SQL Kompendium

http://kickme.to/tiger/

B Transact-SQL in der Praxis

C Referenzteil

D Die CD zum Buch

Vorwort

Datenbanken begegnet man heute in fast allen - nicht nur den kommerziellen - Bereichen des Lebens. Dabei ist den Benutzern der Datenbanken nicht immer bewußt, welcher Aufwand sich hinter Administration und Entwicklung verbirgt. Der Einsatz von intelligenten Datenbank-Managementsystemen ist hier unerläßlich: Mit SQL Server 7.0 bringt Microsoft ein Produkt auf den Markt, das gegenüber den Vorgängerversionen radikal überarbeitet wurde.

Grundsätzlich kann man alle Aufgaben in bezug auf eine Datenbank in zwei Kategorien einordnen: die eigentliche Arbeit mit den Daten und die administrativen Aufgaben. Währen d bei einem einfachen Datenbank-Managementsystem diese beiden Kategorien eng beieinander liegen und sich praktisch nicht voneinander trennen lassen, hat man es bei Datenbanken, die sich wie im vorliegenden Fall auf SQL Server stützen, normalerweise mit zwei tatsächlich getrennten Personenkreisen zu tun.

Auf der einen Seite stehen die Benutzer, die mühelos mit den Daten arbeiten wollen, und die Anwendungsentwickler. Auf der anderen Seite müssen die Datenbankadministratoren Datenbanken erstellen, verwalten, sichern etc. - kurz: alle administrativen Aufgaben wahrnehmen.

Dieses Buch wendet sich vor allem an Entwickler und Administratoren von Datenbanken, die professionell mit SQL Server 7.0 arbeiten wollen.

SQL Server ist kein Produkt, das man zu Hause im stillen Kämmerlein mal so eben ausprobiert, um ein paar Achtungserfolge in der Hobbyprogrammierung zu erzielen. Mitarbeiter, die in ihren Firmen mit SQL Server in Berührung kommen, sollen befähigt werden, die Abläufe hinter den Kulissen zu verstehen, gegebenenfalls selbst in die Programmierung einzugreifen oder als Administrator einer Datenbank die richtigen Schritte zu unternehmen. Als Stichwort sei hier nur die sorgfältige Planung von Sicherungskonzepten genannt.

Der Aufbau des Inhalts

Gleich dem berühmten roten Faden, den Ariadne ihrem geliebten Theseus gab, damit er den Rückweg aus dem Labyrinth des Minotauros fände, soll Ihnen die folgende Übersicht als Wegweiser durch dieses Buch dienen:

Kapitel 1 bringt eine allgemeine Einführung zu Datenbanken, nennt geschichtliche Hintergründe zu SQL Server und der Sprache SQL, beschäftigt sich mit der Aufgabenverteilung in einem Datenbanksystem und gibt einen Überblick über den architektonischen Aufbau von SQL Server.

Am Anfang von Kapitel 2 erfahren Sie, welche Punkte vor der Installation von SQL Server zu berücksichtigen sind. Daran schließt sich die eigentliche Installation von SQL Server an. Es wird gezeigt, wie man SQL Server startet, anhält und beendet sowie die Funktionsfähigkeit des SQL-Server-Systems testet.

Kapitel 3 erläutert die Komponenten der SQL-Server-Installation und führt in die grafische

Benutzeroberfläche sowie ausgewählte Dienstprogramme der Befehlszeile ein.

Bevor Sie beginnen, Datenbanken zu erstellen und mit Daten zu füllen, sollten Sie sich in Kapitel 4 einen Überblick über die Objekte einer Datenbank und die Charakteristika eines relationalen Datenbanksystems verschaffen.

Der Sprache Transact-SQL kommt sowohl bei der Entwicklung von Anwendungen als auch bei der Administration eine zentrale Bedeutung zu. Kapitel 5 beschreibt die Grundlagen dieser Sprache, zeigt, wie man Befehle mit dem Befehlszeilenprogramm osql und dem grafischen Pendant SQL Server Query Analyzer ausführt und Skripts erstellt, um Anweisungen bearbeiten und erneut ausführen zu können.

In Kapitel 6 erstellen und verwalten Sie Datenbanken. Anhand der zu SQL Server gelieferten Beispiele wird gezeigt, wie man Datenbankdiagramme erstellt, um sich einen Überblick über die Struktur einer Datenbank zu verschaffen.

Zu den wichtigsten Aufgaben des Datenbankadministrators gehört das Sichern und Wiederherstellen von Datenbanken. Wenn Sie die Beispiele dieses Buches ausprobieren oder eigene Experimente anstellen, um Erfahrungen mit SQL Server zu sammeln, sollten Sie von vornherein damit vertraut sein, wie sich der Zustand von SQL Server und der Datenbanken gegebenenfalls wiederherstellen läßt. Aus diesem Grund beschäftigt sich Kapitel 7 bereits an dieser Stelle mit diesem Thema.

Grundlage für die Speicherung von Daten in einem relationalen Datenbank-Managementsystem sind Tabellen. Kapitel 8 zeigt, wie man Tabellen erstellt, die Datentypen der Spalten festlegt und Tabellen mit Daten füllt.

Wenn die Entscheidung für den Einsatz von SQL Server fällt, liegen oftmals bereits größere Datenbestände vor, die in Datenbanken von SQL Server übernommen werden sollen. Kapitel 9 beschäftigt sich mit dem Importieren und Exportieren von Daten. Es wird gezeigt, wie man die Datentransformationsdienste einsetzt und mit dem Befehlszeilenprogramm bcp arbeitet.

Nach diesen Vorbereitungen können Sie nun mit den eigentlichen Daten arbeiten. Kapitel 10 zeigt, wie man Daten abruft, einfügt, ändert und löscht. Diese Aufgaben lassen sich mit den wichtigsten Befehlen der Sprache Transact-SQL erledigen: SELECT, INSERT, UPDATE und DELETE.

Kapitel 11 ist den Funktionen der Sprache Transact-SQL gewidmet. Anhand zahlreicher Beispiele werden die Funktionen und ihr Einsatz erläutert.

Einen weiteren Schwerpunkt bei der Arbeit mit Datenbanken bildet sowohl für den Administrator als auch den Anwendungsentwickler das Thema Gespeicherte Prozeduren. In Kapitel 12 erfahren Sie, was gespeicherte Prozeduren sind, wie man die zum Lieferumfang gehörenden Prozeduren einsetzt und eigene Prozeduren schreibt. Zu diesem Thema gehören auch die Anweisungen zur Flußsteuerung.

Sichten sind eine andere Darstellungsform von Abfrageergebnissen. Kapitel 13 erläutert, wie man Sichten sinnvoll einsetzt.

Die Benutzer von Datenbanken erwarten, daß die Ergebnisse auf ihre Abfragen in kürzester Zeit eintreffen. Indizes und Schlüssel können dazu beitragen. Kapitel 14 zeigt aber auch, daß Indizes nicht in jedem Fall einen Geschwindigkeitsvorteil bringen.

Die in Kapitel 15 behandelten Cursor spielen eine Außenseiterrolle. Strenggenommen passen sie nicht in

das relationale Datenbankmodell, wurden aber eingeführt, damit auch Anwendungen nach herkömmlichen Entwurfsmustern mit relationalen Datenbanken effektiv arbeiten können.

Kapitel 16 beschäftigt sich mit Aspekten der Datenintegrität, mit Regeln, Bedingungen und Standardwerten.

Wenn Aktionen auszuführen sind, die vom Einfügen, Aktualisieren oder Löschen von Daten abhängen, kann man dies mit Triggern realisieren. Kapitel 17 erläutert auch, wie man mit Triggern Geschäftsregeln durchsetzt, die referentielle Integrität erzwingt und die Datenintegrität gewährleistet.

Transact-SQL-Anweisungen werden nur selten einzeln ausgeführt. Meist faßt man mehrere Anweisungen in sogenannten Stapeln zusammen oder führt Operationen aus, die entweder geschlossen (im Erfolgsfall) oder insgesamt überhaupt nicht (beim Scheitern mindestens einer Anweisung) ausgeführt werden. Transaktionen sind Schwerpunkt von Kapitel 18. In diesem Zusammenhang erfahren Sie auch, was Prüfpunkte sind, wie man Sperren realisiert und welche Rolle der Distributed Transaction Coordinator bei verteilten Transaktionen spielt.

Gemäß den Prinzipien des relationalen Datenbankmodells erfolgt die Verwaltung eines relationalen Datenbank-Managementsystems selbst über Tabellen - den sogenannten Systemkatalog. Kapitel 19 bringt eine Übersicht über wichtige Details des Systemkatalogs und zeigt vor allem, wie man ihn nutzbringend einsetzt.

Die ständige Verfügbarkeit von Datenbanken setzt voraus, daß der Datenbankadministrator regelmäßige Wartungen durchführt. Kapitel 20 zeigt, wie man Wartungspläne erstellt, um diese Routineaufgaben automatisch ausführen zu lassen.

Das Thema Sicherheit würde man intuitiv der Administration zuordnen. Doch auch auf der Anwendungsseite sind sicherheitstechnische Aspekte zu berücksichtigen, und Anwendungen können bis zu einem gewissen Umfang auch auf den Sicherheitsmechanismus von SQL Server Einfluß nehmen. Kapitel 21 geht umfassend auf diese Bereiche ein.

Zentral verwaltete Daten überall griffbereit zu haben, auf dem neuesten Stand vorzufinden und synchronisieren zu können - die in Kapitel 22 behandelte Replikation macht es möglich.

Mit dem SQL Server-Agenten lassen sich Aufträge und Warnungen erstellen sowie Operatoren benachrichtigen. Kapitel 23 beschreibt den Einsatz dieses Werkzeugs.

Kapitel 24 gibt Hinweise, wie sich die Leistung eines SQL-Server-Systems optimieren läßt, wie man mit dem Query Analyzer Abfragen analysiert, mit dem SQL Server Profiler Ablaufverfolgungen erstellt und den Indexoptimierungs-Assistenten einsetzt.

Kapitel 25 konzentriert sich auf die Fehlersuche in SQL-Server-Systemen.

Data Warehouses und Entscheidungssysteme sind für die Wettbewerbsfähigkeit eines Unternehmens von zentraler Bedeutung. Kapitel 26 geht auf die Unterschiede zwischen Online-Transaktionsverarbeitung (OLTP) und analytischer Online-Verarbeitung (OLAP) ein, erläutert die Komponenten eines Data Warehouse und stellt die zum Lieferumfang von SQL Server gehörenden OLAP Services vor.

Kapitel 27 zeigt am Beispiel von Visual C++, wie man eine Anwendung erstellt, die auf eine SQL-Server-Datenbank zugreift.

Viele Unternehmen gehen dazu über, Produktinformationen aus firmeneigenen Datenbanken im Internet zu veröffentlichen. Kapitel 28 zeigt, wie man diese Aufgaben mit dem Web-Assistenten realisiert.

Kapitel 29 beschäftigt sich mit den neuen Merkmalen von SQL Server 7.0 und geht auf die Versionsumstellung ein.

Drei Anhänge runden die Themen zu SQL Server ab: Anhang A bringt die wichtigsten Abkürzungen, denen Sie auf Schritt und Tritt bei Ihrer Arbeit mit SQL Server begegnen. Anhang B geht auf das Skript InstPubs.sql ein. Dieses Skript gehört zum Lieferumfang von SQL Server. Durch die eingefügten Kommentare gewinnen Sie einen Überblick über die Aufgaben des Skripts und können sich vor allem ansehen, wie Transact-SQL-Anweisungen in der Praxis eingesetzt werden. Eine Referenz zur Sprache Transact-SQL, zu den wichtigsten Systemprozeduren von SQL Server 7.0 und ein Überblick über DBCC-Anweisungen sind im Anhang C zu finden.

Das Paket SQL Server ist so vielschichtig und umfangreich, daß es sich in einem Buch nicht komplett abhandeln läßt. Deshalb handelt es sich hier um eine Gratwanderung zwischen einem allgemeinen Überblick über alle Elemente und einem tieferen Einblick in konkrete Realisierungen.

Die meisten Beispiele in diesem Buch wurden mit der folgenden Konfiguration erstellt:

- Server-Computer \\EINS mit Windows NT Server 4.0
- Client-Computer \\ZWEI mit Windows 98
- Verbindung über lokales Netz mit NE2000-kompatiblen Schnittstellenkarten

Abschließend ein Wort in eigener Sache: Von der Idee zu diesem Buch bis zur Fertigstellung ist fast ein Jahr ins Land gegangen. In dieser Zeit ist das Familienleben wieder einmal zu kurz gekommen - man möge es mir nachsehen. Meine Frau Regina hat nichtsdestotrotz Nervenstärke bewiesen und die Korrekturarbeiten übernommen, wofür ich ihr sehr dankbar bin.

Ebenfalls bedanken möchte ich mich bei Angelika Ritthaler und Erik Franz vom Verlag Markt&Technik, die mir bei Bedarf kurzfristig mit der Beschaffung von Recherchematerial geholfen haben.

Besonderer Dank gilt meinem Lektor Dr. Rainer Fuchs für die großartige Unterstützung - nicht zuletzt auch für seine Geduld, die er gerade in der Endphase dieses Projekts aufbringen mußte.

Frank Langenau

Kapitel 1 Einführung

1.1 Wat is en Dampfmaschin?

Diese Frage und die sich anschließende Erläuterung des Lehrers Bömmel im Film »Feuerzangenbowle« sind ebenso bekannt wie einfach. Fast genauso einfach läßt sich die Frage »Was ist eine Datenbank?« beantworten:

Im weitesten Sinne kann man jede Sammlung von Daten, die miteinander in Beziehung stehen und in einer bestimmten Struktur vorliegen, als *Datenbank* bezeichnen. Die Hauptaufgabe einer Datenbank besteht darin, Daten zu speichern und dann diese Daten für autorisierte Anwendungen und Benutzer verfügbar zu machen.

Damit sich die gesammelten Informationen nutzbringend und schnell auswerten und wiederfinden lassen, kommt der Organisation der Datenbank eine entscheidende Rolle zu. Mit *Datenbank-Managementsystemen* (DBMS) lassen sich die erforderlichen Datenstrukturen anlegen und verwalten. Mit speziellen *Abfragesprachen* ruft man die Daten der Datenbank ab.

1.2 Datenbanksysteme

Für das Strukturieren und die physikalische Speicherung der Daten in Datenbanken sind verschiedene Modelle - oder Architekturen - entwickelt worden. Datenbankarchitekturen verfolgen das Ziel, einer großen Zahl von Benutzern die zentral verwalteten Datenbanken möglichst effizient zugänglich zu machen. Wesentliche Bedeutung kommt dabei der Integrität und Sicherheit der Daten zu, wobei gleichzeitig die Systemkosten niedrig zu halten sind.

Ein Datenbanksystem läßt sich grob in drei Komponenten gliedern:

- Benutzeroberfläche
- Geschäftsregeln
- Datenbank-Managementsystem

Die einzelnen Datenbankarchitekturen unterscheiden sich vornehmlich darin, wie und wo diese Komponenten untergebracht sind.

1.2.1 Dateiorientiertes Modell

In älteren Datenbankprodukten wie dBase, Access und vielen anderen ist eine Datenbank normalerweise in einer einzigen Datei untergebracht. Der Zugriff auf die Daten erfolgt sequentiell. Man spricht bei diesem Typ von linearen (oder flachen), dateibasierten Datenbanken. Dieses Modell ist für einfache Datenstrukturen durchaus geeignet, führt aber zu Problemen, sobald die Datenstrukturen komplizierter werden. Außerdem ist die Wartung in dateibasierten Datenbanken schwieriger zu handhaben.

1.2.2 Relationales Modell

Im relationalen Modell baut sich eine Datenbank aus mehreren Tabellen - sogenannten Relationen - auf, die miteinander in Beziehung stehen. Die Probleme der effizienten Strukturierung von Daten werden mit Hilfe der Mengenlehre beschrieben. Daher bezeichnet man auch die von einer Abfrage zurückgelieferten Ergebnisse als *Ergebnismengen*. Ein wesentliches Merkmal des relationalen Modells ist die Normalisierung der Tabellen, so daß keine doppelten Daten zu speichern sind. Im Zuge der Normalisierung mit den verschiedenen Methoden der Normalformen wird eine Tabelle (Relation) durch Projektion in zwei oder mehr Relationen aufgeteilt. Diese Vereinfachung erkauft man sich mit einer größeren Anzahl von Tabellen.

1.2.3 Mainframe

Größere Datenbanksysteme der 80er und frühen 90er Jahre wurden ausschließlich auf Mainframes realisiert. Die Benutzer kommunizierten mit der Datenbank auf dem Mainframe-Computer über primitive Terminals, die lediglich Text anzeigen und Daten vom Benutzer entgegennehmen konnten. Die eigentliche Arbeit konzentrierte sich auf den Mainframe-Computer, der sowohl das relationale Datenbank-Managementsystem als auch die für den Zugriff auf das RDBMS erforderlichen Anwendungen und die Kommunikation zwischen Mainframe und den Terminals implementieren mußte.

Auch wenn sich die Mainframe-Architektur schon allein aufgrund ihrer Zuverlässigkeit über Jahrzehnte hinweg behaupten konnte, weist sie doch einige Nachteile auf. Vor allem sind dabei die in die Millionen gehenden Kosten zu nennen, die mit der Anschaffung und dem laufenden Betrieb des Mainframe-Computers verbunden sind. Außerdem handelt es sich sowohl bei der Hardware als auch bei der Software um proprietäre Systeme, die keine allgemein üblichen Komponenten verwenden, so daß das Unternehmen in der Regel an einen Hersteller gebunden ist.

1.2.4 PC-Dateiserver

Mit der stürmischen Verbreitung der PCs Ende der 80er Jahre zeichneten sich Alternativen zur Datenbankverwaltung auf Mainframes ab. Die Benutzer begrüßten Datenbankprogramme wie dBASE und auch die Möglichkeit, eigene Programme in höheren Programmiersprachen entwickeln zu können.

Beim PC-Dateiserver-Modell sind die Datenbankdateien auf einem zentralen Computer, dem Dateiserver, untergebracht, während die Anwendungen und das RDBMS auf den Client-Computern - PCs - laufen.

Wenn der Benutzer eine Abfrage ausführt, erhält er nicht einfach die Ergebnisse vom Dateiserver zurück, sondern den gesamten erforderlichen Datenbestand. Die Abfrage muß die Anwendung auf dem Client-Computer dann selbst realisieren. Die Netzwerkbelastung ist damit extrem hoch, bei vielen gleichzeitigen Anfragen an den Dateiserver muß man mit entsprechend langen Antwortzeiten rechnen.

1.2.5 Client-Server-Architektur

Die Client-Server-Architektur übernimmt das beste aus beiden Welten, d.h. die leistungsfähige Verarbeitung der Mainframe-Architektur und die mit niedrigen Kosten gepaarte Flexibilität von PCs. Die oben genannten drei Bestandteile eines Datenbanksystems sind damit logisch zwischen Client- und Server-Computer aufgeteilt.

Die Benutzeroberfläche ist auf dem Client-Computer realisiert. Hier gibt der Benutzer seine Abfragen ein und nimmt die Ergebnisse entgegen.

Auf dem Server-Computer befindet sich das Datenbank-Managementsystem. Hier werden die Daten der Datenbank gespeichert, manipuliert und abgerufen. Die gesamte Verarbeitung findet auf dem Server statt.

Je nachdem, wie und wo die Geschäftsregeln untergebracht und realisiert werden, unterscheidet man bei der Client-Server-Architektur das Zwei-Schichten- und das Mehr-Schichten-Modell.

Zwei-Schichten-Modell

In einem Client-Server-System nach dem Zwei-Schichten-Modell läuft auf dem lokalen Computer des Benutzers - dem Client - eine Anwendung, die eine Verbindung zur Datenbank auf dem Server herstellt. Die Anwendung realisiert sowohl die Geschäftsregeln als auch die Anzeige der Ergebnisse. Die Geschäftsregeln können allerdings auch auf dem Server oder sogar auf beiden Computern implementiert sein.

Mehr-Schichten-Modell

Bei einem mehrschichtigen Client-Server-System ist die Anwendungslogik des Clients auf mehrere Computer aufgeteilt. Auf dem Client läuft nur noch die Anwendung, die für die Anzeige der Ergebnisse verantwortlich ist. Die *Geschäftsregeln* - d.h., die in einer Organisation gültigen Vorschriften für die Arbeit mit der Datenbank - sind in Anwendungen realisiert, die auf Servern zwischen dem Client-Computer (der die Benutzeroberfläche realisiert) und dem RDBMS-Server (der die Datenbankverarbeitung realisiert) laufen.

Vorzüge des Client-Server-Modells im Datenbankbetrieb

Der Trend zum Client-Server-Modell ergibt sich aus den folgenden Vorzügen:

- Kostengünstige Lösung: Datenbanken nach dem Client-Server-Modell basieren auf einer offenen Architektur. Der dadurch mögliche Wettbewerb von Datenbankanbietern führt fast automatisch zu niedrigeren Anschaffungskosten. Bei Mainframe-Datenbanken handelt es sich meist um proprietäre Systeme, die nur von einem Anbieter bezogen werden können.
- Geschwindigkeit: Die Aufteilung der Arbeitslast zwischen Server und Client verringert Engpässe der Netzwerkübertragung. Damit erreicht man das Leistungsvolumen von Mainframe-Computern auf PC-basierten Client-Server-Systemen.
- Anpassungsfähigkeit: Durch die offenere Architektur des Client-Server-Modells gegenüber der proprietären Mainframe-Architektur ist es möglich, das RDBMS von einem Hersteller, die Hardware von einem anderen und die Entwicklungssoftware von einem dritten Anbieter zu

beziehen.

- Einfacher Datenzugriff: Während Datenbanken auf Mainframes eher etwas für Spezialisten sind, die sich mit der Datenbanksprache und der Programmierung auskennen, macht die Client-Server-Architektur die Daten einem breiten Publikum verfügbar. Der Benutzer - sprich der Konsument der Daten - greift über zugeschnittene Anwendungen auf die Datenbank zu und merkt praktisch nichts von der Komplexität des Datenzugriffs.
- Betriebssysteme: Der Datenbankzugriff kann von Client-Computern aus erfolgen, die mit verschiedenen Betriebssystemen arbeiten, solange diese ein gemeinsames Netzwerkprotokoll verwenden.

1.2.6 Architektur von SQL Server

Microsoft SQL Server kann als Client-Server-Datenbanksystem nach dem Zwei- oder Mehr-Schichtenmodell oder als Desktop-Datenbanksystem arbeiten.

Bei Desktop-Datenbanksystemen liegt die gesamte Verarbeitungslast beim Client-Rechner. In derartigen Systemen sind die Client-Anwendungen, SQL Server und die Datenbanken auf ein und demselben Computer vereinigt. SQL Server kann sich selbst dynamisch konfigurieren, um effektiv mit den zur Verfügung stehenden Ressourcen des Clients zu laufen. Für diese eigenständigen Einsatzgebiete ist es nicht erforderlich, jedem Client einen Datenbankadministrator zuzuordnen.

Wie der Name bereits verrät, ist SQL Server vorrangig für die Arbeit auf einem Server konzipiert. In derartigen Systemen befinden sich die Datenbankdateien und das Datenbank-Managementsystem auf einem zentralen Computer - dem Server. Dieser stellt Schnittstellen bereit, die den Anwendungen auf separaten Client-Computern eine Verbindung zu den Datenbanken ermöglichen. Die Programme der Client-Computer liefern vor allem die Benutzeroberfläche und können auch Programmkomponenten wie Berichte, Abfragen oder Formulare realisieren.

1.3 Relationale Datenbank-Managementsysteme

Bei der Verwaltung von Datenbanken in der Client-Server-Architektur haben sich Datenbank-Managementsysteme, die auf dem relationalen Modell basieren, durchgesetzt. Dieses Modell stützt sich auf die mathematischen Methoden der Mengenlehre, um die Daten möglichst effizient zu organisieren. Sämtliche Informationen werden in Tabellen (den sogenannten Relationen) zusammengestellt. Die Tabellen bestehen aus Spalten (den Attributen) und Zeilen (den Tupeln).

1.3.1 Kriterien für relationale Datenbanken

Das relationale Modell definiert ein einfaches, aber streng formuliertes Konzept, wie der Benutzer die Daten empfängt. Die Daten werden in Form von zweidimensionalen Tabellen dargestellt. Jede Tabelle repräsentiert ein reales Element, über das Informationen gesammelt werden - Kunden, Artikel, Bestellungen oder auch Ereignisse. Eine relationale Datenbank ist eine Sammlung von zweidimensionalen Tabellen.

Die zwölf goldenen Regeln des Dr. Codd

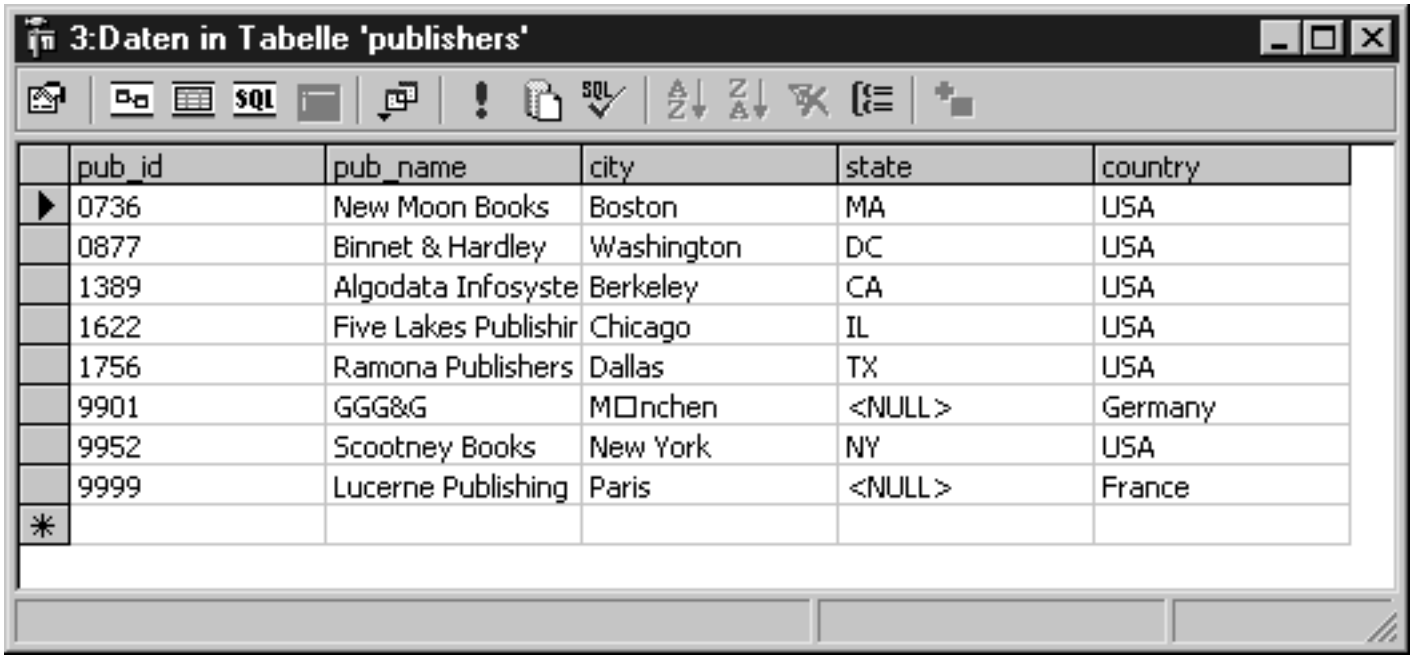
Als Vater des relationalen Datenbankmodells gilt Dr. E. F. Codd, der 1970 im Association of Computer Machinery Journal den Artikel »A Relational Model of Data for Large Shared Data Banks« veröffentlichte. Seine zwölf Regeln sind in die Datenbankgeschichte eingegangen - es handelt sich eigentlich inklusive der Regel 0 um 13 Regeln.

0. Regel - Grundregel

Ein relationales Datenbank-Managementsystem (RDBMS) muß in der Lage sein, Datenbanken vollständig über seine relationalen Fähigkeiten zu verwalten. Wenn ein DBMS von einem Werkzeug zur Datenmanipulation, das datensatzweise arbeitet, abhängt, ist es kein echtes relationales Datenbank-Managementsystem.

1. Regel - Darstellung von Informationen

Alle Daten in einer relationalen Datenbank werden (auf der logischen Ebene) ausschließlich als Werte in Tabellen dargestellt (siehe Abbildung 1.1). Daten dürfen nicht in irgendeiner anderen Weise gespeichert werden.



pub_id	pub_name	city	state	country
0736	New Moon Books	Boston	MA	USA
0877	Binnet & Hardley	Washington	DC	USA
1389	Algodata Infosyste	Berkeley	CA	USA
1622	Five Lakes Publishir	Chicago	IL	USA
1756	Ramona Publishers	Dallas	TX	USA
9901	GGG&G	M□nchen	<NULL>	Germany
9952	Scootney Books	New York	NY	USA
9999	Lucerne Publishing	Paris	<NULL>	France

Abbildung 1.1: Die Tabellenstruktur in einem relationalen Datenbankmodell

2. Regel - Zugriff auf Daten

Jedes Datenelement muß logisch über eine Kombination aus seinem Primärschlüsselwert, Tabellennamen und Spaltennamen zugänglich sein.

3. Regel - Behandlung von Nullwerten

Nullwerte werden ausdrücklich unterstützt. Nullwerte repräsentieren fehlende oder ungeeignete Informationen.

4. Regel - Datenbankstruktur

Die Datenbankbeschreibung (der Systemkatalog) wird ebenfalls auf der logischen Ebene in Tabellenform gespeichert. Die relationale Sprache - wie zum Beispiel SQL - muß auf den Entwurf der Datenbank genauso einwirken können wie auf die Daten, die in der Struktur gespeichert sind.

5. Regel - Abfragesprache

Ein RDBMS muß eine eindeutig definierte Datenmanipulationssprache unterstützen, mit der sich Manipulation und Definition von Daten, Definition von Ansichten (Views), Integritätsbedingungen und Definition von Transaktionen umfassend realisieren lassen. SQL ist die bekannteste dieser Sprachen. SQL Server stellt mit Transact-SQL eine erweiterte Form von ANSI SQL bereit.

6. Regel - Ansichten

Alle aktualisierbaren Ansichten müssen durch das System aktualisierbar sein. In einem echten RDBMS sind die meisten, wenn auch nicht alle, Ansichten aktualisierbar.

7. Regel - Abfragen und Bearbeiten von Tabellen

Zu einem RDBMS gehört mehr als nur die Fähigkeit, relationale Datensätze abrufen zu können. Es muß auch Daten als relationale Menge einfügen und aktualisieren können.

Das heißt, daß als Operanden in Abfrage- und Bearbeitungsoperationen auch ganze Tabellen und nicht nur einzelne Datensätze erlaubt sind.

8. Regel - Physikalische Unabhängigkeit der Daten

Daten müssen physikalisch unabhängig von Anwendungsprogrammen sein. Das zugrundeliegende RDBMS-Programm oder der »Optimierer« sollte physikalische Änderungen an den Daten verfolgen können. Zum Beispiel sollten keine Änderungen an den Anwendungsprogrammen eines RDBMS erforderlich sein, wenn eine Tabelle indiziert wird.

9. Regel - Logische Unabhängigkeit der Daten

Soweit möglich müssen Anwendungsprogramme unabhängig von Änderungen an den zugrundeliegenden Tabellen sein. Beispielsweise sollte es nicht notwendig sein, Code neu zu schreiben, wenn Tabellen zu einer Ansicht kombiniert werden.

10. Regel - Unabhängige Integrität

Die Integritätsbedingungen für die Daten müssen sich in einer relationalen Sprache definieren und im Systemkatalog speichern lassen. Diese Regel läßt sich nur schwer in die Praxis umsetzen. Die Integritätsbedingungen können in einer Anwendung integriert sein. Diese Lösung widerspricht dem relationalen Modell, bei dem die Integrität zum Datenbankentwurf gehören sollte.

11. Regel - Unabhängigkeit der Verteilung von Daten

Ein RDBMS ist unabhängig von einer Verteilung der Daten.

12. Regel - Unterlaufen der Abfragesprache

Wenn ein RDBMS eine Low-Level-Abfragesprache unterstützt (mit der sich einzelne Datensätze bearbeiten lassen), dann darf diese Sprache nicht die Integritätsbedingungen der relationalen (High-Level-)Sprache unterlaufen. Folglich genügt es nicht, wenn sich ein RDBMS durch relationale Regeln steuern läßt, sondern diese Regeln müssen auch die primären Gesetze sein.

1.3.2 Vorzüge des relationalen Modells für Client-Server-Datenbanken

In Client-Server-Datenbanken hat sich das relationale Modell aus folgenden Gründen durchgesetzt:

- Datenintegrität: Die Fehlerfreiheit, Genauigkeit und Zuverlässigkeit - d.h. die Qualität - von Daten ist eines der Hauptziele von relationalen Datenbanken. In einem RDBMS läßt sich die Datenintegrität zentral auf der Server-Seite durchsetzen und greift damit automatisch auf die Anwendungsseite durch.
- Strukturierte Abfragesprache: Die Sprache SQL (bzw. der erweiterte Dialekt Transact-SQL von SQL Server) bietet eine einheitliche Methode für den Zugriff und die Bearbeitung von Daten in einer relationalen Datenbank, so daß sich diese Sprache quasi als Industriestandard etabliert hat.
- Flexibilität: Die Struktur einer Datenbank läßt sich praktisch im Vorübergehen ändern, ohne daß man das Datenbanksystem herunterfahren und neu starten muß.
- Effiziente Datenspeicherung: Mit den Konzepten der Normalisierung (siehe Kapitel 4) vermeidet man redundante Daten.
- Sicherheit: In einem relationalen Datenbank-Managementsystem lassen sich auch die Sicherheitsaspekte einer Datenbank realisieren. Das bedeutet, daß man die Sicherheit zentral auf der Server-Seite verwalten kann und nicht dem Anwendungsentwickler überlassen muß.

1.4 Aufgabengebiete

Um die Funktionsfähigkeit des gesamten Komplexes Datenbanksystem sicherzustellen, müssen das Wartungspersonal für die Computertechnik, der Netzwerkadministrator, der Systemadministrator, der Web-Master und der Datenbankadministrator Hand in Hand arbeiten. In kleineren Unternehmen verschmelzen diese Verantwortungsbereiche und werden gegebenenfalls von nur einer Person wahrgenommen.

Läßt man die hardwarespezifischen Aufgaben einmal außer acht, haben mit einer Datenbank im wesentlichen drei Personengruppen zu tun:

- Benutzer
- Administratoren
- Entwickler

1.4.1 Benutzer

Nicht von ungefähr steht der Benutzer an erster Stelle. Ihn zufriedenzustellen, sollte zum Ehrenkodex der Administratoren und Entwickler gehören.

Allerdings geht es in diesem Buch weniger um den Benutzer an sich, sondern darum, wie er auf die Daten einer Datenbank zugreifen und sie abrufen kann. Dazu muß der Anwendungsentwickler eine Anwendung mit geeigneter Benutzeroberfläche bereitstellen, während der Datenbankadministrator die Verfügbarkeit und Sicherheit des Datenbanksystems gewährleistet. Außerdem sollen viele Benutzer gleichzeitig auf ein und dieselbe Datenbank zugreifen können.

Auf der Seite der Datenbank (d.h. der Server-Seite) existieren verschiedene Dienstprogramme, mit denen sich Datenstrukturen erstellen und bearbeiten lassen. Neben diesen allgemeinen Aufgaben sind in Datenbanksystemen noch viele weitere Dinge zu realisieren, beispielsweise Geschäftsregeln durchzusetzen, Daten zu sichern und wiederherzustellen oder Sicherheitsaspekte wahrzunehmen.

1.4.2 Datenbankadministrator

Der Datenbankadministrator ist unter anderem dafür verantwortlich, die Datenbanken zu erstellen und zu verwalten. Die von SQL Server für die Administration bereitgestellten Anwendungen unterstützen den Systemadministrator bei allen Aufgaben, die sich auf die Verwaltung und die Überwachung der Systemleistung und -aktivitäten beziehen. Im einzelnen gehören dazu folgende Punkte:

- SQL Server installieren
- Versionsumstellung durchführen
- Starten, Anhalten und Stoppen von SQL Server
- Datenbanken verwalten und Funktionsfähigkeit sicherstellen, Service Packs installieren
- Feinabstimmung durchführen
- Daten importieren und exportieren
- Speicherbedarf anpassen
- Datenbanken sichern und wiederherstellen
- Server und Clients verwalten
- Benutzer von SQL Server einrichten und verwalten
- Sicherheitsaufgaben wahrnehmen
- Mit Entwicklern zusammenarbeiten
- Standards aufstellen und durchsetzen
- Daten übertragen
- Replikationsumgebung verwalten
- Data Warehousing
- Ereignisse planen
- Serverleistung und -aktivitäten überwachen
- Fehlersuche und Fehlerbeseitigung

- Durchgehende Betriebsfähigkeit gewährleisten

1.4.3 Entwickler

Die Aufgaben des Entwicklers lassen sich in etwa folgendermaßen umreißen:

- Systemanalyse
- Datenbanken entwerfen und erstellen
- Replikationsszenarien entwerfen
- Datenbanken programmieren
- Grundlegende Sicherheit der Datenbank programmieren
- Benutzeroberfläche entwerfen und programmieren

1.5 Das Datenbank-Managementsystem: SQL Server

Microsoft SQL Server 7.0 ist ein relationales Datenbank-Managementsystem, das mit SQL als Abfragesprache arbeitet. Es gehört zur BackOffice-Produktfamilie und ist in Windows NT Server integriert. Zu BackOffice gehören folgende Komponenten:

- Windows NT Server
- Exchange Server
- SQL Server
- Proxy Server
- SNA Server
- Site Server
- Systems Management Server

In diesem Buch geht es vor allem um Windows NT Server als zugrundeliegendes Betriebssystem und natürlich SQL Server selbst.

1.5.1 Geschichtliche Einordnung

Die Entwicklung von SQL Server ist durch drei Hauptversionen gekennzeichnet. Die Anfänge von SQL Server gehen auf eine Zusammenarbeit von Microsoft und Sybase für das Betriebssystem OS/2 zurück. Bis zur Version 4.21 wurde SQL Server zusammen mit Sybase entwickelt. In dieser Version fehlt die deklarative referentielle Integrität (DRI), mit der Benutzer von Microsoft Access vertraut sind. Die vollständig neu gefaßte Version 6.0 enthält viele neue Merkmale einschließlich DRI und führte den praktischen SQL Enterprise Manager für die Verwaltung des Servers ein. Der Übergang von der Version 6.0 zur Version 6.5 ist weniger spektakulär und wird als Wartungsversion angesehen. An den grundlegenden Merkmalen haben sich nur wenige Änderungen ergeben, obwohl auch in der Version 6.5 zahlreiche neue Merkmale hinzugekommen sind. Dazu gehört die verbesserte Leistung der Datenbank-Engine, das Sperren von Zeilen, die Replikation in Microsoft-Access-Datenbanken,

Internet/Intranet-Features und der Distributed Transaction Coordinator (DTC).

Mit dem Produkt SQL Server 7.0 liegt nun ein völlig neu konzipiertes Datenbank-Managementsystem vor. Auf die verbesserten Eigenschaften geht Kapitel 29 näher ein.

1.5.2 Entwicklungsziele

Mit SQL Server 7.0 verfolgt Microsoft das Ziel, zu den führenden Anbietern auf dem Sektor der Datenbanktechnologie zu avancieren. In diesen Bereich fallen E-Commerce-Anwendungen, Unterstützung mobiler PCs, Branchenautomatisierung, geschäftliche Anwendungen und Data Marts.

Die folgende Übersicht beinhaltet die von Microsoft angeführten Entwicklungsziele von SQL Server und die jeweiligen Kapitel, die sich im Buch mit diesen Themenschwerpunkten befassen.

- Durchgängige Technologie vom Laptop bis zum Cluster. SQL Server 7.0 ist vom Laptop unter Windows 95/98 bis hin zu verteilten Multiprozessor-Cluster-Systemen unter Windows NT Server Enterprise Edition skalierbar. Die über den gesamten Bereich gleiche Codebasis gewährleistet eine nahezu hundertprozentige Kompatibilität zwischen den verschiedenartigsten Anwendungen.
- Automatische Konfiguration und selbständiges Tuning und damit Minimierung der Wartungsarbeiten (Kapitel 20).
- Integrierte analytische Onlineverarbeitung (OLAP) (Kapitel 26).
- Integrierte Werkzeuge zur Extraktion und Transformation von Daten zwischen heterogenen Datenquellen (Kapitel 9).
- Unterstützung des kompletten Data-Warehouse-Lebenszyklus mit speziellen Tools (Kapitel 26).
- Einzigartige Auswahl von Replikations-Szenarien (Kapitel 22).
- Integration mit Windows NT Server und den Microsoft-BackOffice-Komponenten.
- Neue OLE-DB-Schnittstelle für universellen und performanten Datenzugriff, auch auf nicht-relationale Dateien.
- Zentraler Metadaten-Speicher im Microsoft Repository.

Weitere Schlagworte der von Microsoft angeführten Entwicklungsziele sind

- vereinfachter Umgang
- skalierbar und zuverlässig
- Data Warehousing
- Verfügbarkeit

1.6 Die Abfragesprache: SQL

1.6.1 Geschichtliche Entwicklung

In den Jahren 1973 bis 1976 arbeiteten die Entwickler von IBM mit dem System R an einem Prototyp RDBMS, um Adhoc-Abfragen und die Transaktionsverarbeitung zu realisieren. Im Zuge dieser Entwicklung entstand auf der Basis des Modells von Codd die Sprache SEQUEL, was für »Structured English Query Language« steht. Der Name wurde später aufgrund von Warenzeichengesetzen in SQL für

»Structured Query Language« geändert.

Eng verbunden mit der Sprache SQL ist das American National Standards Institute (ANSI), das mehrere international anerkannte Standards zu SQL erarbeitete und auch heute noch erarbeitet (ANSI 1986, 1989 und 1992). Dieser internationale Standard definiert die Datenstrukturen und grundlegenden Operationen auf SQL-Daten. Er beschreibt die funktionellen Fähigkeiten für das Erzeugen, den Zugriff, die Verwaltung, die Steuerung und den Schutz von SQL-Daten.

Im Laufe der Zeit hat sich die Sprache SQL von einer reinen Abfragesprache zu einer Sprache entwickelt, mit der sich auch Datenbanken erstellen und die Sicherheit des Datenbanksystems verwalten lassen.

SQL2 oder SQL-92

Der Standard SQL2 oder SQL-92 wurde im Jahre 1992 verabschiedet. SQL-92 definiert drei mögliche Ebenen des Sprachumfangs:

- Eingangsstufe (Entry Level)
- Zwischenstufe (Intermediate Level)
- Stufe des vollständigen Standards (Full Level)

Eingangsstufe

Enthält die gesamte Funktionalität von SQL1 einschließlich der referentiellen Integrität und der Unterstützung für Modulsprache und eingebettete SQL-Schnittstellen für sieben verschiedene Programmiersprachen.

Zwischenstufe

Zusätzliche Funktionalität für Schemaänderungen, dynamisches SQL, Isolationsebenen für die Transaktionsverarbeitung, kaskadierte referentielle Löschaktionen, Zeilen- und Tabellenausdrücke, Union Joins, Schnittmengen von Tabellen (Intersection), Differenzoperationen, Domänen, CASE-Ausdrücke, Typumwandlungen, mehrere Zeichensätze, Datums-/Zeit-Typen und Zeichenstrings variabler Länge.

Stufe des vollständigen Standards

Diese Stufe schließt aufgeschobene Bedingungsprüfung, Namensbedingungen, selbstreferenzierende Aktualisierungen und Löschungen, kaskadierte referentielle Aktualisierungsaktionen, Unterabfragen in Testklauseln, bildlauffähige Cursor, Zeichenüber-setzungen, Bitstring-Datentypen, temporäre Tabellen und einfache Annahmen (Assertions) ein.

SQL3

Diese Spezifikation ist gerade in Arbeit und soll folgende Merkmale aufweisen, die zum Teil bereits in SQL Server 7.0 umgesetzt sind:

- Unterstützung aktiver »Regeln«, sogenannter Trigger
- Unterstützung abstrakter Datentypen

- Unterstützung von mehreren Nullzuständen
- Unterstützung für PENDANT referentielle Integrität
- Rekursive UNION-Operation für Abfrageausdrücke
- Unterstützung für Aufzählungs- und boolesche Datentypen
- Unterstützung für SENSITIVE Cursor (d.h. Cursor, die Änderungen an den zugrundeliegenden Daten widerspiegeln)

1.6.2 Grundzüge von SQL

Microsoft SQL Server verwendet eine erweiterte Version der Sprache SQL: Transact-SQL. Der Sprachumfang entspricht der Eingangsstufe des Standards SQL-92. Außerdem werden verschiedene Merkmale der Zwischenstufe und der Stufe des vollständigen Standards unterstützt.

Die Anweisungen der Sprache Transact-SQL lassen sich den allgemeinen Kategorien Datendefinitionssprache, Datenmanipulationssprache und Datensteuerungssprache zuordnen.

Datendefinitionssprache

Alle Objekte einer SQL-Server-Datenbank lassen sich mit der Datendefinitionssprache (DDL - Data Definition Language) definieren. Dieser Teil des Sprachumfangs von Transact-SQL basiert auf DDL-Anweisungen von SQL-92 mit Erweiterungen. Die Datendefinitionssprache kommt vornehmlich für administrative Zwecke zum Einsatz. Typische Vertreter der DDL sind die SET-Anweisungen zum Einstellen von Optionen und die CREATE-Anweisungen zum Erstellen von Datenbankobjekten.

Datenmanipulationssprache

Die Datenmanipulationssprache (DML - Data Manipulation Language) ist eine Teilmenge der Datenbanksprache, mit der sich Daten abrufen und bearbeiten lassen. Mit den Anweisungen der Datenmanipulationssprache werden Daten in Objekten ausgewählt, aktualisiert und gelöscht oder in die Objekte eingefügt. Dabei handelt es sich um die Objekte, die mit Hilfe der DDL definiert wurden. Beispiele für die DML sind SELECT, INSERT, UPDATE.

Datensteuerungssprache

Mit den Anweisungen der Datensteuerungssprache (DCL - Data Control Language) steuert man die Berechtigungen für Datenbankobjekte. Typische Anweisungen sind GRANT und REVOKE.

---

kapitel 2 installieren und einrichten

2.1 vor dem start

in computerbüchern, die sich mit standardsoftware beschäftigen, wird die installation gewöhnlich in einen anhang verbannt. oft ist es ja auch damit getan, die cd einzulegen, die installation zu starten, ein paar angaben wie seriennummer, eigener name und name der firma einzugeben sowie gegebenenfalls ein paar optionen auszuwählen.

bei sql server sind die dinge etwas komplizierter. die installation erfordert eine gründliche vorbereitung, da spätere änderungen bei bestimmten optionen nur mit großem aufwand zu bewerkstelligen sind. selbst wenn sie sql server als separates system in der desktop-edition installieren, um hier die entwicklung von anwendungsprogrammen oder den test von gespeicherten prozeduren durchzuführen, müssen sie die gleichen vorbereitenden schritte unternehmen, damit im praktischen einsatz ihrer kreationen keine probleme auftreten.

2.2 vorbereitungen

bevor sie mit der installation von sql server beginnen, sollten sie sich zunächst über die nachstehend beschriebenen punkte klarheit verschaffen. einmal gewählte einstellungen wie sortierreihenfolge oder zeichensatz lassen sich später nur mit großem aufwand ändern. bei der installation von sql server sollten sie also auf experimente verzichten und bereits im vorfeld eine installationsstrategie entwickeln. am ende dieses abschnitts finden sie eine checkliste mit den für die installation erforderlichen angaben. auch wenn sie sql server zu testzwecken oder zur programmentwicklung auf einer separaten workstation installieren, sollten sie von vornherein die optionen der sql-server-installationen auf den übrigen rechnern des firmennetzes übernehmen, damit bei der späteren überführung etwaiger gespeicherter prozeduren oder anderer komponenten in die produktionsumgebung keine probleme entstehen.

