 566
 Volltextsuche 76
 Vorlagenabfragen 873

W

Währungsdatentypen 382
 –, mathematische Funktionen 406
 WAIT 512
 Warnungen 780
 –, erstellen 782
 –, Replikation 744
 –, testen 785
 Warnungssystem 786
 Warteschlangenlese-Agent 682
 Wartungsplanverlauf 793
 Wartungstasks 142
 Webbrowser 877
 Werte
 –, automatische 679
 –, NULL 421, 465
 –, numerische Datentypen 371
 –, Standard 466
 –, unbekannte 421
 WHILE 506
 Wiederherstellung
 –, Abonnementserver 747
 –, automatische 331
 –, bis zum Zeitpunkt 291
 –, Dateien 351
 –, Dateigruppen 351

- , Dateistandorte bestimmen 339
 - , differenzielle Sicherung 346
 - , einfache 292
 - , Konfiguration 333
 - , manuelle 334
 - , Markierung 347, 349
 - , massenprotokollierte 291
 - , master 359
 - , parallele 284
 - , Publikationsserver 746
 - , Registrierung 142
 - , Registrierungsschlüssel 331
 - , sp_change_users_login 799
 - , Systemdatenbanken 362
 - , Szenarios 357
 - , Transaktionsprotokoll 346
 - , Transaktionsprotokolle und differenzielle Sicherungen 350
 - , Verteilungsserver 747
 - , vollständige 291, 340
 - , zu bestimmtem Zeitpunkt 348
 - Wiederherstellungsintervall 333
 - Wiederherstellungsmodelle 290
 - , ALTER DATABASE 293
 - Wiederherstellungsoptionen
 - , MOVE 345
 - , REPLACE 345
 - Wiederherstellungsprozess 286
 - Windows 2000
 - , Dateisysteme 280
 - , Kennwortrichtlinien 196
 - , Optimierung 834
 - , RAID 280
 - , Ressource Kit 754
 - , Sicherheit 50
 - , Sicherheitsmodus 205
 - , SQL Server-Authentifizierung 192
 - , Systemmonitor 741, 787
 - , Zeitplandienst 753
 - Windows 2000 Server 280
 - Windows 98
 - , RAID 280
 - , Windows 9x 47
 - Windows 9x 47
 - , Dateisysteme 280
 - , Dienste automatisch starten 102
 - , Dienst-Manager 102
 - , Einschränkungen 49
 - , Named Pipes 190
 - , Sicherheitsmodus 205
 - , SQL Server automatisch starten 113
 - , SQL Server-Authentifizierung 192
 - , Standardnetzwerkbibliothek 106
 - Windows ME Siehe Windows 9x
 - Windows NT
 - , Benutzermanager (für Domänen) 199
 - , Dateisysteme 280
 - , Dienste 32
 - , Sicherheit 50
 - , Standardnetzwerkbibliothek 106
 - , Systemmonitor 741
 - Windows-Authentifizierung 50, 67, 197
 - Winsockproxy 112
 - WITH CHECK OPTION 624
 - WITH ENCRYPTION 624, 643
 - WITH RECOMPILE 641
 - WITH SCHEMABINDING 624
 - WITH VIEW_METADATA 624
 - Workflows 816
 - WRITETEXT 378
- X**
- XML 866
 - , DTD (Document Type Definition) 867
 - , Extensible Stylesheet Language (XSL) 867
 - , FOR XML-Klausel 886
 - , IIS 868
 - , Stylesheets 867
 - , virtuelles Verzeichnis 869
 - , Webbrowser 877
 - , Zugriff auf SQL Server 868
 - XMLDATA 890
 - xp_sendmail 769
 - XPath 873
 - XSL-Stylesheets 867

Y

YEAR 411

Z

Zeichenfolgen 372

–, abrufen 420

–, Funktionen 407

–, Platzhalter 420

Zeichenmodus 805

Zeilen

–, aktualisieren 474

–, auswählen 416

–, Blattebene 529

–, Füllfaktor 535

–, gesperrte überspringen 510

–, Index 529

–, löschen 470

–, mehrere einfügen 466

–, Sperren 508

–, Tabellen 368

Zeilenlokator 567

Zeitplandienst 753

Zeitstempel 679

Zugriffstoken 197