2.2.1 anforderungen an hardware und software

der folgende überblick nennt die voraussetzungen bezüglich der hardware- und software-ausstattung, die für den betrieb eines sql-server-systems erforderlich bzw. empfohlen sind.

betriebssysteme

sql server ist vorrangig auf das betriebssystem windows nt ausgerichtet. unter diesem betriebssystem sind vor allem die aspekte der sicherheit hervorragend gelöst. es gibt aber auch editionen von sql server, die sich unter windows 95/98 installieren lassen.

speicher

der minimale speicherbedarf ist 32 mbyte. allerdings läßt sich die leistung mit einem größeren hauptspeicher wesentlich verbessern. eine speicherverwaltung bzw. das abstimmen des speicherbedarfs ist in der version 7.0 nicht mehr erforderlich, da sql server den vorhandenen speicher intelligent nutzt.

laufwerke

lese- und schreibvorgänge stellen einen typischen engpaß in datenbanksystemen dar. deshalb spielt die wahl des geeigneten datenträgersystems eine wichtige rolle. statt einer einzigen großen festplatte ist es oftmals günstiger, mehrere kleinere einzusetzen und die datenbanken mit den zugehörigen transaktionsprotokollen auf verschiedene physikalische laufwerke aufzuteilen. weiterhin sollten sie einen eventuell vorhandenen schreibcache abschalten, da die datenbankwiederherstellung von sql server mit durchsetzung der datenintegrität zu problemen führen kann.

dateisystem

wenn sie mit windows nt arbeiten und die wahl zwischen den dateisystemen ntfs und fat haben, spielt die entscheidung für ein dateisystem allein aus gründen der leistungsbilanz praktisch keine rolle. mit ntfs laufen die lesevorgänge zügiger ab, während bei fat die schreibvorgänge schneller sind. ntfs empfiehlt sich vor allem dann, wenn sie die sicherheits- und überwachungsmechanismen von windows nt nutzen wollen. bei einem dual-boot-computer sollten sie dagegen bei fat bleiben.

empfohlene software- und hardware-ausstattung

für die installation von sql server gibt microsoft die in tabelle 2.1 aufgeführten anforderungen an die hardware und software an.

kategorie	anforderungen
plattform	dec alpha intel und kompatible systeme (pentium ab 166 mhz, pentium pro oder pentium ii)
arbeitsspeicher	enterprise edition: 64 mbyte (minimum) andere ausgaben: 32 mbyte (maximum)
festplattenplatz	vollständige installation: 180 mbyte standardinstallation: 170 mbyte minimale installation: 65 mbyte nur verwaltungstools: 90 mbyte
betriebssysteme	enterprise: windows nt server enterprise edition ab 4.0 mit sp4 oder höher standard/sbs: windows nt server 4.0 oder höher mit sp4 oder höher. desktop/nur verwaltungstools: windows nt workstation 4.0 oder höher mit sp4 oder höher, microsoft windows 95/98
internet-software	internet explorer 4.01 mit sp1 oder höher
netzwerksoftware	integrierte netzwerksoftware von windows nt oder windows 95/98

unterstützte clients	windows 95/98, windows nt workstation, unix, apple macintosh und os/2

tabelle 2.1: hardware- und software-anforderungen

zeichensatz

mit dem zeichensatz legen sie die gültigen zeichen für ihre sql-server-datenbanken fest.

wenn sie den zeichensatz während der installation festlegen, ist das quasi eine entscheidung fürs leben, denn der zeichensatz läßt sich nur mit großem aufwand umstellen. um den zeichensatz zu ändern, müssen sie alle daten aus den datenbanken exportieren, die datenbanken neu erstellen und anschließend die exportierten daten in die neu erstellten datenbanken einlesen.

jeder zeichensatz besteht aus 256 zeichen, wobei die ersten (druckbaren) 128 zeichen mit den zeichencodes von 0 bis 127 in allen zeichensätzen gleich sind. die einzelnen zeichensätze unterscheiden sich bei den erweiterten zeichen in den zeichencodes 128 bis 255.

aus den buchstaben, ziffern und symbolen dieser zeichensätze lassen sich die werte der zeichenbasierten datentypen char, varchar und text darstellen. der speicherbedarf beträgt bei diesen datentypen 1 byte pro zeichen.

tabelle 2.2 listet die in sql server verfügbaren zeichensätze auf. die ersten drei zeichensätze sind die gebräuchlichsten.

codepage	beschreibung
1252 (iso, 8859-1, lateinisch 1 oder ansi)	dieser iso-zeichensatz ist kompatibel mit dem unter windows nt und windows 95/98 verwendeten ansi-zeichensatz. die codeseite 1252 gilt als standardeinstellung für sql server der versionen 6.x und 7.0.
850	mehrsprachig. dieser zeichensatz enthält alle zeichen für die europäischen, nord- und südamerikanischen länder.
437	us-englisch. allgemeiner zeichensatz in den vereinigten staaten. diese codeseite enthält auch verschiedene grafikzeichen, die normalerweise nicht in einer datenbank gespeichert werden.
874	thai
932	japanisch
936	chinesisch (vereinfacht)
949	koreanisch
950	chinesisch (traditionell)
1250	mitteleuropäisch
1251	kyrillisch

1253	griechisch
1254	türkisch
1255	hebräisch
1256	arabisch
1257	baltisch

tabelle 2.2: verfügbare zeichensätze in sql server

sortierreihenfolge

die gewählte sortierreihenfolge bestimmt, wie die daten in sql-abfragen mit den klauseln group by, order by und distinct dargestellt werden. weiterhin legen sie mit der sortierreihenfolge fest, ob die groß-/kleinschreibung zu berücksichtigen ist.

die wahl der sortierreihenfolge ist genauso wichtig wie die entscheidung für einen zeichensatz.

wenn sie sich für eine sortierreihenfolge mit unterscheidung der groß-/kleinschreibung - dazu gehört auch die binärreihenfolge - entscheiden, müssen sie auch die schreibweise von objektnamen genau einhalten.

zu den einzelnen codeseiten gehört jeweils ein spezieller satz von sortierreihenfolgen. die folgende übersicht erläutert die für die codeseite 1252 verfügbaren optionen.

wörterbuchreihenfolge, keine unterscheidung der groß-/kleinschreibung

diese sortierreihenfolge ist die standardeinstellung für sql server der versionen 6.x und 7.0. in einer sortierten ergebnismenge sind die einträge in genau der reihenfolge angeordnet, wie man sie aus wörterbüchern kennt. bei vergleichen werden groß- und kleinbuchstaben gleichwertig behandelt. buchstaben mit diakritischen zeichen sind selbständige zeichen. ein umlaut wie ä unterscheidet sich demnach vom buchstaben a oder einem zeichen mit akzent wie á oder à. außerdem ist nicht vorherzusehen, ob der großbuchstabe oder der kleinbuchstabe zuerst eingeordnet wird.

binärreihenfolge

diese art der sortierung basiert ausschließlich auf den numerischen werten (0 bis 255) der zeichen des installierten zeichensatzes und ist demnach die schnellste sortierreihenfolge. da aber in allen zeichensätzen die großbuchstaben vor den kleinbuchstaben eingeordnet sind und damit wörter wie »zwei« vor »eins« stehen, kommt die binärreihenfolge sicherlich nur für spezielle datenbanken in frage.

wörterbuchreihenfolge, unterscheidung nach groß-/kleinschreibung

diese sortierreihenfolge behandelt groß- und kleinbuchstaben sowie buchstaben mit diakritischen zeichen unterschiedlich.

wörterbuchreihenfolge, keine unterscheidung nach groß-/kleinschreibung, großschreibung bevorzugt

bei wörtern mit gleichen anfangsbuchstaben stehen wörter mit großbuchstaben vor wörtern mit den entsprechenden kleinbuchstaben, also »eins« vor »eins«, und sind ansonsten lexikalisch geordnet, beispielsweise »eins« vor »zwei«. großbuchstaben mit diakritischen zeichen stehen vor den entsprechenden kleinbuchstaben. buchstaben mit diakritischen zeichen kommen nach buchstaben ohne diakritische zeichen und stehen ansonsten an der position in der sortierreihenfolge, die ihrer position im zeichensatz entspricht.

wörterbuchreihenfolge, keine unterscheidung nach groß-/kleinschreibung oder akzenten

diese sortierreihenfolge behandelt groß- und kleinbuchstaben als identische zeichen. das gleiche gilt für buchstaben mit und ohne diakritische zeichen.

weitere sortierreihenfolgen

in der online-dokumentation finden sie weitere sortierreihenfolgen, die für die schreibweise in den skandinavischen ländern zutreffend sind.

unicode-reihenfolge

die unicode-daten wurden erst mit sql server 7.0 eingeführt. dieser datentyp belegt den doppelten platz gegenüber den normalen ansi-zeichen, d.h. 2 byte pro zeichen. während sich im ansi-zeichensatz mit einem byte pro zeichen lediglich 256 verschiedene zeichen darstellen lassen, sind es im unicode-zeichensatz 65536 zeichen. in sql server sind die datentypen nchar, nvarchar und ntext für die speicherung von unicode-daten vorgesehen. das *n* vor den datentypen kommt vom standard sql-92 und bezeichnet nationale datentypen.

sie sollten beim upgrade von sql server nicht die unicode-reihenfolge ändern, sondern die standardeinstellung übernehmen. die datenbank master der version 7.0 enthält unicode-spalten, die aufgrund der in 6.x nicht vorhandenen unicode-unterstützung im normalen ansi-zeichensatz vorlagen. wenn sie eine abweichende unicode-reihenfolge wählen, kann es beim sortieren von 6.x-datenbanken probleme geben.

netzwerk

sql server unterstützt mehrere netzwerkoptionen gleichzeitig. somit können sich clients mittels tcp/ip parallel zu clients mit ipx/spx mit sql server verbinden. in der voreinstellung installiert sql server die protokolle tcp/ip, multiprotokoll und named pipes. bei der entscheidung für eine bestimmte netzwerkbibliothek sollten sie die verschiedenen sicherheitsmodi von sql server berücksichtigen.

- windows-nt-authentifizierung: nutzt die mechanismen der sicherheit von windows nt. es ist eine vertraute verbindung erforderlich. die windows-nt-authentifizierung läßt sich über named pipes oder multiprotokoll realisieren. wenn sql server unter windows 95/98 läuft, steht die windows-nt-authentifizierung nicht zur verfügung.
- gemischter modus: die benutzer können wählen, ob sie sich per windows-nt- oder sql-server-authentifizierung anmelden wollen. bei sql-server-authentifizierung sind alle von sql server unterstützten netzwerkprotokolle möglich, bei windows-nt-authentifizierung nur named pipes oder multiprotokoll.

kapitel 21 geht näher auf die sicherheitsaspekte ein.

2.2.2 release notes

in der datei readme.txt finden sie wichtige hinweise, die sie vor der installation lesen sollten. unter anderem geht es um folgende punkte:

- installationsprobleme in verbindung mit anderen programmen
- umstieg von sql server 6.x auf 7.0
- übergang von beta 3 auf endgültige version des sql server 7.0
- hinweise zu verschiedenen assistenten
- technische details zu transact-sql-anweisungen
- sql-dmo

außerdem finden sie in readme.txt informationen, die nicht in der sql-server-online-dokumentation enthalten sind:

- microsoft english query
- microsoft sql server olap services

2.2.3 sql-server-editionen

microsoft liefert sql server in drei verschiedenen editionen aus, die sich vor allem hinsichtlich der vorgesehenen betriebssysteme und des darauf zugeschnittenen leistungsumfangs unterscheiden. aufgrund der skalierbarkeit vom notebook bis zum cluster sind die grundlegenden funktionen und der programmcode bei allen editionen gleich. die einzelnen editionen werden als sbs (small business server), standard und enterprise bezeichnet.

daneben gibt es noch eine sogenannte desktop-ausgabe, die es als einzige ausgabe erlaubt, die server-komponenten unter windows 95/98 zu installieren. das programmiermodul der desktop-ausgabe ist mit den ausgaben small business server, standard und enterprise von sql server identisch. jeder benutzer, der über eine pro-arbeitsplatz-client-zugriffslizenz für sql server verfügt, kann die server-komponenten von sql server desktop edition unter windows 95/98 installieren.

auch wenn sql server nach außen hin als einheitliches produkt erscheint, muß man grundsätzlich zwischen client-software und server-software unterscheiden.

tabelle 2.3 bringt eine übersicht über die unterstützten bzw. nicht unterstützten leistungsmerkmale dieser editionen.

merkmal	sbs	standard	enterprise
läuft auf microsoft windows small business server	ja	ja	nein
läuft auf microsoft windows nt server	nein	ja	nein

läuft auf windows nt enterprise	nein	ja	ja
maximale datenbankgröße	10 gbyte	unbegrenzt	unbegrenzt
anzahl von smp cpus	4	4	32
unterstützung von erweitertem speicher	nein	nein	ja
fehlertolerantes clustern	nein	nein	ja
unterstützung von microsoft search service, volltextkataloge und volltextindizes	ja	ja	ja
unterstützt microsoft sql server olap services	nein	ja (keine benutzerdefinierten cube-partitionen)	ja (einschließlich benutzerdefinierter cube-partitionen)

tabelle 2.3: unterschiede der editionen von sql server 7.0

2.2.4 desktop edition

die small business server (sbs) edition ist für etwa 50 gleichzeitig angemeldete benutzer ausgelegt. allerdings können bei bestimmten sbs-installationen auch weniger lizenzen vergeben sein. die desktop-edition kann remote-verbindungen akzeptieren und ist vor allem für mobile clients ohne ständige verbindung zu sql server der »großen« editionen oder für clients, deren anwendungen gelegentlich datenbanken lokal speichern, vorgesehen.

2.2.5 checkliste zur installation von sql server

anhand der folgenden checkliste können sie die voraussetzungen für eine sql-server-installation prüfen bzw. die erforderlichen angaben bereitstellen oder während der installation ergänzen.

hardware/betriebssystem

- computer: alpha axp oder intel-pentium ab 166 mhz aufwärts; in der hardware-kompatibilitätsliste von windows nt aufgeführt.
- speicherausstattung: 32 mbyte (standard) oder 64 mbyte (enterprise edition).
- betriebssystem: windows nt 4.0 server oder workstation mit installiertem service pack 4; windows 95/98
- speicherbedarf auf festplatte (siehe tabelle 2.4):

laufwerk	minimum	standard	benutzerdefiniert (alle optionen)
programmdateien	15794	83754	91269
system	37916	67251	91458
datendateien	22347	32497	32497

tabelle 2.4: speicherbedarf (in kbyte)

- dateisystem: fat oder ntfs

anmeldung

- benutzername: _____
- firmenname (optional): _____
- cd-key: ___-_____
- servername: _____
- anmeldekonto für mssqlserver: _____ (nur windows nt)
- anmeldekonto für sqlserveragent: _____ (nur windows nt)

verzeichnisse

- stammverzeichnis für sql server: _____
- verzeichnis für programmdateien: _____
- verzeichnis für datendateien: _____

zeichensatz/sortierreihenfolge

- codepage 1252 (iso-zeichensatz, standard)
- codepage 850 (mehrsprachig)
- codepage 437 (us-englisch)
- andere: _____
- gewählte sortierreihenfolge: _____
- gewählte unicode-sortierreihenfolge: _____

netzwerkbibliotheken

- named pipes (standard und bei windows nt erforderlich, unter windows 95 nicht verfügbar)
- tcp/ip-sockets (standard bei windows 95/98)
- multiprotokoll
- nwlink ipx/spx
- banyan vines
- appletalk adsp

sonstiges

- internet explorer 4.01 mit service pack 1 installiert

2.3 installation

wenn auf ihrem computer der automatische start von cd aktiviert ist, erscheint der startbildschirm für die installation von sql server nach dem einlegen der cd gemäß abbildung 2.1. gehen sie andernfalls im cd-laufwerk in das verzeichnis, das ihrer plattform entspricht, und doppelklicken sie auf **setup**.

bild

abbildung 2.1: der startbildschirm der sql-server-installation

2.3.1 sql server 7.0-voraussetzungen installieren

für eine desktop-installation unter windows 98 sind keine voraussetzungen erforderlich. wenn sie die installation unter windows 95 oder windows nt vornehmen, klicken sie auf die option *sql server 7.0-voraussetzungen installieren*. im nächsten dialogfeld wählen sie das betriebssystem, unter dem sie sql server installieren wollen.

voraussetzungen für windows 95 installieren

bei der installation unter windows 95 zeigt das nächste dialogfeld vier schaltflächen (siehe abbildung 2.2).

bild

abbildung 2.2: auswahl der zu installierenden voraussetzungen für windows 95

wenn sie den mauszeiger über die schaltflächen führen, erscheint im linken fensterbereich eine kurze erläuterung. da internet explorer ohnehin für die anzeige von hilfedateien erforderlich ist, bleibt ihnen praktisch die entscheidung zwischen minimalinstallation und vollständiger installation (option *setup-assistent starten*) von internet explorer 4.01 sp1. beide optionen starten das setup-programm von internet explorer, bei der minimalinstallation wird nur das dialogfeld installationsoptionen (in dem sie zwischen standardinstallation und vollständiger installation von internet explorer wählen können) und das dialogfeld der web-erweiterungen für den active desktop übersprungen.

voraussetzungen für windows nt installieren

die installation der voraussetzungen unter windows nt läuft in zwei schritten ab (siehe abbildung 2.3).

bild

abbildung 2.3: auswahl der zu installierenden voraussetzungen für windows nt 4.0

als erstes ist das service pack 4 für windows nt zu installieren. daran schließt sich wie bei windows 95 die installation von internet explorer 4.01 sp1 an.

bild

abbildung 2.4: die startseite für die installation von sp4 für windows nt 4.0

das service pack 4 finden sie auf einer separaten cd, die zum lieferumfang von sql server gehört. wenn sie die cd einlegen und die autostart-funktion aktiviert ist, startet internet explorer und zeigt die startseite der installation an (siehe abbildung 2.4).

klicken sie im linken rahmen auf **installieren von service pack 4**. im dialogfeld **datei-download** wählen sie die option *das programm von diesem ort ausführen*.

nachdem die installation von sp4 abgeschlossen ist, legen sie wieder die erste cd mit sql server ein und fahren mit dem zweiten schritt für die installation der voraussetzungen fort. wie bei windows 95 können sie zwischen minimalinstallation und setup-assistent wählen.

2.3.2 sql server 7.0-komponenten installieren

wenn sie die sql server 7.0-voraussetzungen installiert haben oder diese für ihre installation nicht relevant sind, klicken sie im startbildschirm auf den eintrag *sql server 7.0-komponenten installieren*. es erscheint das in abbildung 2.5 dargestellte dialogfeld, in dem sie die gewünschte edition wählen können.

bild

abbildung 2.5: zu installierende edition auswählen

edition auswählen

zum installieren der vollständigen version von sql server 7.0 (unter windows nt) klicken sie im zweiten dialogfeld auf die erste option *datenbankserver - standard edition*.

setup bereitet den installshield-assistenten vor und zeigt dann das dialogfeld **installationsmethode auswählen** an (siehe abbildung 2.6).

abbildung 2.6: das dialogfeld installationsmethode auswählen

übernehmen sie die vorgegebene option *lokale installation*, wenn sie sql server auf dem computer installieren, von dem aus sie das setup ausführen. klicken sie auf **weiter**.

das dialogfeld **willkommen** (siehe abbildung 2.7) empfiehlt, alle windows-programme zu beenden. außerdem enthält es einen hinweis auf das urheberrecht. diesem dialogfeld begegnen sie in ähnlicher form bei der installation von english query, allerdings ist es dort momentan nur in englisch zu sehen.

bild

abbildung 2.7: das dialogfeld willkommen

das gleiche gilt für den software-lizenzvertrag im nächsten dialogfeld (siehe abbildung 2.8).

bild

abbildung 2.8: das dialogfeld software-lizenzvertrag

klicken sie auf **ja**, um dem software-lizenzvertrag zuzustimmen. (etwas anderes bleibt ihnen nicht übrig, wenn sie sql server installieren wollen.)

daraufhin erscheint das dialogfeld **benutzerinformationen**, in dem sie ihren namen eingeben müssen und optional die firma angeben können.

als nächstes fragt das setup-programm den zehnstelligen »cd key« ab, den sie auf dem gelben aufkleber

der cd-hülle oder im cd-heft finden.

wenn auf dem computer bereits die version 6.x von sql server installiert ist, erscheint als nächstes das dialogfeld **vorhandene sql server-daten konvertieren** (siehe abbildung 2.9).

abbildung 2.9: bei einer vorhandenen installation von sql server version 6.x erscheint dieses dialogfeld

wenn sie das kontrollkästchen einschalten, startet der sql-server-aktualisierungs-assistent nach dem setup von sql server 7.0. die aktualisierung können sie aber auch später durchführen. klicken sie auf **weiter**.

auf die versionsumstellung von sql server 6.x nach 7.0 geht kapitel 29 näher ein.

im dialogfeld **setup-typ** (siehe abbildung 2.10) können sie unter folgenden optionen wählen (in klammern ist der speicherbedarf für programmdateilaufwerk, systemlaufwerk und datendateilaufwerk in kbyte angegeben):

- *standard*: empfohlen für die meisten benutzer. installiert sql server vollständig entsprechend der standardoptionen. nicht zu den standardoptionen gehören die volltextsuche, die entwicklungstools und die codebeispiele (83754, 67251, 32497).
- *minimum*: diese option empfiehlt sich, wenn der computer nur über begrenzten speicherplatz verfügt. installiert nur die kerndienste von sql server und die client-zugriffskomponenten. nicht installiert werden die verwaltungstools (zum beispiel enterprise manager, query analyzer) und die online-dokumentation (15793, 37916, 22347).
- *benutzerdefiniert*: empfohlen für versierte benutzer. hier gelten dieselben standardoptionen wie bei

der installationsart *standard*. allerdings haben sie die möglichkeit, die vorgegebenen optionen zu ändern. wenn sie alle optionen aktivieren, beträgt der speicherbedarf 91269, 91458 bzw. 32497 kbyte.

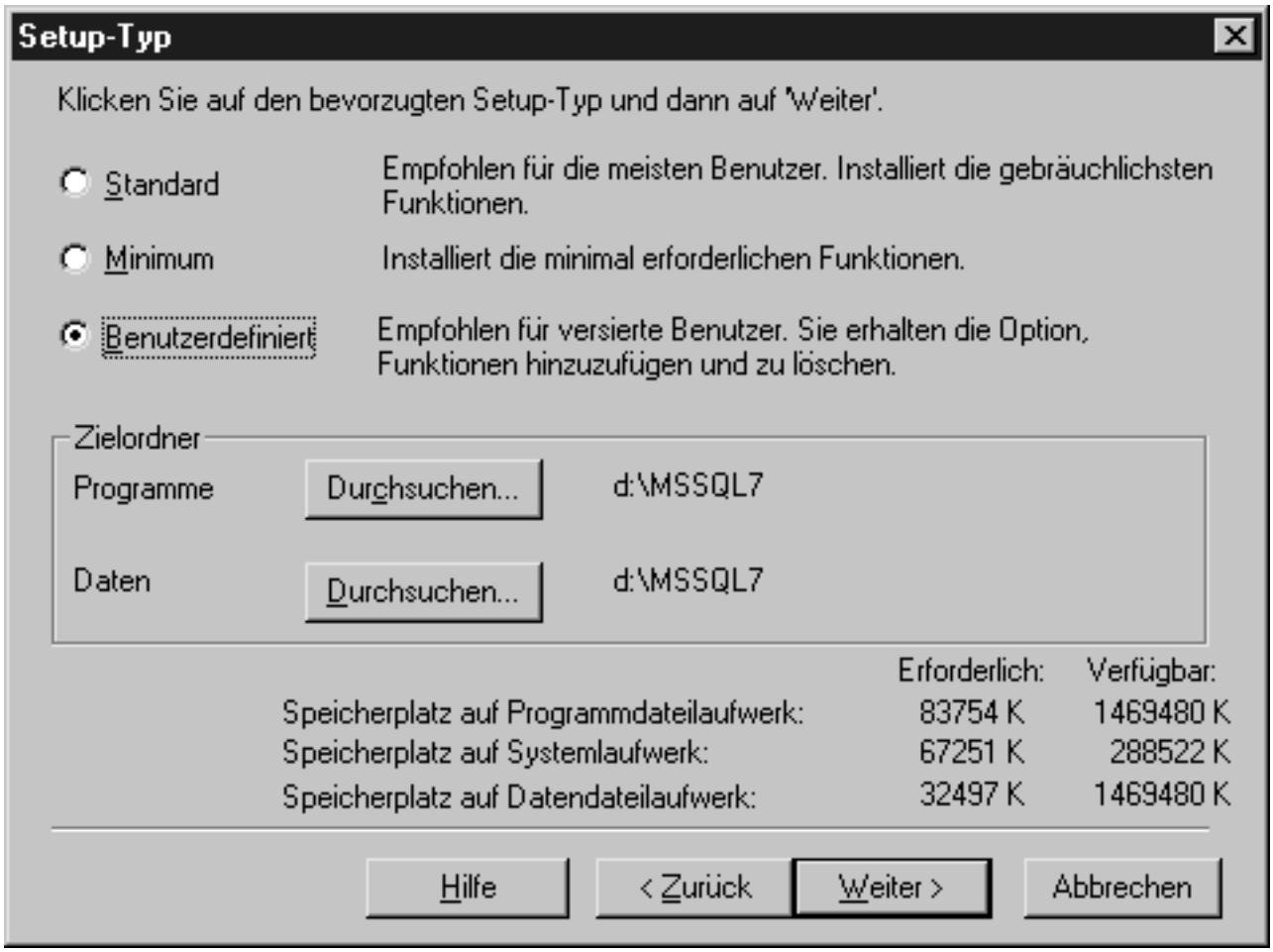


abbildung 2.10: das dialogfeld setup-typ

im abschnitt **zielordner** können sie andere als die vorgegebenen verzeichnisse für programme und daten wählen. für die beispielinstallation wurde lediglich das laufwerk in d: geändert.

benutzerdefinierte installation

die nachstehenden schritte erläutern die benutzerdefinierte installation:

1. wählen sie die option *benutzerdefiniert*, und stellen sie bei bedarf im abschnitt **zielordner** andere verzeichnisse ein. klicken sie auf **weiter**. (bei der standardinstallation und der minimalinstallation überspringen sie die schritte 2 bis 4.)
2. im dialogfeld **komponenten auswählen** legen sie fest, welche komponenten sie installieren möchten. abbildung 2.11 zeigt die per voreinstellung ausgewählten komponenten. wenn sie sich näher mit sql server beschäftigen oder programme entwickeln möchten, sollten sie zusätzlich die komponenten *entwicklungstools* und *codebeispiele* aktivieren. im fenster **unterkomponenten** können sie die auswahl an ihre bedürfnisse anpassen. im unteren teil des dialogfelds erscheint der erforderliche und vorhandene speicherplatz auf den gewählten laufwerken. klicken sie auf **weiter**.

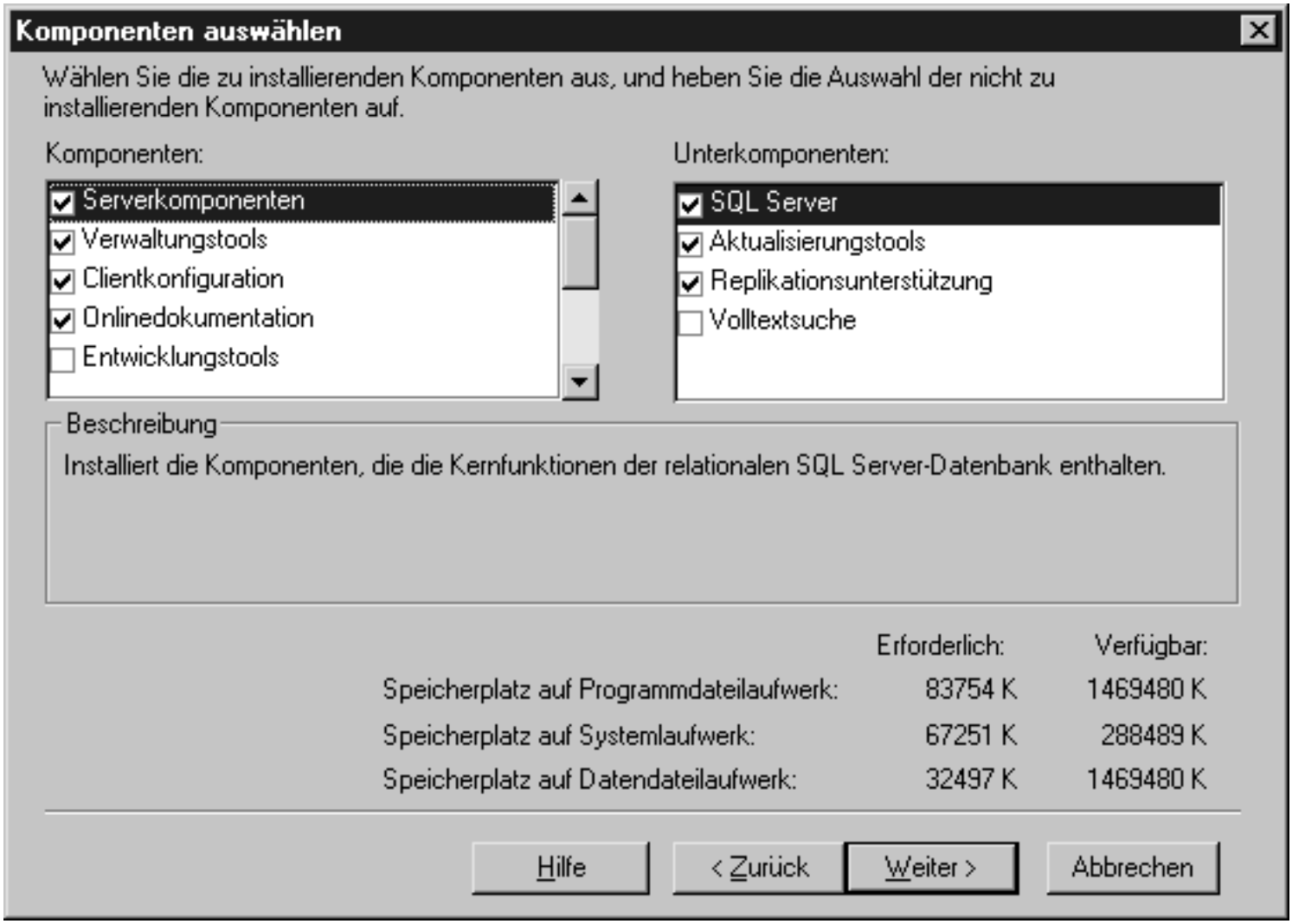


abbildung 2.11: das dialogfeld komponenten auswählen

3. im dialogfeld **zeichensatz/sortierreihenfolge/unicode-sortierreihenfolge** sollten sie die vorgegebenen einstellungen übernehmen, sofern ihre bereits vorhandenen daten keine anderen einstellungen erfordern (siehe abbildung 2.12). klicken sie auf **weiter**.

bild

abbildung 2.12: einstellen von zeichensatz und sortierreihenfolge

4. im dialogfeld **netzwerkbibliotheken** richten sie die bibliotheken für die kommunikation mit den clients ein. für die installation unter windows nt ist die unterstützung von named pipes erforderlich und läßt sich nicht deaktivieren. die netzwerkbibliothek tcp/ip-sockets wird von windows 95/98 als standard verwendet. die vorgegebene anschlußnummer 1433 ist die offizielle socketnummer der internet assigned number authority für sql server. weiterhin ist das kontrollkästchen **multiprotokoll** aktiviert. diese netzwerkbibliothek nutzt die vorteile der remote-prozeduraufrufe von windows und benötigt im unterschied zu anderen bibliotheken keine konfigurationsparameter. es empfiehlt sich, die vorgegebenen einstellungen in diesem dialogfeld zu übernehmen, falls ihnen der netzwerkadministrator keine anderen forderungen unterbreitet.
5. unter windows nt müssen sie jedem dienst von sql server ein anmeldekonto zuweisen, damit sich die dienste mssqlserver und sqlserveragent starten und ausführen lassen. während der installation nehmen sie diese einstellungen im dialogfeld **dienstkonten** (siehe abbildung 2.13) vor. da der

sqlserveragent-dienst ohnehin vom dienst mssqlserver abhängig ist und nur automatisch gestartet werden kann, wenn auch für mssqlserver der automatische start eingestellt ist, können sie die voreinstellung *dasselbe konto für jeden dienst verwenden* übernehmen. im abschnitt diensteinstellungen stehen die optionen *konto 'lokales system' verwenden* und *domänenbenutzerkonto verwenden* zur auswahl. es wird empfohlen, die vorgegebene zweite option zu übernehmen, da sonst die interaktion von sql server mit anderen servern eingeschränkt ist (siehe dazu die online-dokumentation, die sie direkt aus dem dialogfeld über die schaltfläche **hilfe** aufrufen können). im feld **benutzername** ist bereits der name eingetragen, unter dem sie sich bei windows nt angemeldet haben. gleiches gilt für das kennwort.

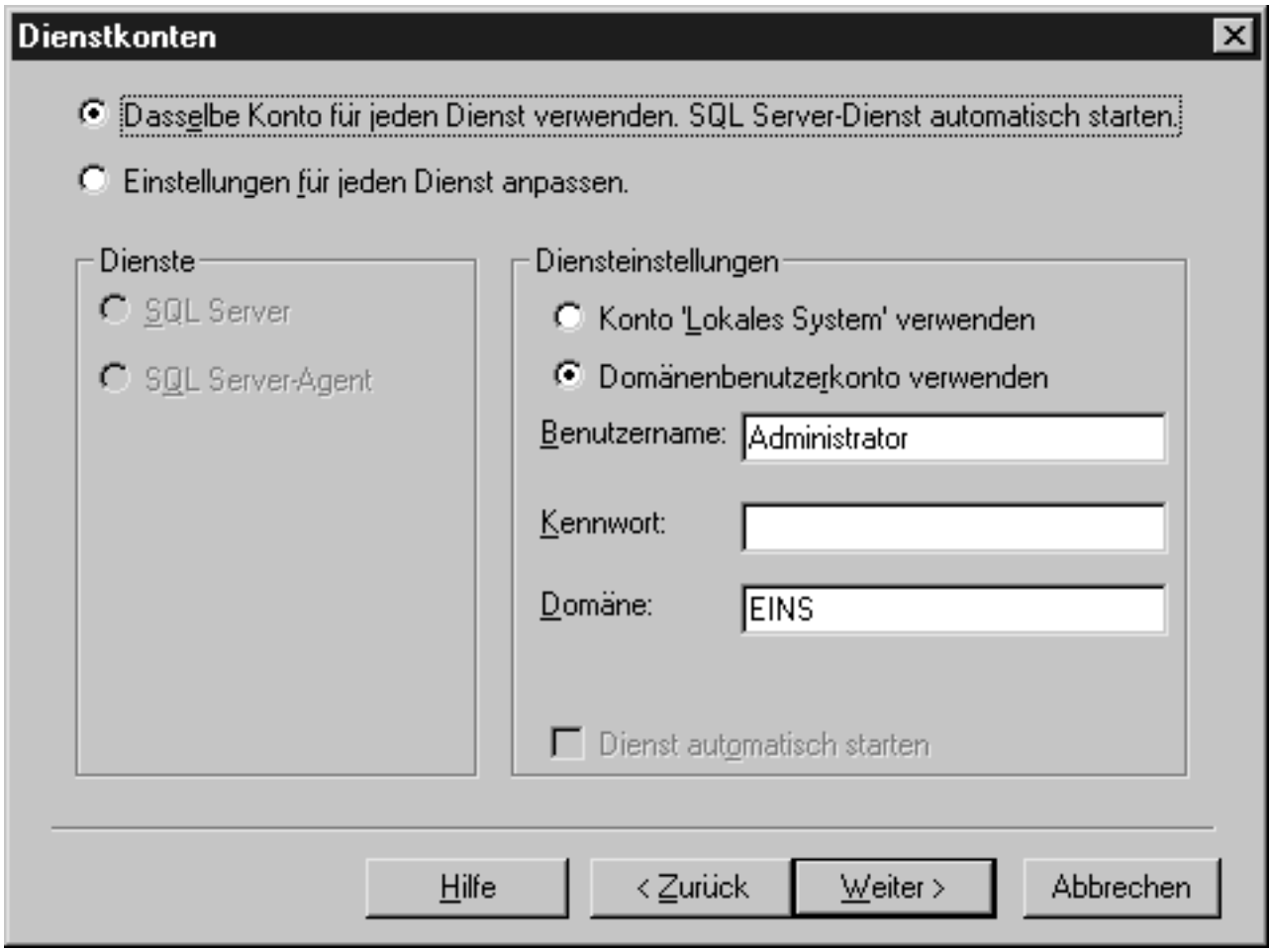


abbildung 2.13: im dialogfeld dienstkonten weisen sie den diensten von sql server anmeldekonten zu

6. klicken sie auf **weiter**, um zum dialogfeld **kopiervorgang starten** zu gelangen. das dialogfeld weist darauf hin, daß als nächstes die informationen zum lizenzierungsmodus abgefordert werden. klicken sie auf **weiter**.
7. im dialogfeld **lizenzierungsmodus wählen** (siehe abbildung 2.14) legen sie den typ der lizenzierung fest. im zweifelsfall wählen sie die option *pro server*, da ihnen der lizenzvertrag die möglichkeit einräumt, zu einem späteren zeitpunkt einmalig in den lizenzierungsmodus *pro arbeitsplatz* zu wechseln. der pro-arbeitsplatz-modus ist endgültig.

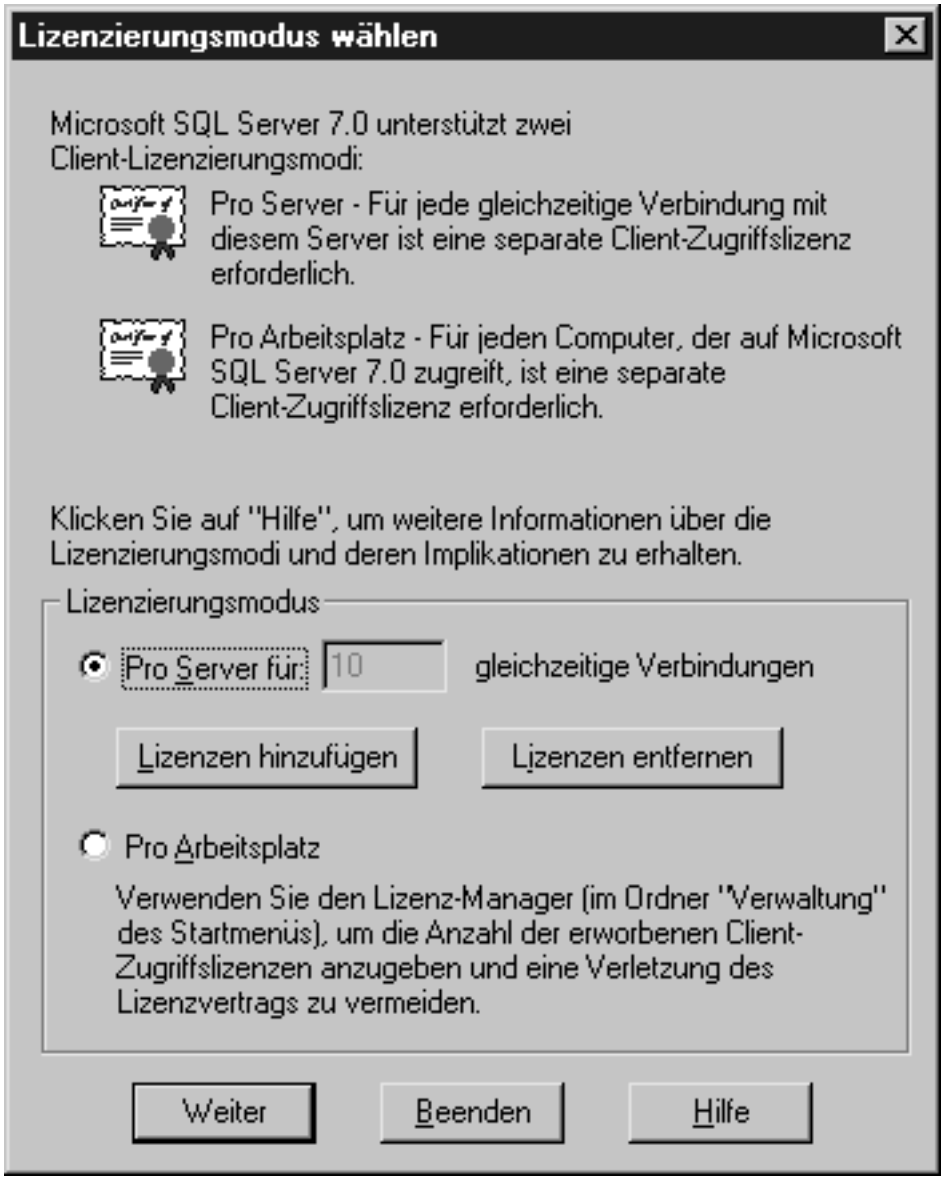


abbildung 2.14: das dialogfeld lizenzierungsmodus wählen

8. klicken sie auf **weiter**, um das kopieren der dateien zu starten.
9. nachdem sql server die installation abgeschlossen hat, erscheint ein dialogfeld mit der erfolgsmeldung. klicken sie auf **beenden**, um das setup abzuschließen.

es empfiehlt sich, den computer neu zu starten, um temporäre dateien der installation zu entfernen.

2.4 olap services installieren

die olap services installieren sie in folgenden schritten:

1. klicken sie im startbildschirm der sql-server-installation (siehe abbildung 2.1) auf *sql server 7.0-komponenten installieren*. im zweiten dialogfeld wählen sie *sql server 7.0 olap services*. daraufhin startet der installshield-assistent, der im ersten dialogfeld **willkommen** wie üblich das beenden aller windows-programme empfiehlt (was wir an dieser stelle voraussetzen). klicken sie auf **weiter**.
2. im dialogfeld **software-lizenzvertrag** klicken sie auf **ja**, wenn sie mit den bestimmungen

einverstanden sind (andernfalls ist keine installation möglich).

3. jetzt müssen sie den zehnstelligen cd-key eingeben, den sie auch für die sql-server-installation verwendet haben. klicken sie auf ok. wenn der cd-key richtig war, erscheint ein dialogfeld mit der product id dieses programms. ansonsten werden sie aufgefordert, den schlüssel erneut einzugeben.
4. im dialogfeld **zielpfad wählen** ist der pfad c:\programme\olap services vorgegeben. klicken sie auf **durchsuchen**, wenn sie olap in einem anderen verzeichnis installieren wollen (in der beispielinstallation wurde lediglich das laufwerk in d: geändert).

bild

abbildung 2.15: zielpfad für die installation von olap services festlegen

5. im nächsten dialogfeld legen sie auf gleiche weise den speicherort der daten fest. vorgegeben ist das verzeichnis \data unterhalb des im schritt 4 festgelegten verzeichnisses.
6. im nächsten dialogfeld wählen sie die komponenten aus, die sie für olap services installieren möchten (siehe abbildung 2.16).

Bild

abbildung 2.16: das dialogfeld komponenten wählen

7. im dialogfeld **programmordner auswählen** legen sie die programmgruppe fest, über die sie olap services starten können. vorgegeben ist ein unterordner von microsoft sql server 7.0.
8. nachdem sie auf **weiter** geklickt haben, installiert das setup-programm die microsoft data access components und die dateien für die im schritt 6 ausgewählten komponenten.
9. als letztes erscheint die meldung über die erfolgreiche installation. klicken sie auf **ok**. dann erscheint die frage, ob sie den computer sofort oder später neu starten wollen. es empfiehlt sich, die vorgegebene option (sofort neu starten) zu wählen.

auf die olap services geht kapitel 26 näher ein.

2.5 english query installieren

wenn sie anwendungen entwickeln wollen, in denen der benutzer abfragen in englischer umgangssprache formulieren kann, installieren sie sql server english query in folgenden schritten:

1. klicken sie im startbildschirm der sql-server-installation (siehe abbildung 2.1) auf *sql server 7.0-komponenten installieren*. im zweiten dialogfeld wählen sie *english query*.
2. das setup-programm installiert zunächst die microsoft data access components.
3. die installation läuft ab hier in englischer sprache, da english query momentan nur in einer englischen version verfügbar ist. das erste dialogfeld weist sie darauf hin, alle geöffneten anwendungen zu schließen, bevor sie mit der installation fortfahren (siehe abbildung 2.17). außerdem finden sie hier einen hinweis auf das urheberrecht und die erlaubnis, english query auf einem computer zu installieren. klicken sie auf **continue** (entspricht der schaltfläche **weiter** in deutschen windows-programmen).

abbildung 2.17: willkommen-dialogfeld bei der installation von english query

4. im nächsten dialogfeld (siehe abbildung 2.18) legen sie den zielordner für die installation fest. vorgegeben ist c:\programme\microsoft english query. die beispielinstallation verwendet ein gleichnamiges verzeichnis auf laufwerk d: klicken sie auf **change folder**, wenn sie den zielordner ändern möchten. falls der zielordner nicht vorhanden ist, fragt das setup-programm, ob es den ordner anlegen soll (siehe abbildung 2.19). klicken sie auf **yes**. schließen sie dann das dialogfeld zur ordnerauswahl.

abbildung 2.18: frage, ob nicht vorhandener ordner angelegt werden soll

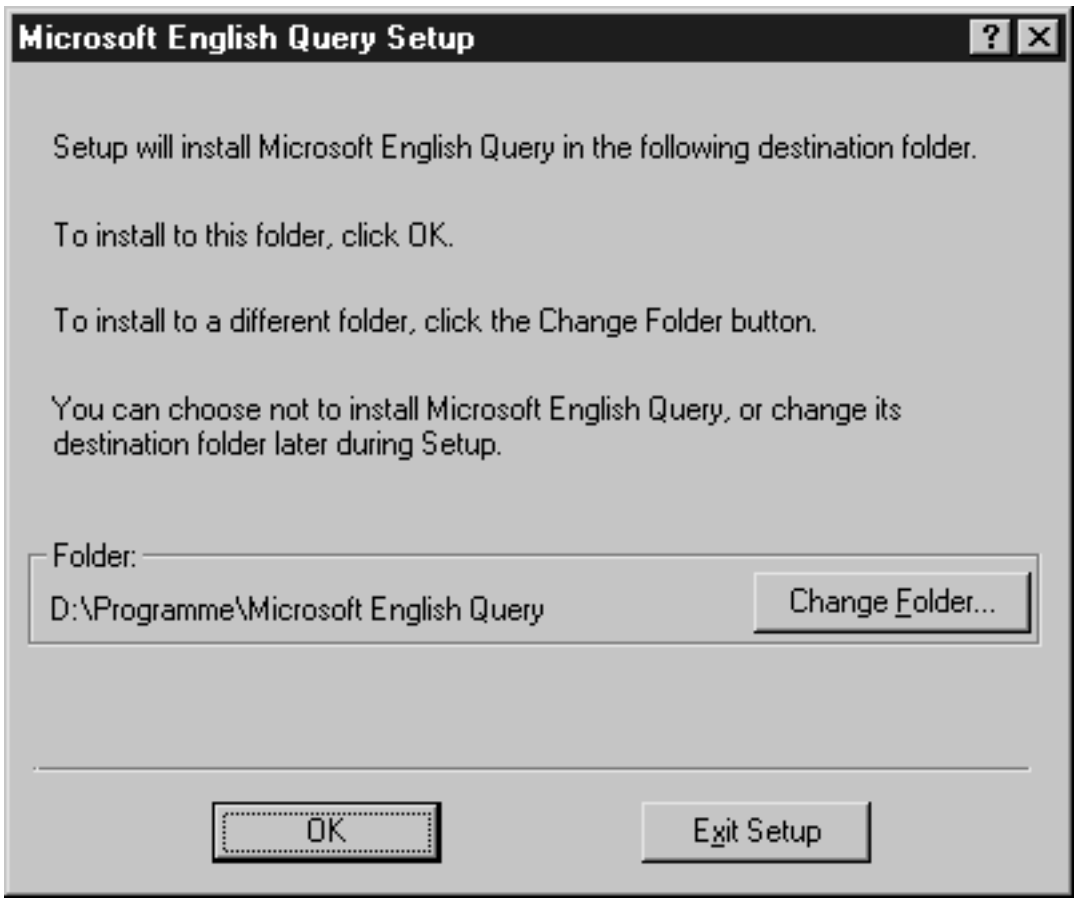


abbildung 2.19: auswahl des zielordners für die installation

5. als nächstes erscheint der software-lizenzvertrag (in englischer und französischer sprache), den sie über die schaltfläche **i agree** akzeptieren.
6. im nächsten dialogfeld (siehe abbildung 2.20) können sie zwischen vollständiger (complete) und benutzerdefinierter (custom) installation wählen. die folgenden erläuterungen beziehen sich auf die benutzerdefinierte installation. außerdem haben sie hier ein letztes mal die möglichkeit, den installationsordner zu ändern (über die schaltfläche **change folder**).

klicken sie auf die schaltfläche **custom**.

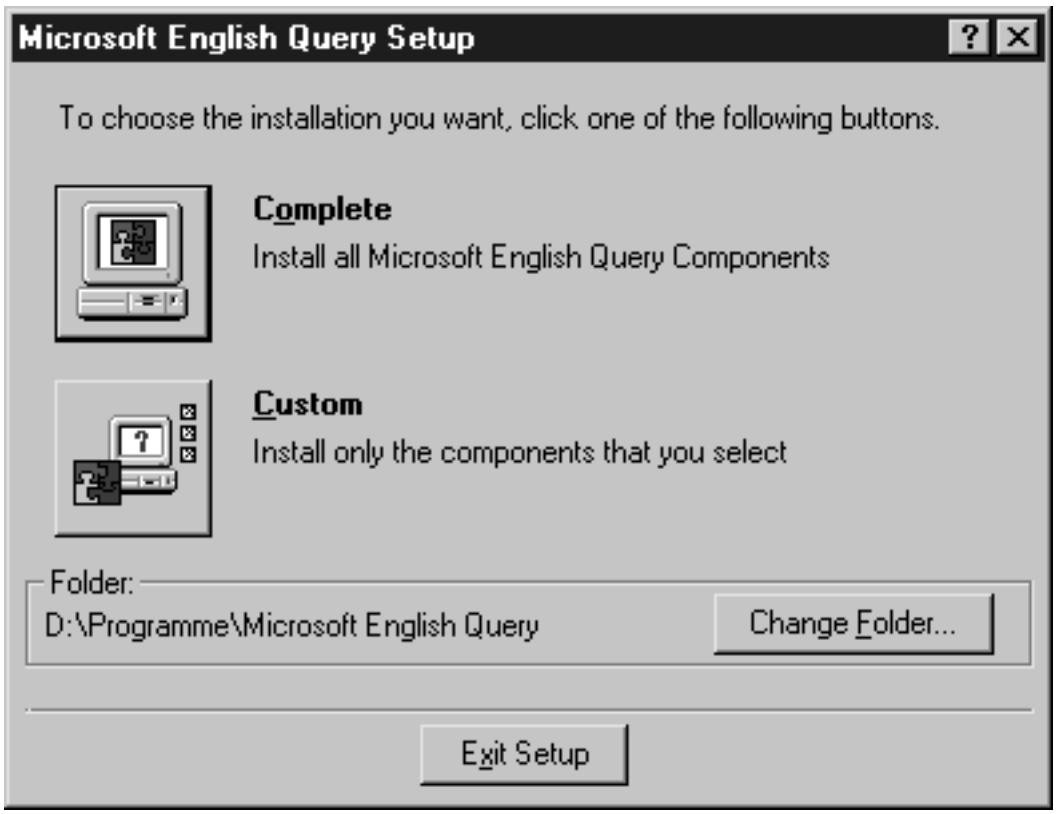


abbildung 2.20: auswahl der installationsart

7. bei der benutzerdefinierten installation wählen sie im nächsten dialogfeld die zu installierenden komponenten (siehe abbildung 2.21):
   - *english query core*: kernkomponenten von english query
   - *english query developer*: werkzeug für die anwendungsentwicklung mit english query
   - *english query help*: online-hilfe für english query
   - *english query samples*: beispielanwendungen mit english query

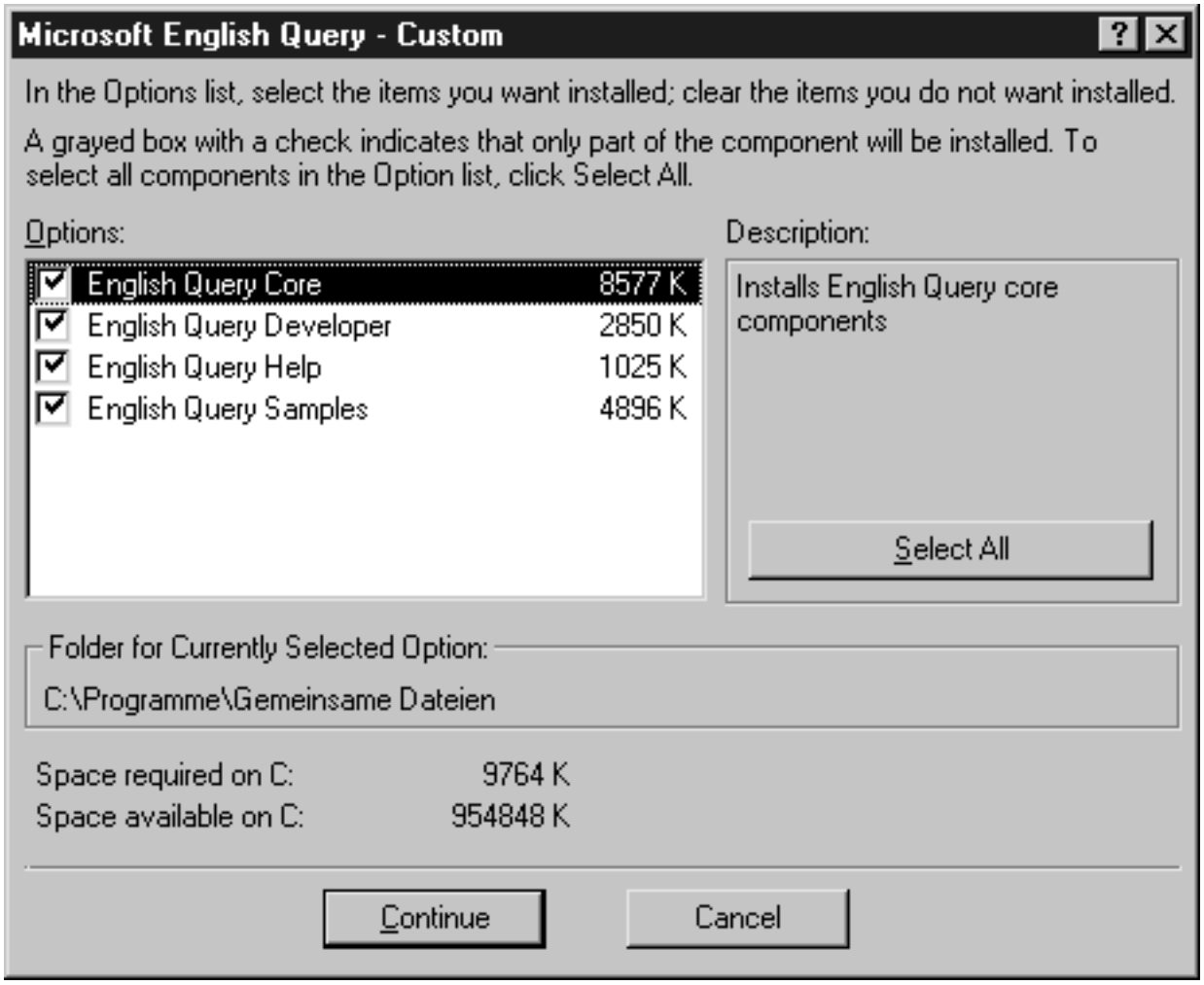


abbildung 2.21: auswahl der zu installierenden komponenten

8. klicken sie auf **continue**, um die ausgewählten komponenten zu installieren. im verlauf der installation erscheint ein meldungsfeld mit der frage, ob sie eine vorhandene hilfedatei ersetzen möchten (siehe abbildung 2.22). wenn sie die hilfe lieber in deutsch beibehalten möchten, klicken sie auf **no**. das nächste dialogfeld fragt dann, ob sie alle dateien unverändert übernehmen wollen (siehe abbildung 2.23). hier ist es egal, ob sie **yes** oder **no** wählen, da english query keine derartigen fragen mehr stellt.

die versionsfrage bezieht sich nur auf den rahmen, in dem die hilfe erscheint - beispielsweise die beschriftung der registerkarten mit **index**, **suchen** usw. die hilfe zu english query selbst ist unabhängig davon trotzdem nur in englisch verfügbar.

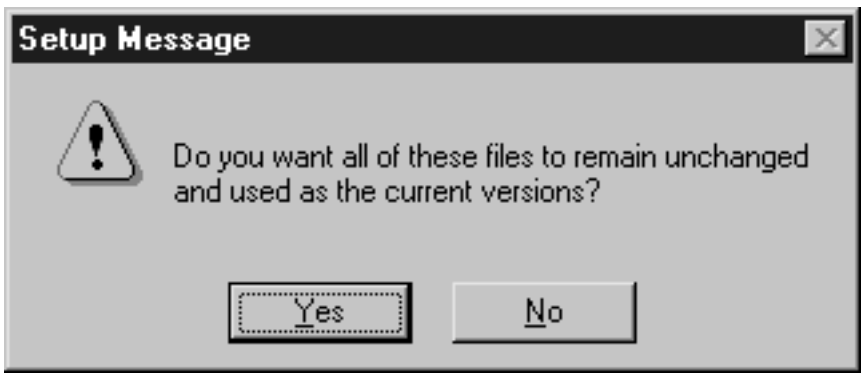


abbildung 2.22: klicken sie auf »yes«, um alle deutschen dateien beizubehalten

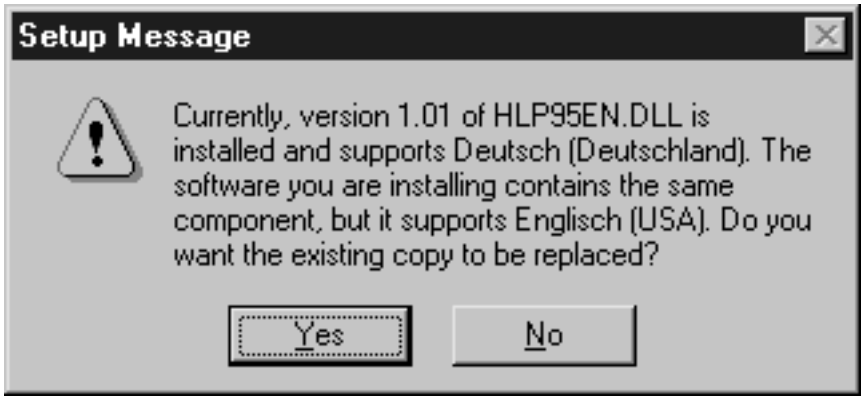


abbildung 2.23: klicken sie auf »no«, um die deutsche version der hilfe beizubehalten

normalerweise ist der fall genau umgekehrt. bei der installation eines deutschen (microsoft-)programms wird zunächst eine englische version auf die platte gebracht, die das setup-programm später durch die lokalisierte version ersetzt. gegebenenfalls erscheint ebenfalls eine derartige frage wie in abbildung 2.22. dann müssen sie natürlich auf **ja** (bzw. **yes**) klicken, um die englische durch die deutsche datei zu ersetzen.

9. schließlich erscheint die erfolgsmeldung, daß english query erfolgreich installiert wurde (siehe abbildung 2.24). klicken sie auf **ok** und im startbildschirm der sql-server-installation auf **beenden**, um die installation abzuschließen. wie immer zu empfehlen: computer neu starten, auch wenn sie das programm nicht ausdrücklich dazu auffordert.

abbildung 2.24: erfolgreicher abschluß der installation

2.6 server registrieren

bevor sie mit einem server arbeiten können, müssen sie ihn zunächst registrieren. bei der installation von sql server geschieht das automatisch für den computer, auf dem sie sql server installieren.

führen sie folgende schritte aus, um einen neuen server zu registrieren:

1. starten sie den enterprise manager über **start / programme / microsoft sql server 7.0 / enterprise manager**. erweitern sie im enterprise manager die strukturansicht bis zu *sql server-gruppe*, und

klicken sie mit der rechten maustaste auf diesen eintrag.

2. im kontextmenü wählen sie den befehl **neue sql server-registrierung**. daraufhin wird der sql-server-registrierungs-assistent gestartet. klicken sie im startdialogfeld auf **weiter**.
3. im zweiten dialogfeld des registrierungs-assistenten sind die verfügbaren (noch nicht registrierten) server aufgelistet. markieren sie den zu registrierenden server (sie können auch mehrere auf einmal auswählen und registrieren), und klicken sie auf die schaltfläche **hinzufügen** (siehe abbildung 2.25). gegebenenfalls können sie eine auswahl zurücknehmen, indem sie den/die server in der rechten liste markieren und auf **entfernen** klikken. sollte in der liste **verfügbare server** kein server aufgeführt sein, können sie den namen direkt in das feld **verfügbare server** eingeben.

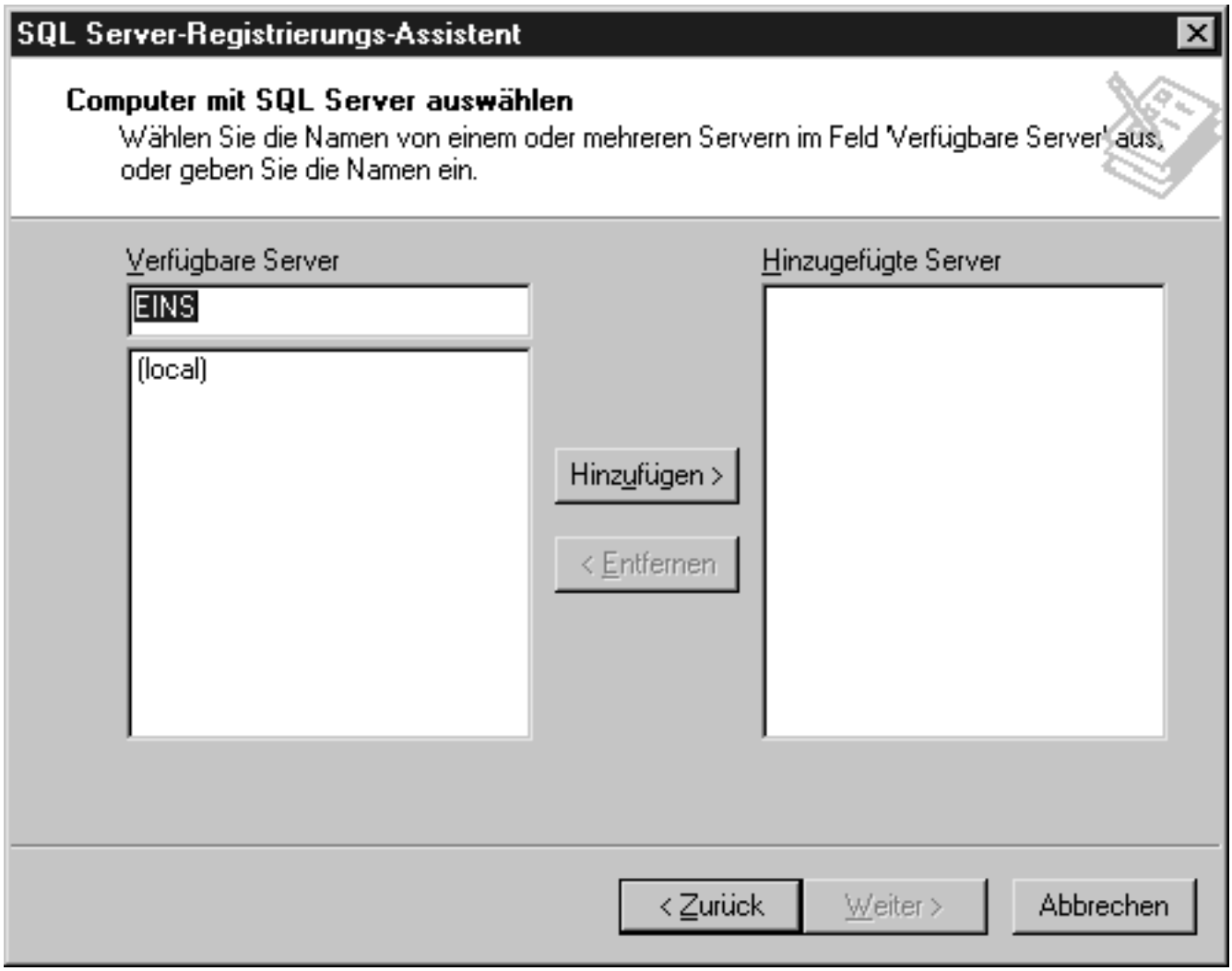


abbildung 2.25: im zweiten dialogfeld des registrierungs-assistenten wählen sie die zu registrierenden server aus

4. klicken sie auf **weiter**. im dritten dialogfeld wählen sie den authentifizierungsmodus für die anmeldung (siehe abbildung 2.26). kapitel 21 geht näher auf die authentifizierung ein.

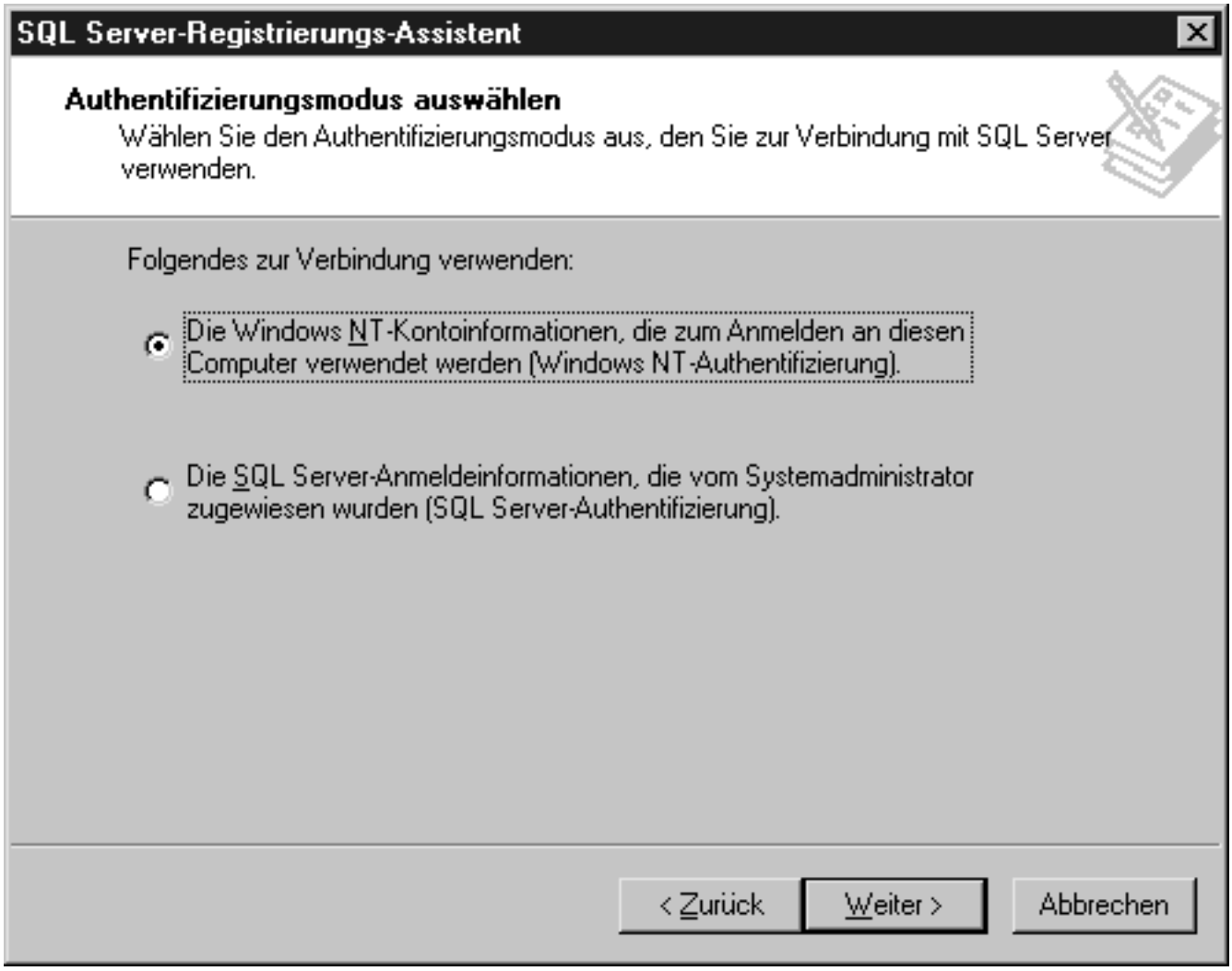


abbildung 2.26: authentifizierungsmodus für die anmeldung bei sql server auswählen

5. wenn sie die option *sql server-authentifizierung* gewählt haben, gelangen sie zunächst zum dialogfeld **verbindungsoptionen auswählen** (siehe abbildung 2.27). hier geben sie den benutzernamen ein, unter dem sie sich bei sql server anmelden. nach der installation von sql server ist der benutzername sa ohne kennwort eingestellt.

bei der option *automatisch anmelden* speichert sql server den benutzernamen und das kennwort in der registrierung. andernfalls muß der benutzer bei jeder anmeldung den benutzernamen und das kennwort eingeben.

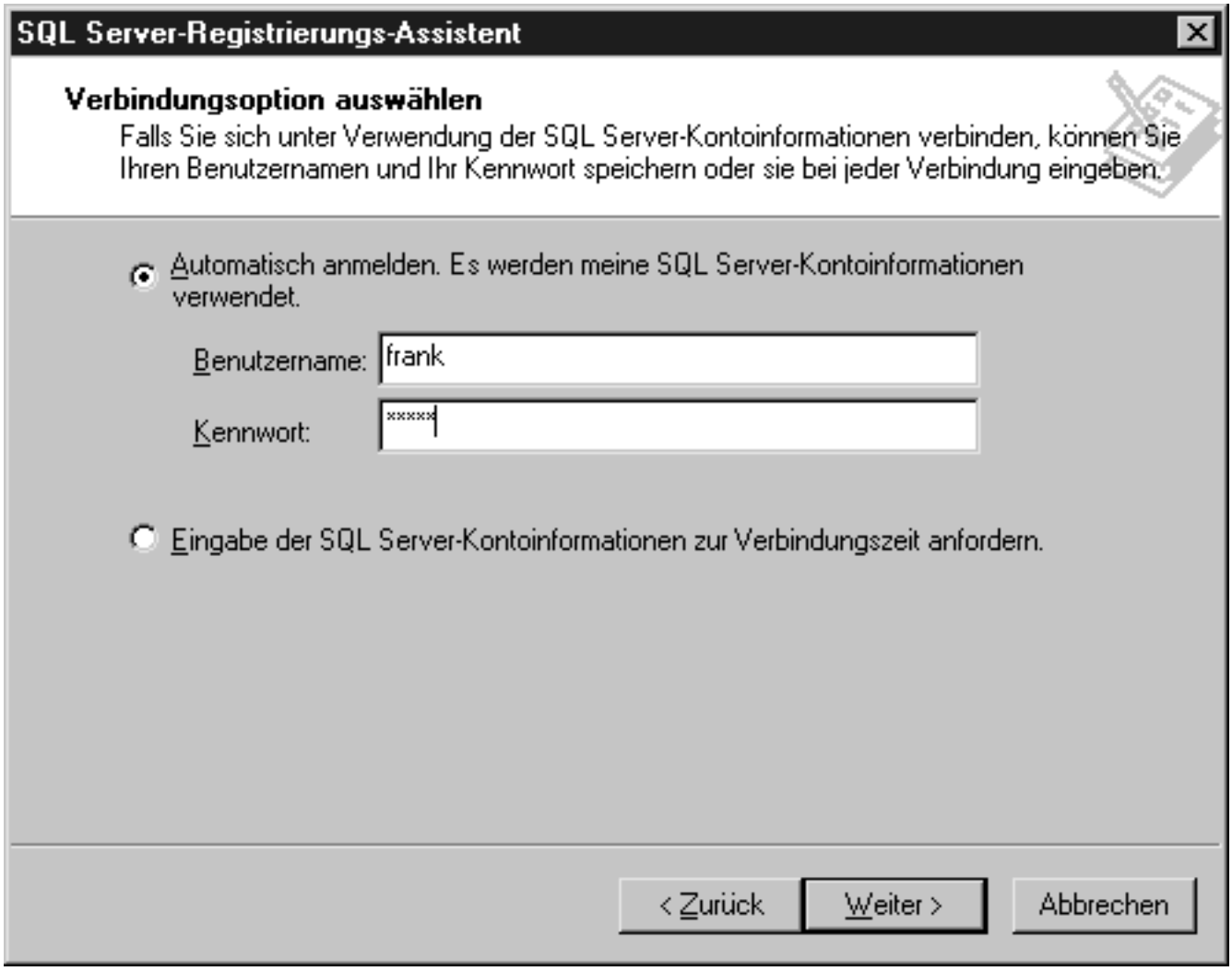


abbildung 2.27: bei sql-server-authentifizierung müssen sie die verbindungsoptionen festlegen

6. falls sie im dritten dialogfeld *windows-nt-authentifizierung* gewählt haben, gelangen sie direkt, andernfalls über den zwischenschritt **verbindungsoptionen auswählen** zum dialogfeld **sql server-gruppe auswählen**. hier können sie den server einer vorhandenen server-gruppe zuordnen (sql-server-gruppe in der voreinstellung) oder eine neue gruppe unterhalb von microsoft sql server anlegen (siehe abbildung 2.28).

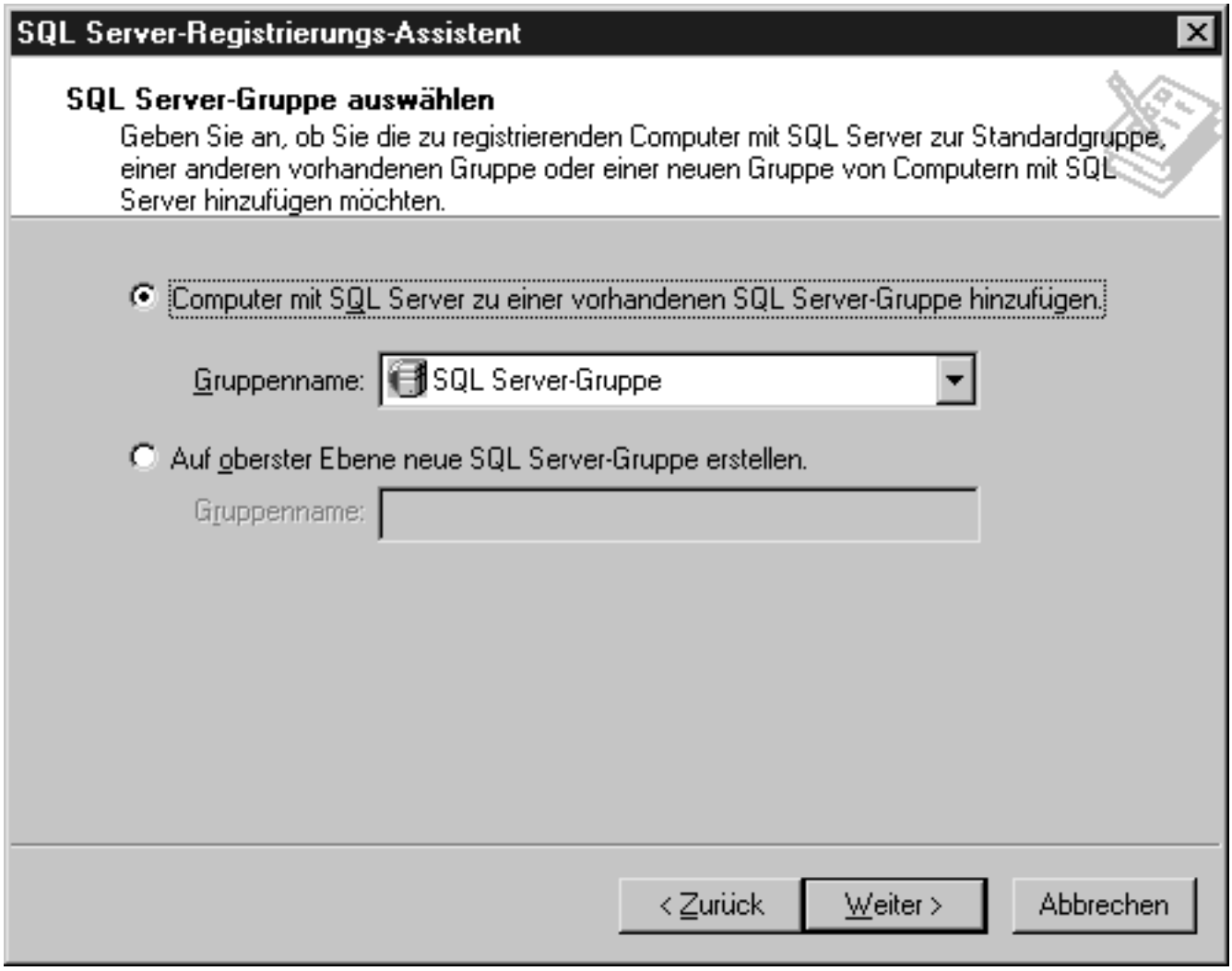


abbildung 2.28: das dialogfeld sql server-gruppe auswählen

7. klicken sie im letzten dialogfeld des sql-server-registrierungs-assistenten (siehe abbildung 2.29) auf **fertigstellen**.

abbildung 2.29: das abschließende dialogfeld des sql-server-registrierungs-assistenten

8. nach kurzer zeit erscheint die meldung, daß die registrierung erfolgreich verlaufen oder gescheitert ist (siehe abbildung 2.30).

abbildung 2.30: abschlußmeldung des sql-server-registrierungs-assistenten

9. wenn die registrierung gescheitert ist, klicken sie im meldungsfeld auf die schaltfläche **eigenschaften**, um das dialogfeld **registrierter sql server - eigenschaften** zu öffnen. hier können sie die einstellungen noch einmal prüfen und gegebenenfalls ändern.

gegenüber den vorgängerversionen ist es nicht mehr erforderlich, den für sql server vorgesehenen speicher zu konfigurieren. der speicherbedarf wird automatisch je nach anforderung angepaßt.

2.7 client-komponenten installieren

wenn sie unter windows nt server/workstation 4.0/5.0 oder windows 95/98 arbeiten, können sie auf einem client-computer die folgenden tools installieren und eine verbindung zu sql server herstellen:

- sql server query analyzer
- sql server enterprise manager
- sql server client-konfiguration
- sql-server-aktualisierungs-assistent
- sql-server-profiler
- ms dtc
- sql server web-assistent
- odbc-treiber
- bcp

2.8 sql-server-client-konfigurationsprogramm

mit den installierten client-komponenten sollten sie ohne weitere schritte eine verbindung zu sql server herstellen können. bei abweichenden protokollen müssen sie allerdings den client entsprechend konfigurieren. rufen sie dazu über **start / programme / microsoft sql server 7.0** das programm **sql server-clientkonfiguration** (siehe abbildung 2.31) auf.

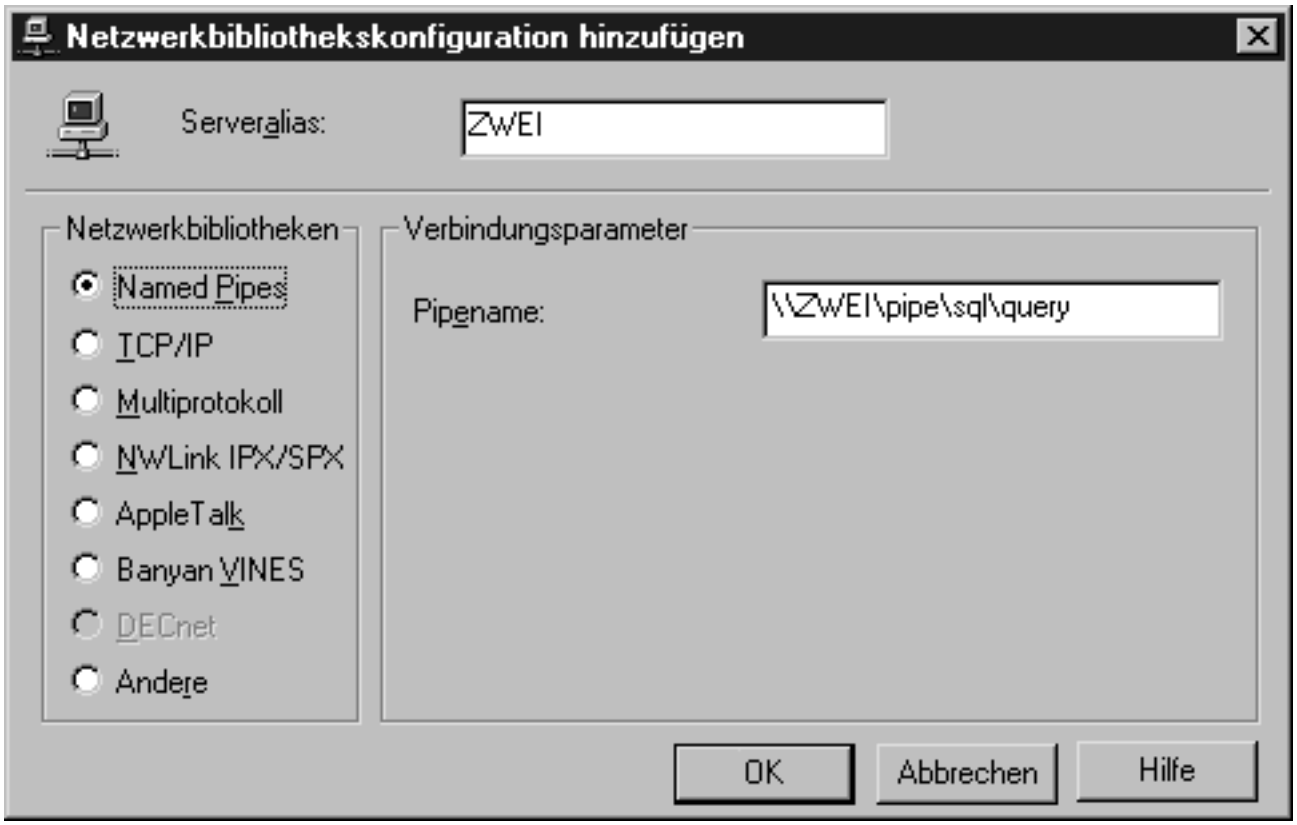


abbildung 2.31: auswahl der netzwerkbibliothek

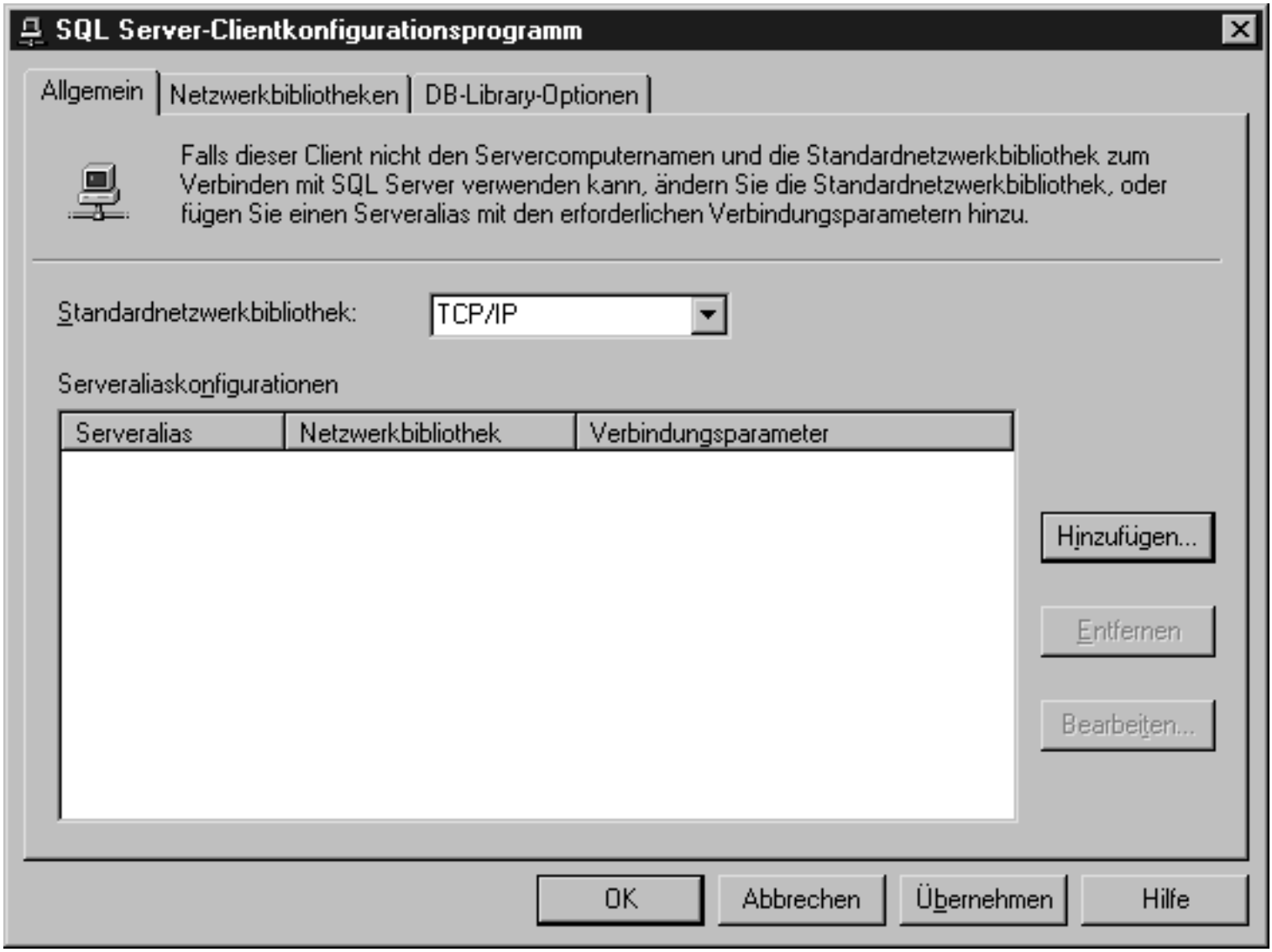


abbildung 2.32: auf der registerkarte allgemein können sie die standardnetzwerkbibliothek ändern

um eine neue client-konfiguration zu erstellen, klicken sie auf die schaltfläche **hinzufügen**. geben sie dann im dialogfeld **netzwerkbibliothekskonfiguration hinzufügen** den server-namen in das textfeld **serveralias** ein (siehe abbildung 2.32).

klicken sie auf **ok**, um den neuen server hinzuzufügen.

2.8.1 named pipe testen

named pipe ist eine art direkter leitung zwischen server und client. sql server und open data services greifen auf diesen mechanismus zurück, um die kommunikation zwischen clients und servern zu ermöglichen. für die installation verwendet sql server immer named pipes.

um das named-pipe-protokoll zu testen, brauchen sie einen server und einen client (ersatzweise auch ein zweites fenster mit der ms-dos-eingabeaufforderung auf dem server). sql server stellt die dienstprogramme makepipe und readpipe bereit. zum testen führen sie die folgenden schritte aus:

1. öffnen sie auf dem server über **start / programme / eingabeaufforderung** ein ms-dos-fenster.

2. geben sie an der eingabeaufforderung den befehl makepipe ein. daraufhin erscheint folgende meldung:

```
making pipe:\\.\pipe\abc
read to write delay (seconds):0
waiting for client to connect...
```

(zu deutsch: pipe... wird erstellt. lese-schreib-verzögerung in sekunden: 0. warten auf client-verbindung...)

der server wartet nun darauf, daß ein client über named pipe eine verbindung herstellt.

3. öffnen sie ein ms-dos-fenster auf dem client (oder ein zweites ms-dos-fenster auf dem server), und geben sie an der eingabeaufforderung einen befehl nach folgendem muster ein:

```
readpipe /sserver /dtestzeichenfolge
```

in der beispielkonfiguration heißt der windows-nt-server eins. wenn sie eine testzeichenfolge angeben, die leerzeichen enthält, schreiben sie sie in anführungszeichen. geben sie nun folgenden befehl auf dem client (oder im zweiten ms-dos-fenster des servers) ein:

```
readpipe /seins /d"das ist ein test"
```

auf dem server erscheint daraufhin folgende meldung:

```
waiting for client to sent...  1
data read:
das ist ein test
waiting for client to sent...  2
pipe closed
waiting for client to connect...
```

der server hat also das verbindungsgesuch erkannt, wartet daraufhin auf eine client-sendung (waiting for client to sent) und zeigt dann die gelesenen daten an. da keine weitere sendung folgt, schließt der server die named pipe (pipe closed) und wartet erneut auf eine verbindung.

auf dem client sind folgende meldungen zu sehen:

```
svrname:\\eins
pipe   :\\eins\pipe\abc
data   :das ist ein test
data sent: 1 :das ist ein test
```

```
data read: 1 :das ist ein test
```

in der ersten zeile gibt der client den servernamen aus, in der zweiten zeile die verwendete named pipe (abc ist die voreinstellung), die dritte zeile zeigt die eingegebene testzeichenfolge. in der vierten zeile (data sent) erscheint die an den server gesendete, in der fünften zeile (data read) die als echo zurückerhaltene zeichenfolge.

4. die named pipe auf dem server können sie mit **ç**+**c**, **ç**+**£** oder durch schließen des ms-dos-fensters schließen. im ms-dos-fenster des clients (bzw. im zweiten fenster des servers) brauchen sie keine weiteren schritte zu unternehmen und können bei bedarf unmittelbar in diesen fenstern weiterarbeiten.

die dienstprogramme makepipe und readpipe bieten noch ein paar konfigurationsoptionen. da diese weniger wichtig sind, verweisen wir hier auf die online-dokumentation von sql server.

2.9 mit einem server verbinden

wenn sie osql starten, um mit datenbanken auf dem server zu arbeiten, muß der dienst mssqlserver auf dem server laufen. auf dem client, von dem aus der start von osql erfolgt, ist das nicht erforderlich. ist zum beispiel auf \\zwei eine desktop-version von sql server installiert und sie führen hier den befehl

```
osql -usa -p -seins
```

aus, um eine verbindung zum server \\eins, der die datenbankkomponenten enthält, herzustellen, muß der dienst mssqlserver auf \\eins laufen, während das auf \\zwei nicht erforderlich ist (aber auch nichts schadet).

2.10 sql server starten, anhalten und beenden

dieser abschnitt zeigt, welche schritte erforderlich sind, um sich bei microsoft sql server anzumelden, und wie man sql server startet, anhält und stoppt. nach der anmeldung kann der benutzer sowohl administrative aufgaben wahrnehmen als auch datenbanken abfragen.

2.10.1 windows nt dienste

unter windows nt wird sql server als dienst installiert und läßt sich damit als windows-nt-dienst (mssqlserver) entweder lokal oder entfernt starten. um die dienste von sql server zu starten, anzuhalten oder zu beenden und die startart der dienste festzulegen, rufen sie den windows-nt-dienst-manager über **start / einstellungen / systemsteuerung / dienste** auf. es erscheint das dialogfeld **dienste** (siehe abbildung 2.33), in dem sie die genannten einstellungen vornehmen können.

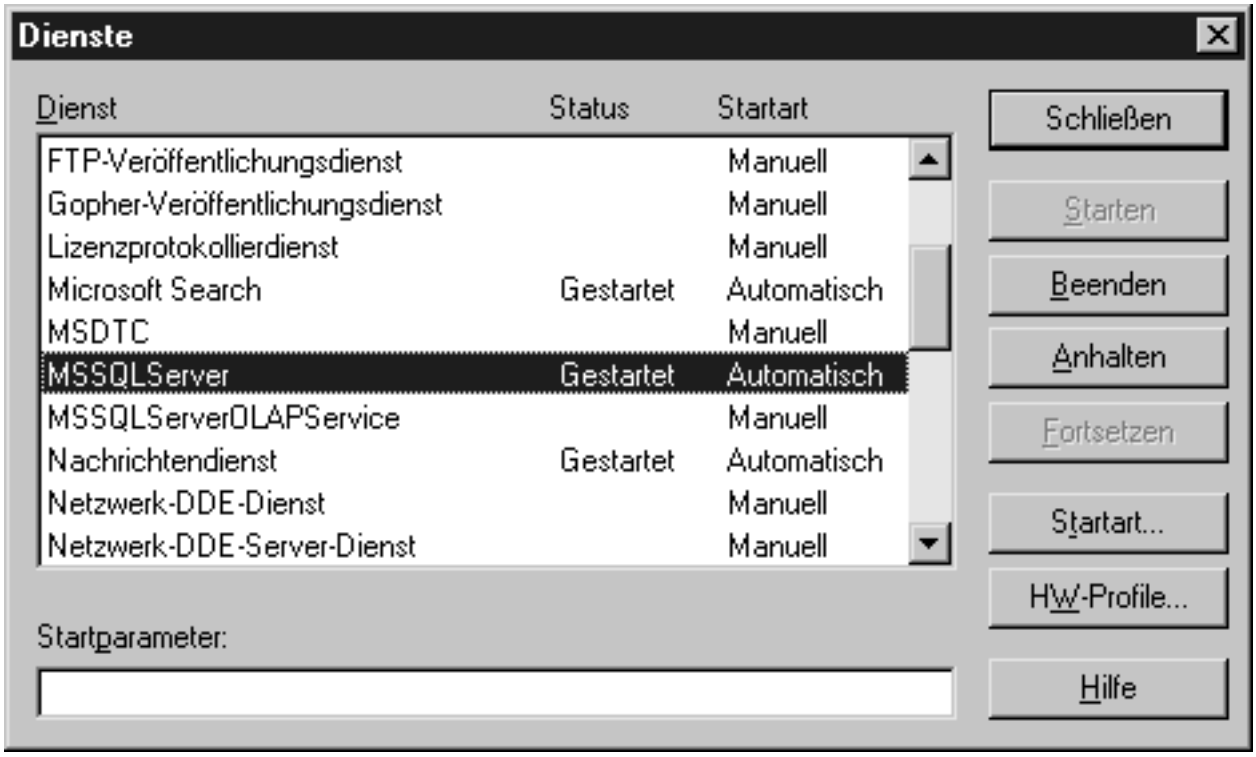


abbildung 2.33: das dialogfeld dienste der windows-nt-systemsteuerung

im dialogfeld **dienste** markieren sie den eintrag *mssqlserver* und klicken dann auf die schaltfläche **startart**. es erscheint das dialogfeld **dienst** (siehe abbildung 2.34). hier legen sie die einstellungen für den jeweiligen dienst - im beispiel mssqlserver - fest.

abbildung 2.34: im dialogfeld dienst legen sie die startart für den jeweiligen dienst fest

die durch windows nt definierten startarten erläutert tabelle 2.5.

startart	bedeutung
automatisch	der dienst startet automatisch, wenn das system gestartet wird.
manuell	ermöglicht das starten eines dienstes durch den benutzer oder einen abhängigen dienst.
deaktiviert	verhindert das starten eines dienstes durch einen benutzer oder einen abhängigen dienst.

tabelle 2.5: die startarten für dienste unter windows nt

wenn der computer als datenbankserver vorgesehen ist, empfiehlt sich die option *automatisch*.

im abschnitt **anmelden als** wählen sie das konto aus, unter dem sie sich bei sql server anmelden möchten. das systemkonto ist das integrierte lokale konto des systemadministrators. dieses konto wählen sie, wenn ihre aufträge ausschließlich ressourcen des lokalen systems benötigen. mit der option *dieses konto* können sie ein domänenkonto von windows nt angeben, in dem der dienst sqlserveragent ausgeführt werden soll. das domänenkonto muß zur rolle sysadmin des lokalen computers mit sql server gehören.

2.10.2 dienst-manager

windows 95/98 unterstützt keine windows-nt-dienste. um einen vergleichbaren mechanismus bereitzustellen, simuliert sql server die dienste mssqlserver und sqlserveragent.

mit dem dienst-manager (siehe abbildung 2.35) können sie die dienste msdtc, mssqlserver und sqlserveragent (falls installiert auch microsoft search zur volltextsuche) für die vorhandenen server starten/fortsetzen, anhalten und beenden.

wenn sie unter windows nt die desktop-version von sql server installiert haben, können sie die dienste von sql server sowohl über das dialogfeld **dienste** der systemsteuerung als auch über den dienst-manager starten/fortsetzen, anhalten und beenden. beide mechanismen laufen synchron.

abbildung 2.35: der sql-server-dienst-manager

läuft sql server unter windows 95/98, sollten sie das kontrollkästchen **dienst bei betriebssystemstart automatisch starten** einschalten, da es hier keine dienste gibt, die sich für den automatischen start konfigurieren lassen.

nachdem sie in der drop-down-liste *server* einen anderen server ausgewählt oder eingegeben haben, klicken sie auf die drop-down-liste *dienste*. der dienst-manager versucht dann, die verbindung zum angegebenen server herzustellen.

anhalten

die funktion *anhalten* im sql-server-dienst-manager bewirkt folgendes:

- sql server läuft weiter.
- es können sich keine neuen benutzer bei sql server anmelden.

wenn sie zum beispiel wartungsarbeiten ausführen wollen, die ein herunterfahren von sql server erfordern, halten sie sql server an und schicken allen benutzern, die noch mit sql server verbunden sind, eine nachricht. diese benutzer können dann ihre arbeiten abschließen.

automatischer oder manueller start

der sql-server-dienst-manager wird entweder automatisch oder manuell gestartet.

2.10.3 sql server von der befehlszeile starten

wenn sie sql server nicht auf normalem weg - sprich über den dienst-manager von windows nt oder den sql-server-dienst-manager - starten können, probieren sie den start von der befehlszeile. dafür stellt sql server das programm sqlservr bereit. die syntax lautet:

```
sqlservr <argumente>
```

tabelle 2.6 zeigt die wichtigsten argumente.

argument	beschreibung
`-c`	startet sql server unabhängig vom dienstkontroll-manager von windows nt und verkürzt damit die zeit für das starten von sql server. allerdings läßt sich sql server bei dieser option nicht mit herkömmlichen mitteln (sql-server-dienst-manager) beenden, sondern nur durch herunterfahren von windows nt oder mit **ç**+**c** im dos-fenster, von dem aus sql server gestartet wurde.
-d*pfad_master_ datenbank*	spezifiziert pfad und dateiname für die master-datenbankdatei
`-f`	startet sql server mit einer minimalkonfiguration
-e*pfad_fehlerprotokolldatei*	spezifiziert pfad und dateiname der fehlerprotokolldatei
-l*pfad_master_ protokolldatei*	spezifiziert pfad und dateiname für die transaktionsprotokolldatei der master-datenbank
`-m`	startet sql server im einzelbenutzermodus
-t*ablaufverfolgungsflag*	startet sql server mit einem ablaufverfolgungsflag (siehe kapitel 25)

tabelle 2.6: argumente des dienstprogramms sqlservr

die folgende anweisung startet sql server von der befehlszeile mit einer minimalkonfiguration, unabhängig vom dienstkontroll-manager und unter angabe der verschiedenen pfad- und dateinamen:

```
sqlservr -c -dd:\mssql7\data\master.mdf
-ld:\mssql7\data\master.ldf -ed:\mssql7\log\errorlog -f
```

2.11 sql server testen

auf dem server starten sie sql server an der eingabeaufforderung mit dem befehl:

```
net start mssqlserver
```

der befehl

```
osql /usa /p
```

stellt eine verbindung zu sql server her. achten sie auf die groß-/kleinschreibung der parameter. /u bezeichnet die login-id des benutzers. per vorgabe ist sa für systemadministrator festgelegt. anfänglich ist noch kein kennwort zugewiesen. mit /p gibt man das kennwort ein, in diesem fall das standardkennwort null. sobald sql server von der administrativen seite her eingerichtet ist, sind die parameter entsprechend einzugeben.

2.11.1 odbcping

mit dem dienstprogramm odbcping testet man die integrität einer odbc-datenquelle oder die fähigkeit eines clients, eine verbindung zu einem server herzustellen.

zum beispiel testet der befehl

```
odbcping -usa -p -seins
```

ob sich eine verbindung zum server eins mit dem benutzernamen sa und dem standardkennwort (null) herstellen läßt. man erhält entweder eine fehlermeldung, daß der angegebene sql server nicht gefunden wurde, oder - im erfolgsfall - zum beispiel folgende ausgabe:

```
connected to sql server
odbc sql server driver version: 03.70.0623
sql server version: microsoft sql server   7.00 - 7.00.623
(intel x86)
        nov 27 1998 22:20:07
        copyright (c) 1998-1998 microsoft corporation
        standard edition on windows nt 4.0 (build 1381: service
pack 4)
```

2.12 technische daten

tabelle 2.7 stellt die in zahlen auszudrückenden technischen spezifikationen von sql server 6.5 und sql server 7.0 gegenüber.

objekt	sql server 6.5	sql server 7.0
stapelgröße	128 k	65536 * netzwerkpaketgröße

bytes pro zeichen- oder binärspalte	255	8000
bytes pro spalte der datentypen text, ntext oder image	2 gbyte - 2	2 gbyte - 2
bytes pro group by, order by	900	8060
bytes pro index	900	900
bytes pro fremdschlüssel	900	900
bytes pro primärschlüssel	900	900
bytes pro zeile	1962	8060
bytes im quelltext einer gespeicherten prozedur	65025	stapelgröße oder 250 mbyte (niedrigerer wert)
gruppierte indizes oder beschränkungen pro tabelle	1	1
spalten in group by, order by	16	nur durch zahl der bytes beschränkt
spalten oder ausdrücke in einer group by with cube- oder with rollup-anweisung	10	10
spalten pro index	16	16
spalten pro fremdschlüssel	16	16
spalten pro primärschlüssel	16	16
spalten pro basistabelle	250	1024
spalten pro select-anweisung	4096	4096
spalten pro insert-anweisung	250	1024
verbindungen je client	maximalwert der konfigurierten verbindungen	maximalwert der konfigurierten verbindungen
datenbankgröße	1 tbyte	1,048,516 tbyte
daten pro datenbank	32	32767
dateigröße (daten)	32 gbyte	32 tbyte
dateigröße (protokoll)	32 gbyte	4 tbyte
references pro tabelle	31	63

verweise auf fremdschlüssel pro tabelle	16	253
länge von bezeichnern (in zeichen)	30	128
größe von indexschlüsseln (in bytes)	900	900
sperren pro verbindung	maximalwert der konfigurierten sperren	maximalwert der konfigurierten sperren
verschachtelte unterabfragen	16	64
verschachtelte trigger-ebenen	16	32
nicht gruppierte indizes oder beschränkungen pro tabelle	249	249
objekte in einer datenbank (tabellen, sichten, gespeicherte prozeduren, erweiterte gespeicherte prozeduren, trigger, regeln, standardwerte und einschränkungen)	2 milliarden	2,147,483,647
parameter pro gespeicherter prozedur	255	1024
zeilen pro tabelle	durch verfügbaren speicher begrenzt	durch verfügbaren speicher begrenzt
sql-stringlänge	128 kbyte	128 * tds-paketgröße
tabellen pro datenbank	2 milliarden	beschränkt durch die anzahl der objekte in einer datenbank
tabellen pro select-anweisung	16	256
trigger pro tabelle	3	beschränkt durch die anzahl der objekte in einer datenbank
unique-indizes oder -einschränkungen pro tabelle	249	249 nicht gruppierte und 1 gruppierter

tabelle 2.7: maximale größen und zahlen von sql server 6.5 und 7.0 im überblick

2.13 sql server deinstallieren

für eine deinstallation von sql server gibt es die unterschiedlichsten gründe. vielleicht haben sie sql server nur vorübergehend installiert, um sich einzuarbeiten. oder sie wollen einfach nur platz schaffen. es kann aber auch sein, daß sie von der deinstallationsoption eher gebrauch machen müssen, als ihnen lieb ist. wenn sie das kennwort für den systemadministrator geändert haben - was sie auf jeden fall im anschluß an die installation tun sollten - und sich partout nicht mehr daran erinnern können, bleibt ihnen nichts weiter übrig, als sql server zu deinstallieren und erneut zu installieren.

wie bei jedem programm, das man unter windows 95/98 installiert und das sich an die spielregeln von microsoft hält, läßt es sich entweder über die systemsteuerung oder einen speziellen menübefehl deinstallieren. sql server erlaubt beide möglichkeiten. auf keinen fall sollte man einfach das installationsverzeichnis von sql server löschen. die registrierung wird dann nicht auf den neuesten stand gebracht, und letztendlich verschlechtert sich die systemleistung.

schließen sie vor der deinstallation alle anwendungen. das betrifft auch die ereignisanzeige von windows nt, den registrierungs-editor, alle sql-server-anwendungen und alle von sql server abhängigen anwendungen.

über **start / einstellungen / systemsteuerung / software** öffnen sie das dialogfeld **eigenschaften von software**, gehen auf die registerkarte **installieren/deinstallieren**, markieren den eintrag **microsoft sql server 7.0** und klicken auf **hinzufügen/entfernen**. es erscheint das dialogfeld **löschen einer datei bestätigen** (siehe abbildung 2.36).

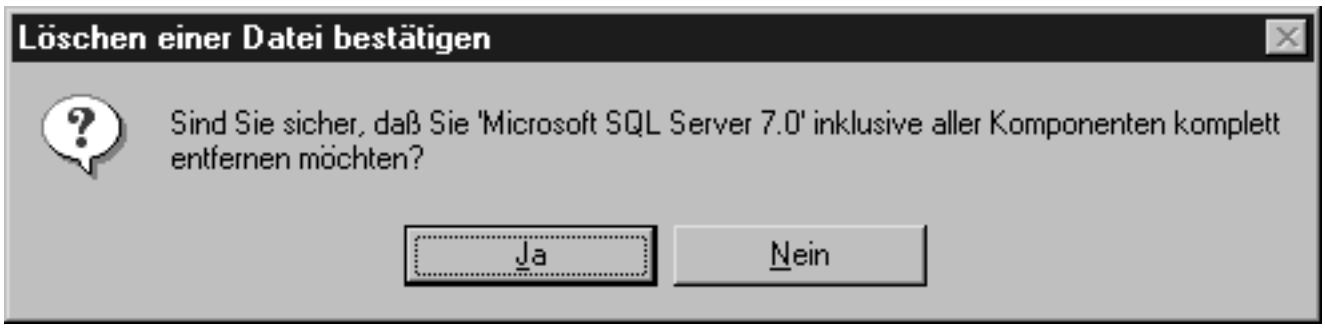


abbildung 2.36: klicken sie auf »ja«, um sql server 7.0 zu deinstallieren

zu diesem dialogfeld gelangen sie auch über **start / programme / microsoft sql server 7.0 / sql server 7.0 deinstallieren**.

wenn sie mit **ja** bestätigen, beginnt die deinstallationsroutine von sql server mit der deinstallation und zeigt den fortschritt an.

falls sie benutzerdatenbanken im verzeichnis \mssql7 bzw. \mssql7\data\ angelegt haben, müssen sie dieses verzeichnis nach der deinstallation manuell löschen.

nachdem die deinstallation abgeschlossen ist, erscheint ein dialogfeld, das ihnen den neustart des computers nahelegt. klicken sie auf **ok**, und starten sie danach den computer neu.

2.14 unbeaufsichtigte installation / deinstallation

2.14.1 installieren

die unbeaufsichtigte installation von sql server bietet den vorteil, daß sie beim ablauf des setup-programms keine weiteren informationen eingeben müssen. wenn ihnen die standardeinstellungen genügen, brauchen sie nicht einmal eine eigene initialisierungsdatei zu erstellen. im stammverzeichnis der sql-server-cd befinden sich die in tabelle 2.8 aufgeführten installationsdateien.

stapeldatei	initialisierungsdatei	beschreibung
sql70cli.bat	sql70cli.iss	installiert die verwaltungstools von sql server
sql70ins.bat	sql70ins.iss	führt eine standardinstallation von sql server durch
sql70cst.bat	sql70cst.iss	führt eine benutzerdefinierte installation von sql server durch

tabelle 2.8: installationsdateien

wenn sie die initialisierungsdateien von der cd-rom verwenden, werden die sql-server-dienste dem lokalen systemkonto zugewiesen. wollen sie während der unbeaufsichtigten installation ein domänenbenutzerkonto zuweisen oder einstellungen wie zeichensatz und sortierreihenfolge ändern, müssen sie eine eigene initialisierungsdatei für das setup-programm erstellen. als basis können sie dabei die initialisierungsdateien der cd-rom verwenden.

weitere informationen zur unbeaufsichtigten installation entnehmen sie bitte der online-dokumentation. das format der initialisierungsdatei und weitere einzelheiten zum setup finden sie unter *installieren von sql server / initialisierungsdatei*.

2.14.2 deinstallieren

im stammverzeichnis der sql-server-cd befindet sich die datei sql70rem.bat. beim aufruf dieser stapeldatei von der befehlszeile ist als parameter der installationspfad anzugeben. die im buch angegebenen beispiele wurden mit einer sql-server-installation im verzeichnis d:\mssql7 erstellt. die unbeaufsichtigte deinstallation rufen sie demnach wie folgt auf:

```
cd:\>sql70rem d:\mssql7
```

hierbei steht cd für den laufwerksbuchstaben des cd-laufwerks.

bevor sie sql server auf diese weise deinstallieren können, müssen sie sämtliche dienste von sql server und auch den dienst-manager beenden.


Microsoft
SQL Server
Installieren der Komponenten von SQL Server 7.0.
Release Notes lesen
SQL Server 7.0-Voraussetzungen installieren (P)
SQL Server 7.0-Komponenten installieren (C)
Unsere Website besuchen (V)
Onlinedokumentation durchsuchen (B)
(Erfordert Internet Explorer 4.01 SP 1)
Beenden (X)

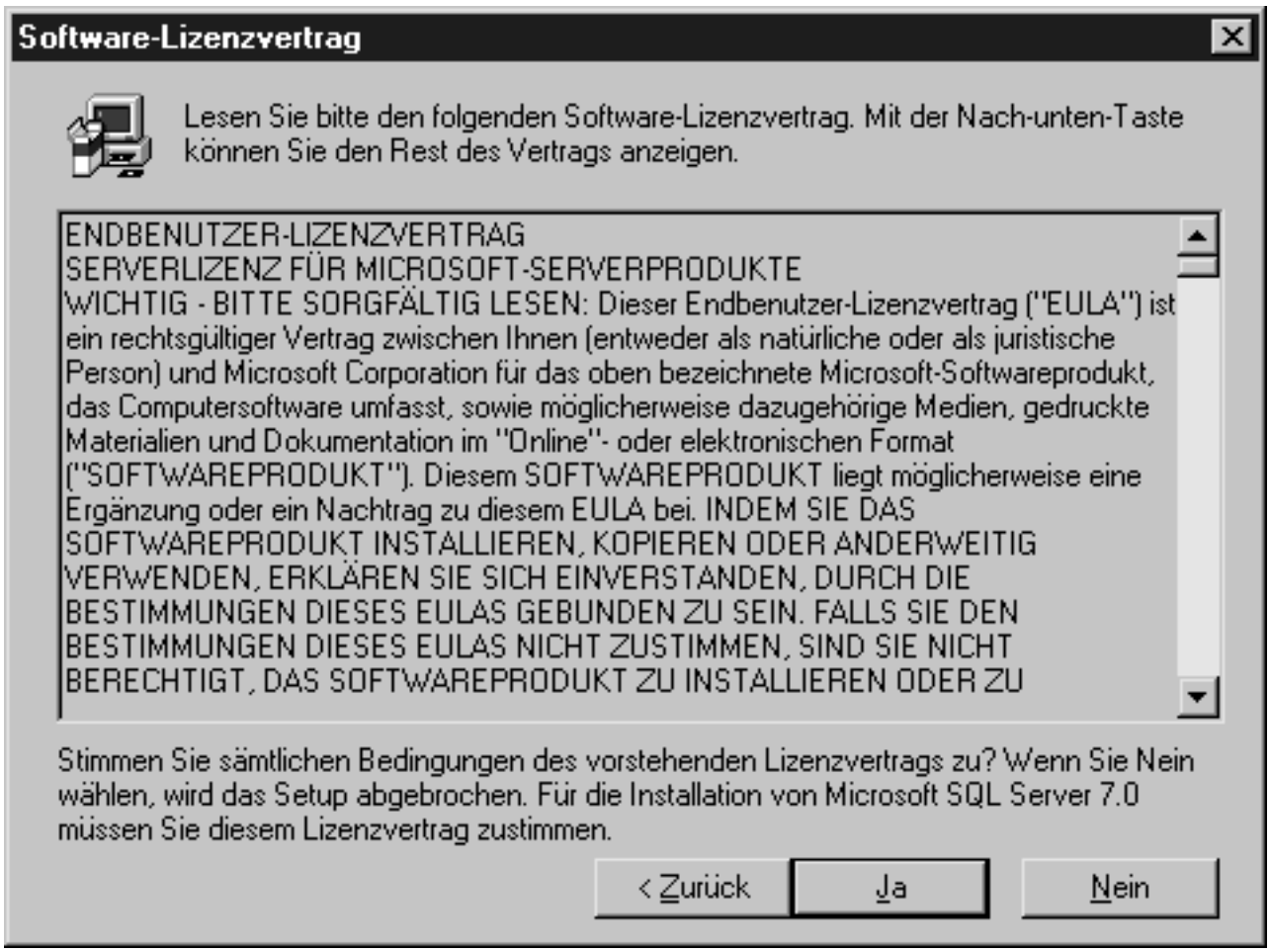

Microsoft
SQLServer
SQL Server 7.0-Voraussetzungen
Windows 95
DCOM95
(Erforderlich, wenn Internet Explorer 4.01 SP1 nicht verwendet wird)
Internet Explorer 4.01 SP 1
(Nicht erforderlich für Installationen vom Typ Minimal, Connectivity oder Nur Server)
Minimalinstallation
Setup-Assistent starten (L)
Jahr-2000-Aktualisierung (Y)
Zurück (B) Beenden (X)

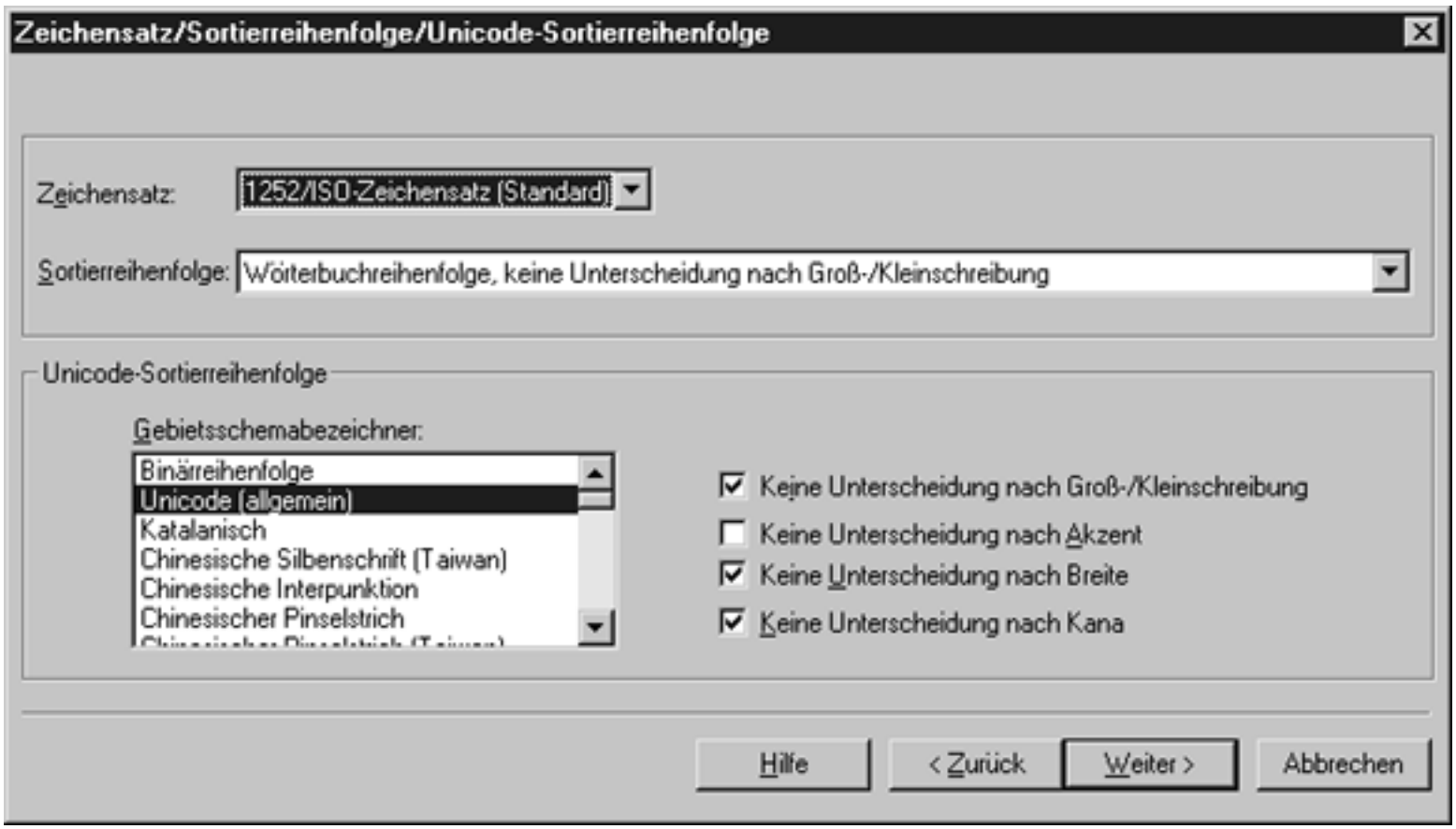

Microsoft
SQLServer
Führt den Standardinstallations-Assistenten für Internet Explorer 4.01 Service Pack 1 aus.
SQL Server 7.0-Voraussetzungen
Windows NT 4.0
Schritt 1: Windows NT 4.0 SP 4
Das Service Pack befindet sich auf einer separaten, in diesem Produkt eingeschlossenen CD. Instruktionen für die Installation finden Sie in den Release Notes.
Schritt 2: Internet Explorer 4.01 SP 1
(Nicht erforderlich für Installationen vom Typ Minimal, Connectivity oder Nur Server)
Minimalinstallation
Setup-Assistent starten (L)
Zurück (B) Beenden (X)


Microsoft Windows NT 4.0 Service Pack 4 - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Wechseln zu Favoriten ?
Zurück Vorwärts Abbrechen Aktualisieren Startseite Suchen Favoriten Verlauf
Adresse K:\NTSP4.HTM
Links
Microsoft Windows NT 4.0
Service Pack
INHALT
Einführung
Versionshinweise
Installieren von Service Pack 4
Microsoft Windows NT Server NetShow Services
EINFÜHRUNG
Diese Service Pack-CD enthält das Standard-Service Pack für Windows NT® sowie Aktualisierungen für Funktionen von Windows NT und Produkte, die mit Windows NT 4.0 geliefert werden.
Microsoft Windows NT 4.0 Service Pack 4 ist einfach zu installieren und aktualisiert nur die Dateien, die in der ursprünglichen Betriebssystemkonfiguration von Windows NT Workstation bzw. Windows NT Server eingerichtet wurden. Service Pack-Versionen sind kumulativ, d.h. sie enthalten neben den neuen
file:///K:/s
Arbeitsplatz


Microsoft
SQLServer
Installieren der vollständigen Version von SQL Server 7.0.
Installation der SQL Server 7.0-Komponenten.
Datenbankserver - Standard Edition
Datenbankserver - Desktop Edition
SQL Server 7.0 OLAP Services
English Query
Zurück (B)
Beenden (X)


Willkommen
Willkommen zum Microsoft SQL Server 7.0-Setup. Mit diesem Programm wird Microsoft SQL Server 7.0 auf Ihrem Computer installiert.
Es wird dringend empfohlen, daß Sie alle Windows-Programme beenden, bevor Sie das Setup ausführen.
Klicken Sie auf Abbrechen, um Setup zu beenden, und schließen Sie danach alle geöffneten Programme. Wählen Sie Weiter, um mit dem Setup fortzufahren.
WARNUNG: Diese Anwendung ist durch Urheberrecht und internationale Vereinbarungen geschützt.
Unberechtigte Reproduktion oder nicht genehmigter Vertrieb dieser Anwendung oder einer ihrer Komponenten wird gerichtlich verfolgt und kann zu erheblichen Strafen führen.
< Zurück
Weiter >
Abbrechen


Software-Lizenzvertrag
Lesen Sie bitte den folgenden Software-Lizenzvertrag. Mit der Nach-unten-Taste können Sie den Rest des Vertrags anzeigen.
ENDBENUTZER-LIZENZVERTRAG
SERVERLIZENZ FÜR MICROSOFT-SERVERPRODUKTE
WICHTIG - BITTE SORGFÄLTIG LESEN: Dieser Endbenutzer-Lizenzvertrag ("EULA") ist ein rechtsgültiger Vertrag zwischen Ihnen (entweder als natürliche oder als juristische 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 ("SOFTWAREPRODUKT"). Diesem SOFTWAREPRODUKT liegt möglicherweise eine Ergänzung oder ein Nachtrag zu diesem EULA bei. INDEM SIE DAS SOFTWAREPRODUKT INSTALLIEREN, KOPIEREN ODER ANDERWEITIG VERWENDEN, ERKLÄREN SIE SICH EINVERSTANDEN, DURCH DIE BESTIMMUNGEN DIESES EULAS GEBUNDEN ZU SEIN. FALLS SIE DEN BESTIMMUNGEN DIESES EULAS NICHT ZUSTIMMEN, SIND SIE NICHT BERECHTIGT, DAS SOFTWAREPRODUKT ZU INSTALLIEREN ODER ZU
Stimmen Sie sämtlichen Bedingungen des vorstehenden Lizenzvertrags zu? Wenn Sie Nein wählen, wird das Setup abgebrochen. Für die Installation von Microsoft SQL Server 7.0 müssen Sie diesem Lizenzvertrag zustimmen.
< Zurück
Ja
Nein


Zeichensatz/Sortierreihenfolge/Unicode-Sortierreihenfolge
Zeichensatz:
1252/ISO-Zeichensatz (Standard)
Sortierreihenfolge:
Wörterbuchreihenfolge, keine Unterscheidung nach Groß-/Kleinschreibung
Unicode-Sortierreihenfolge
Gebietsschemabezeichner:
Binärreihenfolge
Unicode (allgemein)
Katalanisch
Chinesische Silbenschrift (Taiwan)
Chinesische Interpunktion
Chinesischer Pinselstrich
Keine Unterscheidung nach Groß-/Kleinschreibung
Keine Unterscheidung nach Akzent
Keine Unterscheidung nach Breite
Keine Unterscheidung nach Kana
Hilfe
< Zurück
Weiter >
Abbrechen


Zielpfad wählen
Setup wird OLAP Services in folgendem Ordner installieren.
Klicken Sie auf Weiter zur Installation in diesem Ordner, auf Durchsuchen zur Auswahl eines anderen Ordners.
Wählen Sie Abbrechen, um Setup zu beenden, wenn OLAP Services nicht installiert werden soll.
Zielordner
d:\Programme\OLAP Services
Durchsuchen...
< Zurück
Weiter >
Abbrechen


Komponenten wählen
Wählen Sie die Komponenten, die Sie installieren möchten, und löschen Sie die Komponenten, die Sie nicht installieren möchten.
Komponenten
OLAP-Server 30408 K
OLAP-Manager 17871 K
Clientkomponenten 5791 K
Beispielanwendungen 1673 K
Beschreibung
Diese Komponente beinhaltet die ausführbaren Binärdateien des OLAP-Server und verwandte Dateien.
Ändern...
Benötigt: 56269 K
Verfügbar: 658240 K
< Zurück
Weiter >
Abbrechen

kapitel 3 komponenten der sql-server-installation

3.1 bestandsaufnahme

nachdem sie die installation von sql server hinter sich gebracht haben, sollten sie sich zunächst einen überblick verschaffen, welche komponenten zu einer sql-server-installation gehören. außerdem erfahren sie in diesem kapitel, wie sie mit den wichtigsten programmen der benutzeroberfläche von sql server arbeiten.

3.2 installierte komponenten

bei der benutzerdefinierten installation haben sie im dialogfeld **komponenten auswählen** (siehe kapitel 2) die verschiedenen kategorien gesehen, die zu einer installation von sql server gehören:

- server-komponenten
- verwaltungstools
- client-konfiguration
- online-dokumantation
- entwicklungstools
- codebeispiele

die folgenden abschnitte zeigen, wo die verschiedenen komponenten untergebracht sind und um welche programme es sich handelt.

3.2.1 verzeichnisse

tabelle 3.1 gibt einen überblick über die verzeichnisse, die das setup-programm angelegt hat. zum beispiel benötigen sie das verzeichnis, in dem ihre datenbanken abgelegt sind, wenn sie eine odbc-datenquelle erstellen.

ordner	beschreibung
\mssql7	standardverzeichnis von sql server, sofern sie bei der installation keinen anderen pfad angegeben haben
\mssql7\backup	sicherungsdateien
\mssql7\binn	programmdateien, hilfedateien und dll-dateien für erweiterte gespeicherte prozeduren

\mssql7\books	kompilierte html-dateien der online-dokumentation
\mssql7\data	dateien der system- und beispieldatenbanken. das ist auch das standardverzeichnis für die von ihnen angelegten datenbanken.
\mssql7\devtools	include-dateien, bibliotheksdateien und beispieldateien für die entwicklungsunterstützung
\mssql7\ftdata	dateien der volltextkataloge
\mssql7\html	html-dateien für sql server und die microsoft management console (mmc). löschen sie diese dateien nicht, da sql server sie für den ordnungsgemäßen betrieb benötigt!
\mssql7\install	während des setup-programms ausgeführte skripts und die dabei angelegten ausgabedateien
\mssql7\jobs	temporäre ausgabedateien für aufträge
\mssql7\log	fehlerprotokolldateien
\mssql7\repldata	arbeitsverzeichnis für aufgaben in verbindung mit der replikation
\mssql7\upgrade	dateien für die aktualisierung von sql server der version 6.x auf die version 7.0

tabelle 3.1: standardverzeichnisse von sql server

3.2.2 dienste

das setup-programm richtet unter windows nt die folgenden dienste ein:

- mssqlserver: der eigentliche datenbankserver.
- sqlserveragent: unterstützt die ausführung administrativer aufgaben auf der basis eines terminplans.
- msdtc: manager für verteilte transaktionen.
- microsoft search: unterstützt die volltextindizierung von textfeldern.

3.2.3 datenbanken

das setup-programm erzeugt datenbank- und protokolldateien für die systemdatenbanken master, model, tempdb und msdb sowie die als beispiel gelieferten benutzerdatenbanken pubs und northwind. der pfad zu diesen dateien sollte sich auf einem laufwerk befinden, das für das wachstum der betreffenden datenbanken genügend reserven bietet. in der voreinstellung werden die datenbank- und protokolldateien im verzeichnis \mssql7\data installiert.

3.2.4 die programmgruppe microsoft sql server 7.0

nach der installation befinden sich im startmenü zwei einträge, die sich auf sql server 7.0 beziehen: die programmgruppe **microsoft sql server 7.0** und die gruppe **microsoft sql server - versionsumstellung**. auf die versionsumstellung, d.h. auf den übergang von einer vorherigen version des sql server (versionen 6.x) zur version 7.0, geht kapitel 29 näher ein. das vorliegende kapitel befaßt sich mit den komponenten der sql-server-installation, die in der programmgruppe **microsoft sql server 7.0** verfügbar sind (siehe abbildung 3.1).

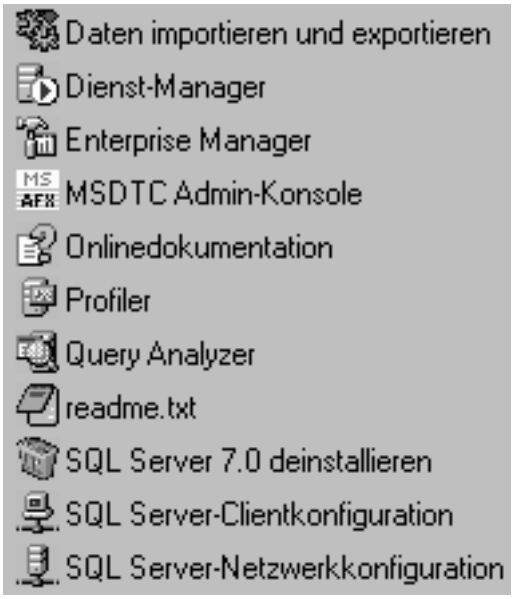


abbildung 3.1: die programmgruppe microsoft sql server 7.0

3.2.5 online-dokumentation

auch wenn ihnen dieses kompendium einen umfassenden einblick in die praktische arbeit mit sql server 7.0 bietet - ohne die online-dokumentation werden sie nicht auskommen. bei ihrer täglichen arbeit stellt die online-dokumentation eine wichtige quelle für detaillierte informationen dar. es handelt sich um eine sammlung von html-seiten, die sich mit hilfe des internet explorers ab version 4.01 (service pack 1) betrachten lassen.

3.3 grafische benutzeroberfläche

in sql server 7.0 hat die grafische benutzeroberfläche erhebliche erweiterungen erfahren. nahezu alle aspekte der administration lassen sich über die bereitgestellten grafischen dienstprogramme realisieren. die folgenden abschnitte gehen auf die wichtigsten ein.

3.3.1 microsoft management console

die microsoft management console (mmc) dient als einheitliche benutzeroberfläche und umgebung für microsoft backoffice-server. die konsole für sql server ist der enterprise manager.

3.3.2 sql server enterprise manager

der enterprise manager ist ihr regiezentrum für sql server. von hier aus können sie den größten teil der administrativen aufgaben abwickeln, d.h. sql server und sql-server-objekte konfigurieren und verwalten. im einzelnen lassen sich folgende funktionen realisieren:

- starten, anhalten/fortsetzen, beenden und konfigurieren der dienste sql server, sql agent, sql mail, dts, microsoft search
- sicherungen verwalten
- datenbanken verwalten
- verwaltungsaufgaben der datenbank verwalten
- anmeldungen und berechtigungen verwalten
- replikation verwalten
- tabellen, sichten, gespeicherte prozeduren, trigger, indizes, regeln, standardwerte und benutzerdefinierte datentypen verwalten
- aufgaben planen
- webseiten erzeugen
- sql-skripts erstellen

den enterprise manager starten sie über **start / programme / microsoft sql server 7.0 / enterprise manager**.

taskpad

im menü **ansicht** des enterprise managers ist je nach markiertem element in der konsolenstruktur der befehl **taskpad** verfügbar. mit diesem befehl aktivieren sie für den rechten fensterausschnitt - den *detailbereich* - des enterprise managers eine übersichtsdarstellung mit verweisen zu aufgaben, die sich auf den kontext des in der konsolenstruktur markierten objekts beziehen (siehe abbildung 3.2). damit erhalten sie einen einfachen einstieg in die welt von sql server.

der befehl **taskpad** ist nur verfügbar, wenn mindestens ein server registriert ist.

bild

abbildung 3.2: das taskpad erleichtert ihnen den einstieg in sql server

3.3.3 msdtc-admin-konsole

die msdtc-admin-konsole dient der verwaltung des microsoft distributed coordinator (dtc). der microsoft transaction coordinator (mtc) bedient sich des dtc bei verteilten transaktionen. kapitel 18 geht näher auf msdtc ein.

3.3.4 profiler

sql server profiler ist ein grafisches werkzeug, mit dem sich folgende server-ereignisse überwachen und auflisten lassen:

- hergestellte und getrennte verbindungen zu servern
- transact-sql-stapel
- ausführung von anweisungen innerhalb gespeicherter prozeduren
- deadlocks
- fehler, die in das sql-server-fehlerprotokoll geschrieben werden

kapitel 24 geht näher auf sql server profiler ein.

3.3.5 query analyzer

sql server query analyzer ist ein grafisches tool, mit dem sich folgende aufgaben realisieren lassen:

- sql-skripts und abfragen bearbeiten und erstellen.
- sql-server-datenbanken abfragen.
- eine grafische darstellung des (geschätzten) ausführungsplans anzeigen.
- abfrageergebnisse in tabellenform oder als text anzeigen.
- eine indexanalyse durchführen.
- kontextbezogene hilfe zur transact-sql-syntax aufrufen.
- statistikinformationen zu einer ausgeführten abfrage anzeigen.
- mehrere transact-sql-skripts oder gespeicherte prozeduren gleichzeitig ausführen.

3.3.6 readme.txt

über diesen menüpunkt öffnen sie die datei `readme.txt`. hier finden sie wichtige hinweise, die sie vor der installation lesen sollten (siehe abschnitt »release notes« in kapitel 2).

3.3.7 sql server 7.0 deinstallieren

über diesen menüpunkt können sie sql server direkt deinstallieren, ohne zuerst **systemsteuerung / software** aufzurufen. auf die deinstallation wurde bereits in kapitel 2 eingegangen.

3.3.8 sql-server-client-konfiguration

mit der sql-server-clientkonfiguration lassen sich die netzwerkbibliotheken des clients verwalten und aliasnamen für server definieren. außerdem können sie standardoptionen für anwendungen, die mit der db-library arbeiten, festlegen.

3.3.9 sql-server-netzwerkkonfiguration

auf die sql-server-netzwerkkonfiguration müssen sie zurückgreifen, wenn der server ein bestimmtes netzwerkprotokoll verwendet, das sql server standardmäßig nicht abfragt, und die standardnetzwerkbibliothek von sql server nicht aktiviert ist, um sql-server-clients abzufragen.

3.3.10 systemmonitor

mit dem systemmonitor lassen sich die ressourcen eines computers überwachen. dieses werkzeug ist in das betriebssystem windows nt integriert und nicht unter windows 95/98 verfügbar. der windows-nt-systemmonitor läßt sich erweitern. damit kann man für jede server-anwendung eigene leistungsindikatoren hinzufügen. auf die überwachung von sql server geht kapitel 24 näher ein.

3.3.11 microsoft sql server - versionsumstellung

wenn auf ihrem computer bereits sql server der version 6.x installiert ist, entfernt das setup-programm die programmgruppe **microsoft sql server 6.x**, da sich alte und neue versionen von sql server nicht gleichzeitig ausführen lassen. an die stelle dieser programmgruppe tritt der eintrag **microsoft sql server - versionsumstellung** mit den untereinträgen **microsoft sql server 6.x**, **sql server 6.x deinstallieren** und **sql server-aktualisierungs-assistent**. der assistent überträgt den inhalt einer 6.x-datenbank nach version 7.0. weitere angaben zu diesem assistenten finden sie in kapitel 29.

3.3.12 sql-server-agent

dieses dienstprogramm dient der automatischen ausführung administrativer aufgaben. außerdem überwacht es ereignisse und führt gegebenenfalls die angegebenen warnungsreaktionen aus. kapitel 23 beschäftigt sich ausführlich mit dem sql-server-agenten.

3.4 die assistenten von sql server

für komplexe administrative aufgaben stellt sql server zahlreiche assistenten bereit. die assistenten von sql server sind im enterprise manager verfügbar, wenn sie die konsolenstruktur mindestens bis zu einem server erweitert haben. um einen assistenten aufzurufen, wählen sie entweder den befehl **assistenten** im menü **extras** oder klicken auf die schaltfläche **assistenten auswählen** in der symbolleiste (siehe abbildung 3.3).

bild

abbildung 3.3: die schaltfläche assistenten in der symbolleiste des enterprise manager

daraufhin wird das dialogfeld **assistent auswählen** geöffnet (siehe abbildung 3.4).

klicken sie auf das pluszeichen, um die strukturansicht der assistenten in der zutreffenden kategorie zu erweitern. markieren sie dann den gewünschten assistenten, und klicken sie auf **ok**.

abbildung 3.5 zeigt das startdialogfeld des anmeldungserstellungs-assistenten. die startdialogfelder der anderen assistenten sind gleich aufgebaut. um ihnen die auswahl des passenden assistenten zu

erleichtern, sind diese mit den texten der startbildschirme und einem verweis, in welchem kapitel der jeweilige assistent näher behandelt wird, nach kategorien zusammengestellt. die kategorien entsprechen bis auf die zusätzlich aufgeführten assistenten der zuordnung, die sie im dialogfeld **assistent auswählen** (siehe abbildung 3.4) vorfinden.

abbildung 3.4: startdialogfeld des anmeldungserstellungs-assistenten

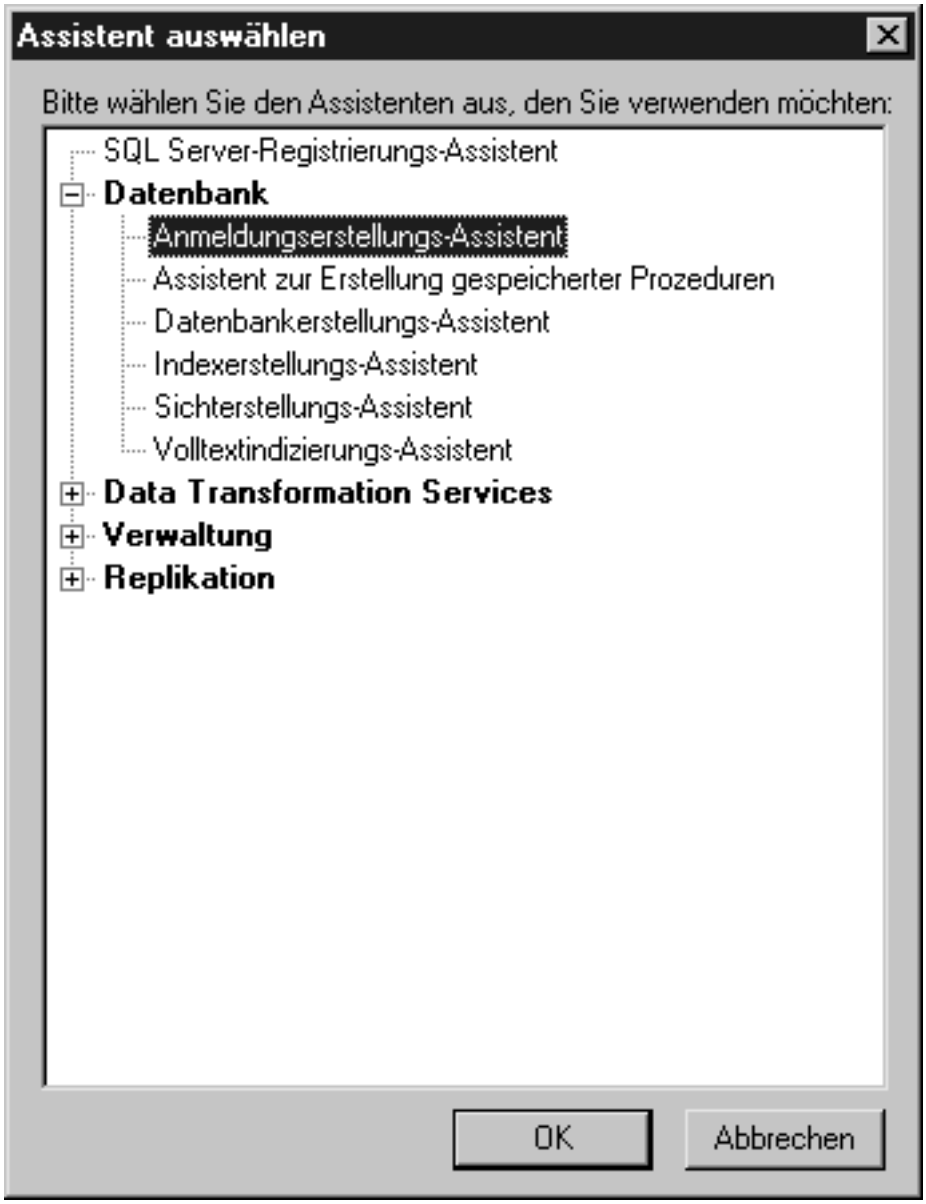


abbildung 3.5: das dialogfeld assistent auswählen

3.4.1 datenbanken

anmeldungserstellungs-assistent

dieser assistent hilft ihnen dabei, einem benutzer den sql-server-anmeldezugriff zu gewähren. mit dem assistenten können sie:

- einen authentifizierungsmodus auswählen.
- zugriff auf sicherheitsrollen gewähren.
- zugriff auf datenbanken gewähren.

siehe kapitel 20.

assistent zur erstellung gespeicherter prozeduren

dieser assistent hilft ihnen beim erstellen gespeicherter prozeduren mit transact-sql. mit dem assistenten

können sie:

- eine datenbank auswählen, die die gespeicherten prozeduren speichert.
- gespeicherte prozeduren generieren, die zeilen einfügen, aktualisieren oder löschen.
- die eigenschaften von gespeicherten prozeduren bearbeiten.
- transact-sql-anweisungen bearbeiten, die die gespeicherten prozeduren erstellen.

siehe kapitel 12.

datenbankerstellungs-assistent

dieser assistent hilft beim erstellen einer neuen datenbank. mit dem assistenten können sie:

- die datenbank benennen.
- die datei(en) erstellen, aus denen die datenbank bestehen soll.
- die vergrößerungsinformationen für datenbankdateien angeben.
- die datei(en) erstellen, aus denen das transaktionsprotokoll bestehen soll.
- die vergrößerungsinformationen für transaktionsprotokolldateien angeben.

siehe kapitel 6.

datenbankdiagrammerstellungs-assistent

dieser assistent hilft beim erstellen eines neuen diagramms. mit dem assistenten können sie:

- ein neues diagramm erstellen und diesem tabellen hinzufügen.
- verknüpfte tabellen zum diagramm hinzufügen.
- die tabellen im diagramm automatisch anordnen.

siehe kapitel 6.

indexerstellungs-assistent

dieser assistent hilft ihnen beim erstellen eines neuen index. mit dem assistenten können sie:

- die datenbank und die tabelle auswählen, die sie indizieren möchten.
- informationen über die aktuellen indizes anzeigen.
- die spalte(n) auswählen, die in den index einbezogen werden soll(en).

siehe kapitel 14.

sichterstellungs-assistent

dieser assistent hilft ihnen beim erstellen einer neuen sicht. mit dem assistenten können sie:

- die datenbank auswählen, auf die die sicht verweisen soll.
- die tabelle(n) auswählen, auf die die sicht verweisen soll.
- die spalte(n) auswählen, die diese sicht anzeigen soll.

- die sicht und beschränkungen definieren.

siehe kapitel 13.

3.4.2 data transformation services

dts-export-assistent

der dts-export-assistent erleichtert das exportieren und transformieren heterogener daten. der assistent führt durch die schritte zum exportieren von daten in zahlreiche gebräuchliche datenformate, einschließlich datenbanken, kalkulationstabellen und textdateien.

siehe kapitel 9.

dts-import-assistent

der dts-import-assistent erleichtert das importieren und transformieren heterogener daten. der assistent führt durch die schritte zum importieren von daten aus zahlreichen gebräuchlichen datenformaten, einschließlich datenbanken, kalkulationstabellen und textdateien.

siehe kapitel 9.

3.4.3 verwaltung

ablaufverfolgungserstellungs-assistent

dieser assistent unterstützt sie beim erstellen und starten einer sql server profiler-ablaufverfolgung, um die ursache häufig auftretender probleme von datenbankanwendungen zu diagnostizieren. mit diesem assistenten können sie folgendes ausführen:

- identifizieren des problems
- angeben von filtereinstellungen
- ablaufverfolgungsdefinition vervollständigen

nachdem sie eine ablaufverfolgung erstellt haben, können sie sie jederzeit von sql server profiler ausführen.

siehe kapitel 24.

auftragserstellungs-assistent

dieser assistent hilft beim erstellen eines auftrags, der wiederholt ausgeführt werden kann. mit dem assistenten können sie

- einen transact-sql-auftrag, einen auftrag für einen befehl der betriebssystem-shell oder einen activescript-auftrag erteilen.
- die auftragshäufigkeit planen.
- operatoren über den auftragsstatus benachrichtigen.

siehe kapitel 23.

datenbankwartungsplanungs-assistent

mit hilfe dieses assistenten können sie einen wartungsplan erstellen, der vom sql-server-agenten regelmäßig ausgeführt wird. mit diesem assistenten können sie

- datenbankintegritätsprüfungen ausführen.
- datenbankstatistiken aktualisieren.
- datenbanksicherungen durchführen.

siehe kapitel 20.

indexoptimierungs-assistent

der assistent unterstützt sie bei der analyse der aktuellen indizes und schlägt indizes zur leistungssteigerung bei abfragen und aktualisierungen vor. der assistent führt die folgenden schritte aus:

- server und zu optimierende datenbanken festlegen.
- zu analysierende arbeitsauslastung festlegen.
- zu optimierende tabellen auswählen.
- daten analysieren und indexempfehlungen formulieren.
- indexempfehlungen implementieren.

siehe kapitel 24.

masterservererstellungs-assistent

dieser assistent hilft ihnen dabei, einen server als masterserver (msx) einzurichten. mit diesem assistenten können sie:

- einen neuen operator namens 'msxoperator' erstellen.
- die sicherheit von sql server auf windows-nt- und sql-server-authentifizierung festlegen (betrifft windows nt server).
- sql-server-agenten starten.
- andere computer mit sql server als zielserver (tsx) eintragen.
- die startkonten für den sql-server- und den sql-server-agent-dienst auf den zielservern in gültige windows-nt-konten (betrifft nur windows nt server) ändern.

sicherungs-assistent

dieser assistent hilft ihnen bei der durchführung einer datenbanksicherung. der assistent stellt ihnen verschiedene fragen über die zu sichernden daten und bietet dann verschiedene sicherungsoptionen an. mit dem assistenten können sie:

- die durchzuführende sicherung angeben.
- die zur datensicherung zu verwendenden medien auswählen.

- daten an die sicherungsmedien anhängen oder diese überschreiben.
- die sicherung überprüfen.

siehe kapitel 7.

web-assistent

dieser assistent hilft ihnen beim veröffentlichen der daten einer sql-server-tabelle auf einer webseite. mit dem assistenten können sie:

- daten von einem server, auf dem sql server ausgeführt wird, auf einer webseite veröffentlichen.
- die aktualisierungshäufigkeit von webseiten festlegen.
- das format der veröffentlichten webseite angeben.

siehe kapitel 28.

warnungserstellungs-assistent

dieser assistent hilft ihnen beim erstellen einer neuen warnung. mit dem assistenten können sie:

- die warnung und die reaktion für die warnung definieren.
- eine datenbank angeben, um die warnungsdefinition zu verfeinern.
- fehlerschlüsselwörter angeben, um die warnungsdefinition zu verfeinern.

siehe kapitel 23.

zielservererstellungs-assistent

dieser assistent hilft ihnen dabei, einen server als zielserver (tsx) einzurichten. mit diesem assistenten können sie folgendes ausführen:

- eintragen beim masterserver.

sql-server-registrierungs-assistent

dieser assistent hilft ihnen dabei, einen oder mehrere sql server zu registrieren. mit diesem assistenten können sie

- einen computer mit sql server auswählen.
- einen authentifizierungsmodus auswählen.
- eine sql-server-gruppe angeben.

siehe kapitel 2.

sql-server-aktualisierungs-assistent

diesen assistenten rufen sie unter windows nt über **start / programme / microsoft sql server - versionsumstellung / sql server-aktualisierungs-assistent** auf.

der assistent dient zur übertragung ihrer server-konfiguration und ihrer datenbanken von einem einzelnen

6.x-sql-server zu der auf diesem computer installierten kopie von sql server 7.0.

darüber hinaus führt der assistent gültigkeitsprüfungen auf daten- und objektebene durch, um sicherzustellen, daß die migration der 6.x-datenbank in die version 7.0 korrekt ist. weitere informationen zur versionsumstellung finden sie in kapitel 29.

bild

abbildung 3.6: der sql-server-aktualisierungs-assistent

3.4.4 replikation

publikationserstellungs-assistent

dieser assistent hilft ihnen, daten zu publizieren, so daß sie gemeinsam mit abonnenten genutzt werden können.

mit diesem assistenten können sie:

- eine publikation aus den daten in folgender datenbank erstellen: *'datenbankname'*.
- daten in der publikation filtern.
- eigenschaften der publikation einstellen.

siehe kapitel 22.

publizierungs- und verteilungskonfigurations-assistent

mit diesem assistenten können sie:

- *'servername'* oder einen anderen server als verteiler angeben.
- eigenschaften von *'servername'* als verteiler konfigurieren.
- eigenschaften von *'servername'* als verleger konfigurieren (optional).

siehe kapitel 22.

publizierungs- und verteilungsdeaktivierungs-assistent

dieser assistent hilft, die publizierung, die verteilung oder beides auf *'servername'* zu deaktivieren.

das deaktivieren von publizierung, verteilung oder beidem wirkt sich wie folgt aus:

- alle publikationen auf diesem server werden gelöscht.
- alle abonnements für die gelöschten publikationen werden gelöscht.
- abonnements, die *'servername'* von anderen verlegern empfängt, sind nicht betroffen.

siehe kapitel 22.

pullabonnement-assistent

dieser assistent hilft ihnen, ein pullabonnement auf dem abonnenten *'servername'* zu erstellen.

mit diesem assistenten können sie:

- den verleger und die publikation aus wählen.
- die datenbank auswählen, in der das abonnement erstellt wird.
- den initialisierungs- und synchronisationsterminplan für das abonnement festlegen.
- andere eigenschaften des abonnements einstellen.

siehe kapitel 22.

pushabonnement-assistent

dieser assistent hilft beim erstellen von push für ein abonnement von publikationen *'name'*, verlegt von *'servername'*, an einen oder mehrere server oder server-gruppen.

- mit diesem assistenten können sie:
- einen oder mehrere abonnenten auswählen.
- die datenbank auswählen, in der das abonnement erstellt wird.
- den initialisierungs- und synchronisationsterminplan für das abonnement festlegen.
- andere eigenschaften des abonnements einstellen.

siehe kapitel 22.

3.4.5 verschiedenes

volltextindizierungs-assistent

der assistent hilft ihnen beim erstellen eines volltextindexkatalogs für die datenbank. mit diesem assistenten können sie:

- eine zu indizierende tabelle auswählen.
- zeichenbasierte spalten auswählen, die indiziert werden sollen.
- weitere datenbanktabellen zu einem vorhandenen katalog hinzufügen.

siehe kapitel 14.

3.5 integration in das betriebssystem

sql server ist auf die betriebssysteme windows nt und windows 95/98 zugeschnitten. aufgrund der skalierbarkeit sind die wesentlichen funktionen unter diesen betriebssystemen gleich. spezielle einrichtungen, wie zum beispiel dienste oder sicherheit, sind dagegen nur unter windows nt verfügbar.

3.5.1 taskleiste

der sql-server-dienst-manager zeigt den momentanen zustand der dienste mssqlserver, msdtc, sqlserveragent und microsoft search als symbol in der taskleiste an. die genannten dienste können sie manuell starten/fortsetzen, anhalten und beenden, indem sie auf das symbol des

sql-server-dienst-managers doppelklicken und den jeweiligen dienst aus dem drop-down-listenfeld **dienste** auswählen.

3.5.2 systemsteuerung

windows nt listet die dienste von sql server in der systemsteuerung unter **dienste** auf (siehe abbildung 3.7). dabei handelt es sich um mssqlserver, sqlserveragent, msdtc und microsoft search.

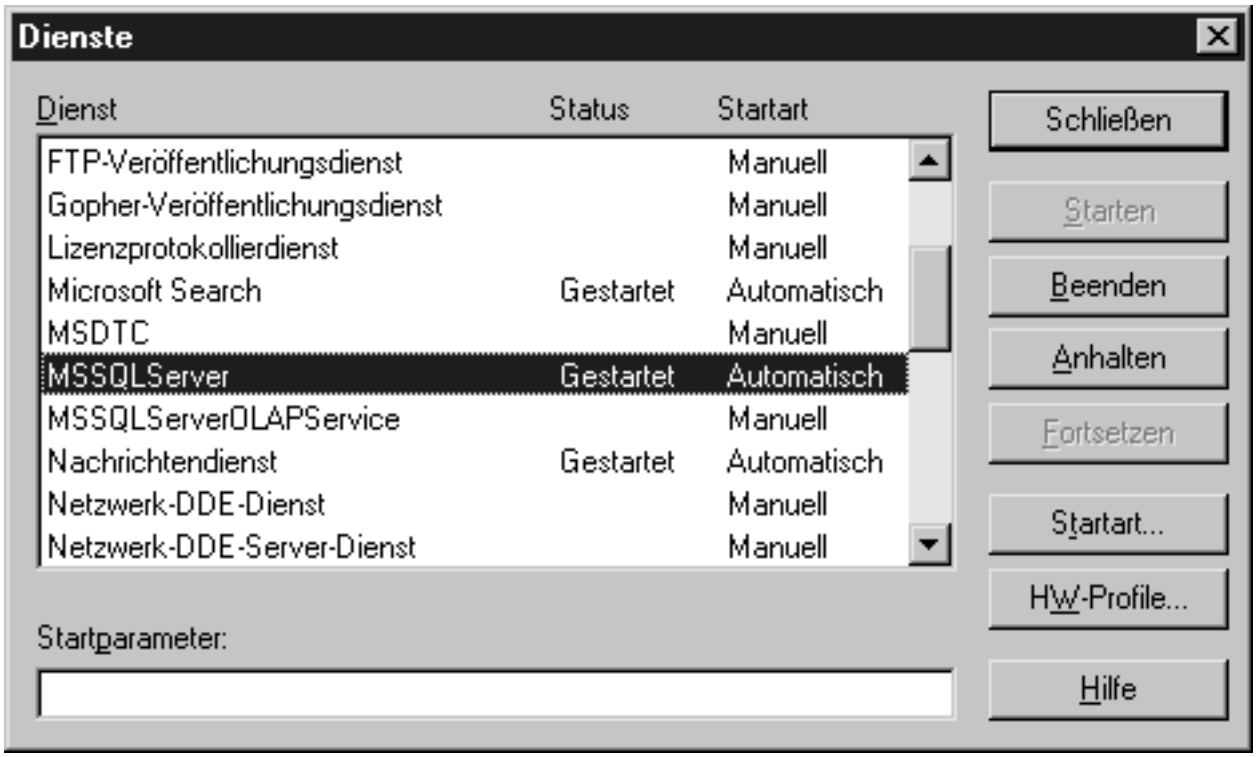


abbildung 3.7: das programm dienste aus der systemsteuerung von windows nt

3.5.3 benutzerkonten von windows nt

wenn sich ein benutzer über die windows-nt-authentifizierung anmeldet, greift sql server auf die benutzerkonten und kennwörter von windows nt zurück.

damit läßt sich mit ein und demselben benutzerkonto sowohl der zugriff auf windows nt als auch sql server regeln. dadurch vereinfachen sich die anmeldeprozeduren und die verwaltung von benutzerkonten. außerdem läßt sich eine »doppelte kontoführung« vermeiden. den benutzer-manager von windows nt (siehe abbildung 3.8) starten sie über **start / programme / verwaltung (allgemein) / benutzer-manager für domänen**.

bild

abbildung 3.8: der windows-nt-benutzer-manager für domänen

3.5.4 systemmonitor

mit dem windows-nt-systemmonitor läßt sich die nutzung der ressourcen auf einem windows-nt-computer grafisch überwachen. jeder leistungsindikator gibt ein maß für die nutzung einer bestimmten ressource an. die standardmäßig von windows nt bereitgestellten leistungsindikatoren erweitert sql server um spezielle indikatoren wie zum beispiel:

- allgemeine statistik (abmeldungen und anmeldungen pro sekunde, benutzerverbindungen)
- abfragen
- cache-verwendung
- datenbanken (u.a. massenkopieren, transaktionen pro sekunde)
- replikationsaktivität
- sperren

abbildung 3.9 zeigt die anzeige des systemmonitors, der über sql-server-profiler - mit speziellen leistungsindikatoren für sql server - gestartet wurde.

bild

abbildung 3.9: der windows-nt-systemmonitor

3.5.5 ereignisanzeige

in der windows-nt-ereignisanzeige (siehe abbildung 3.10) protokolliert sql server informationen, fehler und warnungen. mit der erweiterten gespeicherten prozedur xp_logevent können sie eigene meldungen in die protokolldatei von sql server schreiben und in der windows-nt-ereignisanzeige protokollieren.

Bild

abbildung 3.10: die windows-nt-ereignisanzeige

3.6 beispieldatenbanken

in der version 6.5 von sql server gehörten die beispieldatenbanken nordwind und verleger zum lieferumfang. die englischen namen lauten northwind und pubs. die version 7.0 bringt nur noch die englischen datenbanken mit. die verlegerdatenbank wird im buch durchgängig mit pubs referenziert, um verwechslungen mit den ebenfalls als verlegerdatenbanken bezeichneten objekten im zusammenhang mit der replikation zu vermeiden. (verleger sind server, die anderen servern daten für die replikation zur verfügung stellen.)

3.7 tools der befehlszeile

im zeitalter der grafischen benutzeroberflächen mag vielen schon der gedanke an die eingaben auf der befehlszeile mit den teilweise komplizierten schaltern und parametern angst einjagen, und es stellt sich die frage, warum in dieser hinsicht überhaupt derartige werkzeuge in sql server verfügbar sind. es sind

vor allem folgende gründe:

- ausführungsgeschwindigkeit
- minimaler overhead und damit geringerer speicherbedarf als vergleichbare programme mit grafischer benutzeroberfläche
- starten von sql server im einzelbenutzermodus oder mit der minimalkonfiguration zum beheben von problemen
- wiederherstellen der master-datenbank

die folgenden abschnitte erläutern die wichtigsten tools der befehlszeile im überblick und nennen gegebenenfalls die kapitel, die sich näher mit den jeweiligen programmen beschäftigen.

3.7.1 bcp

kopiert eine datenbanktabelle in eine oder aus einer datendatei, wobei der benutzer ein format vorgeben kann. kapitel 9 geht im zusammenhang mit dem import und export von daten näher auf bcp ein.

3.7.2 isql

mit dem dienstprogramm isql lassen sich transact-sql-anweisungen, systemprozeduren und skriptdateien ausführen. das programm isql greift auf sql server über eine db-library-schnittstelle zu, während osql über odbc arbeitet. da isql aus gründen der abwärtskompatibilität zu sql server 6.5 vorgesehen ist, verzichten wir aus platzgründen auf die behandlung. die wesentlichen befehle entsprechen dem dienstprogramm osql. in isql sind verschiedene features von sql server 7.0 nicht zugänglich.

3.7.3 makepipe / readpipe

das dienstprogramm makepipe setzen sie in verbindung mit readpipe ein, um das named-pipe-protokoll zu testen (siehe kapitel 2).

3.7.4 odbcping

mit dem dienstprogramm odbcping läßt sich testen, ob die odbc-datenquelle in ordnung ist und sich der client mit dem server verbinden kann.

syntax:

```
odbcping [/?] |
  [{ -sservername | -ddatenquelle } [-ulogin_id] [-pkennwort ] ]
```

tabelle 3.2 erläutert die argumente des befehls odbcping.

argument	beschreibung

/?	zeigt die syntax des befehls odbcping an.
-s*servername*	name des servers, zu dem eine verbindung herzustellen ist. die verbindung erfolgt ohne testen von odbc-datenquellen.
-d*datenquelle*	name einer definierten odbc-datenquelle, die den microsoft-sql-server-odbc-treiber verwenden soll. odbcping prüft die korrektheit der datenquelle, indem mit hilfe dieses treibers eine verbindung zu dem in der datenquelle genannten server hergestellt wird.
-u*login_id*	ein gültiger anmeldename für den server.
-p*kennwort*	kennwort für den anmeldenamen.

tabelle 3.2: argumente des befehls odbcping

ein beispiel für den einsatz von odbcping haben sie bereits in kapitel 2 kennengelernt, um die verbindung zum server nach der installation von sql server zu prüfen.

3.7.5 osql

ein dienstprogramm zum ausführen von transact-sql-anweisungen. kapitel 5 geht näher auf dieses programm ein.

3.7.6 rebuild master

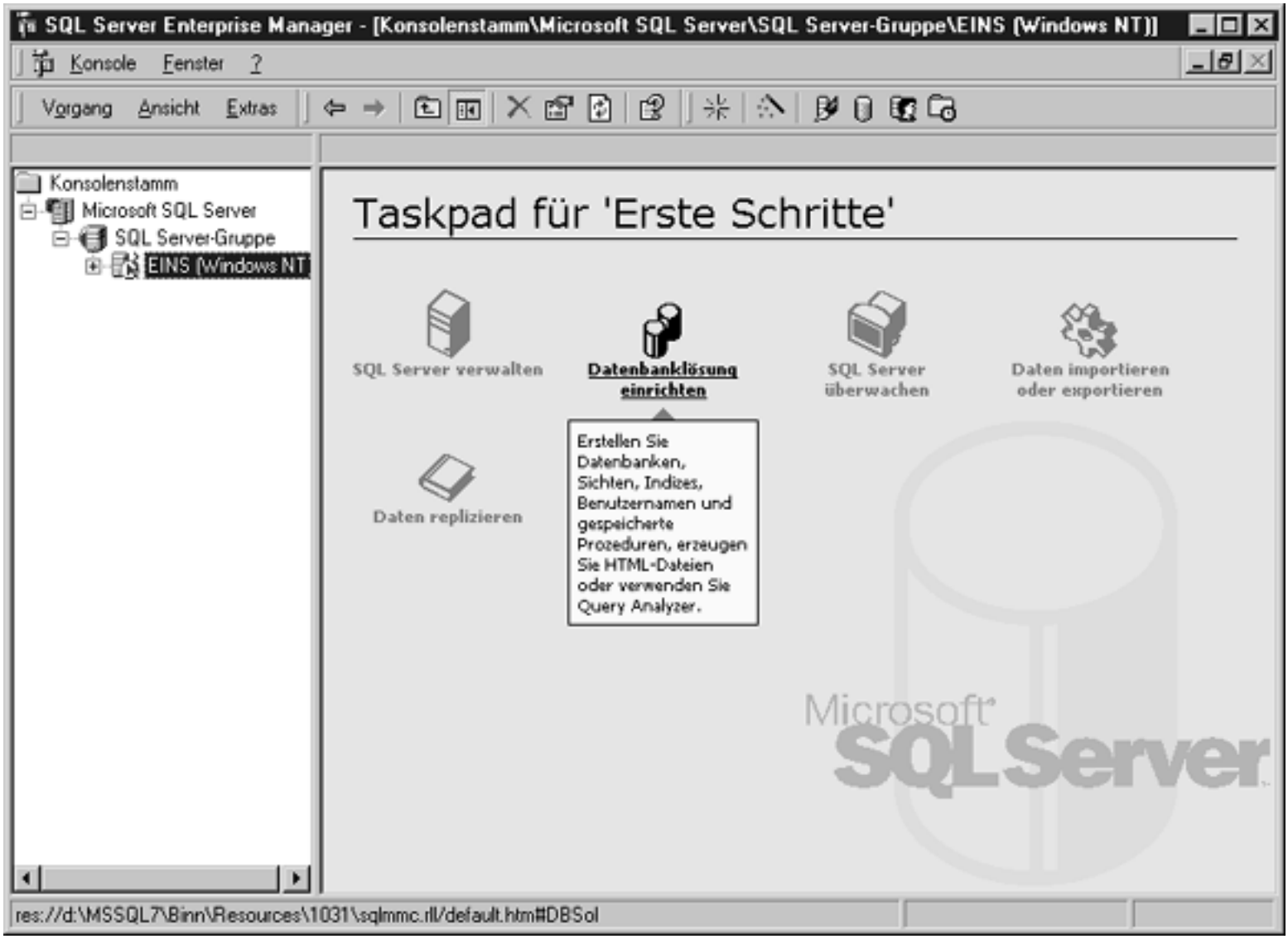
mit diesem dienstprogramm läßt sich unter anderem der zeichensatz, die sortierreihenfolge oder die unicode-sortierreihenfolge wechseln. vor allem aber kann man die master-datenbank bei einem systemabsturz neu erstellen. darauf geht kapitel 7 ausführlich ein.

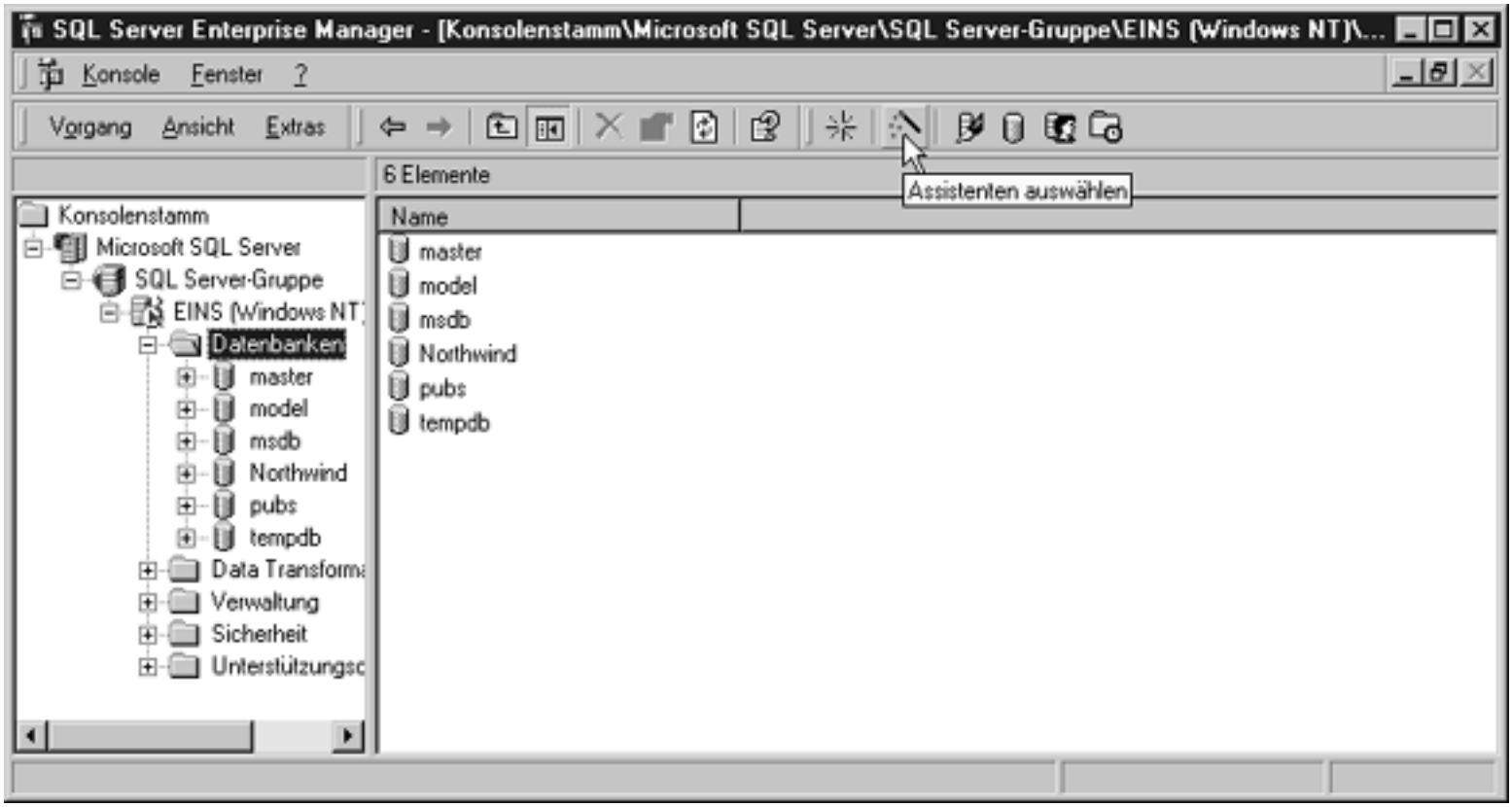
3.7.7 sqlmaint

das dienstprogramm sqlmaint führt für eine oder mehrere datenbanken eine gruppe bestimmter wartungsoperationen aus. kapitel 20 geht im rahmen der wartung auf dieses dienstprogramm ein.

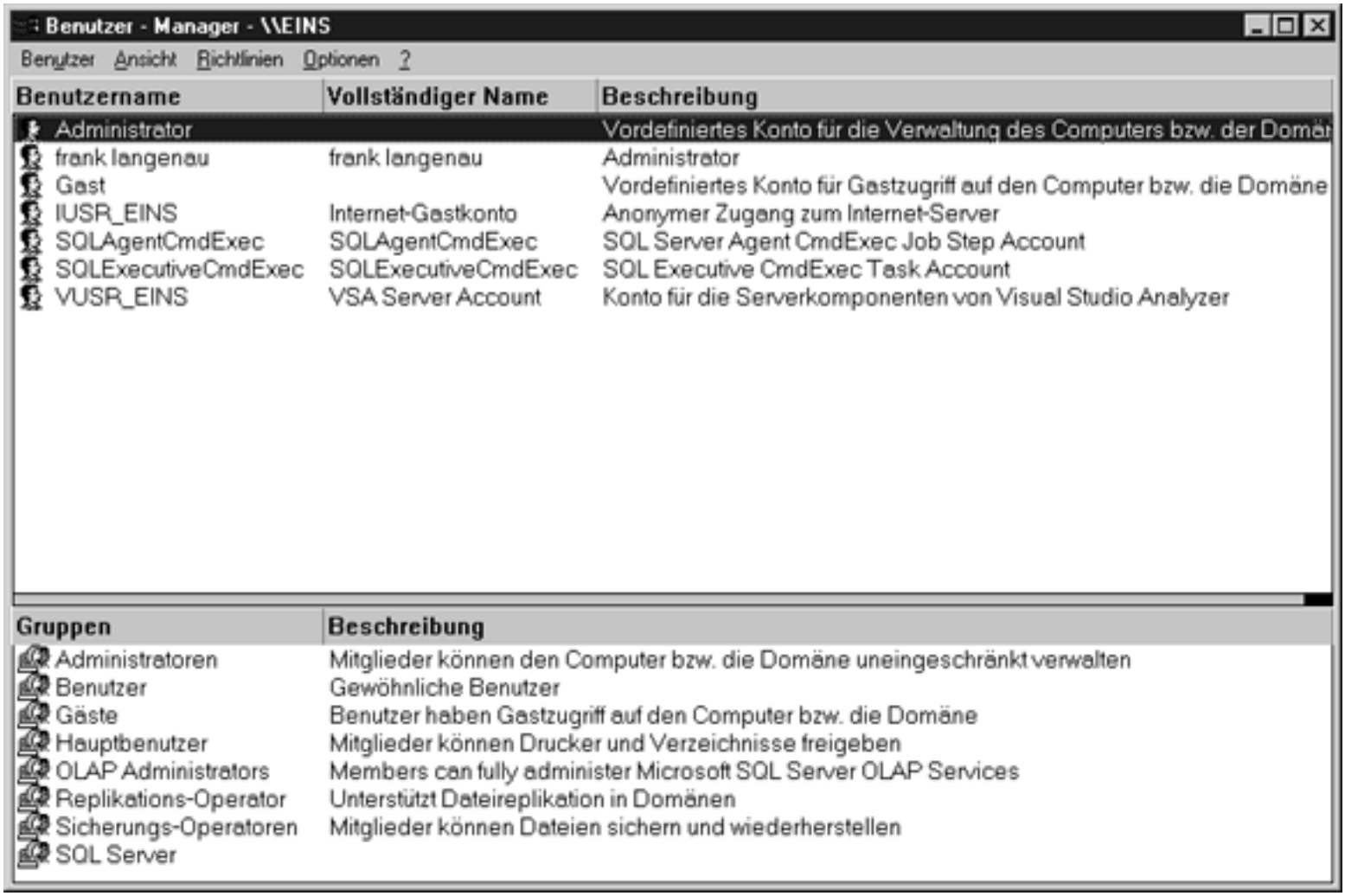
3.7.8 sqlservr

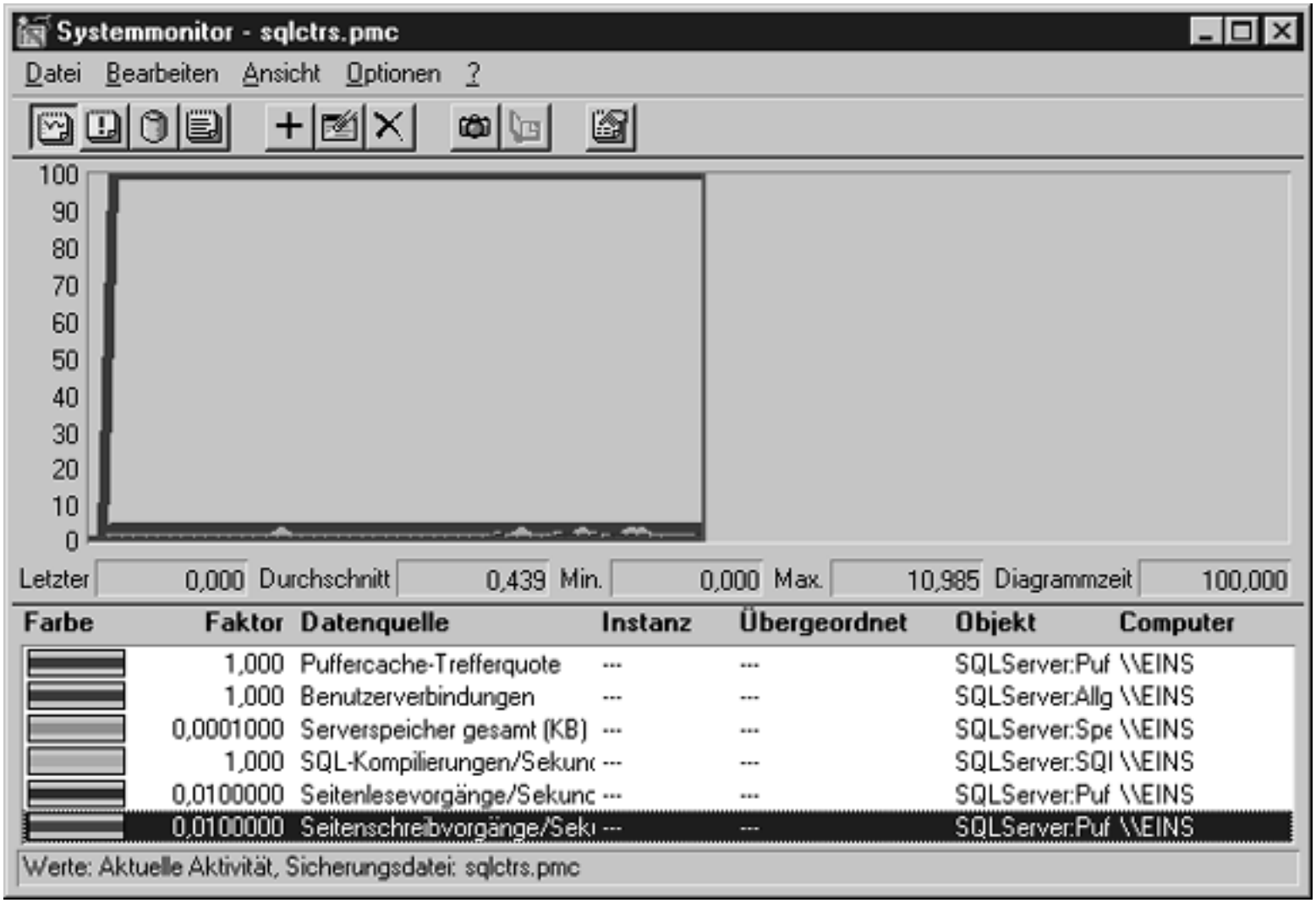
mit diesem dienstprogramm können sie sql server beispielsweise bei der fehlersuche von der befehlszeile aus starten, wie es bereits in kapitel 2 gezeigt wurde.

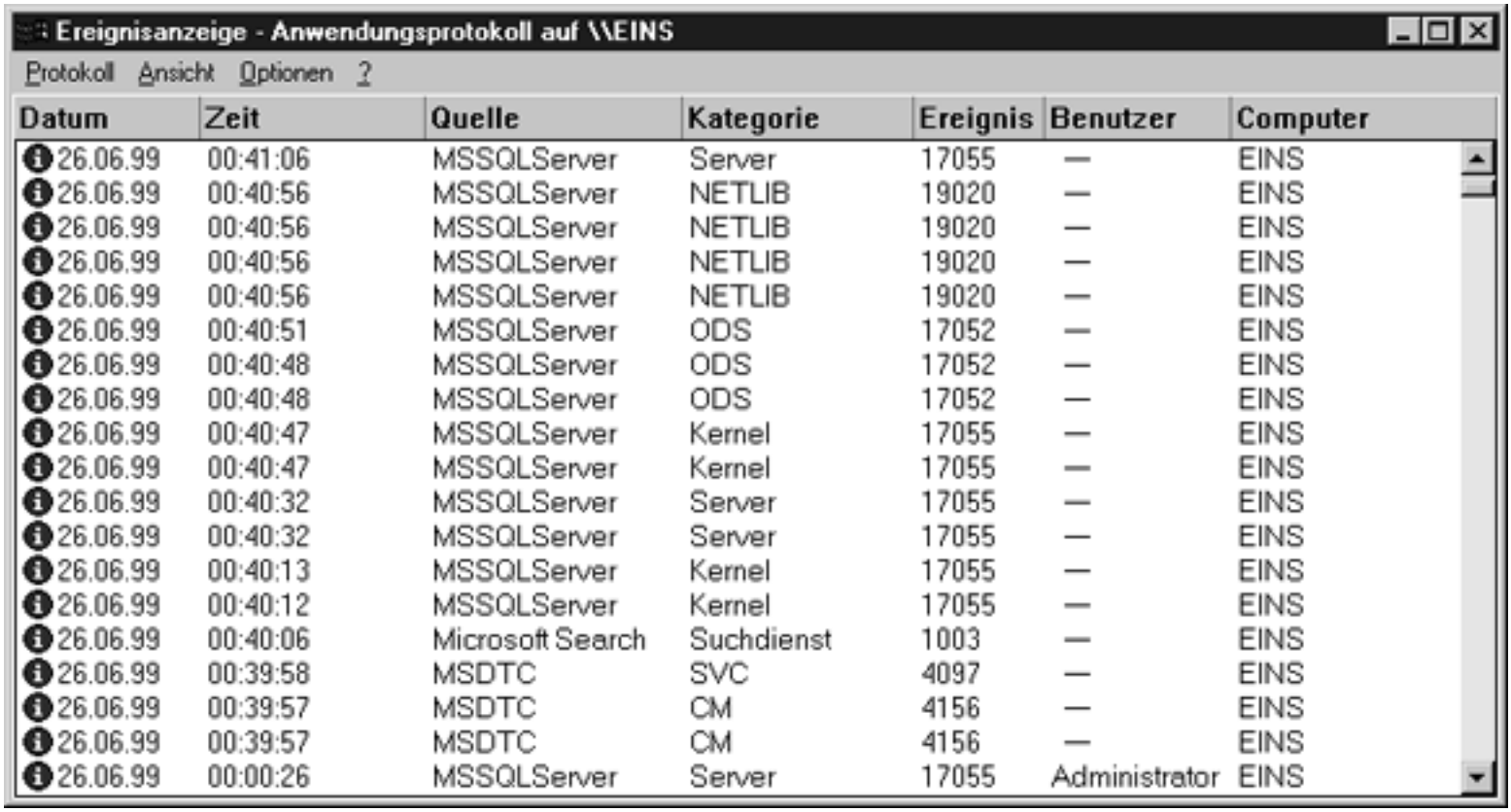

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)]
Konsole
Fenster
?
Vorgang
Ansicht
Extras
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT
Taskpad für 'Erste Schritte'
SQL Server verwalten
Datenbanklösung einrichten
SQL Server überwachen
Daten importieren oder exportieren
Daten replizieren
Erstellen Sie Datenbanken, Sichten, Indizes, Benutzernamen und gespeicherte Prozeduren, erzeugen Sie HTML-Dateien oder verwenden Sie Query Analyzer.
Microsoft SQL Server
res://d:\MSSQL7\Binn\Resources\1031\sqlmmc.rll/default.htm#DBSol


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
Assistenten auswählen
6 Elemente
Name
master
model
msdb
Northwind
pubs
tempdb
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
master
model
msdb
Northwind
pubs
tempdb
Data Transforma
Verwaltung
Sicherheit
Unterstützungsc


SQL Server-Aktualisierungs-Assistent
Willkommen beim SQL Server-Aktualisierungs-Assistenten
Dieser Assistent dient zur Übertragung Ihrer Serverkonfiguration und Ihrer Datenbanken von einem einzelnen 6.x-SQL Server zu der auf diesem Computer installierten Kopie von SQL Server 7.0.
Während dieses Prozesses muss diese Anwendung sowohl die SQL Server-Installation der Version 6.x als auch der Version 7.0 beenden und erneut starten. Bevor Sie fortfahren, sollten Sie die Themen zur Aktualisierung in der Onlinedokumentation sorgfältig lesen.
Klicken Sie auf 'Weiter', um fortzufahren.
Hilfe
Abbrechen
< Zurück
Weiter >
Fertig stellen


Benutzer - Manager - \\EINS
Benutzer Ansicht Richtlinien Optionen ?
Benutzername
Vollständiger Name
Beschreibung
Administrator
Vordefiniertes Konto für die Verwaltung des Computers bzw. der Domär
frank langenau
frank langenau
Administrator
Gast
Vordefiniertes Konto für Gastzugriff auf den Computer bzw. die Domäne
IUSR_EINS
Internet-Gastkonto
Anonymer Zugang zum Internet-Server
SQLAgentCmdExec
SQLAgentCmdExec
SQL Server Agent CmdExec Job Step Account
SQLExecutiveCmdExec
SQLExecutiveCmdExec
SQL Executive CmdExec Task Account
VUSR_EINS
VSA Server Account
Konto für die Serverkomponenten von Visual Studio Analyzer
Gruppen
Beschreibung
Administratoren
Mitglieder können den Computer bzw. die Domäne uneingeschränkt verwalten
Benutzer
Gewöhnliche Benutzer
Gäste
Benutzer haben Gastzugriff auf den Computer bzw. die Domäne
Hauptbenutzer
Mitglieder können Drucker und Verzeichnisse freigeben
OLAP Administrators
Members can fully administer Microsoft SQL Server OLAP Services
Replikations-Operator
Unterstützt Dateireplikation in Domänen
Sicherungs-Operatoren
Mitglieder können Dateien sichern und wiederherstellen
SQL Server


Systemmonitor - sqlctrs.pmc
Datei Bearbeiten Ansicht Optionen ?
100
90
80
70
60
50
40
30
20
10
0
Letzter 0,000 Durchschnitt 0,439 Min. 0,000 Max. 10,985 Diagrammzeit 100,000
Farbe Faktor Datenquelle Instanz Übergeordnet Objekt Computer
1,000 Puffercache-Trefferquote --- --- SQLServer:Puf \\EINS
1,000 Benutzerverbindungen --- --- SQLServer:Allg \\EINS
0,0001000 Serverspeicher gesamt (KB) --- --- SQLServer:Spe \\EINS
1,000 SQL-Kompilierungen/Sekunc --- --- SQLServer:SQI \\EINS
0,0100000 Seitenlesevorgänge/Sekunc --- --- SQLServer:Puf \\EINS
0,0100000 Seitenschreibvorgänge/Seku --- --- SQLServer:Puf \\EINS
Werte: Aktuelle Aktivität, Sicherungsdatei: sqlctrs.pmc



Datum	Zeit	Quelle	Kategorie	Ereignis	Benutzer	Computer
26.06.99	00:41:06	MSSQLServer	Server	17055	—	EINS
26.06.99	00:40:56	MSSQLServer	NETLIB	19020	—	EINS
26.06.99	00:40:56	MSSQLServer	NETLIB	19020	—	EINS
26.06.99	00:40:56	MSSQLServer	NETLIB	19020	—	EINS
26.06.99	00:40:56	MSSQLServer	NETLIB	19020	—	EINS
26.06.99	00:40:51	MSSQLServer	ODS	17052	—	EINS
26.06.99	00:40:48	MSSQLServer	ODS	17052	—	EINS
26.06.99	00:40:48	MSSQLServer	ODS	17052	—	EINS
26.06.99	00:40:47	MSSQLServer	Kernel	17055	—	EINS
26.06.99	00:40:47	MSSQLServer	Kernel	17055	—	EINS
26.06.99	00:40:32	MSSQLServer	Server	17055	—	EINS
26.06.99	00:40:32	MSSQLServer	Server	17055	—	EINS
26.06.99	00:40:13	MSSQLServer	Kernel	17055	—	EINS
26.06.99	00:40:12	MSSQLServer	Kernel	17055	—	EINS
26.06.99	00:40:06	Microsoft Search	Suchdienst	1003	—	EINS
26.06.99	00:39:58	MSDTC	SVC	4097	—	EINS
26.06.99	00:39:57	MSDTC	CM	4156	—	EINS
26.06.99	00:39:57	MSDTC	CM	4156	—	EINS
26.06.99	00:00:26	MSSQLServer	Server	17055	Administrator	EINS

kapitel 4 daten und objekte

4.1 tabellen, sichten, indizes & co.

in sql server treffen sie ständig auf den begriff *objekt*. dabei handelt es sich aber nicht um objekte im sinne der objektorientierten programmierung, sondern um die komponenten einer datenbank. auf der einen seite gibt es objekte, die physisch in der datenbank vorhanden - d.h. gespeichert - sind, auf der anderen seite objekte, die sql server nur temporär anlegt. dieses kapitel zeigt, wie sich die datenbankobjekte von sql server in das relationale datenbankmodell einordnen, welche fragen in diesem zusammenhang beim datenbankdesign zu beachten sind und wo die objekte von sql server gespeichert werden.

4.2 logische datenbankkomponenten

die daten in einer sql-server-datenbank sind in *objekten* organisiert. einen überblick über die elemente einer datenbank erhalten sie, wenn sie im enterprise manager die konsolenstruktur bis zu einer datenbank erweitern. abbildung 4.1 zeigt als beispiel den erweiterten ordner der datenbank pubs. zu den hauptkomponenten einer datenbank gehören:

- diagramme
- tabellen
- sichten
- gespeicherte prozeduren
- db-benutzernamen
- rollen
- regeln
- standards
- benutzerdefinierte datentypen
- volltextkataloge

auf diese komponenten hat der benutzer zugriff, wenn er eine verbindung zur datenbank herstellt.

bild

abbildung 4.1: elemente einer datenbank im enterprise manager

zu den datenbankobjekten zählen laut definition tabellen, indizes, trigger, sichten, schlüssel, einschränkungen, standardwerte, regeln, benutzerdefinierte datentypen und gespeicherte prozeduren. in abbildung 4.1 sind also nicht alle datenbankobjekte zu sehen. das hängt unter anderem damit zusammen, daß sich bestimmte objekte nicht unabhängig von anderen erstellen oder ändern lassen, zum beispiel:

- cursor
- primärschlüssel
- fremdschlüssel
- indizes
- trigger
- bedingungen
- einschränkungen
- sperren

die folgenden abschnitte geben einen überblick zu den einzelnen komponenten bzw. objekten und nennen die kapitel, die sich näher damit beschäftigen.

datenbankdiagramme

mit datenbankdiagrammen lassen sich datenbankobjekte in grafischer form anzeigen, erstellen und verwalten. abbildung 4.2 zeigt das beispiel eines datenbankdiagramms, das für die datenbank pubs erstellt wurde. kapitel 6 zeigt die vielfältigen möglichkeiten, die sich mit datenbankdiagrammen ergeben.

Bild

abbildung 4.2: datenbankdiagramm für die beispieldatenbank pubs

tabellen

sämtliche informationen einer datenbank sind in tabellen gespeichert. tabellen bestehen aus spalten und zeilen. die eigenschaften der in den tabellen abgelegten daten werden durch metadaten beschrieben, die ihrerseits in systemtabellen von sql server gespeichert sind. dazu gehören zum beispiel angaben zur struktur der daten bzw. tabellen oder den datentypen der spalten. abbildung 4.3 zeigt die tabelle publishers der datenbank pubs. dem thema tabellen widmet sich kapitel 8.

3:Daten in Tabelle 'publishers'

pub_id	pub_name	city	state	country
0736	New Moon Books	Boston	MA	USA
0877	Binnet & Hardley	Washington	DC	USA
1389	Algodata Infosyste	Berkeley	CA	USA
1622	Five Lakes Publishir	Chicago	IL	USA
1756	Ramona Publishers	Dallas	TX	USA
9901	GGG&G	M□nchen	<NULL>	Germany
9952	Scootney Books	New York	NY	USA
9999	Lucerne Publishing	Paris	<NULL>	France

abbildung 4.3: die tabelle publishers der datenbank pubs

sichten

eine sicht bietet den blick auf eine oder mehrere tabellen. man spricht daher auch von einer *virtuellen tabelle*. im grunde handelt es sich um eine gespeicherte abfrage, wobei nicht die sicht selbst - d.h. die ergebnismenge - als eigenständiges objekt, sondern lediglich die zugrundeliegende select-anweisung in der datenbank gespeichert wird. mit sichten kann man ausgewählte daten aus mehreren tabellen bereitstellen, ohne daß man die meist komplexe select-anweisung jedesmal neu formulieren muß. sichten bieten sich auch als sicherheitsmechanismus an. beispielsweise kann man bestimmte spalten von der ergebnismenge ausschließen oder änderungen an den zugrundeliegenden tabellen verbieten.

gespeicherte prozeduren

transact-sql-anweisungen lassen sich wie in höheren programmiersprachen zu einer prozedur zusammenfassen und wie eine einzige anweisung ausführen. dazu ruft man die prozedur unter ihrem namen auf. gespeicherte prozeduren können parameter übernehmen und werte zurückgeben. echte rückgabewerte, wie sie funktionen in höheren programmiersprachen liefern, sind bei gespeicherten prozeduren allerdings nicht möglich. da sql server die gespeicherten prozeduren zu einem ausführungsplan kompiliert, läuft eine gespeicherte prozedur in der regel schneller ab als die einzelnen in der prozedur enthaltenen transact-sql-anweisungen. kapitel 12 behandelt gespeicherte prozeduren und geht auch auf die steuerungsstrukturen ein, die man zwar in jedem skript verwenden kann, vor allem aber in prozeduren einsetzt.

trigger

trigger sind spezielle gespeicherte prozeduren, die sql server beim hinzufügen, ändern oder löschen von daten automatisch aufruft. mit triggern lassen sich zum beispiel geschäftsregeln durchsetzen oder abläufe automatisieren. näheres zu triggern finden sie in kapitel 17.

db-benutzernamen

benutzernamen regeln den zugriff auf eine sql server-datenbank. der db-(datenbank-)benutzername identifiziert einen benutzer innerhalb einer datenbank. kapitel 21 geht im rahmen der sicherheit ausführlich auf db-benutzernamen ein. beachten sie dort auch den hinweis zur terminologie von benutzer, benutzername und benutzerkonto.

rollen

rollen fassen mehrere benutzer mit gleichen berechtigungen zusammen und sind vergleichbar mit den gruppen im sicherheitsmechanismus von windows nt. die sicherheitssysteme von sql server und windows nt sind prinzipiell eigenständige einrichtungen, auch wenn es viele gemeinsamkeiten und verflechtungen gibt. die sicherheit ist das thema von kapitel 21.

regeln

wenn die werte einer spalte bestimmten bedingungen genügen müssen, kann man diese bedingungen in

regeln festlegen. allerdings sind regeln in sql server 7.0 aus gründen der abwärtskompatibilität vorhanden und wurden durch die leistungsfähigeren check-einschränkungen ersetzt. regeln sind separate objekte, die erst an eine oder mehrere spalten gebunden werden müssen. mit regeln kann man zum beispiel sicherstellen, daß eine spalte für telefonnummern tatsächlich nur ziffern und gegebenenfalls klammern für die vorwahl enthält. kapitel 16 geht im rahmen der datenintegrität auf regeln ein.

standards

für die spalten einer tabelle kann man werte festlegen, die sql server automatisch einsetzt, wenn der benutzer beim einfügen von daten keinen wert für die betreffende spalte bereitstellt. standardwerte behandelt kapitel 16.

benutzerdefinierte datentypen

mit benutzerdefinierten datentypen lassen sich einheitliche definitionen für gleichartige spalten in verschiedenen tabellen realisieren. wenn ein benutzerdefinierter datentyp für eine bestimmte datenbank erstellt wird, ist er nur in dieser datenbank gültig. damit ein benutzerdefinierter datentyp für alle neu zu erstellenden datenbanken verfügbar ist, müssen sie ihn in der systemdatenbank model erstellen. datentypen im allgemeinen und benutzerdefinierte datentypen im speziellen sind vor allem bei der definition der spalten in tabellen von bedeutung und werden deshalb in kapitel 8 besprochen.

volltextkataloge

ein volltextkatalog faßt die volltextindizes der tabellen einer datenbank zusammen. volltextindizes erlauben die schnelle suche nach zeichendaten. diese funktionalität ist neu in der version 7.0 von sql server. die volltextsuche wird von einem unabhängigen modul - dem microsoft-search-dienst - realisiert. der dienst steht nur unter windows nt server zur verfügung und gehört nicht zur standardinstallation. kapitel 14 geht im rahmen des themas indizes auch auf die volltextindizes ein.

4.3 datenbankentwurf

mit der entscheidung für sql server steht quasi auch das zugrundeliegende modell fest - das relationale datenbankmodell. damit ist auch klar, daß sämtliche daten in tabellen zu speichern sind. von diesem punkt an liegt der eigentliche entwurf einer datenbank im verantwortungsbereich des datenbankentwicklers.

dieses buch beschäftigt sich in erster linie mit den instrumenten von sql server und kann deshalb schon aus platzgründen keine umfassende anleitung für einen guten datenbankentwurf bieten. zu diesem thema sei auf die einschlägige literatur verwiesen.

ein paar grundsätzliche punkte sollen dennoch angerissen werden: bevor sie eine datenbank erstellen, müssen sie unter anderem folgende fragen klären:

- welche ziele werden mit der datenbank verfolgt?
- wie sieht der vorgesehene benutzerkreis aus?
- wie erfolgt der zugriff auf die daten?

- welche daten sind in der datenbank zu speichern?
- welche geschäftsregeln sind zu realisieren?
- welche sicherheitsaspekte sind zu berücksichtigen?
- wer darf welche daten abrufen, ändern, löschen?
- sollen die daten im internet veröffentlicht werden?
- wie lassen sich die daten effizient bereitstellen?

wie bereits kapitel 1 erwähnt hat, steht der benutzer der datenbank im mittelpunkt. deshalb sollten sie beim entwurf einer datenbank immer die wünsche der benutzer im auge behalten und sich in die verschiedenen szenarien des datenbankzugriffs hineinversetzen können.

der benutzer muß schnell, problemlos und direkt auf die daten zugreifen können. überflüssige informationen sind von vornherein auszublenden. wenn etwa ein mitarbeiter der abteilung telefonmarketing eine kurze information an alle kunden weitergeben muß und somit nur an den namen und telefonnummern der kunden interessiert ist, genügt es, die entsprechenden spalten der kundentabelle abzurufen und bereitzustellen. in diesem fall ist es nicht angebracht, vollständige kundendatensätze mit kompletten adressen und weiteren angaben zu übertragen. einerseits würde das die netzwerkbelastung unnötig erhöhen, andererseits hätte der mitarbeiter die für ihn interessanten daten nicht auf einen blick zur verfügung und müßte sie praktisch aus einer zu großen liste mit vielen nebensächlichen informationen herausfiltern.

wenn sie die antworten zu den eingangs aufgeworfenen fragen zusammengetragen haben, steht als nächstes die konzeptionelle gestaltung der datenbank - der logische datenbankentwurf - auf der tagesordnung. dazu müssen sie klären, wo und in welcher form die daten zu speichern sind. hier interessiert in erster linie die aufteilung der daten auf tabellen sowie die datentypen der spalten in diesen tabellen. kommt alles in eine tabelle, oder sollen es mehrere sein? das zauberwort heißt normalisierung.

4.3.1 normalisierung

relationale datenbank-managementsysteme speichern alle informationen in tabellen, die miteinander in beziehung stehen. dabei sollen redundante informationen möglichst vermieden werden. das läßt sich erreichen, indem man umfangreiche tabellen in mehrere kleinere tabellen aufteilt.

angenommen, eine datenbank speichert kundendaten und bestellungen. ein kunde löst aber (hoffentlich) nicht nur eine bestellung aus. wenn man in ein und derselben tabelle sowohl kundendaten als auch bestellungen speichert, werden die kundendaten mit jeder bestellung wiederholt. im zuge der normalisierung teilt man die tabelle in zwei tabellen auf: eine tabelle für bestellungen und eine für kundendaten. in der tabelle für bestellungen schreibt man statt der vollständigen angaben zum kunden lediglich einen verweis auf die kundentabelle - ein gutes beispiel dafür ist die verwendung einer kundennummer.

das programm, das den lieferschein bzw. die rechnung druckt, holt dann die angaben zur bestellung aus der tabelle bestellungen, ermittelt aus dem feld kundennummer den verweis auf die kundentabelle und holt schließlich die kundendaten mit name und lieferanschrift aus der kundentabelle.

die tabelle bestellungen wird wahrscheinlich artikelbezeichnungen, mengenangaben und ähnliches enthalten. wenn verschiedene kunden die gleichen artikel bestellen, steht in jeder tabelle bestellungen die

vollständige artikelbezeichnung. deshalb legt man auch hier nur eine artikelnummer ab, die auf eine dritte tabelle verweist, in der die artikelbeschreibungen gespeichert sind.

die aufteilung von datensätzen auf mehrere tabellen bezeichnet man als *normalisierung*. dabei unterscheidet man verschiedene ebenen der aufteilung - die normalformen. die niedrigste ebene der normalisierung wird durch die 1. normalform beschrieben. alle darauf folgenden normalformen schließen die bedingungen aller niedrigeren normalformen ein. in der regel begnügt man sich mit der 3. normalform, denn eine weitere aufgliederung führt eher zu entwurfsproblemen als zu funktionellen vorteilen.

die folgenden abschnitte zeigen anhand einer fiktiven datenbank immobilien, wie man eine tabelle normalisieren kann. ausgangspunkt ist die in abbildung 4.4 wiedergegebene tabelle.

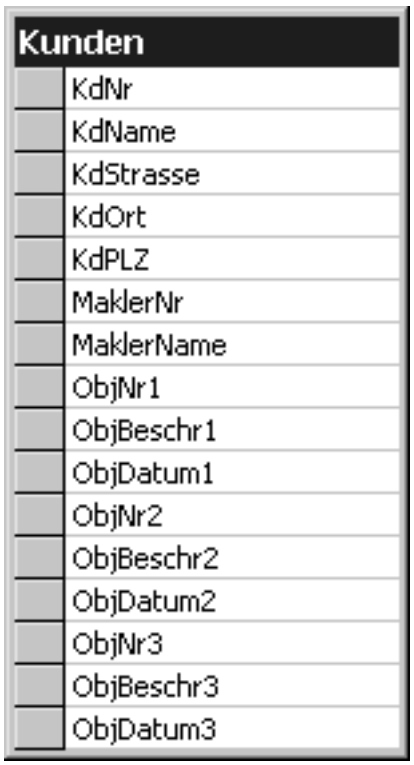


abbildung 4.4: nicht normalisierte tabelle

die tabelle kunden der datenbank immobilien enthält folgende spalten:

- kdnr: eindeutige nummer für einen kaufinteressenten
- kdname: name des kunden
- kdstrasse: anschrift des kunden (straße und hausnummer)
- kdort: wohnort des kunden
- kdplz: postleitzahl für den wohnort
- maklernr: eindeutige nummer für den makler, der den kunden betreut
- maklername: name des maklers
- objnr1: eindeutige nummer eines objekts, für das sich der kunde interessiert
- objbeschr1: beschreibung des objekts
- objdatum1: datum, an dem eine reservierung vorgenommen wurde

- objnr2, objbeschr2, objdatum2, objnr3, objbeschr3, objdatum3: objektangaben für zwei weitere objekte

1. normalform

die erste normalform verlangt, daß eine tabelle keine wiederholten gruppen und keine doppelten zeilen enthält. in der - bisher einzigen - tabelle der datenbank immobilien sind aber derartige gruppen vorhanden: die spalten objnr, objbeschr und objdatum tauchen als gruppe dreimal auf.

um die tabelle in die erste normalform zu bringen, lagert man die objektdaten in eine eigene tabelle aus. jede tabelle erhält einen primärschlüssel, der jede zeile einer tabelle eindeutig identifiziert, so daß keine doppelten zeilen auftreten können. wenn sich das nicht mit einer einzigen spalte realisieren läßt, schafft ein aus mehreren spalten zusammengesetzter primärschlüssel abhilfe. enthält die tabelle nur daten, aus denen kein eindeutiger primärschlüssel gebildet werden kann, ist eine zusätzliche spalte mit einem eindeutigen wert einzufügen.

abbildung 4.5 zeigt, wie die tabelle kunden aufgeteilt wurde.

abbildung 4.5: tabellen, die der 1. normalform entsprechen

in der tabelle vormerkungen sind jetzt die objektbeschreibungen untergebracht. die spalte kdnr dient als verweis auf die tabelle kunden.

2. normalform

die zweite normalform verlangt, daß keine spalte funktional nur von einem teil des (zusammengesetzten) primärschlüssels abhängig ist. eine spalte b ist von einer spalte a funktional abhängig, wenn es zu jedem wert von a genau einen wert von b gibt.

in der tabelle vormerkungen ist die spalte objnr nur von objnr und nicht von kdnr abhängig. wenn man annimmt, daß sich der primärschlüssel aus den spalten kdnr und objnr zusammensetzt, haben wir es mit einer verletzung der 2. normalform zu tun: die spalte objnr ist nur von einem teil des primärschlüssels

abhängig.

verschieben sie also die objektbeschreibung in eine separate tabelle objekte. die beziehung zwischen der neuen tabelle objekte und der tabelle vormerkungen wird über die spalte objnr hergestellt. abbildung 4.6 zeigt die beiden tabellen, die jetzt der 2. normalform entsprechen.

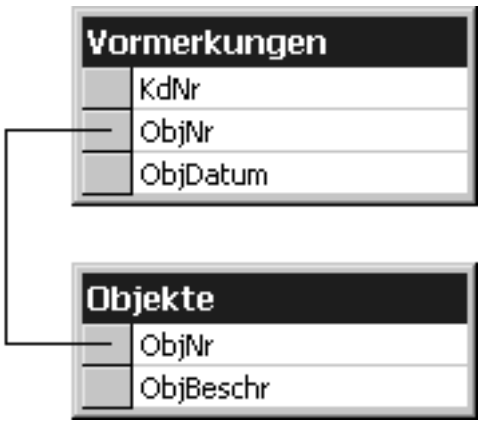


abbildung 4.6: tabellen, die der 2. normalform entsprechen

3. normalform

bei einer tabelle nach der dritten normalform dürfen keine spalten von beliebigen anderen nicht zum schlüssel gehörenden spalten abhängig sein.

in der tabelle kunden nach abbildung 4.5 hängt aber die spalte maklername nicht von kdnr (dem primärschlüssel der tabelle), sondern von maklernr ab. verschieben sie daher die spalte maklername in eine eigene tabelle makler, und stellen sie den bezug zur tabelle kunden über die spalte maklernr her. die resultierenden tabellen zeigt abbildung 4.7.

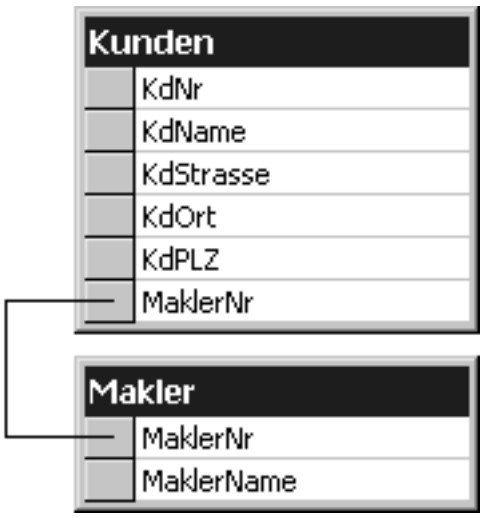


abbildung 4.7: tabellen, die der 3. normalform entsprechen

nach diesen schritten ist die normalisierung der datenbank immobilien abgeschlossen. das ergebnis zeigt abbildung 4.8.

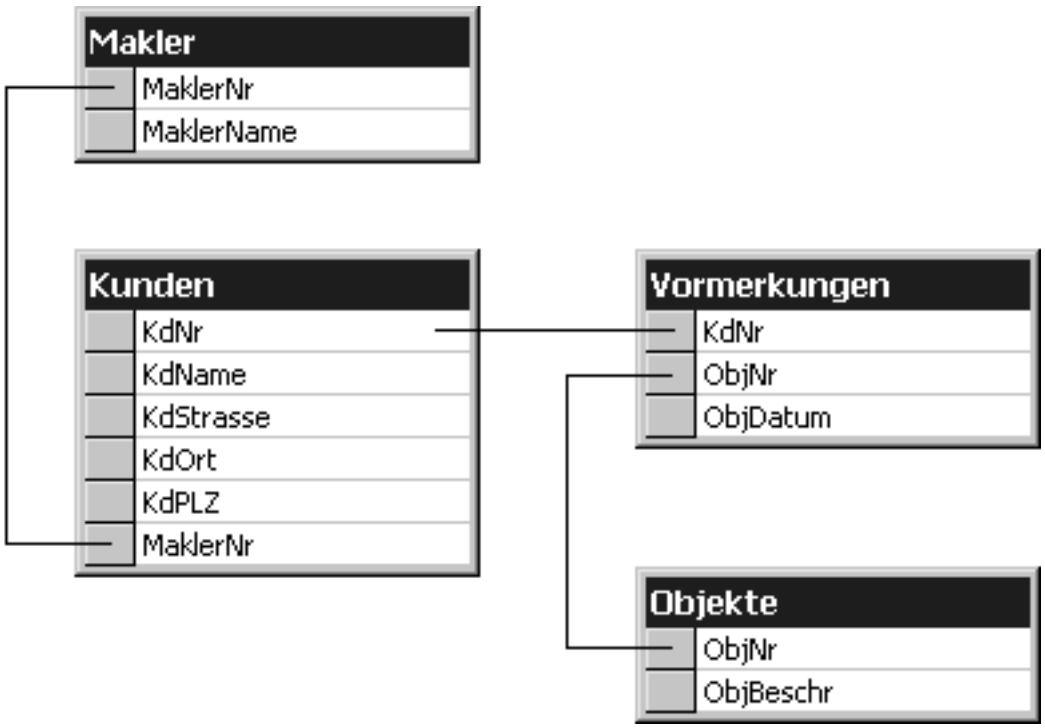


abbildung 4.8: die datenbank immobilien in normalisierter form

übernormalisierung

man kann die normalisierung auch zu weit treiben. wenn etwa in einer tabelle für kundenadressen eine spalte für telefonnummern vorgesehen ist, wird man in der regel die telefonnummer inklusive vorwahl eingeben. man kann natürlich auch noch die telefonnummer in getrennten tabellen - eine für die vorwahlen und eine andere für die eigentlichen rufnummern - speichern. das ist aber wenig sinnvoll und führt letztendlich nur zu unnötig komplexen abfrageanweisungen.

4.3.2 denormalisierung

bei einem gut durchdachten datenbankentwurf sollte man die normalisierung nach der 3. normalform anstreben. es gibt aber fälle, in denen eine straffe normalisierung nachteilig ist, beispielsweise wenn die abfrageleistung absolute priorität hat oder wenn man die bereitstellung der daten für berichte vereinfachen will.

wenn sie zur 2. oder sogar 1. normalform zurückkehren, müssen sie auch bereit sein, auf die strukturellen vorteile einer vollständigen normalisierung zu verzichten. man sollte diesen weg also nur in ausnahmefällen beschreiten.

wenn große datenmengen abzufragen sind und die daten aus mehreren unterschiedlichen tabellen kommen, müssen entsprechend viele tabellenverknüpfungen hergestellt werden. das schlägt sich in der erforderlichen rechenzeit nieder und kann auch eine stärkere beanspruchung der datenträgerein-/-ausgaben - mithin eine höhere belastung der ressourcen - bedeuten. in diesem fall ist eine denormalisierung zu empfehlen.

4.4 physische datenbankkomponenten

4.4.1 daten- und protokolldateien

sql server speichert eine datenbank in mindestens zwei dateien: in der einen datei stehen die eigentlichen daten, in der anderen ein protokoll über die änderungen an der datenbank. je nach bedarf kann man die komplette datenbank in nur einer datendatei und einer protokolldatei unterbringen oder auf mehrere datendateien und protokolldateien aufteilen. sql server unterscheidet dabei zwischen primärer datendatei und sekundären datendateien. jede datenbank verfügt über eine primäre datendatei, die den ausgangspunkt der datenbank bildet und gegebenenfalls auf sekundäre datendateien verweist. im systemkatalog (siehe dazu kapitel 19) ist lediglich der pfad zur primären datei verzeichnet.

sql server unterscheidet zwischen *logischen dateinamen* und *betriebssystemdateinamen*. über die logischen dateinamen greift man in transact-sql-anweisungen auf die datenbankdateien zu. die betriebssystemdateinamen sind für den benutzer in der regel nicht relevant und werden von sql server intern verwaltet. wenn sie allerdings datenbanksicherungen durchführen und dabei betriebssystemdateien kopieren oder verschieben müssen, ist die kenntnis der betriebssystemdateinamen erforderlich. auf diesen punkt geht kapitel 7 ein.

das *transaktionsprotokoll* spiegelt die geschichte der datenveränderungen an einer datenbank wider. wenn man eine datenbank erstellt, legt sql server gleichzeitig ein zugehöriges transaktionsprotokoll (mit der erweiterung .ldf) an. auf transaktionsprotokolle geht kapitel 18 näher ein.

4.4.2 systemdatenbanken

für die funktionsweise von sql server selbst sind die vier systemdatenbanken master, model, tempdb und msdb mit folgenden aufgaben vorgesehen:

- master: speichert alle systeminformationen von sql server. dazu gehören anmeldekonten, einstellungen der systemkonfiguration, verzeichnis aller benutzerdatenbanken mit speicherort der primären dateien und initialisierungsinformationen für sql server.
- model: diese datenbank dient als vorlage für alle neu zu erstellenden datenbanken. in der datenbank model sind zum beispiel auch die global verfügbaren benutzerdefinierten datentypen untergebracht.
- tempdb: diese datenbank dient als zwischenspeicher für alle tabellen und gespeicherten prozeduren, die nur temporär existieren. beim start von sql server ist tempdb leer und weist die festgelegte standardgröße auf. wenn sie sql server beenden, gehen alle inhalte der datenbank tempdb verloren. die datenbank tempdb zeichnet also keine informationen von einer sql-server-sitzung zur nächsten auf.
- msdb: in der datenbank msdb zeichnet der sql-server-agent die geplanten termine und aufträge sowie die operatoren auf.

4.4.3 seiten und blöcke (extents)

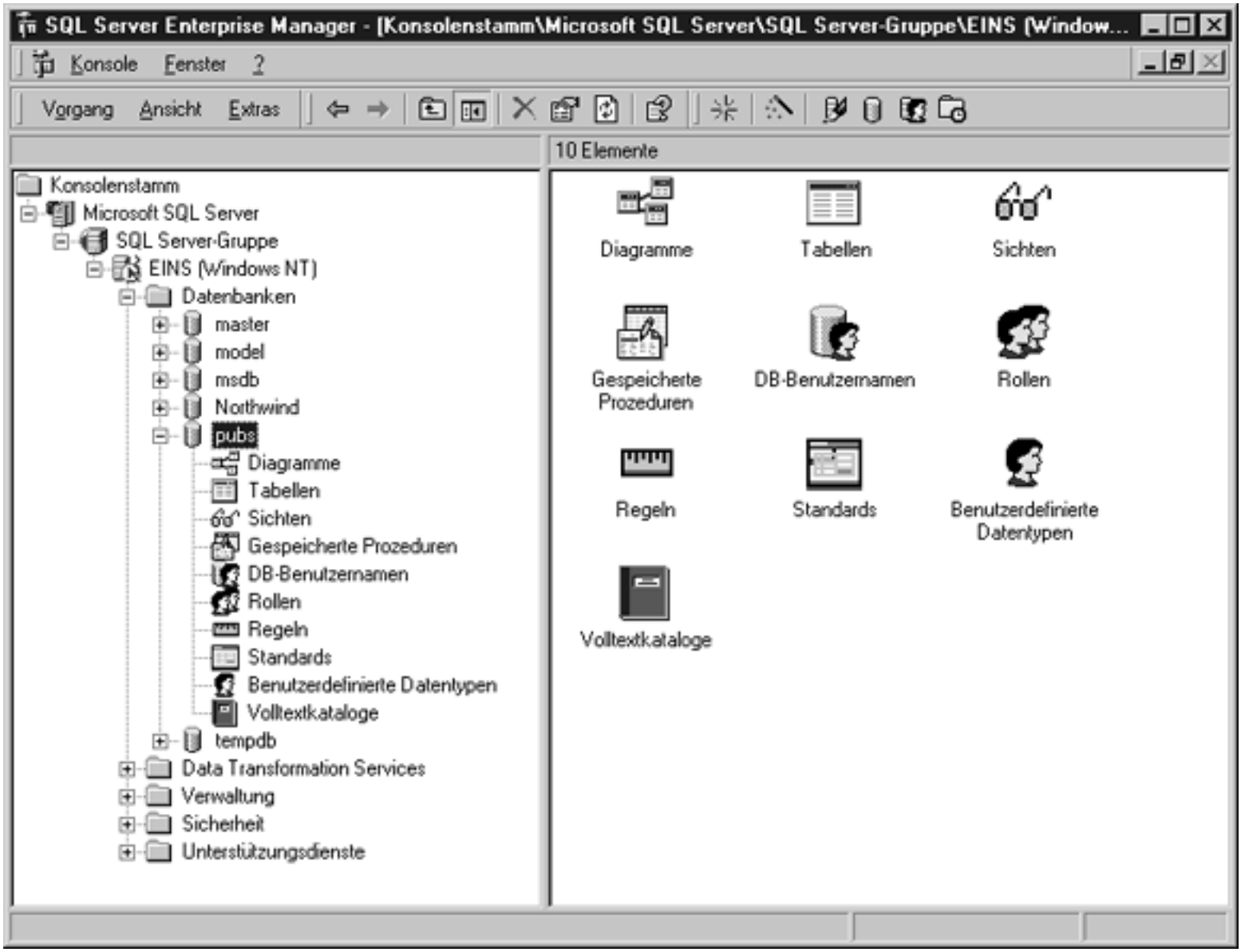
die grundlegende speichereinheit in microsoft sql server ist die seite mit einem umfang von 8 kbyte. eine aufeinanderfolgende gruppe von 8 daten- oder indexseiten bildet einen block (extent). von den 8192 byte einer seite verwendet sql server 96 byte für den header der seite und 36 byte für interne datenstrukturen, so daß 8060 nutzbytes verfügbar sind. das ist gleichzeitig die maximale größe einer einzelnen zeile (ohne text- und bilddaten).

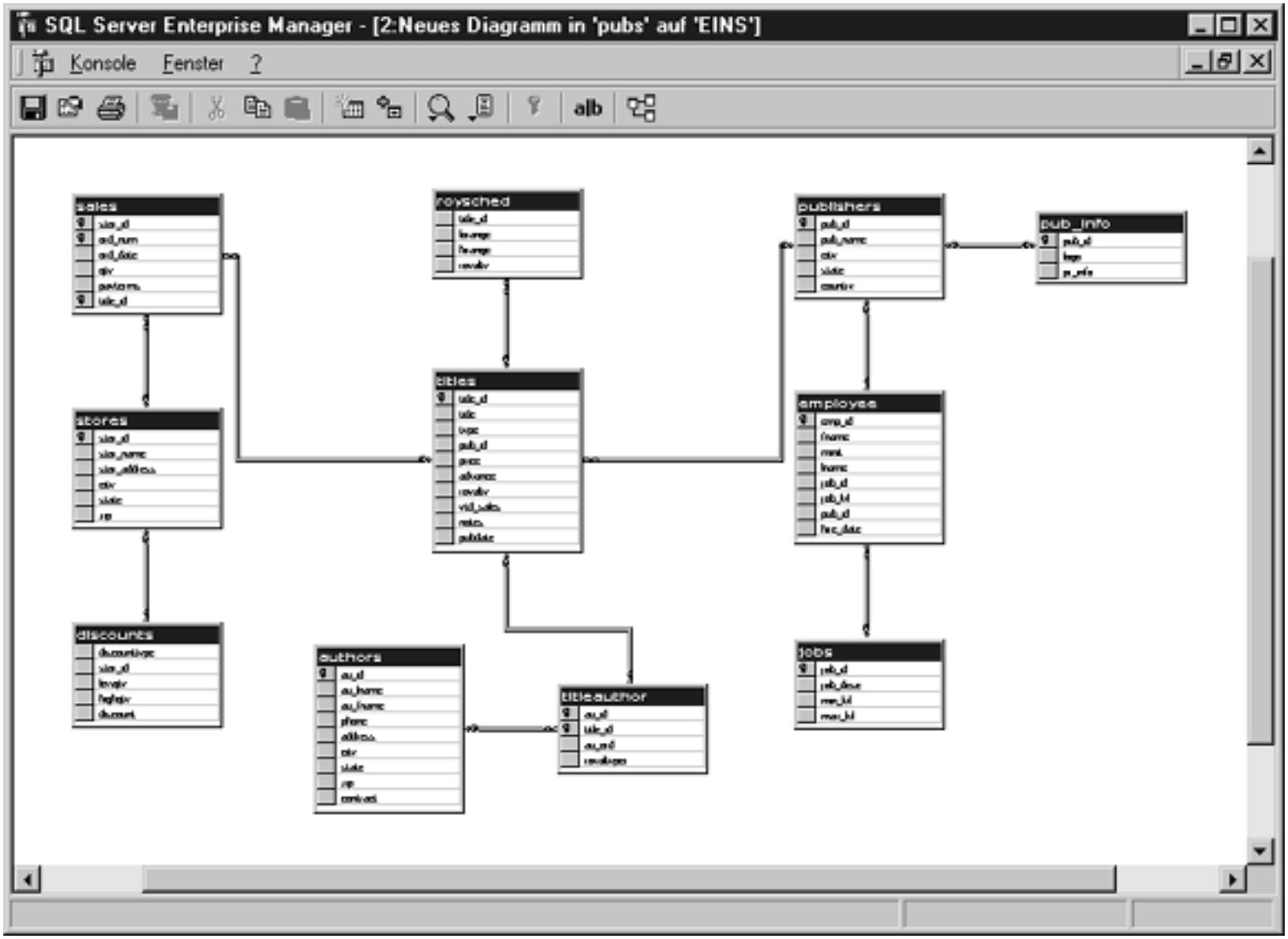
diese seitenstruktur ist unter anderem für die effiziente gestaltung von indizes (siehe dazu kapitel 14) von bedeutung.

4.4.4 dateigruppen

datenbankdateien lassen sich so auf mehrere dateien aufteilen, daß man objekte in sogenannten dateigruppen logisch gruppieren kann. das gilt für tabellen, indizes und daten der datentypen text, ntext und image. bei bestimmten datenbanken kann man mit hilfe von dateigruppen eine leistungsverbesserung erreichen. beispielsweise bietet es sich an, die indizes einer datenbank in einer eigenen dateigruppe unterzubringen, die auf einer sehr schnellen festplatte gespeichert wird. umfangreiche bilddateien legt man auf einer zweiten festplatte ab. alle anderen objekte kommen in die standarddateigruppe.

da sich sql server auch ohne dateigruppen effizient einsetzen läßt, geht dieses buch nicht weiter auf dateigruppen ein.


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Window...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
10 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
master
model
msdb
Northwind
pubs
Diagramme
Tabellen
Sichten
Gespeicherte Prozeduren
DB-Benutzernamen
Rollen
Regeln
Standards
Benutzerdefinierte Datentypen
Volltextkataloge
tempdb
Data Transformation Services
Verwaltung
Sicherheit
Unterstützungsdienste
Diagramme
Tabellen
Sichten
Gespeicherte Prozeduren
DB-Benutzernamen
Rollen
Regeln
Standards
Benutzerdefinierte Datentypen
Volltextkataloge


SQL Server Enterprise Manager - [2:Neues Diagramm in 'pubs' auf 'EINS']
Konsole
Fenster
?
alb
sales
roysched
publishers
pub_info
stores
titles
employee
discounts
authors
titleauthor
jobs

kapitel 5 transact-sql

5.1 an transact-sql kommt keiner vorbei

die produktbezeichnung läßt keine zweifel aufkommen: sql ist wesentlicher bestandteil des datenbank-managementsystems sql server 7.0. selbst wenn sie ausschließlich mit den grafischen werkzeugen arbeiten, kommen sie früher oder später mit transact-sql - dem speziell erweiterten sql-dialekt von sql server - in berührung. zum beispiel bietet der sichterstellungs-assistent die möglichkeit, mit hilfe von transact-sql beschränkungen zu definieren, d.h. die von der sicht anzuzeigenden informationen einzuschränken. diese möglichkeit brauchen sie natürlich nicht wahrzunehmen. doch spätestens im letzten schritt des assistenten werden sie mit einer transact-sql-anweisung konfrontiert (siehe abbildung 5.1).

bild

abbildung 5.1: im letzten schritt des sichterstellungs-assistenten erscheint eine transact-sql-anweisung, die sie bearbeiten können

der assistent hat diese anweisung automatisch generiert. sie können sie unverändert übernehmen, ohne sich weiter darum zu kümmern. manchmal ist es aber notwendig, bestimmte anpassungen vorzunehmen.

dieses kapitel macht sie deshalb bereits an dieser stelle mit dem aufbau und den syntaktischen elementen der sprache transact-sql bekannt und zeigt, mit welchen dienstprogrammen sie transact-sql-anweisungen ausführen, bearbeiten und speichern können.

5.2 einführung

sql steht für structured query language, also etwa strukturierte abfragesprache. das ist aber nur die halbe wahrheit. genaugenommen müßte man diese sprache als »strukturierte abfrage-, definitions-, manipulations- und verwaltungssprache« bezeichnen, weil nicht nur abfragen, sondern auch alle denkbaren anderen aufgaben in bezug auf datenbanken möglich sind. sql server verwendet einen dialekt der sprache ansi-sql oder ansi-92 - transact-sql. microsoft hat sql unter anderem um befehle zur transaktionsverarbeitung erweitert.

die sprache transact-sql ist dreh- und angelpunkt bei der arbeit mit microsoft sql server. alle anwendungen, die mit sql server kommunizieren, senden transact-sql-anweisungen an den server, unabhängig von der benutzeroberfläche der jeweiligen anwendung. allerdings ist es von der art des zugriffs auf die daten in sql server abhängig, welche kenntnisse der benutzer über transact-sql benötigt:

- benutzer, die mit grafischen abfragewerkzeugen oder allgemeinen firmenanwendungen arbeiten, benötigen keine oder nur geringe kenntnisse über transact-sql.
- benutzer, die mit sql-anwendungen wie sql server query analyzer oder dem dienstprogramm osql

arbeiten, sowie anwendungsprogrammierer müssen sich mit dem einsatz von transact-sql auskennen.

5.3 bestandteile von transact-sql

die anweisungen der sprache transact-sql lassen sich drei kategorien zuordnen:

- datendefinitionssprache (ddl - data definition language)
- datenbearbeitungssprache (dml - data manipulation language)
- datensteuerungssprache (dcl - data control language)

5.3.1 datendefinitionssprache

in diese kategorie fallen die befehle, mit denen sich datenbankobjekte definieren oder deklarieren lassen, zum beispiel create table, create index und drop table.

5.3.2 datenbearbeitungssprache

mit der datenbearbeitungssprache lassen sich die in tabellen gespeicherten daten einer datenbank manipulieren. der wichtigste befehl dieser kategorie ist select zum abrufen - oder auswählen (englisch: select) - von daten. weitere markante vertreter dieser kategorie sind insert (einfügen neuer zeilen in eine tabelle), update (aktualisieren von tabelleninhalten) und delete (löschen von daten).

5.3.3 datensteuerungssprache

die anweisungen dieser teilmenge von transact-sql steuern die berechtigungen für datenbankobjekte. als beispiele seien die anweisungen grant, revoke, lock, commit und rollback genannt.

5.4 stapelverarbeitung

wie eingangs erwähnt, senden alle anwendungen, die mit sql server kommunizieren, transact-sql-anweisungen an den server. die anweisungen müssen nicht einzeln gesendet werden, sondern können als gruppe in einem sogenannten *stapel* zum server geschickt werden.

das konzept des stapels ist für die ausführung der transact-sql-anweisungen durch sql server von grundlegender bedeutung. sql server kompiliert die anweisungen eines stapels zu einer einzigen ausführbaren einheit - dem sogenannten *ausführungsplan*. dann werden die anweisungen des ausführungsplans nacheinander abgearbeitet.

treten laufzeitfehler auf, wird die aktuelle anweisung beendet, und (außer bei bestimmten laufzeitfehlern wie zum beispiel einschränkungsverletzungen) die übrigen anweisungen im stapel gelangen nicht mehr zur ausführung.

wenn man eine tabelle verändert, kann man im selben stapel nicht auf die neuen spalten verweisen.

sind anweisungen mit execute auszuführen (beispielsweise gespeicherte prozeduren), kann das

schlüsselwort execute fehlen, wenn es sich um die erste anweisung im stapel handelt. bei allen folgenden anweisungen ist das schlüsselwort execute erforderlich.

in einem stapel dürfen folgende anweisungen nicht zusammen mit anderen transact-sql-anweisungen kombiniert werden:

- create default
- create procedure
- create rule
- create trigger
- create view

5.5 transact-sql-anweisungen ausführen

mit den dienstprogrammen von sql server lassen sich transact-sql-anweisungen von der eingabeaufforderung oder einer grafischen benutzeroberfläche ausführen.

von interesse sind hier die dienstprogramme isql und osql für die befehlszeile sowie sql server query analyzer als grafisches werkzeug. das dienstprogramm isql arbeitet mit der älteren db-library und wurde durch das neuere programm osql, das odbc zur kommunikation mit sql server verwendet, ersetzt. beide programme weisen aber viele gemeinsamkeiten auf.

in den älteren versionen von sql server gab es noch die windows-version des programms isql namens isql/w. dieses programm hört jetzt auf den namen sql server query analyzer und verwendet genau wie osql die odbc-schnittstelle. der name der ausführbaren datei von sql server query analyzer lautet isqlw.exe.

mit den genannten programmen lassen sich ad-hoc-abfragen genauso wie gespeicherte prozeduren oder als skript gespeicherte transact-sql-anweisungsfolgen (stapel) ausführen.

5.5.1 osql

mit dem dienstprogramm osql lassen sich transact-sql-anweisungen, systemprozeduren und skriptdateien ausführen. die kommunikation mit dem server läuft über eine odbc-schnittstelle ab. das programm starten sie direkt von der eingabeaufforderung des betriebssystems. die argumente sind in der richtigen schreibweise (groß/klein) einzugeben.

syntax:

die syntax des befehls osql lautet:

```
osql -u anmeldename [-e] [-e][-p][-d datenbankname]
[-q "abfrage"][-q "abfrage"]
[-c cmd_end][-h header][-w spaltenbreite][-s spaltentrennung]
[-t timeout][-m fehlerstufe][-l][-?][-r {0 | 1}]
[-h workstation][-p kennwort][-r]
```

```
[-s servername][-i eingabedatei][-o ausgabedatei][-u]
[-a paketgröße][-b][-o][-l timeout]
```

tabelle 5.1 erläutert die argumente des befehls osql.

parameter	beschreibung
-u *anmeldename*	der anmeldename des benutzers. groß-/kleinschreibung ist zu beachten.
-e	liefert die eingaben als echo zurück.
-e	verwendet eine vertraute verbindung, statt nach dem kennwort zu fragen.
-p	gibt leistungsstatistiken aus.
-n	entfernt numerierungen und das aufforderungssymbol (>) aus eingabezeilen.
-d *datenbankname*	führt implizit den befehl use *datenbankname* beim start von osql aus.
-q "abfrage"	führt eine abfrage beim start von osql aus. (die abfrageanweisung darf nicht den befehl go enthalten.) die abfrageanweisung ist in anführungszeichen einzuschließen. in die abfrage eingebettete strings, die in anführungszeichen zu schreiben sind, müssen in apostrophe (einfache anführungszeichen) eingeschlossen werden.
-q "abfrage"	führt eine abfrage aus und beendet osql sofort wieder. ansonsten gelten die hinweise wie bei der option -q.
-c	spezifiziert die abschlußzeichenfolge für befehle. in der standardeinstellung schickt man befehle an sql server mit dem befehl go, der allein auf einer zeile steht. eine andere zeichenfolge darf nicht aus reservierten wörtern oder zeichen mit spezieller bedeutung für das betriebssystem bestehen, egal ob ihnen ein backslash vorangeht oder nicht.
-h *header*	spezifiziert die anzahl der zeilen, die zwischen zwei spaltenüberschriften auszugeben sind. in der voreinstellung wird für jede ergebnismenge einer abfrage nur eine spaltenüberschrift ausgegeben. der wert -1 legt fest, daß keine überschriften erscheinen. achten sie darauf, keine leerzeichen zwischen -h und -1 zu schreiben (also nicht -h -1).
-w *spaltenbreite*	legt die spaltenbreite für bildschirmausgaben fest. die voreinstellung ist 80 zeichen. überschreitet eine ausgabezeile die maximale bildschirmbreite, wird sie auf mehrere zeilen aufgeteilt
-s *spaltentrennung*	legt das zeichen fest, mit dem spalten getrennt werden (in der voreinstellung ein leerzeichen). zeichen mit spezieller bedeutung für das betriebssystem (wie etwa | ; & < >) sind in anführungszeichen (") einzuschließen.

-t *timeout*	gibt die anzahl der sekunden an, bevor eine zeitüberschreitung für einen befehl in kraft tritt. ist kein timeout-wert angegeben, läuft ein befehl auf unbestimmte zeit. die zeitüberschreitung für die anmeldung in osql beträgt in der voreinstellung 8 sekunden.
-m *fehlerstufe*	paßt die anzeige von fehlermeldungen an. für die angegebene oder eine größere schwere des fehlers werden fehlernummer, zustand und fehlerstufe anzeigt. bei fehlern unterhalb der angegebenen fehlerstufe werden überhaupt keine meldungen angezeigt. beim wert -1 werden alle zurückgegebenen header mit meldungen angezeigt, sogar informative meldungen. zwischen dem parameter -m und der -1 darf kein leerzeichen stehen (also nicht -m -1).
-l	listet die lokal konfigurierten server und die namen der über das netzwerk erreichbaren server auf.
-?	zeigt die syntaxzusammenfassung der osql-schalter an.
-r	leitet die ausgabe der meldungen auf den bildschirm um. ohne parameter oder 0: nur fehlermeldungen parameter gleich 1: alle meldungen
-h	name einer arbeitsstation. wenn nicht angegeben, wird der computername verwendet.
-p	das kennwort, mit dem sich der benutzer bei sql server anmeldet. fehlt die option -p, fordert osql zur eingabe eines kennwortes auf. wenn sie -p ohne kennwort angeben, verwendet osql das standardkennwort (null). das standardkennwort gilt nach der installation von sql server.
-r	legt fest, daß der odbc-treiber von sql server bei der umwandlung von währungs-, datums- oder zeitdaten in zeichenfolgen auf die client-einstellungen zurückgreift.
-s	bezeichnet den server, auf dem sql server installiert ist und zu dem die verbindung hergestellt werden soll.
-i *eingabedatei*	bezeichnet die eingabedatei, die einen stapel von sql-anweisungen oder gespeicherten prozeduren enthält. anstelle von -i kann man auch den operator kleiner als (<) schreiben.
-o *ausgabedatei*	bezeichnet die datei, die die ausgaben von osql empfängt. anstelle von -o kann man auch das zeichen für größer als (>) schreiben. wenn die *eingabedatei* nicht im unicode-format vorliegt und -u nicht angegeben ist, wird die *ausgabedatei* im oem-format gespeichert. ist die *eingabedatei* im unicode-format gespeichert oder ist -u angegeben, wird die *ausgabedatei* im unicode-format abgelegt.
-u	legt fest, daß die *ausgabedatei* im unicode-format gespeichert wird, unabhängig vom format der *eingabedatei*.

`-a` *paketgröße*	legt eine andere paketgröße fest. standardwert ist die vom server verwendete größe.
`-b`	beendet osql bei einem fehler und gibt einen dos errorlevel-wert zurück (1 bei einem schweregrad größer gleich 10, sonst 0).
`-o`	deaktiviert verschiedene features von osql, um das verhalten an isql der früheren versionen von sql server anzupassen.
`-l`	zeitüberschreitung in sekunden für die anmeldung. standardwert ist 15 sekunden.

tabelle 5.1: befehlszeilenparameter des dienstprogramms osql

um das dienstprogramm zu starten und sich bei sql server als systemadministrator ohne kennwort anzumelden, geben sie folgenden befehl ein:

```
osql -usa -p
```

damit wird osql gestartet. der anmeldename sa ist die standardeinstellung, nachdem sie sql server installiert haben. sollte bereits ein anderer name festgelegt worden sein (siehe kapitel 21), müssen sie statt dessen diesen namen angeben. die option -p ohne kennwort bezieht sich ebenfalls auf die standardeinstellung der installation.

wenn sie ein eigenes verzeichnis für ihre sql-skripts angelegt haben, sollten sie zunächst in dieses verzeichnis wechseln und von hier aus osql starten. damit können sie die skripts mit :r skript.sql aufrufen, ohne ein verzeichnis angeben zu müssen.

wenn sie osql ohne benutzernamen starten, prüft sql server die umgebungsvariablen und verwendet diese werte. sind keine umgebungsvariablen festgelegt, wird der benutzername der arbeitsstation verwendet. ist kein server spezifiziert, verwendet osql den namen der arbeitsstation. wenn sie weder die option -u noch die option -p angeben, versucht sql server, eine verbindung unter verwendung des windows-nt-authentifizierungsmodus herzustellen. bei osql -usa ohne kennwort erscheint die aufforderung zur eingabe des kennwortes. drücken sie einfach **è**, um das standardkennwort (null) zu verwenden.

konnte eine verbindung zu sql server hergestellt werden, erscheint die aufforderung

```
>1
```

jetzt können sie transact-sql-befehle, systemprozeduren und skriptdateien eingeben und ausführen lassen.

befehle des dienstprogramms osql

das dienstprogramm osql umfaßt befehle, die nicht zum sprachumfang von transact-sql gehören. tabelle

5.2 führt die befehle von osql auf.

befehl	beschreibung
go	führt alle befehle aus, die nach dem letzten go eingegeben wurden.
reset	löscht alle bisher eingegebenen anweisungen.
ed	ruft den editor auf.
!! *befehl*	führt einen befehl des betriebssystems aus.
quit oder exit()	beendet osql.
ç+c	beendet eine abfrage, ohne osql zu beenden.

tabelle 5.2: befehle des dienstprogramms osql

die befehle müssen allein auf einer zeile und unmittelbar nach dem aufforderungszeichen (>) stehen. kommentare können sich anschließen.

anweisungsstapel abschließen und ausführen (go)

im gegensatz zu anderen sql-dialekten (wie etwa bei oracle) gibt es kein zeilenabschlußzeichen. eine anweisung wird mit dem befehl go abgeschlossen und ausgeführt. der befehl go signalisiert damit den dienstprogrammen von sql server das ende eines stapels von transact-sql-anweisungen. go ist das sogenannte tsql-stapeltrennzeichen.

es sind die regeln für stapel zu befolgen. insbesondere ist der gültigkeitsbereich lokaler (benutzerdefinierter) variablen auf einen stapel beschränkt. nach der ausführung von go kann man die variablen des ausgeführten stapels nicht mehr referenzieren. gespeicherte prozeduren sind mit dem befehl exec(ute) aufzurufen, außer wenn es sich um die erste anweisung im stapel handelt.

um beispielsweise alle im systemkatalog gespeicherten datenbanknamen aufzulisten, geben sie folgende befehle ein:

```
1>select name from sysdatabases
2>go
```

befehle von osql wie auch von transact-sql kann man groß, klein oder auch gemischt schreiben. im laufenden text werden die befehle normalerweise in durchgängiger großschreibung angegeben, damit sie sich auf einen blick als solche erkennen lassen. fast alle beispiele verwenden dagegen die kleinschreibung. das hat keine funktionelle bedeutung und ist eher geschmackssache.

beenden von osql (exit, quit)

das dienstprogramm osql beenden sie durch eingabe von exit oder quit. während der befehl quit allein auf einer zeile stehen muß und osql einfach nur beendet, kann man bei exit einen ganzzahligen wert mit einer länge von 4 byte an den aufrufer zurückgeben. zum beispiel liefert die anweisung:

```
>1 exit (select -2147483648)
```

das ergebnis:

```
-------------
 -2147483648
```

den befehl exit kann man in folgenden formaten verwenden:

- exit
- führt den stapel nicht aus, beendet osql und liefert auch keinen rückgabewert.
- exit()
- führt den stapel aus, beendet osql und liefert keinen rückgabewert.
- exit(*abfrage*)
- führt den aktuellen stapel und die in klammern angegebene *abfrage* aus, beendet osql und gibt das ergebnis der abfrage (einen ganzzahligen wert zwischen -2147483648 und 2147483647) zurück.

weitere hinweise finden sie in der online-dokumentation zum dienstprogramm osql.

editor aufrufen (ed)

mit dem befehl ed ruft man den standardeditor auf. normalerweise wird der ms-dos-editor gestartet. um beispielsweise den windows-editor als standardeditor festzulegen, geben sie an der ms-dos-eingabeaufforderung - d.h. nicht an der aufforderung von osql - den befehl set editor = notepad ein oder schreiben ihn in die datei autoexec.bat.

nach ausführung von

```
ed
```

erscheinen die zuletzt eingegebenen befehle im editor und können hier bearbeitet werden.

bisher eingegebene anweisungen löschen (reset)

um neue anweisungen einzugeben, setzt man zunächst den anweisungsstapel mit reset zurück und ruft dann ed auf. es erscheint der editor ohne jegliche anweisungen.

sql-dateien laden und ausführen

der befehl

```
:r dateiname
```

lädt die durch *dateiname* spezifizierte datei mit sql-anweisungen. mit go starten sie den geladenen anweisungsstapel, mit ed rufen sie den editor auf, um die geladenen anweisungen zu bearbeiten.

wenn sie ein skript per osql ausführen, darf es keine befehlsabschlußzeichen (sprich go) enthalten. praktisch bedeutet das, daß ein derartiges skript aus nur einem stapel bestehen darf. das go geben sie interaktiv ein, um den stapel auszuführen.

5.5.2 isqlw

das dienstprogramm isqlw starten sie von der ms-dos-eingabeaufforderung mit isqlw[.exe]. tabelle 5.3 faßt die befehlszeilenparameter für isqlw zusammen. wenn sie isqlw mit parametern starten, rufen sie praktisch sql server query analyzer als programm mit vorgegebener konfiguration auf. der start von isqlw ohne parameter ruft sql server query analyzer auf und zeigt das anmeldedialogfeld an.

parameter	beschreibung
`-?`	zeigt eine liste mit den befehlszeilenparametern an.
-s *servername*	der netzwerkname des servers, zu dem eine verbindung hergestellt werden soll. die standardeinstellung ist der lokale computer (local).
-d *datenbank*	gibt eine use-anweisung für die zu verwendende datenbank beim start von isqlw aus. voreinstellung ist die standarddatenbank des benutzers.
`-e`	verwendet eine vertraute verbindung und fordert kein kennwort an.
`-u`	spezifiziert den benutzernamen. dieser parameter ist von der groß-/kleinschreibung abhängig.
-p *kennwort*	gibt das anmeldekennwort an. die standardeinstellung ist null (d.h. kein kennwort).
-i *eingabedatei*	spezifiziert eine eingabedatei mit einem stapel aus sql-anweisungen oder gespeicherten prozeduren. wenn sie eine eingabedatei festlegen, müssen sie auch mit dem parameter -o eine ausgabedatei spezifizieren. die ergebnisse der abfragen werden in die ausgabedatei geschrieben, ohne daß eine benutzeroberfläche erscheint.
-o *ausgabedatei*	spezifiziert die ausgabedatei, in die isqlw die ergebnisse schreibt.

`-f u`
`-f a`
`-f o`
	dateiformat unicode
	dateiformat ansi
	dateiformat oem
	fehlt dieser parameter, wird die datei im automatischen modus geöffnet (bei entsprechender kennzeichnung im unicode, sonst als ansi).

`-s` *"spaltentrennzeichen"*	das spaltentrennzeichen ist in der standardeinstellung ein leerzeichen. zeichen, die das betriebssystem verwendet, sind in anführungszeichen zu schreiben.
-f *dateiliste*	isqlw lädt die in der dateiliste angegebenen dateien in sql server query analyzer, wobei für alle dateien dieselbe verbindung gilt. die einzelnen dateinamen können pfadangaben und platzhalterzeichen enthalten.

tabelle 5.3: befehlszeilenparameter des dienstprogramms isqlw

bei den parametern ist auf die groß-/kleinschreibung zu achten. das dienstprogramm isqlw soll hier nicht weiter behandelt werden, weil es im wesentlichen dem nachstehend beschriebenen sql server query analyzer entspricht (bis auf den modus, der keine benutzeroberfläche anzeigt). weitere hinweise und beispiele finden sie in der online-dokumentation. wenn sie direkt von der befehlszeile arbeiten wollen, können sie auch auf osql zurückgreifen.

5.5.3 sql server query analyzer

sql server query analyzer ist ein grafisches dienstprogramm, mit dem sie unter anderem transact-sql-anweisungen ausführen, die ergebnisse als text oder in tabellenform in einem eigenen fenster anzeigen und eine analyse der abfrage durchführen können.

mit sql server query analyzer ist es darüber hinaus möglich, den ausführungsplan einer abfrage anzuzeigen. anhand dieser informationen kann der programmierer die leistung einer abfrage und den ressourcenverbrauch optimieren. auf diese punkte geht kapitel 24 näher ein.

sql server query analyzer starten

führen sie die folgenden schritte aus, um sql server query analyzer zu starten:

1. wählen sie **start / programme / microsoft sql server 7.0 / query analyzer**, oder rufen sie isqlw von der befehlszeile auf. beim aufruf über das menü oder mit isqlw ohne parameter erscheint zunächst das verbindungsdialogfeld gemäß abbildung 5.2.

abbildung 5.2: beim start von sql server query analyzer erscheint zunächst das verbindungsdialogfeld

2. aus dem drop-down-listenfeld **sql server** wählen sie den server aus, mit dem sie eine verbindung herstellen möchten. bei der verbindung zu einem sql server, der unter windows nt läuft, ist sowohl die option *windows nt-authentifizierung* als auch *sql server-authentifizierung* verfügbar, bei verbindung zu einem windows-95/98-rechner nur die zweite option. tragen sie bei sql-server-authentifizierung den benutzernamen und das kennwort ein, und klicken sie auf **ok**.

bild

abbildung 5.3: nach herstellen der verbindung ist sql server query analyzer bereit zur eingabe von abfragen

3. nachdem sql server query analyzer die verbindung zum server hergestellt hat, ist das dienstprogramm bereit zur eingabe von transact-sql-anweisungen (siehe abbildung 5.3).

skripts laden

um eine vorhandene abfragedatei, ein sogenanntes *sql-skript* (dateierweiterung *.sql) zu laden, wählen sie entweder in der symbolleiste des abfragefensters die schaltfläche **sql-skript laden** (zweite schaltfläche von links) oder im hauptfenster von sql server query analyzer den befehl **datei / öffnen**. es erscheint das dialogfeld **abfragedatei öffnen** (siehe abbildung 5.4). wenn sie zum beispiel die datenbank pubs neu erstellen möchten, wählen sie das skript instpubs.sql im verzeichnis \mssql7\install.

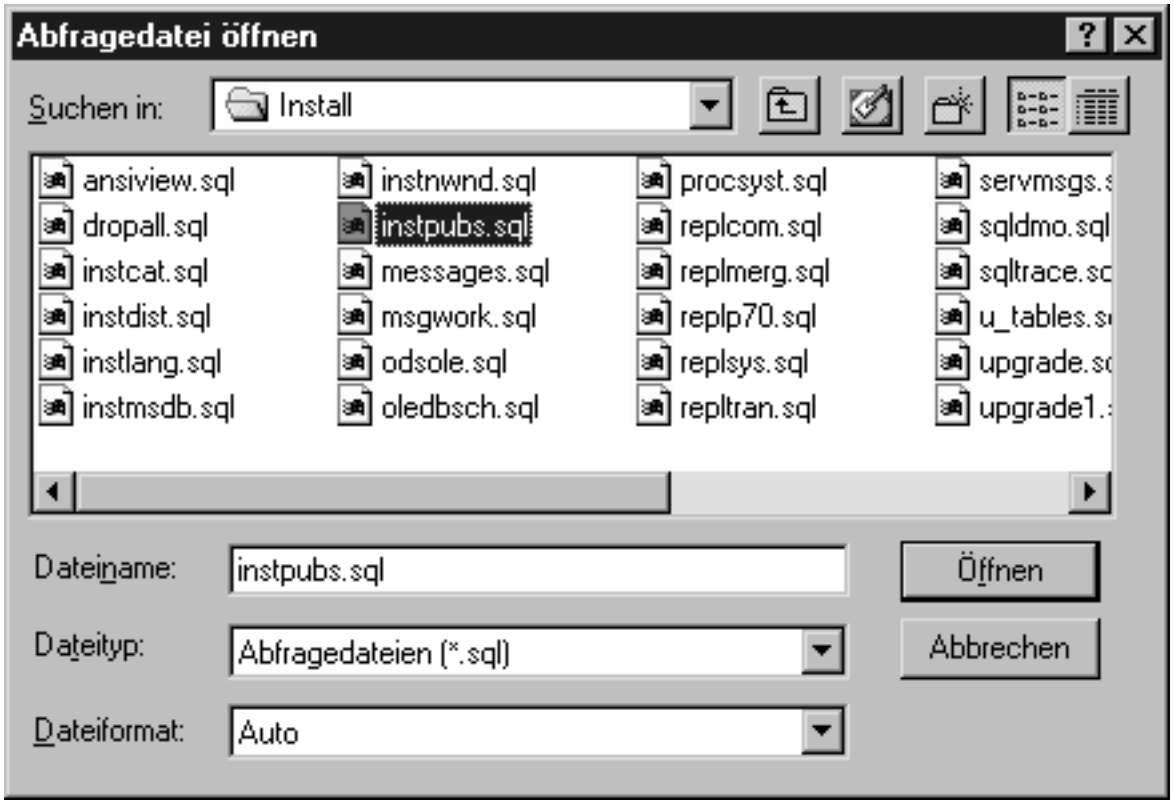


abbildung 5.4: im dialogfeld abfragedatei öffnen wählen sie das auszuführende skript aus

klicken sie auf **öffnen**. im abfragefenster erscheint nun der text des sql-skripts instpubs.sql (siehe abbildung 5.5).

bild

abbildung 5.5: das skript instpubs.sql im abfragefenster von sql server query analyzer

im anhang b finden sie ein vollständig kommentiertes listing des skripts instpubs.sql.

skripts bearbeiten

das abfragefenster dient als editor für transact-sql-anweisungen. neben den üblichen editorbefehlen wie ausschneiden, kopieren oder einfügen stehen zusätzliche befehle wie gehe zu (zeile) oder die umwandlung der markierung in klein-/großbuchstaben zur verfügung.

darüber hinaus markiert der editor die anweisungen mit festgelegten farben, um bestimmte elemente hervorzuheben.

damit können sie zum beispiel auf einen blick erkennen, ob sie eine anweisung korrekt eingegeben haben: während der eingabe erscheinen die zeichen in der textfarbe (schwarz in der voreinstellung). sobald sql server query analyzer einen gültigen befehl erkennt, wechselt die farbe. ein korrekt eingegebenes schlüsselwort erscheint demnach in blau. probieren sie es aus. tippen sie den befehl execute ein. die ersten drei buchstaben erscheinen zunächst in schwarz. sobald sie das c eingegeben haben, wechselt der befehl in blau, weil der transact-sql-befehl sowohl in den schreibweisen exec als auch execute gültig ist. wenn sie weiterschreiben, nimmt der befehl wieder die textfarbe an und wechselt nach dem letzten e wieder zur blauen markierung. abbildung 5.6 zeigt das dialogfeld **schriftart** mit der registerkarte **format**. hier können sie bei bedarf die farben und die schriftart für die einzelnen

syntaxelemente an eigene vorstellungen anpassen. das dialogfeld öffnen sie über **ansicht / schriftart**.

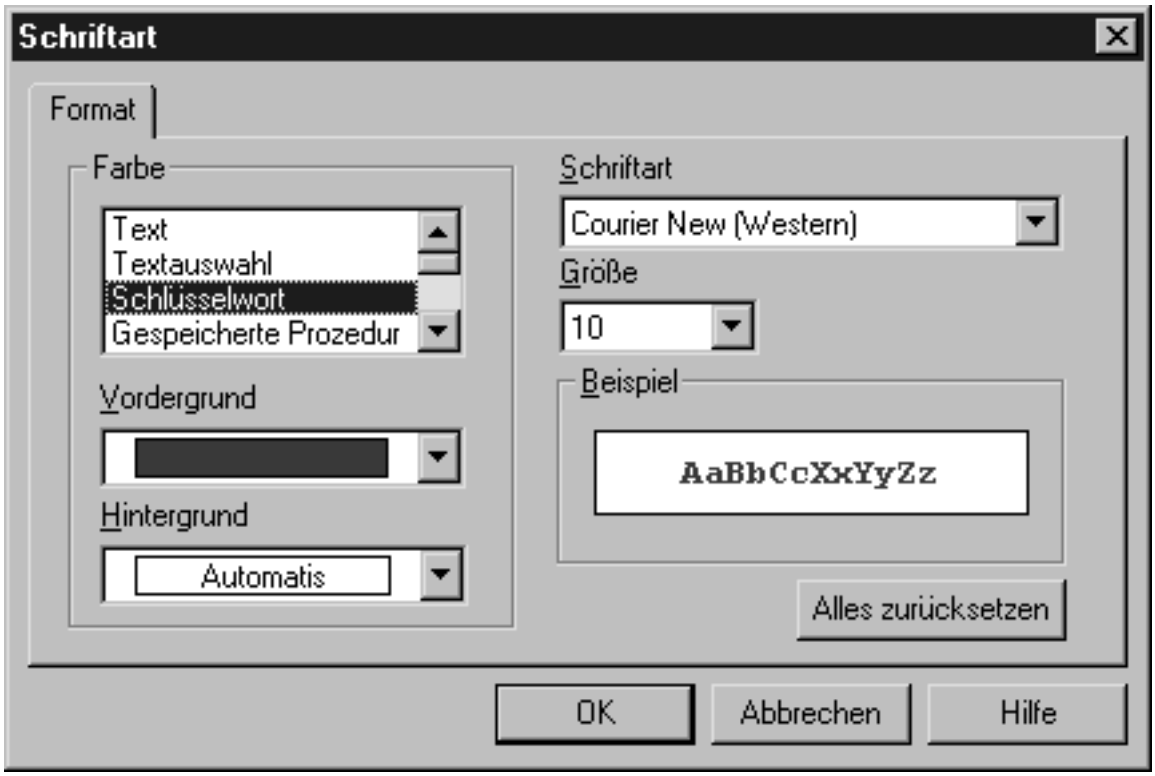


abbildung 5.6: im dialogfeld schriftart lassen sich die farblichen markierungen der syntaxelemente anpassen

um ein bearbeitetes skript zu speichern, wählen sie den befehl **datei / speichern** oder **datei / speichern unter**, wie sie es von einem standardeditor oder einer textverarbeitung gewohnt sind.

skripts ausführen

die transact-sql-anweisungen im abfragefenster beziehen sich auf die aktuelle datenbank, die auf der symbolleiste des abfragefensters im listenfeld **db** angezeigt wird. wenn sie die aktuelle datenbank mit einer use-anweisung ändern, aktualisiert sql server query analyzer das listenfeld, nachdem sie die abfrage ausgeführt haben. (auf die anweisung use geht kapitel 10 näher ein.)

die folgenden schritte gehen davon aus, daß sie das skript instpubs.sql geladen haben.

falls sie bereits änderungen an der beispieldatenbank pubs vorgenommen haben und diese änderungen beibehalten möchten, sollten sie ein anderes skript laden oder ein neues skript erstellen, um die nachstehenden schritte auszuführen. die ausführung von instpubs.sql löscht nämlich die datenbank pubs vollständig und bringt sie wieder in den zustand, den sie bei der installation von sql server hatte.

die anweisungen im abfragefenster führen sie folgendermaßen aus:

1. wählen sie im menü **ansicht** den befehl **ausführen**, oder drücken sie **í**. daraufhin wird das skript gestartet. das geteilte abfragefenster zeigt im unteren teil während der laufenden abfrage die fortschrittsmeldungen an (siehe abbildung 5.7).

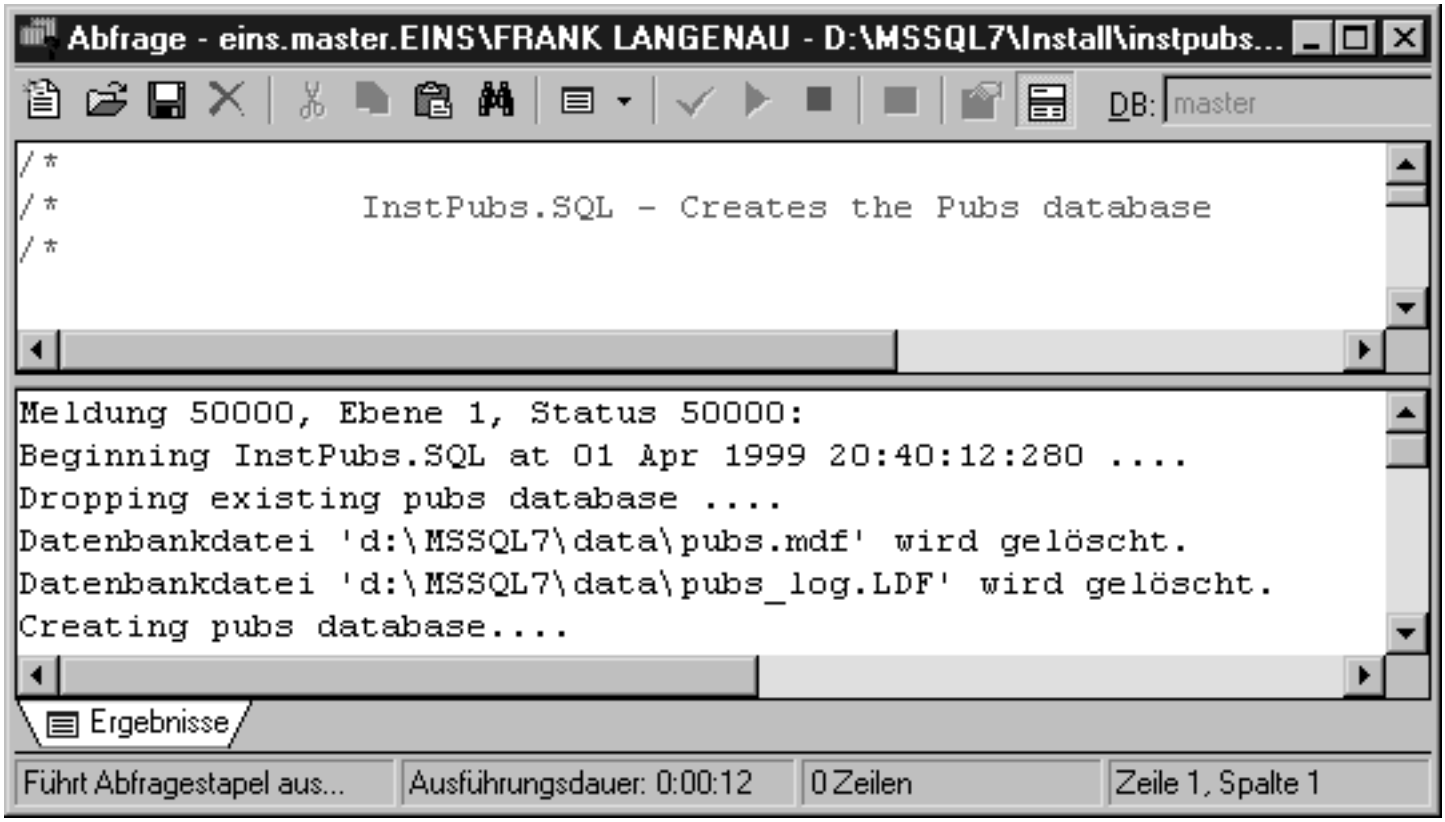


abbildung 5.7: eine abfrage während der ausführung

2. nachdem sql server den ausführungsstapel abgearbeitet hat, erscheint in der statuszeile ein entsprechender hinweis zusammen mit der ausführungszeit.

neues skript erstellen

um eine neue abfrage bzw. ein neues skript zu erstellen, wählen sie die schaltfläche **neue abfrage** (die erste von links in der symbolleiste des abfragefensters) oder drücken **ç**+**n**. es wird ein neues leeres editorfenster geöffnet, in das sie neue transact-sql-anweisungen eingeben können.

skripts speichern

geänderte skripts speichern sie entsprechend der standardbedienung von windows mit dem befehl **datei / speichern** oder durch drücken von **ç**+**s**. wenn sie ein neu angelegtes skript erstmalig speichern, erscheint automatisch das dialogfeld **abfrage speichern**, das einem standarddialogfeld von windows entspricht. im feld **dateityp** ist bereits der standardtyp *abfragedateien (*.sql)* vorgegeben.

verbindung herstellen und trennen

mit jedem öffnen eines abfragefensters richtet sql server query analyzer eine neue verbindung ein. die anzahl der verbindungen können sie der statusleiste im hauptfenster von sql server query analyzer entnehmen. für die neue verbindung werden die aktuellen einstellungen übernommen. um die neuen verbindungen zu einem anderen server oder mit anderen anmeldeoptionen herzustellen, wählen sie den befehl **datei / verbinden** oder die tastenkombination **ç**+**o**. daraufhin erscheint das dialogfeld **verbindung zu sql server herstellen**, wie es bereits abbildung 5.2 gezeigt hat.

wenn sie ein abfragefenster schließen, wird die zugehörige verbindung zum server automatisch getrennt.

die verbindung für das aktuelle fenster können sie auch über den befehl **datei / trennen** aufheben. falls das skript geändert oder neu erstellt und noch nicht gespeichert wurde, erscheint ein bestätigungsdialogfeld (siehe abbildung 5.8).

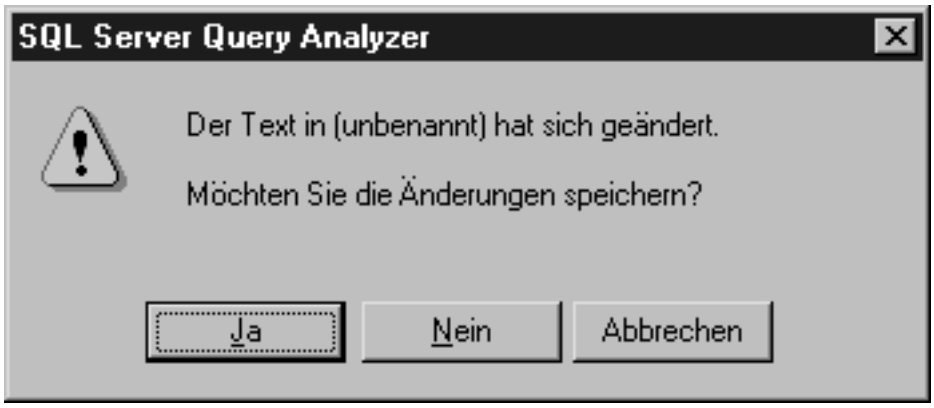


abbildung 5.8: bestätigungsdialogfeld, wenn ein skript noch nicht gespeichert wurde

wenn sie das skript speichern möchten, klicken sie auf **ja**. es erscheint dann das dialogfeld **abfrage speichern**, wie es bereits der abschnitt »skripts speichern« erläutert hat. mit klicken auf die schaltfläche **nein** verwerfen sie die änderungen, bei **abbrechen** kehren sie zum abfragefenster zurück, ohne die verbindung zu trennen.

beim beenden von sql server query analyzer werden alle verbindungen getrennt, wobei sie gegebenenfalls zum speichern aufgefordert werden.

ansichten

im menü **abfrage** des query analyzers können sie einstellen, in welcher form die ergebnisse im abfragefenster erscheinen sollen. wählen sie dazu den befehl **ergebnisse in text** bzw. **ergebnisse in gitternetz**. für die anweisung

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

zeigt abbildung 5.9 zeigt die textdarstellung der ergebnisse und abbildung 5.10 die gitternetzvariante.

bild

abbildung 5.9: ergebnisse in textform

bei der ausgabe im gitternetz erscheinen gegebenenfalls mehrere ergebnisfenster, zwischen denen sie über die register am unteren rand umschalten können. meldungen wie die anzahl der beeinflußten zeilen oder fehlermeldungen kommen immer in das fenster **meldungen**.

Bild

abbildung 5.10: ergebnisse im gitternetz

5.6 syntaxelemente

die folgenden abschnitte geben einen überblick über die syntaxelemente, aus denen sich transact-sql-anweisungen aufbauen:

- bezeichner
- datentypen
- funktionen
- ausdrücke
- operatoren
- kommentare
- reservierte schlüsselwörter

5.6.1 bezeichner

alle objekte in sql server können mit einem bezeichner, d.h. einem namen für das objekt, versehen sein. dazu gehören beispielsweise tabellen, sichten, cursor, gespeicherte prozeduren oder datenbanken und server. transact-sql unterscheidet zwischen regulären und begrenzten bezeichnern. bei beiden muß die anzahl der zeichen zwischen 1 und 128 liegen, ausgenommen lokale temporäre tabellen mit maximal 116 zeichen.

reguläre bezeichner

die namen von regulären bezeichnern müssen folgenden regeln entsprechen:

- erstes zeichen: ein vom unicodestandard 2.0 definierter buchstabe oder eines der sonderzeichen _ (unterstrich), @ (at-zeichen) oder # (nummernzeichen).
- nach dem ersten zeichen: ein vom unicodestandard 2.0 definierter buchstabe, ziffern, symbole (@, $, # oder _).
- reservierte wörter (siehe anhang c) sind nicht erlaubt.
- leerzeichen oder nicht hier aufgeführte sonderzeichen sind nicht zulässig.

verschiedene transact-sql-funktionen beginnen mit zwei at-zeichen (@@). um verwechslungen zu vermeiden, sollten sie in selbstdefinierten namen auf diese zeichenfolge verzichten.

begrenzte bezeichner

wenn sie die regeln für einen regulären bezeichner nicht einhalten können, *müssen* sie ihn in doppelte anführungszeichen oder eckige klammern einschließen, reguläre bezeichner *dürfen* begrenzt sein.

5.7 ausdrücke

ein ausdruck ist eine verknüpfung von symbolen (bezeichnern, werten und operatoren), deren auswertung ein einzelnes ergebnis liefert. ein ausdruck kann aus den elementen

- konstante
- funktion
- spaltenname
- variable
- unterabfrage
- case, nullif oder coalesce

oder einer durch operatoren verbundenen kombination dieser elemente bestehen.

das ergebnis eines ausdrucks, der aus einer einzelnen konstanten, variablen, skalarfunktion oder einem spaltennamen besteht, entspricht in datentyp, genauigkeit, anzahl der dezimalstellen und wert des ausdrucks den betreffenden eigenschaften des elements, auf das sich der ausdruck bezieht. auf die datentypen der ergebnisse geht der abschnitt zu datentypen weiter oben in diesem kapitel ein.

5.8 operatoren

ein operator ist ein symbol oder eine festgelegte zeichenfolge zur kennzeichnung einer bestimmten operation, die auf ein oder mehrere elemente (ausdrücke) anzuwenden ist. die in transact-sql definierten operatoren können sie sowohl in separaten transact-sql-anweisungen als auch in gespeicherten prozeduren einsetzen.

die operatoren von transact-sql lassen sich folgenden kategorien zuordnen:

- arithmetische operatoren
- zuweisungsoperator
- bitweise operatoren
- vergleichsoperatoren
- logische operatoren
- operatoren zum verketten von zeichenfolgen
- unäre operatoren

der begriff *operator* kann in sql server zweierlei bedeuten. in verbindung mit transact-sql versteht man darunter - wie oben beschrieben - ein symbol, das operationen definiert. es kann sich aber auch um eine person handeln, die zum beispiel bei warnungen benachrichtigt wird. das vorliegende kapitel beschäftigt sich mit den symbolen, kapitel 23 geht auf die operatoren als personen ein.

arithmetische operatoren

mathematische operationen lassen sich mit den in tabelle 5.4 gezeigten arithmetischen operatoren durchführen.

operator	beschreibung
+	addition
-	subtraktion
*	multiplikation
/	division
%	modulo-operation, gibt den rest einer ganzzahligen division zurück.

tabelle 5.4: arithmetische operatoren

die operatoren für addition, subtraktion, multiplikation und division verwendet man wie bei den bekannten grundrechenarten. der modulo-operator gibt den rest einer ganzzahligen division zurück. zum beispiel liefert

```
print 13 % 7
```

das ergebnis:

```
6
```

weil 13 geteilt durch 7 gleich 1 mit rest 6 ist.

die operatoren + und - können sie auch in ausdrücken mit datumswerten einsetzen. zum beispiel ermittelt die folgende anweisung mit der systemfunktion getdate() das aktuelle datum und zeigt es in der spalte heute an. für die spalte geburtstag werden zum aktuellen datum 8 tage addiert:

```
select getdate() as heute, getdate() + 8 as geburtstag
```

das ergebnis lautet:

```
heute                       geburtstag
--------------------------- ---------------------------
1999-06-23 15:07:54.880     1999-07-01 15:07:54.880
```

zuweisungsoperator

wie in vielen programmiersprachen fungiert das gleichheitszeichen als zuweisungsoperator. das folgende beispiel deklariert die lokale variable @zahl vom typ int und weist ihr in der zweiten zeile den wert 5 zu:

```
declare @zahl as int
set @zahl = 5
print @zahl
```

in transact-sql sind zuweisungen immer mit set zu formulieren. auf die deklaration von variablen sowie die anweisungen print und set geht kapitel 12 ein.

bitweise operatoren

manchmal ist es erforderlich, einzelne bits zu setzen oder zurückzusetzen. zum beispiel lassen sich standardoptionen in user options mit set-anweisungen einstellen. jedes bit in user options hat eine bestimmte bedeutung. so repräsentiert das bit mit dem wert 28 (= 256) die option quoted_identifier. ist das entsprechende bit gesetzt (d.h. gleich 1), unterscheidet sql server zwischen einfachen und doppelten anführungszeichen bei der auswertung eines ausdrucks.

tabelle 5.5 zeigt die von sql server unterstützten bitweisen operatoren.

operator	beschreibung
&	bitweises and
|	bitweises or
^	bitweises exklusives or

tabelle 5.5: bitweise operatoren

bei einer verknüpfung der operanden mit bitweisem and ist der wert der bitposition im ergebnis nur dann 1, wenn beide operanden auf der entsprechenden bitposition den wert 1 enthalten. beim bitweisen or ist das ergebnis 1, wenn mindestens ein operand an der jeweiligen bitposition eine 1 enthält. das bitweise exklusive or liefert im ergebnis eine 1, wenn die bitwerte der operanden ungleich sind, und eine 0, wenn die bitwerte der operanden gleich sind, d.h. 0 ^ 0 und 1 ^ 1 ergeben 0, während 0 ^ 1 und 1 ^ 0 das ergebnis 1 liefern.

die bitweisen operatoren lassen sich nicht uneingeschränkt auf alle datentypen anwenden. tabelle 5.6 zeigt die kombinationen der möglichen datentypen für linken und rechten operanden.

linker operand	rechter operand
binary	int, smallint, tinyint
bit	int, smallint, tinyint, bit
int	int, smallint, tinyint, binary, varbinary
smallint	int, smallint, tinyint, binary, varbinary

tinyint	int, smallint, tinyint, binary, varbinary
varbinary	int, smallint, tinyint

tabelle 5.6: datentypen der operanden bei bitweisen operatoren

das folgende beispiel deklariert drei variablen, die den linken und rechten operanden sowie das ergebnis aufnehmen. zeile 4 weist der variablen für den linken operanden den wert 120 und zeile 5 der variablen für den rechten operanden den wert 65 zu. den einsatz des operators für exklusives or zeigt zeile 6. schließlich gibt zeile 7 das ergebnis aus.

```
1:  declare @linker as int
2:  declare @rechter as int
3:  declare @ergebnis as int
4:  set @linker=120
5:  set @rechter=65
6:  set @ergebnis = @linker ^ @rechter
7:  print @ergebnis
```

das ergebnis lautet

```
57
```

in binärer darstellung sehen die werte für die beiden operanden und das ergebnis folgendermaßen aus:

```
0000 0000 0111 1000  linker operand
0000 0000 0011 1001  rechter operand
0000 0000 0100 0001  ergebnis
```

vergleichsoperatoren

mit den vergleichsoperatoren lassen sich zwei ausdrücke auf gleichheit oder ungleichheit testen. das ergebnis ist ein boolescher wert - true, false oder unknown. der wert unknown tritt nur in verbindung mit null-werten auf. wenn die option ansi_nulls aktiviert ist und mindestens einer der beiden operanden einen null-wert enthält, lautet das ergebnis bei allen vergleichsoperatoren unknown. wenn ansi_nulls auf off gesetzt ist, gilt das gleiche, allerdings liefert ein test auf gleichheit zwischen zwei null-werten das ergebnis true.

tabelle 5.7 zeigt die von sql server unterstützten vergleichsoperatoren. die zu vergleichenden ausdrücke können einen beliebigen datentyp außer text, ntext und image haben.

operator	beschreibung
=	gleich
>	größer als
<	kleiner als
>=	größer oder gleich
<=	kleiner oder gleich
<>	ungleich
!=	ungleich
!<	nicht kleiner als
!>	nicht größer als

tabelle 5.7: vergleichsoperatoren

die letzten drei operatoren in tabelle 5.7 sind nicht im sql-92-standard definiert.

operator	liefert true, wenn
all	alle vergleiche in der ergebnismenge true ergeben.
and	beide booleschen ausdrücke true ergeben.
any	mindestens ein vergleich in der menge true ergibt.
between	der operand innerhalb eines bereichs liegt.
exists	eine unterabfrage mindestens eine zeile zurückgibt.
in	der operand mit einem element in einer liste von ausdrücken übereinstimmt.
like	der operand einem muster entspricht.
not	kehrt den wert eines booleschen operators um.
or	mindestens einer der beiden booleschen ausdrücke true ergibt.
some	mindestens einer der vergleiche in der menge true ergibt.

tabelle 5.8: logische operatoren

logische operatoren

logische operatoren testen den wahrheitswert einer bedingung und geben einen booleschen wert - true oder false - zurück.

tabelle 5.8 zeigt die von sql server unterstützten logischen operatoren. kapitel 10 geht näher auf die logischen operatoren in verbindung mit where-klauseln ein.

operatoren zum verketten von zeichenfolgen

zeichenfolgen lassen sich in transact-sql mit dem plus-operator verbinden. beispielsweise liefert die anweisung

```
print 'abc' + 'def'
```

das ergebnis

```
abcdef
```

für alle anderen operationen mit zeichenfolgen stehen spezielle funktionen bereit, auf die kapitel 11 näher eingeht.

unäre operatoren

während die bisher behandelten operatoren jeweils zwei operanden verknüpfen, beziehen sich die unären operatoren nur auf einen operanden. die unären operatoren lassen sich auf numerische datentypen anwenden, das bitweise not nur auf zahlen vom datentyp int.

tabelle 5.9 zeigt die unären operatoren von sql server.

operator	beschreibung
+	positiver numerischer wert
-	negativer numerischer wert
~	bitweises not, bildet das einerkomplement einer zahl

tabelle 5.9: unäre operatoren

operatorrangfolge

getreu dem motto »punktrechnung geht vor strichrechnung« stellt die operatorrangfolge eine vorschrift dar, in welcher reihenfolge ein zusammengesetzter ausdruck mit mehreren operatoren ausgewertet wird. tabelle 5.10 gibt die operatorrangfolge in sql server an. an erster stelle stehen in der tabelle die operatoren mit der höchsten priorität, d.h. ausdrücke mit diesen operatoren werden zuerst ausgewertet. die niedrigste priorität hat demnach der zuweisungsoperator. das ist auch verständlich, denn zuerst will man ja einen ausdruck vollständig auswerten und dann erst einer ergebnisvariablen zuweisen. operatoren in derselben zeile haben die gleiche priorität und werden entsprechend ihrer position im ausdruck von links nach rechts ausgewertet.

wenn sie von der automatischen operatorrangfolge abweichen wollen, können sie klammern setzen. ein ausdruck in klammern wird zuerst ausgewertet, bevor die verknüpfung des so erhaltenen ergebnisses mit

den übrigen teilausdrücken stattfindet.

priorität von oben nach unten
+ (positiv), - (negativ), ~ (bitweises not)
* (multiplikation), / (division), % (modulo)
+ (addition), + (zeichenfolgen verketten), - (subtraktion)
=, >, <, >=, <=, <>, !=, !>, !< (vergleichsoperatoren)
^ (bitweises xor), & (bitweises and), | (bitweises or)
not
and
all, any, between, in, like, or, some
= (zuweisung)

tabelle 5.10: operatorrangfolge

5.9 kommentare

mit kommentaren lassen sich im programmcode anmerkungen einfügen, die nicht ausgeführt werden und der erläuterung dienen.

zwei minuszeichen kennzeichnen einen kommentar, der sich bis zum ende der aktuellen zeile erstreckt. einen mehrzeiligen kommentar leitet man mit den zeichen /* ein, mit */ wird er abgeschlossen. innerhalb eines mehrzeiligen kommentars kann ein einzeiliger kommentar verschachtelt sein.

```
1:  -- das ist ein einzeiliger kommentar
2:  /* ein mehrzeiliger kommentar deaktiviert anweisungen:
3:  use pubs
4:  select * from titles   -- wählt alle spalten von titles aus
5:     hier endet der mehrzeilige kommentar */
```

ab zeile 2 beginnt ein mehrzeiliger kommentar. er umschließt die anweisungen in den zeilen 3 und 4, die damit wirkungslos werden. diese eigenschaft von mehrzeiligen kommentaren macht man sich häufig bei der entwicklung von längeren prozeduren und bei der fehlersuche zunutze. wenn man zum beispiel »verdächtige« anweisungen einklammert, kann man die ursache für einen fehler schrittweise eingrenzen.

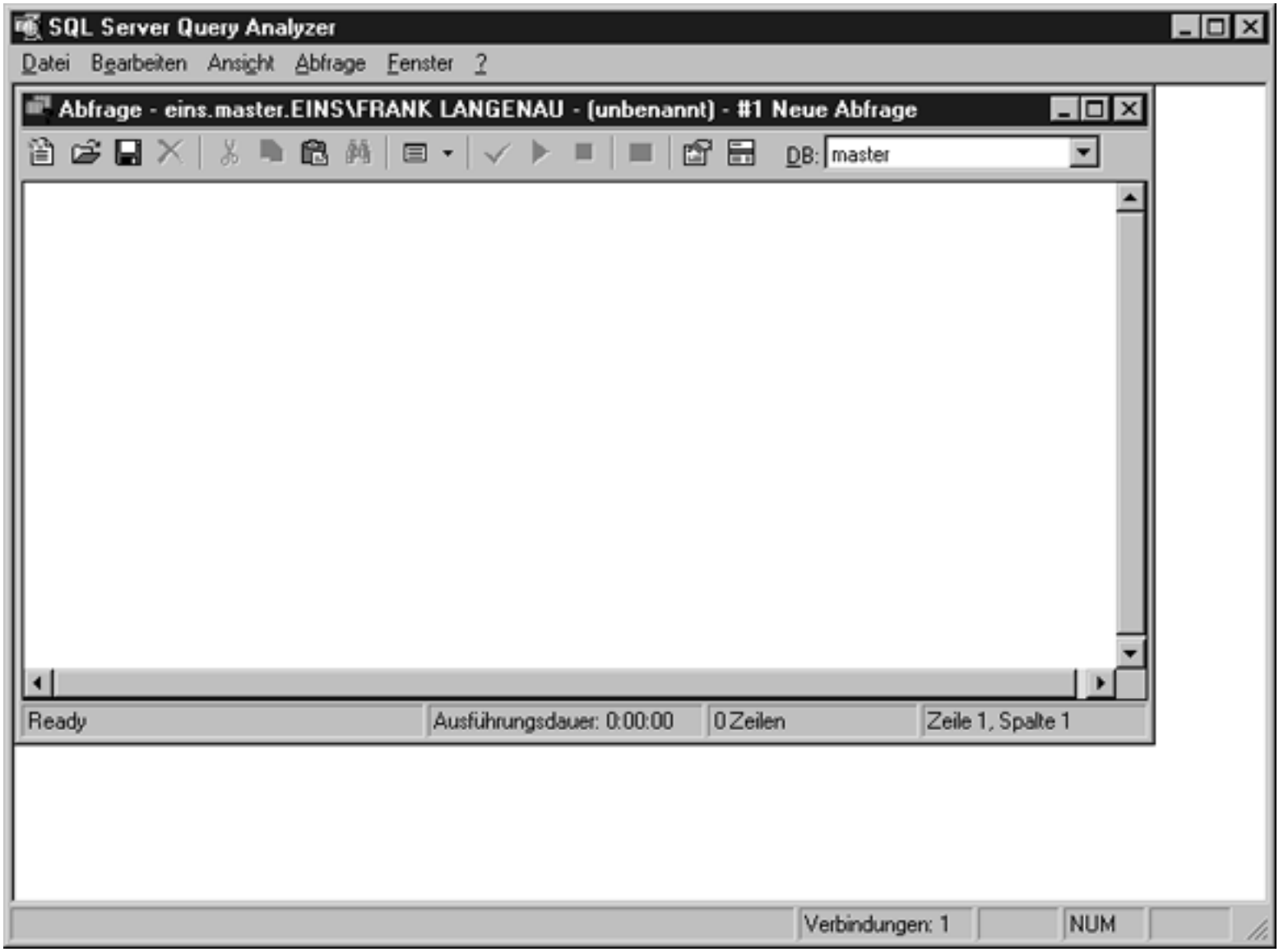
der kommentar in zeile 4 ist in den mehrzeiligen kommentar eingeschachtelt.

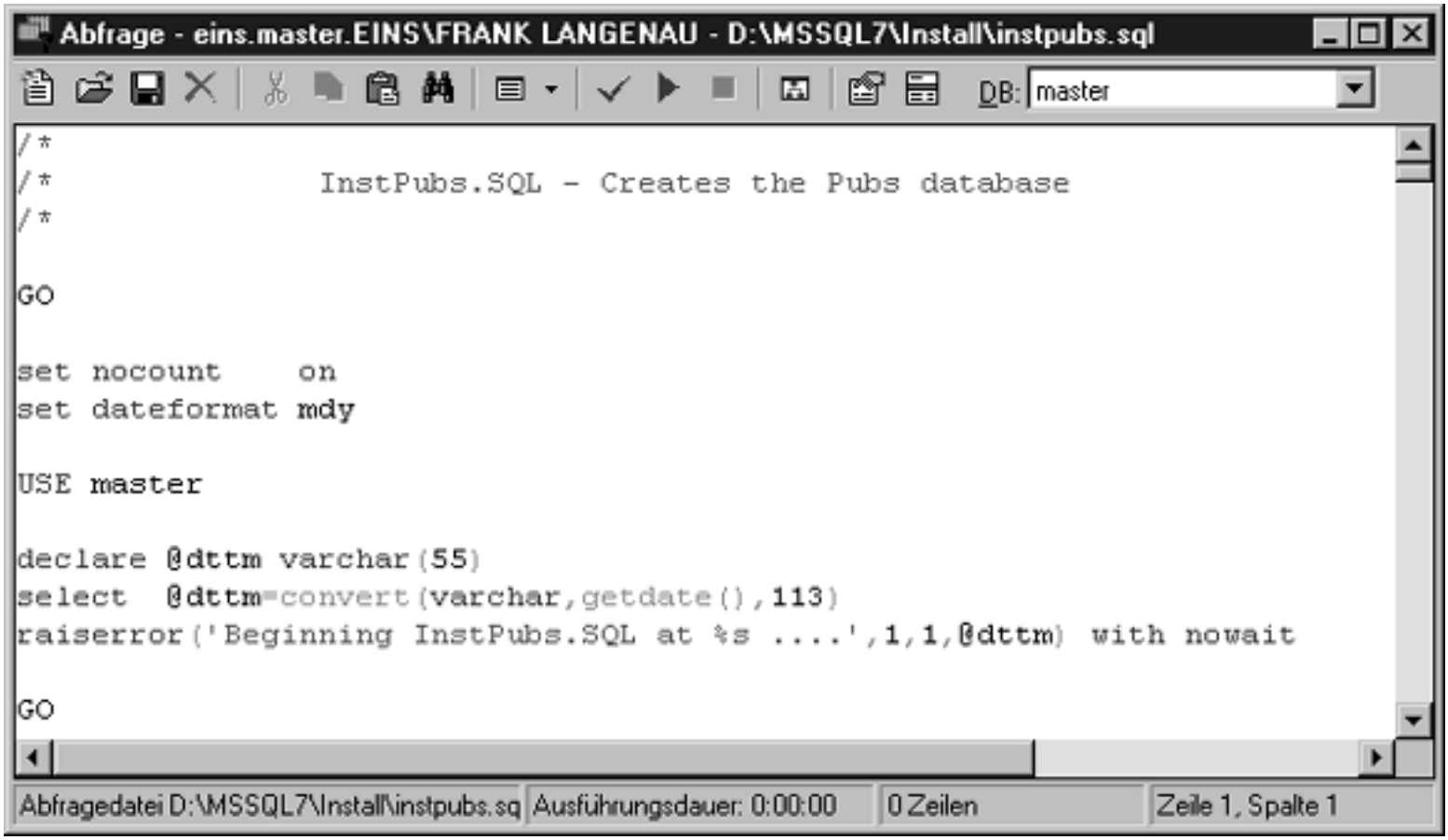
5.10 reservierte schlüsselwörter

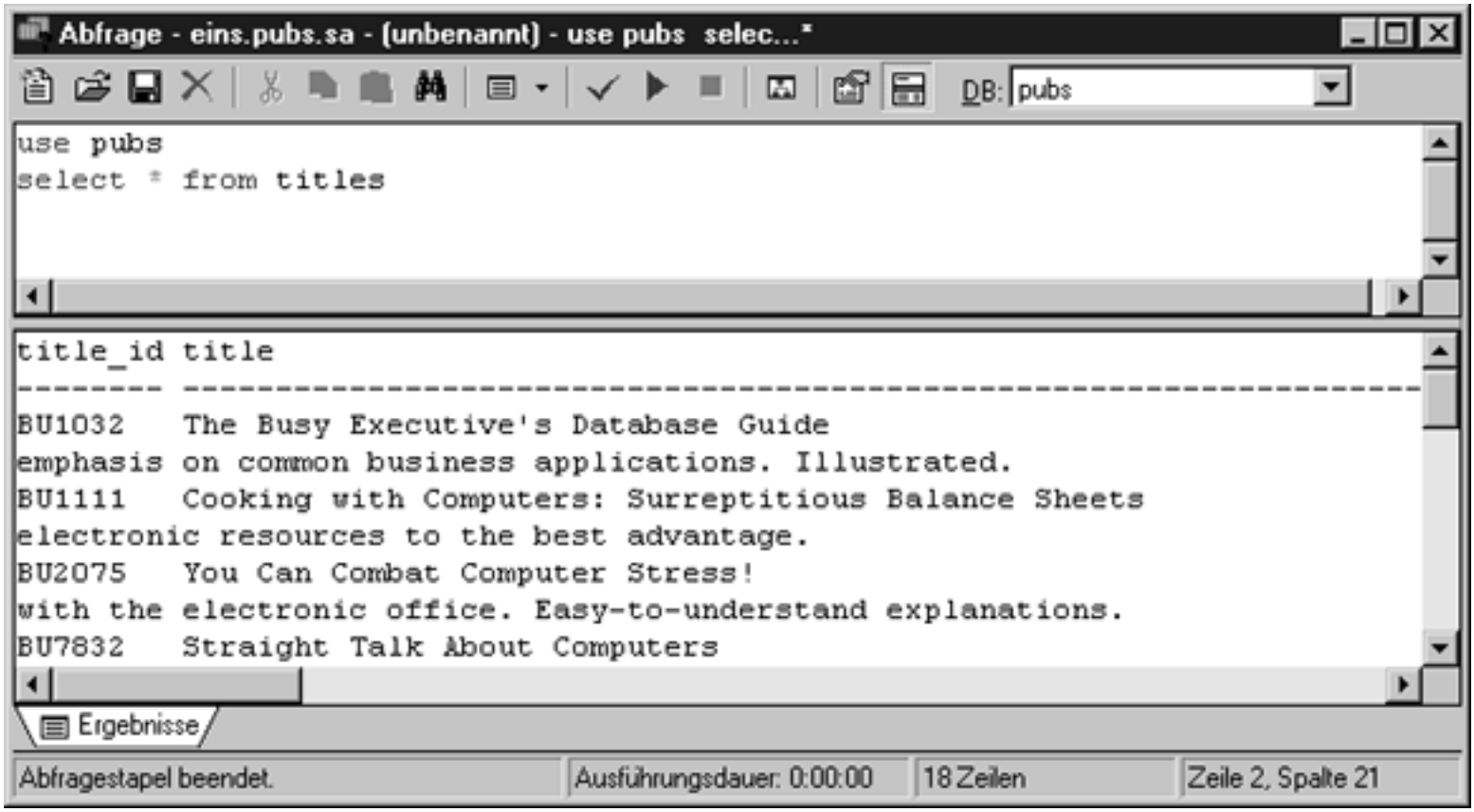
zum sprachumfang von transact-sql gehören bestimmte wörter, die sie normalerweise nicht als bezeichner für datenbankobjekte verwenden dürfen. diese reservierten schlüsselwörter sind im anhang c aufgeführt.

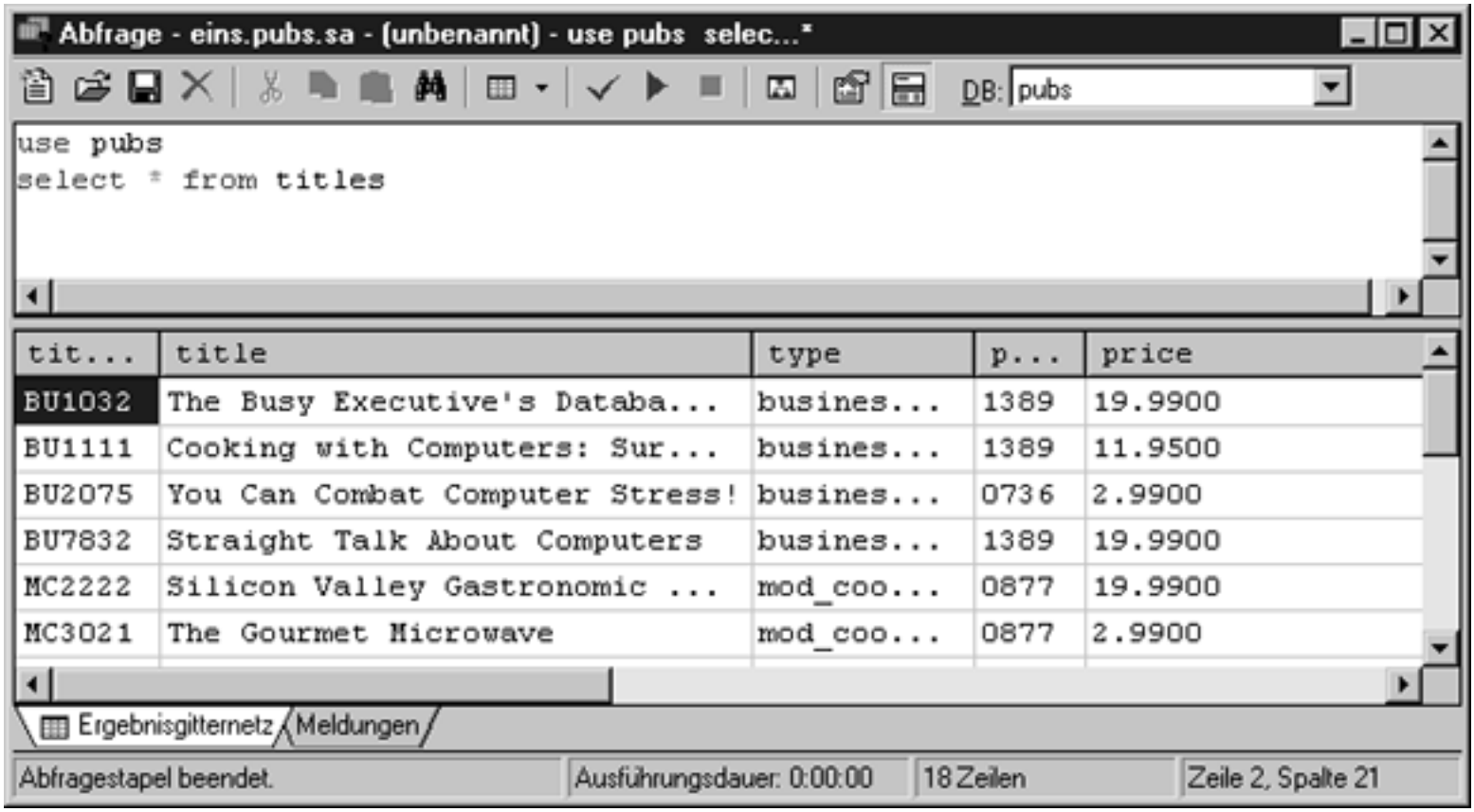
wenn sie schlüsselwörter als bezeichner verwenden wollen, müssen sie sie in anführungszeichen oder eckige klammern einschließen (abgrenzen). damit anführungszeichen gültig sind, muß die option quoted_identifier auf on eingestellt sein, was beim ole-db-provider für sql server und odbc-treiber von sql server per voreinstellung gegeben ist, bei der db-library dagegen nicht.


Sichterstellungs-Assistent - EINS
Sichterstellungs-Assistenten beenden
Sie haben die Schritte beendet, die zum Erstellen der nachfolgend angezeigten Sicht erforderlich sind. Sie können, falls gewünscht, die Anweisungen in diesem Fenster bearbeiten, um sie anzupassen.
USE [pubs]
GO
CREATE VIEW [TitelPreisView]
AS SELECT [titles].[title], [titles].[price]
FROM [titles]
< Zurück
Fertig stellen
Abbrechen


SQL Server Query Analyzer
Datei Bearbeiten Ansicht Abfrage Fenster ?
Abfrage - eins.master.EINS\FRANK LANGENAU - (unbenannt) - #1 Neue Abfrage
DB: master
Ready
Ausführungsdauer: 0:00:00
0 Zeilen
Zeile 1, Spalte 1
Verbindungen: 1
NUM


Abfrage - eins.master.EINS\FRANK LANGENAU - D:\MSSQL7\Install\instpubs.sql
DB: master
/*
/* InstPubs.SQL - Creates the Pubs database
/*
GO
set nocount on
set dateformat mdy
USE master
declare @dttm varchar(55)
select @dttm=convert(varchar,getdate(),113)
raiserror('Beginning InstPubs.SQL at %s ....',1,1,@dttm) with nowait
GO
Abfragedatei D:\MSSQL7\Install\instpubs.sq
Ausführungsdauer: 0:00:00
0 Zeilen
Zeile 1, Spalte 1


Abfrage - eins.pubs.sa - (unbenannt) - use pubs selec...*
DB: pubs
use pubs
select * from titles
title_id title
BU1032 The Busy Executive's Database Guide
emphasis on common business applications. Illustrated.
BU1111 Cooking with Computers: Surreptitious Balance Sheets
electronic resources to the best advantage.
BU2075 You Can Combat Computer Stress!
with the electronic office. Easy-to-understand explanations.
BU7832 Straight Talk About Computers
Ergebnisse
Abfragestapel beendet.
Ausführungsdauer: 0:00:00
18 Zeilen
Zeile 2, Spalte 21


Abfrage - eins.pubs.sa - (unbenannt) - use pubs selec...*
DB: pubs
use pubs
select * from titles
tit... title type p... price
BU1032 The Busy Executive's Databa... busines... 1389 19.9900
BU1111 Cooking with Computers: Sur... busines... 1389 11.9500
BU2075 You Can Combat Computer Stress! busines... 0736 2.9900
BU7832 Straight Talk About Computers busines... 1389 19.9900
MC2222 Silicon Valley Gastronomic ... mod_coo... 0877 19.9900
MC3021 The Gourmet Microwave mod_coo... 0877 2.9900
Ergebnisgitternetz
Meldungen
Abfragestapel beendet.
Ausführungsdauer: 0:00:00
18 Zeilen
Zeile 2, Spalte 21

kapitel 6 datenbanken erstellen und verwalten

6.1 die datenbank an ihrer seite

der begriff datenbanken legt einen vergleich mit banken im sinne von kreditinstituten nahe. wenn sie ihr geld auf einer bank deponieren, entspricht das geld den daten, die das kreditinstitut verwalten soll. das konto würde dann einer datenbanktabelle entsprechen, in der die einzelnen buchungen zeilenweise eingetragen sind. einen direkten zugriff auf ihr konto haben sie nicht, d.h., sie können nicht einfach den kontostand um ein paar mark erhöhen. allerdings können sie sich anhand der auszüge darüber informieren, was sich auf ihrem konto getan hat. die funktion des kontoauszugs läßt sich in einer datenbank mit einer sicht vergleichen. (das beispiel hinkt etwas, weil sie mit aktualisierbaren sichten auch die zugrundeliegenden tabellen ändern können.)

eine datenbank stellt praktisch den rahmen für alle zu speichernden informationen dar. damit sie verstehen, wie man welche informationen in einer datenbank speichern kann, geht dieses kapitel auch auf die grundsätzlichen elemente einer datenbank ein.

6.2 datenbanken erstellen

in diesem kapitel erstellen sie die sehr einfache datenbank lotto, die die ziehungsergebnisse der samstagsziehung 6 aus 49 seit 1956 aufnehmen soll.

die datenbank lotto ist bewußt einfach gehalten. dadurch lassen sich die beispiele schnell nachvollziehen, ohne daß komplexe strukturen zu erstellen sind. auf der begleit-cd finden sie die ziehungsergebnisse im textformat. in den folgenden kapiteln kommt die datenbank lotto verschiedentlich zum einsatz, um die jeweiligen themen mit einfachen beispielen zu vertiefen.

kritiker mögen einräumen, daß auch ein einfaches excel-tabellenblatt genügt hätte. hier geht es aber darum, grundlegende abläufe zu verdeutlichen. der weg vom einfachen zum komplizierten läßt sich dann bestimmt leichter gehen.

viele beispiele greifen auch auf die datenbank pubs zurück, die zum lieferumfang von sql server gehört.

um eine neue datenbank zu erstellen, sind folgende angaben erforderlich:

- name der datenbankdatei und der transaktionsprotokolldatei
- speicherort dieser dateien
- anfängliche größe der dateien

sql server speichert eine datenbank in drei betriebssystemdateien (siehe tabelle 6.1).

dateityp	verwendung	standarderweiterung
primär	ausgangspunkt für jede datenbank. nimmt zeiger auf andere dateien der betreffenden datenbank auf. zu jeder datenbank gehört eine primäre datendatei.	.mdf
sekundär	nimmt daten auf, die nicht in primären dateien untergebracht werden. je nach datenbankkonfiguration kann die datenbank keine, eine oder mehrere sekundäre datendateien haben.	.ndf
protokoll	enthält das protokoll für die datenbank. zu jeder datenbank gehört mindestens eine protokolldatei. datenbanken können aber auch mehrere protokolldateien anlegen.	.ldf

tabelle 6.1: betriebssystemdateien in sql server 7.0

6.2.1 datenbankerstellungs-assistent

der datenbankerstellungs-assistent hilft ihnen mit detaillierten erläuterungen beim erstellen einer datenbank.

die einzelnen schritte des assistenten lassen sich eher als »gemütliche« reise durch das erstellen einer datenbank bezeichnen. schneller geht es, wenn sie eine datenbank direkt über den enterprise manager erstellen. darauf geht der entsprechende abschnitt weiter unten in diesem kapitel ein.

führen sie die folgenden schritte aus:

1. um den assistenten aufzurufen, erweitern sie die konsolenstruktur mindestens bis zum server, wählen dann aus dem menü **extras** des enterprise managers den befehl **assistenten** oder klicken in der symbolleiste auf **assistenten auswählen**. im dialogfeld **assistent auswählen** erweitern sie die kategorie **datenbank**, markieren den eintrag *datenbankerstellungs-assistent* und klicken auf **ok**. sql server ruft den datenbankerstellungs-assistenten auch auf, wenn sie ausgehend von der taskpad-ansicht **datenbanklösung einrichten** und danach **datenbank erstellen** wählen.

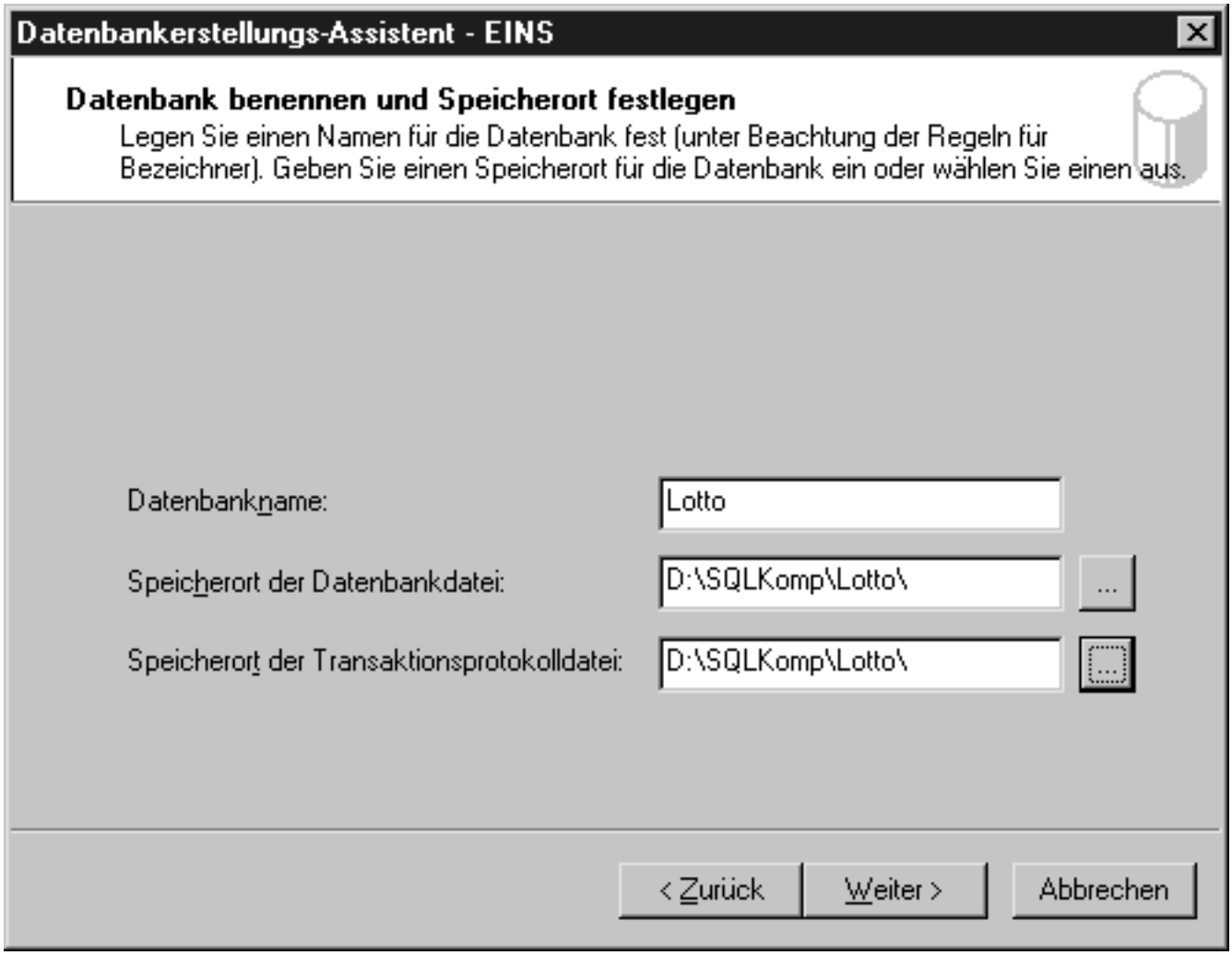


abbildung 6.1: das zweite dialogfeld des datenbankerstellungs-assistenten

2. klicken sie im startdialogfeld auf **weiter**. im zweiten dialogfeld des datenbankerstellungs-assistenten legen sie den namen der datenbank sowie den speicherort für die datenbankdatei und die transaktionsprotokolldatei fest (siehe abbildung 6.1). der speicherort darf sich nicht auf einem komprimierten laufwerk befinden.
3. im nächsten dialogfeld legen sie den namen der datenbankdatei und deren anfängliche größe fest. sql server gibt lotto_data als namen und 1 mbyte als größe vor. diese werte können sie übernehmen, obwohl die größe für das beispiel mehr als reichlich bemessen ist. klicken sie auf **weiter**. (außerdem kann eine datenbank nicht kleiner als die datenbank model werden.)

der name der datenbank lautet lotto. unter diesem namen führen sie alle operationen bezüglich der datenbank aus - zum beispiel daten einfügen, aktualisieren und löschen oder sichten erstellen. die namen der datenbank*datei* und der (noch festzulegenden) transaktionsprotokoll*datei* sind vollkommen unabhängig vom eigentlichen namen der datenbank. mit den dateinamen haben sie nur zu tun, wenn sie operationen auf der ebene des dateisystems ausführen. es empfiehlt sich allerdings, aussagekräftige und einheitliche namenskonventionen zu befolgen, damit sie gegebenenfalls die dateien auf der ebene des dateisystems dem in sql server bekannten namen der datenbank zuordnen können.

4. das vierte dialogfeld bietet verschiedene optionen, die sich auf das wachstum der datenbankdateien beziehen (siehe abbildung 6.2). eines der hervorstechenden merkmale von sql server 7.0 ist die möglichkeit, datenbank- und transaktionsprotokolldateien bei bedarf automatisch zu vergrößern.

wenn sie die option *datenbankdateien nicht automatisch vergrößern* wählen, riskieren sie eine fehlermeldung, falls die größe der datenbank nicht mehr ausreichend platz für neue daten bietet (was für das beispiel zwar nicht zutreffen wird, aber bei »realen« anwendungen wie etwa einer datenbank für bestellungen schnell passieren kann). der abschnitt »datenbanken in der größe ändern« weiter unten in diesem kapitel beschäftigt sich näher mit diesem thema.

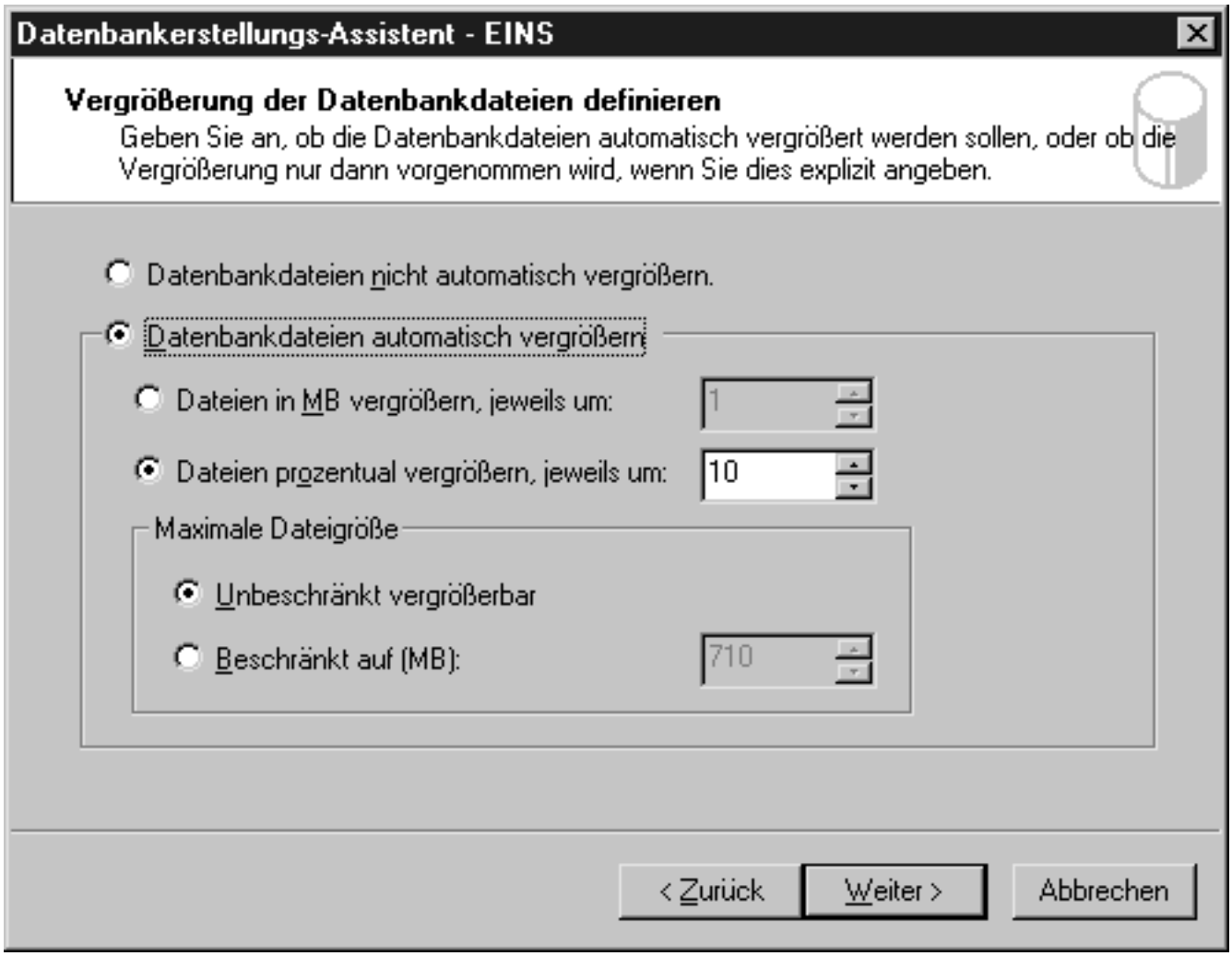


abbildung 6.2: wachstum der datenbankdateien festlegen

5. in den beiden nächsten dialogfeldern des assistenten legen sie analog zum dritten und vierten dialogfeld optionen für die transaktionsprotokolldatei fest.
6. der letzte schritt des assistenten (siehe abbildung 6.3) faßt die von ihnen festgelegten einstellungen zusammen. klicken sie auf **fertigstellen**, um die datenbank lotto anlegen zu lassen.

bild

abbildung 6.3: das letzte dialogfeld des assistenten bringt eine übersicht der gewählten einstellungen

7. abschließend bringt der assistent eine meldung, daß die datenbank erfolgreich erstellt wurde. wenn sie mit **ok** bestätigt haben, erscheint die frage, ob sie einen wartungsplan für die datenbank erstellen möchten. klicken sie an dieser stelle auf **nein**. mit dem thema wartungspläne beschäftigt sich kapitel 20.

6.2.2 datenbanken mit enterprise manager erstellen

falls sie die datenbank lotto mit dem datenbankerstellungs-assistenten angelegt haben und das beispiel in diesem abschnitt nachvollziehen möchten, können sie die datenbank zunächst löschen. markieren sie dazu die datenbank in der konsolenstruktur, drükken sie **ç**, und bestätigen sie den löschvorgang.

mit dem enterprise manager erstellen sie eine neue datenbank praktisch in einem schritt - d.h. in einem dialogfeld mit zwei registerkarten, auf denen sie die gleichen optionen festlegen können, die sie beim datenbankerstellungs-assistenten kennengelernt haben.

erweitern sie in der konsolenstruktur den server, auf dem sie die datenbank erstellen wollen, klicken sie mit der rechten maustaste auf **datenbanken**, und wählen sie den befehl **neue datenbank** aus dem kontextmenü. alternativ können sie auch den befehl **neue datenbank** über das menü **vorgang** aufrufen. haben sie die konsolenstruktur nur bis zum server erweitert, wählen sie **vorgang / neu / datenbank**.

das dialogfeld **datenbankeigenschaften** enthält zwei registerkarten: **allgemein** (siehe abbildung 6.4) und **transaktionsprotokoll** (siehe abbildung 6.5).

bild

abbildung 6.4: auf der registerkarte allgemein legen sie den namen der datenbank und die eigenschaften der datenbankdatei fest

auf der registerkarte **allgemein** müssen sie zunächst einen namen für die datenbank festlegen. während sie den namen eintippen, füllt sql server automatisch die spalten **dateiname** und **speicherort** im abschnitt **datenbankdateien**. die vorgaben müssen sie nicht übernehmen. tragen sie einfach andere werte in die felder ein. wenn sie in der spalte **speicherort** auf die schaltfläche mit den drei punkten klicken, erscheint das dialogfeld **datenbankdatei suchen**. markieren sie hier den pfad zur datenbankdatei, und klicken sie auf **ok**. im abschnitt **dateieigenschaften** legen sie die größenoptionen der datenbankdatei fest. analoge einstellungen nehmen sie auf der registerkarte **transaktionsprotokoll** für die transaktionsprotokolldatei vor.

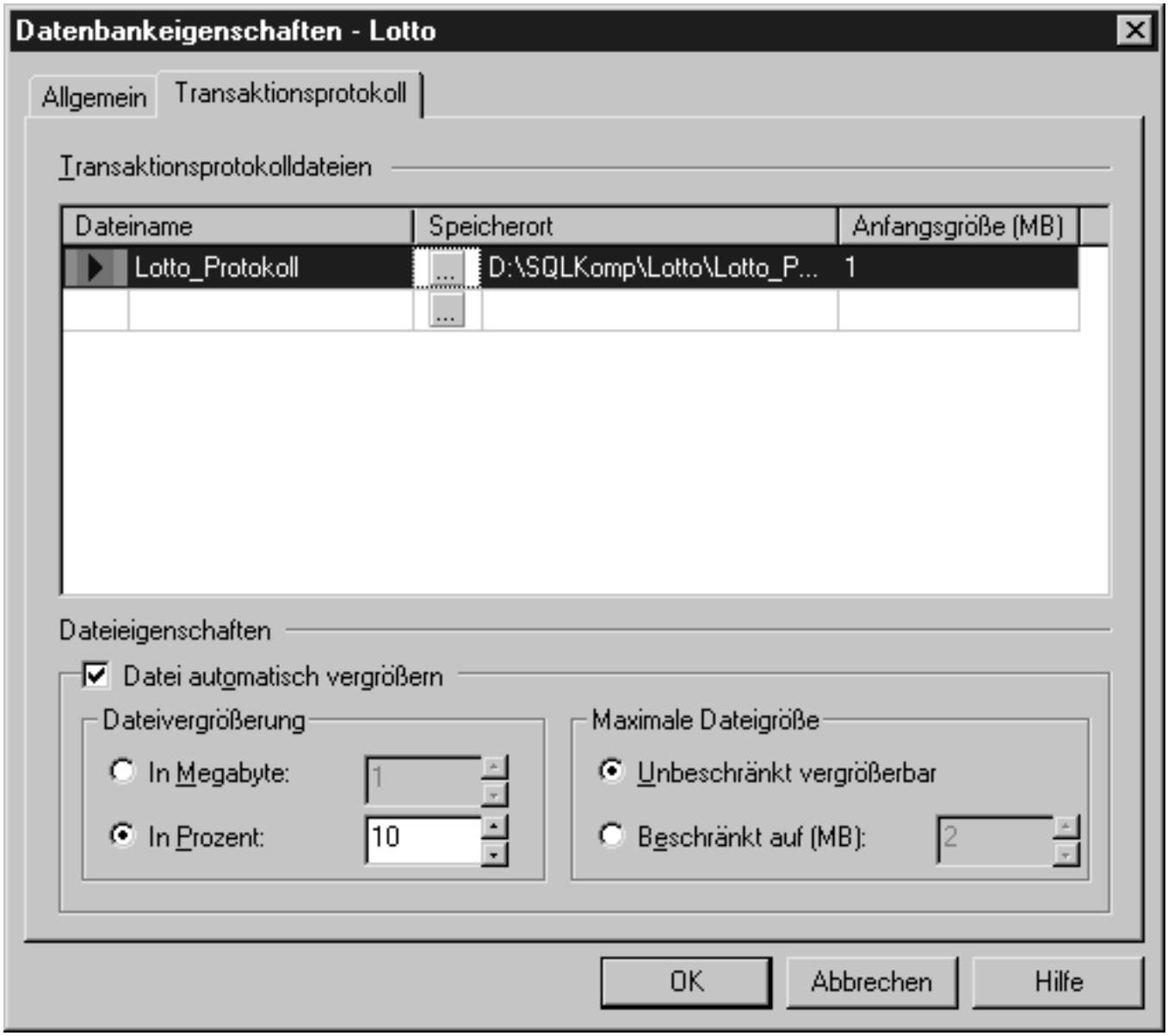


abbildung 6.5: auf der registerkarte transaktionsprotokoll stellen sie die eigenschaften der transaktionsprotokolldatei ein

die vorgegebenen namen für die datenbankdatei lauten datenbank_daten und für die transaktionsprotokolldatei datenbank_protokoll. der datenbankerstellungs-assistent liefert dagegen die namen datenbank_data bzw. datenbank_log. diese unterschiede haben keine tiefgreifende bedeutung. wahrscheinlich wurde die lokalisierung an manchen stellen nicht konsequent umgesetzt.

klicken sie abschließend auf **ok**, um die datenbank erstellen zu lassen.

6.2.3 transact-sql (create database)

die anweisung create database erfüllt zwei aufgaben: zum einen läßt sich damit eine neue datenbank erstellen, zum anderen eine datenbank mit den dateien einer bereits erstellten datenbank verbinden. die vollständige syntax der anweisung create database lautet:

syntax:

```
create database datenbankname
[ on [primary]
        [ <datendateiliste>]
        [, <dateigruppe>]
]
[ log on { <datendatei> } ]
[ for load | for attach ]

<datendatei> ::=

  ( [ name = logischerdateiname, ]
  filename = physischerdateiname'
  [, size = größe]
  [, maxsize = { maxgröße | unlimited } ]
  [, filegrowth = zuwachs] ) [,...n]

<dateigruppe> ::=
filegroup dateigruppenname <datendatei> [,...n]
```

tabelle 6.2 erläutert die argumente.

argument	bedeutung
datenbankname	name der neuen datenbank. innerhalb eines servers müssen datenbanknamen eindeutig sein und den regeln für bezeichner entsprechen. der datenbankname kann eine länge von 128 zeichen haben, sofern kein logischer name für das protokoll spezifiziert wird. gibt man keinen logischen protokolldateinamen an, generiert sql server einen logischen namen, indem ein entsprechendes suffix an den datenbanknamen angehängt wird. in diesem fall ist die länge des datenbanknamens auf 123 zeichen beschränkt, damit die generierten logischen dateinamen nicht die länge von 128 zeichen überschreiten.
on	wenn dieses schlüsselwort angegeben ist, folgen die spezifikationen der (durch komma getrennten) datendateien für die primäre dateigruppe der datenbank. daran kann sich eine liste von benutzerdateigruppen und deren dateien anschließen.
primary	weist darauf hin, daß die zugehörige dateiliste die primärdatei spezifiziert
log on	auf diese schlüsselwörter folgt eine durch komma getrennte liste von explizit definierten protokolldateien. fehlt diese angabe, erzeugt sql server automatisch eine protokolldatei mit einem standardnamen.

for load	aus gründen der abwärtskompatibilität vorhanden und in sql server 7.0 nicht erforderlich
for attach	datenbank wird aus vorhandenem satz von betriebssystemdateien angefügt. die erste primärdatei muß in *dateiliste* angegeben sein.
name	logischer name für die durch *datendatei* spezifizierte datei.
logischerdateiname	name für die datendatei innerhalb von transact-sql-anweisungen. der logische dateiname kann, muß aber nicht mit dem physischen dateinamen übereinstimmen.
filename	betriebssystemdateiname für die durch *datendatei* spezifizierte datei.
physischerdateiname	pfad und dateiname, den das betriebssystem beim erstellen der durch *datendatei* spezifizierten datei verwendet. der logische dateiname kann, muß aber nicht mit dem physischen dateinamen übereinstimmen.
size	größe der in *datendatei* spezifizierten datei. fehlt die angabe, nimmt sql server die standardgröße der primären datei der systemdatenbank model.
größe	anfangsgröße der in *datendatei* spezifizierten datei.
maxsize	maximalgröße der in *datendatei* spezifizierten datei.
maxgröße	maximalgröße, auf die die in *datendatei* spezifizierte datei vergrößert werden darf.
unlimited	datei kann vergrößert werden, bis der speicherplatz erschöpft ist.
filegrowth	schrittweite für die vergrößerung der *datendatei.*
zuwachs	zusätzlicher speicherplatz, den die *datendatei* bei vergrößerung erhält.
filegroup	definiert datendateien für die primäre dateigruppe.

tabelle 6.2: argumente des befehls create database

der zusatz [n] gibt an, daß das entsprechende element mehrfach vorkommen kann.

beispiele:

die folgende anweisung erstellt die datenbank lotto mit standardeinstellungen für alle parameter:

```
use master
go
create database lotto
```

wenn sie die daten- und protokolldateien - wie in den beispielen mit dem

datenbankerstellungs-assistenten und dem enterprise manager gezeigt - im verzeichnis d:\sqlkomp\lotto\ unterbringen möchten, schreiben sie die nachstehende anweisung:

```
use master
go

create database lotto
on
( name = lottodat,
  filename = 'd:\sqlkomp\lotto\lottodat.mdf',
  size = 25mb)
log on
( name = lottolog,
  filename = 'd:\sqlkomp\lotto\lottolog.ldf',
  size = 10 mb)
```

die anweisung legt außerdem die größe der datendatei mit 25 mbyte und die größe der protokolldatei mit 10 mbyte fest. das ist zwar mit kanonen auf spatzen geschossen, wird aber für beispiele weiter hinten in diesem kapitel gebraucht, wenn es um die größenänderung von datenbanken geht.

als bestätigung liefert sql server die meldung:

```
create database-prozess reserviert 25.00 mb auf datenträger
'lottodat'.
create database-prozess reserviert 10.00 mb auf datenträger
'lottolog'.
```

6.3 datenbankeigenschaften ändern

hin und wieder ist es erforderlich, die eigenschaften einer datenbank an neue gegebenheiten anzupassen. die folgenden abschnitte zeigen, wie sie informationen über eine datenbank abrufen und die verschiedenen optionen anzeigen und ändern können.

6.3.1 datenbankinformationen anzeigen

im enterprise manager können sie einen überblick über die technischen daten einer datenbank anzeigen. wenn sie die gewünschte datenbank in der konsolenstruktur markieren und im menü **ansicht** den befehl **taskpad** aktivieren, erscheint im rechten fensterbereich des enterprise managers die taskpad-ansicht der datenbank (siehe abbildung 6.6).

bild

abbildung 6.6: die taskpad-ansicht einer datenbank im enterprise manager

in der taskpad-ansicht lassen sich auch ausgewählte aktionen für eine datenbank aktivieren. wenn sie den mauszeiger auf die einträge in den grau unterlegten blöcken setzen, wird der jeweilige befehl hervorgehoben. um die aktion auszuführen, klicken sie einfach auf den eintrag.

6.3.2 datenbankeigenschaften mit enterprise manager ändern

im dialogfeld **eigenschaften** einer datenbank lassen sich detaillierte informationen zur jeweiligen datenbank anzeigen und optionen festlegen. das dialogfeld rufen sie über den befehl **datenbankeigenschaften** in der taskpad-ansicht auf. wenn die taskpad-ansicht nicht aktiv ist, klicken sie mit der rechten maustaste auf die datenbank in der konsolenstruktur und wählen den befehl **eigenschaften** aus dem kontextmenü.

größeneinstellungen

auf der registerkarte **allgemein** (siehe abbildung 6.7) können sie die größeneinstellungen für die datenbank anpassen. die größeneinstellungen für das transaktionsprotokoll nehmen sie auf der registerkarte **transaktionsprotokoll** (siehe abbildung 6.8) vor.

die vergrößerung der daten- oder protokolldateien können sie von sql server automatisch vornehmen lassen oder selbst manuell einstellen.

wenn das kontrollkästchen **datei automatisch vergrössern** eingeschaltet ist, nimmt ihnen sql server alle arbeiten ab, wenn die datendatei bei bedarf wachsen muß. parallel dazu können sie auch festlegen, ob die datendatei um einen bestimmten prozentsatz oder um eine feste größe wachsen soll. der mindestwert beträgt 1 mbyte. weiterhin läßt sich die maximalgröße beschränken. wenn die datenbank unbeschränkt vergrößerbar ist, kommt es zu einer fehlermeldung, falls der verfügbare speicher aufgebraucht ist.

im eigenschaftsdialogfeld können sie die größe der daten- und protokolldateien auch manuell festlegen. das kann nach zwei methoden geschehen:

- tragen sie den wert für die größe der datei direkt in das jeweilige feld der spalte **reservierter speicher** ein.
- weisen sie der datenbank oder dem protokoll eine neue - physikalische - datei zu. im abschnitt **datenbankdateien** bzw. **transaktionsprotokolldateien** tragen sie einfach in die leeren zeilen eine oder mehrere zusätzliche dateinamen ein.

die zweite methode bietet sich an, wenn sich eine datenbank über mehrere laufwerke erstrecken soll.

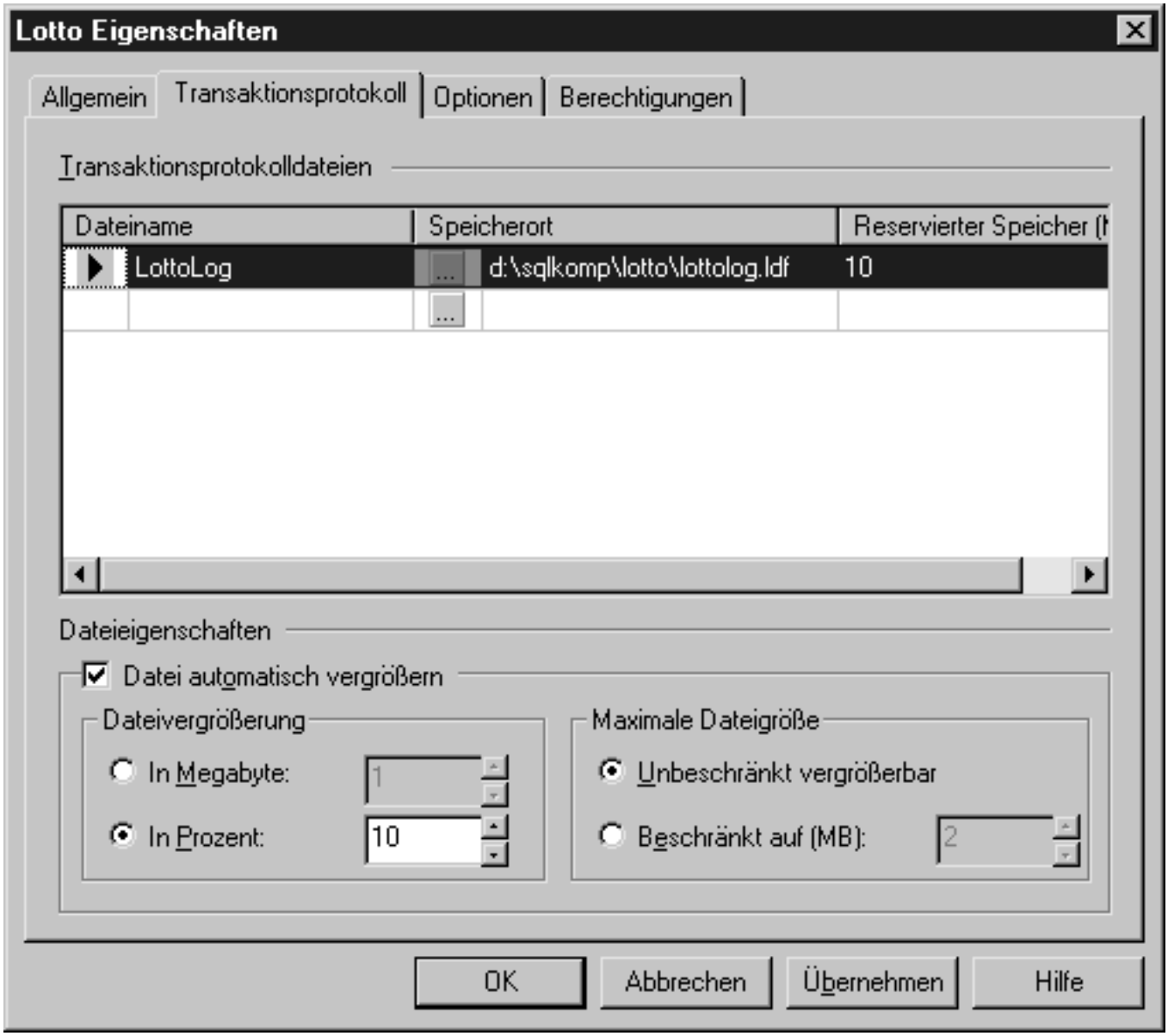


abbildung 6.7: die registerkarte transaktionsprotokoll

bild

abbildung 6.8: die registerkarte allgemein des eigenschaftsdialogfelds einer datenbank

datenbankoptionen

die optionen für eine datenbank sind auf der registerkarte **optionen** des dialogfelds **eigenschaften** zusammengefaßt (siehe abbildung 6.9).

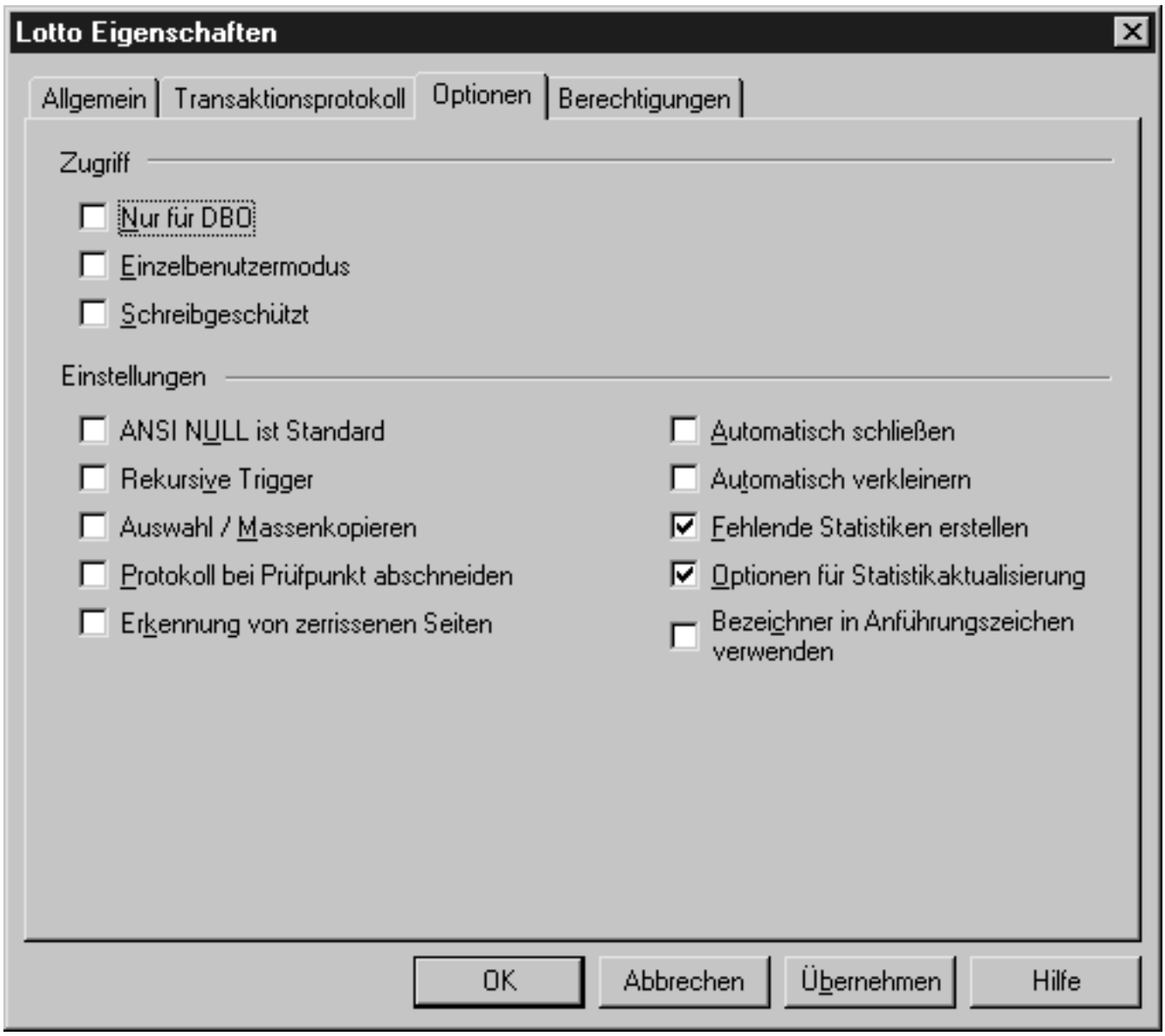


abbildung 6.9: die registerkarte optionen

abbildung 6.9 zeigt die standardeinstellungen der optionen. die folgenden erläuterungen beziehen sich auf die aktivierten einstellungen (kontrollkästchen eingeschaltet). in klammern sind die bezeichnungen angegeben, die sie bei ausführung der gespeicherten prozedur sp_dboption erhalten. (bei der behandlung der transact-sql-anweisungen zum einstellen der konfigurationsoptionen weiter unten in diesem kapitel gibt tabelle 6.3 noch einmal einen überblick über diese optionen.)

nur für dbo (dbo use only)

auf die datenbank können nur der datenbankbesitzer und der systemadministrator zugreifen.

einzelbenutzermodus (single user)

mit der datenbank kann sich nur ein benutzer zu einem bestimmten zeitpunkt verbinden. bereits bestehende verbindungen bleiben erhalten. neuen benutzern wird der zugang verweigert.

schreibgeschützt (read only)

der inhalt der datenbank läßt sich zwar anzeigen, aber nicht ändern.

ansi null ist standard (ansi null default)

für alle benutzerdefinierten datentypen oder spalten, die beim erstellen oder ändern einer tabelle nicht explizit als not null definiert sind, gilt als standardeinstellung die null-zulässigkeit. in sql server ist die voreinstellung not null.

wenn sie in einer create table-anweisung die null-zulässigkeit einer spalte mit null oder not null explizit festlegen, setzt das die standardeinstellung außer kraft.

rekursive trigger (recursive triggers)

läßt den rekursiven aufruf von triggern zu. wenn zum beispiel für eine tabelle ein trigger definiert ist, der daten in dieser tabelle verändert, führt das zum erneuten aufruf dieses triggers (direkte rekursion). ändert ein trigger daten in einer anderen tabelle und löst daraufhin einen trigger in der zweiten tabelle aus, kommt es zu indirekter rekursion, wenn der trigger der zweiten tabelle daten in der ersten tabelle ändert und in der folge den trigger der ersten tabelle auslöst.

auswahl / massenkopieren (select into/bulkcopy)

eine datenbank kann auch nicht protokollierte operationen akzeptieren. nach einer derartigen operation kann keine sicherung des transaktionsprotokolls erstellt werden. diese option sollten sie deshalb bei produktionsdatenbanken deaktivieren, da eine wiederherstellung der datenbank anhand des transaktionsprotokolls sonst nicht möglich ist.

in der voreinstellung (false) sind die operationen select into und schnelles massenkopieren mit bcp nicht möglich. die mit bcp durchgeführten operationen werden protokolliert.

protokoll bei prüfpunkt abschneiden (trunc. log on chkpt.)

das transaktionsprotokoll wird automatisch abgeschnitten, sobald das protokoll zu 70 prozent gefüllt ist. mit dieser option kann man verhindern, daß sql server das transaktionsprotokoll vergrößert.

wenn sie ihre sicherungsstrategie auf dem transaktionsprotokoll aufbauen, sollte diese option deaktiviert bleiben.

erkennung von zerrissenen seiten (torn page detection)

sql server erkennt unvollständige e/a-operationen, die durch störungen im betriebsablauf hervorgerufen wurden, zum beispiel durch einen stromausfall beim aktualisieren einer datenbank. wenn eine zerrissene seite erkannt wird, müssen sie die datenbank aus einer sicherung wiederherstellen.

automatisch schließen (autoclose)

sql server schließt die datenbank automatisch, sobald sich alle benutzer abgemeldet haben und keine prozesse mehr in der datenbank laufen. die voreinstellung bei der desktop edition ist true, bei allen anderen editionen false.

wenn eine datenbank ständig in benutzung ist oder anwendungen immer wieder darauf zugreifen müssen, sollten sie diese option auf false setzen, da sich sonst der zusätzliche aufwand für das öffnen und schließen der datenbank negativ auf die reaktionszeit auswirkt.

automatisch verkleinern (autoshrink)

die daten- und protokolldateien werden falls möglich bei einer periodischen prüfung automatisch verkleinert, um den platz auf der festplatte zu reduzieren. die voreinstellung bei der desktop edition ist true, bei allen anderen editionen false. protokolldateien werden nur dann automatisch verkleinert, wenn sie die option *protokoll bei prüfpunkt abschneiden* auf true setzen.

wenn die option auf true gesetzt ist, kann sich das negativ auf die leistung der datenbank auswirken. wird eine verkleinerung der datenbank erforderlich, setzen sie die option auf true, und nachdem die datenbank verkleinert wurde, wieder zurück auf false.

fehlende statistiken erstellen (auto create statistics)

sql server erstellt für spalten, die in einem prädikat verwendet werden, automatisch statistiken. dadurch verbessert sich die abfrageleistung.

prädikate sind ausdrücke, die true, false oder unknown ergeben. man verwendet prädikate in den suchbedingungen von where- und having-klauseln und in den verknüpfungsbedingungen von from-klauseln.

optionen für statistikaktualisierung (auto update statistics)

bei änderungen der tabelle werden vorhandene statistiken automatisch auf den neuesten stand gebracht.

bezeichner in anführungszeichen verwenden (quoted identifier)

wenn diese option auf true gesetzt ist, kann man bezeichner in doppelte anführungszeichen einschließen. die gesamte zeichenfolge gilt dann als objektbezeichner und kann schlüsselwörter von transact-sql sowie andere zeichen, die sonst nicht in transact-sql-anweisungen zulässig sind, enthalten. literale sind in diesem fall in einfache anführungszeichen zu schreiben.

berechtigungen

abbildung 6.10 zeigt die registerkarte **berechtigungen** des dialogfelds **eigenschaften**. die abkürzung gp in der spalte **gp erstellen** steht für »gespeicherte prozedur«.

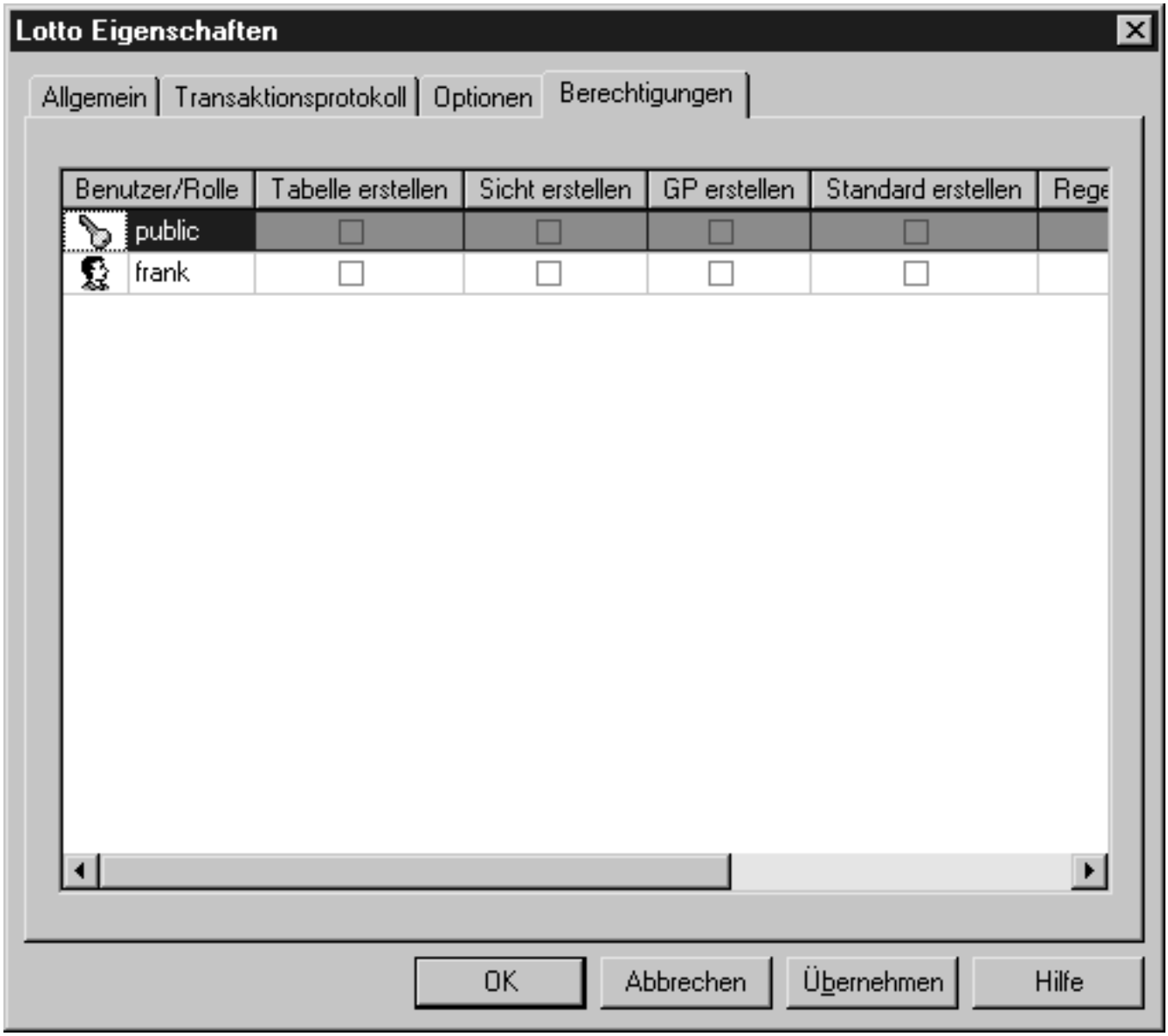


abbildung 6.10: die registerkarte berechtigungen

6.3.3 datenbanken verkleinern

wenn sie den platzbedarf für eine datenbank auf der festplatte reduzieren und die datenbank verkleinern möchten, können sie das manuell erledigen oder sql server die verkleinerung automatisch vornehmen lassen. damit sql server die datenbank automatisch verkleinern kann, müssen sie die option *automatisch verkleinern* (autoshrink) auf true setzen (siehe dazu abbildung 6.9).

eine datenbank läßt sich höchstens auf die größe verkleinern, mit der sie erstellt wurde. außerdem kann sie nicht kleiner als die datenbank model werden. sql server installiert die datenbank model mit einer größe von 0,75 mbyte (= 786432 byte). die zugehörige protokolldatei modellog.ldf weist die gleiche größe auf. als standardgröße gibt sql server beim erstellen einer datenbank 1 mbyte vor. wenn sie eine datenbank zum beispiel mit einer größe von 25 mbyte erstellen, können sie die größe später nicht auf 10 mbyte reduzieren.

im dialogfeld **eigenschaften** einer datenbank können sie zwar einen wert für die neue größe eingeben, dieser muß aber größer als der alte wert sein. andernfalls erhalten sie eine fehlermeldung (siehe

abbildung 6.11).

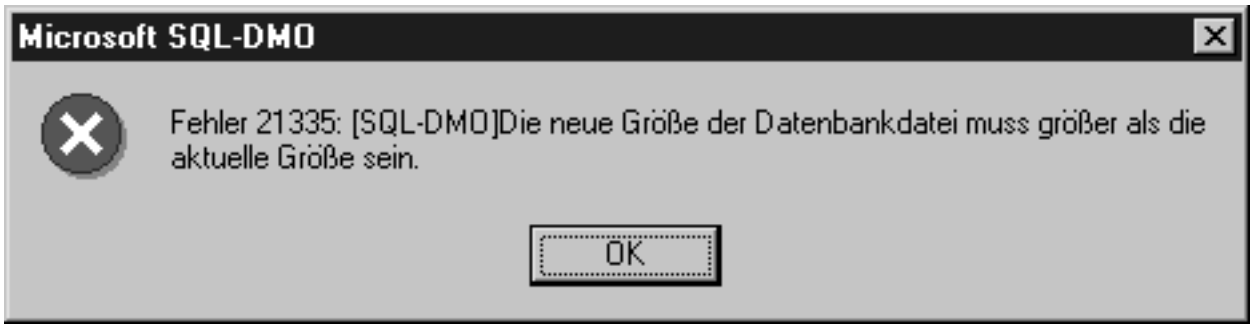


abbildung 6.11: wenn sie im dialogfeld eigenschaften einen kleineren wert für die größe eingeben, erhalten sie diese fehlermeldung

bevor sie im folgenden beispiel die datenbank lotto verkleinern, können sie sich über die derzeitige größe der datenbank in der taskpad-ansicht informieren (siehe abbildung 6.12). wenn sie die datenbank mit einer größe der datendatei von 25 mbyte und einer größe des protokolls von 10 mbyte erstellt haben, wird eine größe von 35 mbyte ausgewiesen. unter diesen wert, mit dem sie die datenbank erstellt haben, läßt sich die datenbank nicht verkleinern, obwohl sie wie im beispiel noch nicht einmal 1 mbyte benötigt.

bild

abbildung 6.12: die taskpad-ansicht zeigt die größe der datenbank an

damit sie sich davon überzeugen können, daß sql server die datenbank tatsächlich verkleinert, setzen sie zunächst die größe der datendatei auf beispielsweise 50 mbyte und die größe der transaktionsprotokolldatei auf 30 mbyte fest. tragen sie diese werte im dialogfeld **eigenschaften** in die spalten **reservierter speicher** ein (siehe dazu abbildung 6.7 für die datendatei und abbildung 6.8 für die transaktionsprotokolldatei weiter vorn im abschnitt »größeneinstellungen«).

die taskpad-ansicht gibt die summe der größen für datendatei und transaktionsprotokolldatei an.

um eine datenbank mit enterprise manager zu verkleinern, führen sie die folgenden schritte aus:

1. erweitern sie die konsolenstruktur bis zur gewünschten datenbank.
2. klicken sie mit der rechten maustaste auf die datenbank, und wählen sie **alle aufgaben / datenbank verkleinern** aus dem kontextmenü. (in der taskpad-ansicht der datenbank ist der befehl **datenbank verkleinern** direkt verfügbar.)

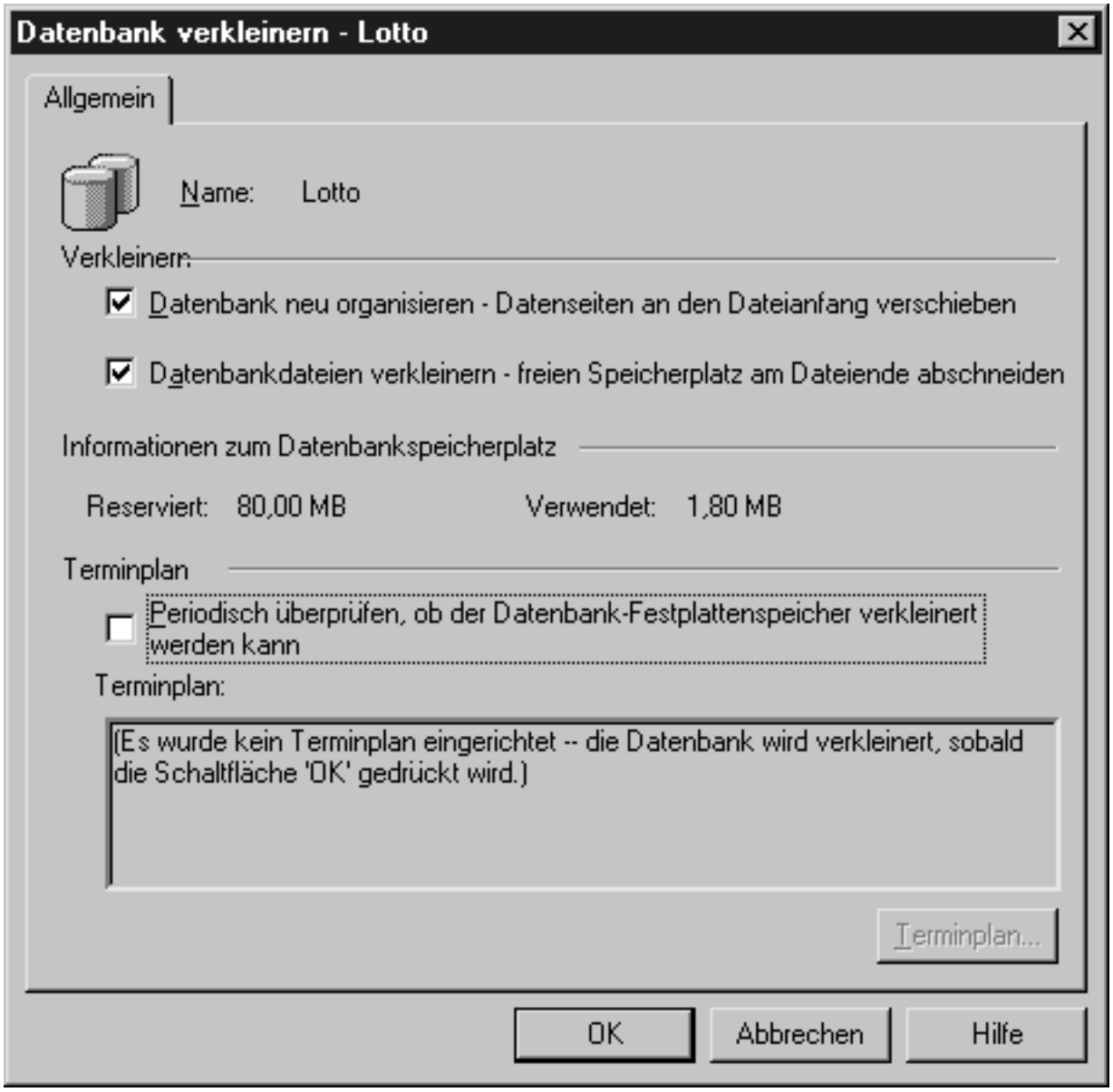


abbildung 6.13: das dialogfeld datenbank verkleinern

3. im dialogfeld **datenbank verkleinern** (siehe abbildung 6.13) können sie festlegen, auf welche weise die größe der datenbank reduziert werden soll. außerdem läßt sich ein terminplan erstellen, nach dem sql server die datenbank periodisch auf eine mögliche verkleinerung hin überprüft.
4. klicken sie auf **ok**, wenn sie die datenbank verkleinern wollen. wenn sie keinen terminplan festgelegt haben, verkleinert sql server die datenbank sofort und zeigt eine meldung gemäß abbildung 6.14 an. andernfalls erscheint eine bestätigung wie in abbildung 6.15.

abbildung 6.14: bestätigung, wenn die verkleinerung der datenbank zu einem späteren zeitpunkt stattfinden soll

abbildung 6.15: bestätigung, wenn die datenbank verkleinert wurde

wenn sie keinen terminplan festgelegt haben und auf **ok** klicken, können sie sich unmittelbar von der wirkung der verkleinerung überzeugen. aber was stellen sie fest? das meldungsfeld und die taskpad-ansicht geben an, daß die datenbank auf 55 mbyte verkleinert wurde. ursprünglich waren es doch nur 35 mbyte. warum ist die datenbank größer als beim erstellen?

noch einmal zum nachrechnen: die datenbank haben sie mit einer größe von 25 mbyte für die datendatei und 10 mbyte für die transaktionsprotokolldatei erstellt. das sind 35 mbyte gesamtgröße, die sich beim verkleinern auch nicht unterschreiten lassen. als nächstes haben sie im dialogfeld **eigenschaften** die größe der datendatei auf 50 mbyte und die größe der protokolldatei auf 30 mbyte erhöht. diese 80 mbyte erscheinen auch im dialogfeld **datenbank verkleinern**. wenn sie von 80 mbyte die vergrößerung der datendatei von 25 mbyte abziehen, erhalten sie 55 mbyte - genau diesen wert zeigt sql server an.

des rätsels lösung: sql server verkleinert die datendatei sofort, die transaktionsprotokolldatei aber erst, wenn sie das protokoll sichern oder die option *truncate on checkpoint* auf true gesetzt haben und das protokoll automatisch abgeschnitten wird.

6.3.4 konfigurationsoptionen via transact-sql

für die gespeicherten prozeduren und transact-sql-anweisungen, mit denen sich datenbankeigenschaften und konfigurationsoptionen anzeigen und ändern lassen, muß man die genaue bezeichnung der einzelnen optionen kennen. tabelle 6.3 zeigt diese optionen im überblick. detaillierte erläuterungen finden sie weiter vorn in diesem kapitel beim festlegen von datenbankoptionen mit dem enterprise manager.

option	kurzbeschreibung
auto create statistics	fehlende statistiken erstellen
auto update statistics	veraltete statistiken automatisch aktualisieren
autoclose	datenbank automatisch beim abmelden aller benutzer schließen
autoshrink	datenbankdateien automatisch verkleinern
ansi null default	null-zulässigkeit als standardeinstellung
ansi nulls	vergleiche mit null-werten liefern unknown

ansi warnings	fehler und warnungen ausgeben, beispielsweise bei division durch null
concat null yields null	ergebnis einer verkettung ist null, wenn einer der operanden null ist
cursor close on commit	alle geöffneten cursor schließen, wenn für eine transaktion ein commit oder ein rollback ausgeführt wird
dbo use only	zugriff auf datenbank nur durch datenbankbesitzer und systemadministrator möglich
default to local cursor	cursordeklarationen standardmäßig auf local einstellen
merge publish	datenbank kann für eine merge-replikation publiziert werden
offline	datenbank ist offline
published	datenbank kann für die replikation publiziert werden
quoted identifier	bezeichner dürfen in doppelte anführungszeichen eingeschlossen werden
read only	datenbank ist schreibgeschützt
recursive triggers	rekursiver aufruf von triggern zulässig
select into/bulkcopy	nicht protokollierte operationen (select into und massenkopieren mit bcp) möglich
single user	datenbank im einzelbenutzermodus
subscribed	datenbank kann für die publikation abonniert werden
torn page detection	zerrissene seiten erkennen
trunc. log on chkpt.	transaktionsprotokoll bei prüfpunkt automatisch abschneiden

tabelle 6.3: datenbankoptionen

die in tabelle 6.3 genannten zustände oder wirkungen gelten für die auf true gesetzte option.

sp_dboption

mit der gespeicherten prozedur sp_dboption lassen sich datenbankoptionen anzeigen und einstellen. die syntax lautet:

```
sp_dboption [[@dbname =] 'datenbank']
[, [@optname =] 'optionsname']
[, [@optvalue =] 'wert']
```

der parameter @dbname bezeichnet den namen der datenbank, für die die angegebene option angezeigt

oder geändert werden soll. in @optname geben sie den namen der option gemäß tabelle 6.3 an. ist der name ein schlüsselwort oder enthält er leerzeichen, schreiben sie ihn in anführungszeichen. der vollständige name ist nicht erforderlich, es genügt ein eindeutiger teilstring des optionsnamens. schließlich legt @optvalue den neuen wert der option fest. fehlt dieser parameter, gibt die prozedur sp_dboption die aktuelle einstellung an.

um zum beispiel die aktuelle einstellung der option select into/bulkcopy für die datenbank pubs abzurufen, führen sie die folgende anweisung aus:

```
exec sp_dboption pubs, sel
```

als optionsname genügt sel, weil diese zeichenfolge bereits eindeutig ist. statt der zeichenfolge select into/bulkcopy könnten sie beispielsweise auch schreiben:

```
exec sp_dboption pubs, 'o/b'
```

das ergebnis lautet in beiden fällen:

```
optionname                      currentsetting
------------------------------- --------------
select into/bulkcopy            off
```

eine neue einstellung legen sie folgendermaßen fest:

```
exec sp_dboption pubs, sel, true
```

sql server gibt daraufhin die meldung aus:

```
die geänderte datenbank wird überprüft.
dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.
```

die ergebnismenge zeigt unter currentsetting die aktuelle einstellung mit off und on an. festlegen müssen sie die werte dagegen mit false bzw. true.

eine liste aller einstellbaren optionen erhalten sie mit

```
exec sp_dboption
```

das ergebnis lautet:

```
settable database options:
-----------------------------------
ansi null default
ansi nulls
ansi warnings
auto create statistics
auto update statistics
autoclose
autoshrink
concat null yields null
cursor close on commit
dbo use only
default to local cursor
merge publish
offline
published
quoted identifier
read only
recursive triggers
select into/bulkcopy
single user
subscribed
torn page detection
trunc. log on chkpt.
```

wenn sie die gespeicherte prozedur nur mit der datenbank als parameter aufrufen, wie zum beispiel

```
exec sp_dboption pubs
```

erhalten sie eine liste der auf on (d.h. true) gesetzten optionen:

```
the following options are set:
----------------------------------
select into/bulkcopy
trunc. log on chkpt.
auto create statistics
```

```
auto update statistics
```

sp_configure

mit der gespeicherten prozedur sp_configure lassen sich globale konfigurationseinstellungen für den aktuellen server anzeigen und ändern. die syntax lautet:

```
sp_configure [[@configname =] 'name'
[,[@configvalue =] 'wert']
```

der parameter @configname bezeichnet den namen einer konfigurationsoption. tabelle 6.4 zeigt die verfügbaren namen. wie bei sp_dboption können sie einen beliebigen teil des konfigurationsnamens angeben, wenn dieser teil eindeutig ist.

option	beschreibung
allow updates	direkte aktualisierung in den systemtabellen möglich
default language	id einer in der systemtabelle syslanguages verfügbaren sprache für meldungen von sql server (1 - deutsch)
language in cache	maximale anzahl der sprachen, die gleichzeitig im sprachencache gespeichert werden können neustart erforderlich
max text repl size	maximale größe (in byte) von daten des typs text und image, die einer replizierten spalte in einer einzelnen anweisung mit insert, update, writetext oder updatetext hinzugefügt werden können
nested triggers	kaskadieren von triggern (bis maximal 16 ebenen) möglich
remote access	anmelden über remoteserver zugelassen neustart erforderlich
remote login timeout	anzahl der sekunden, nach der ein vergeblicher versuch einer remote-anmeldung als fehlgeschlagen gilt
remote proc trans	bereitstellen einer von ms dtc koordinierten verteilten transaktion zum schutz der acid-eigenschaften von transaktionen
remote query timeout	anzahl der sekunden, nach der die verarbeitung einer remote-abfrage aufgrund einer zeitüberschreitung abgebrochen wird
show advanced options	zeigt die erweiterten optionen mit der gespeicherten prozedur sp_configure an
user options	festlegen globaler standardwerte für alle benutzer

tabelle 6.4: konfigurationsoptionen

falls nicht anders angegeben, gelten die in tabelle 6.4 genannten zustände oder wirkungen für die auf 1 gesetzte option. die anmerkung »neustart erforderlich« bedeutet, daß die option erst nach einem neustart von sql server wirksam wird.

wenn show advanced options auf den wert 1 für config_value gesetzt ist, zeigt die gespeicherte prozedur sp_configure auch die erweiterten server-optionen an. diese optionen sollten nur zertifizierte sql-server-techniker ändern. weitere hinweise dazu finden sie in der online-dokumentation unter dem stichwort »konfigurationsoptionen/festlegen von konfigurationsoptionen«.

wenn sie die gespeicherte prozedur sp_configure ohne parameter ausführen, erhalten sie eine ergebnismenge mit folgenden fünf spalten:

- name: name der konfigurationsoption
- minimum: mindestwert der konfigurationsoption
- maximum: höchstwert der konfigurationsoption
- config_value: der wert, auf den sie die konfigurationsoption mit sp_configure eingestellt haben. diesen wert finden sie in der spalte value der systemtabelle sysconfigures.
- run_value: aktueller wert für die konfigurationsoption. dieser wert steht in der spalte value der systemtabelle syscurconfigs.

der datentyp der spalte name ist nvarchar(70), die anderen spalten sind vom typ int.

die werte in config_value und run_value können voneinander abweichen, da manche optionseinstellungen nicht sofort, sondern erst nach dem neustart von sql server oder nach ausführen von reconfigure (siehe weiter unten) wirksam werden.

das ergebnis sieht folgendermaßen aus:

```
name                        min maximum     config_value run_value
--------------------------- --- ----------- ------------ ---------
allow updates               0   1           0            0
default language            0   9999        1            1
language in cache           3   100         3            3
max text repl size (b)      0   2147483647  65536        65536
nested triggers             0   1           1            1
remote access               0   1           1            1
remote login timeout (s) 0   2147483647  5            5
remote proc trans           0   1           0            0
remote query timeout (s) 0   2147483647  0            0
show advanced options       0   1           0            0
user options                0   4095        0            0
```

das folgende beispiel zeigt, wie man die anzeige der erweiterten optionen aktiviert:

```
exec sp_configure show, 1
```

die konfigurationsoptionen sind vom datentyp int. deshalb schalten sie eine option mit 1 (und nicht mit true) ein und mit 0 (nicht mit false) aus.

reconfigure

nachdem sie eine einstellung mit sp_configure geändert haben, steht der neue wert zunächst nur in config_value. damit sql server diesen wert übernimmt, müssen sie die transact-sql-anweisung reconfigure ausführen. danach ist der neue wert wirksam und erscheint in der spalte run_value.

in der anweisung reconfigure können sie optional die schlüsselwörter with override angeben. diese sind eigentlich für die konfiguration der erweiterten optionen vorgesehen und setzen bestimmte überprüfungen außer kraft. für die normalen optionen spielt es keine rolle, ob sie reconfigure oder reconfigure with override ausführen.

auf konfigurationsoptionen, die einen neustart von sql server erfordern, hat die ausführung von reconfigure keinen einfluß.

6.4 datenbanken umbenennen

eine datenbank kann nur der systemadministrator (sa) oder ein mitglied der rolle sysadmin umbenennen. weitere voraussetzungen sind:

- die datenbank darf von niemandem benutzt werden.
- die datenbank muß sich im einzelbenutzermodus befinden.

da es im enterprise manager keinen befehl zum umbenennen einer datenbank gibt, müssen sie entweder mit transact-sql oder mit sql-dmo arbeiten. die folgenden schritte zeigen, wie sie eine datenbank per transact-sql umbenennen:

1. setzen sie die datenbank in den einzelbenutzermodus. das erledigen sie mit der gespeicherten prozedur sp_dboption, wie es der abschnitt zum ändern von konfigurationsoptionen weiter vorn in diesem kapitel gezeigt hat.
2. benennen sie die datenbank mit der gespeicherten prozedur sp_renamedb um.
3. setzen sie die umbenannte datenbank in den multibenutzermodus.

die syntax der gespeicherten prozedur sp_renamedb lautet:

```
sp_renamedb [@old_name =] 'altername',
[@new_name =] 'neuername'
```

als beispiel benennen sie die datenbank lotto in bingo um:

```
-- datenbank lotto in bingo umbenennen
```

```
use master
go
-- datenbank lotto in einzelbenutzermodus setzen
sp_dboption lotto, 'single user', true
go
-- umbenennen
sp_renamedb 'lotto', 'bingo'
go
-- datenbank bingo in multibenutzermodus setzen
sp_dboption bingo, 'single user', false
go
```

wenn sie diese anweisungen ausführen, erscheinen folgende rückmeldungen:

```
dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.
die datenbank ist jetzt im einzelbenutzermodus.

(1 row(s) affected)

dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.
die datenbank wurde umbenannt und befindet sich im
einzelbenutzermodus.
ein mitglied der sysadmin-rolle muss die datenbank mithilfe von
sp_dboption in den multibenutzermodus zurücksetzen.
dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.
die datenbank ist jetzt im multibenutzermodus.
```

6.5 datenbanken löschen

eine datenbank läßt sich ohne viel aufhebens im enterprise manager löschen. erweitern sie dazu die konsolenstruktur bis zur gewünschten datenbank, klicken sie mit der rechten maustaste auf den namen der datenbank, und wählen sie **löschen** aus dem kontextmenü. sobald sie die sicherheitsabfrage mit **ja** bestätigt haben, ist die datenbank im nirwana verschwunden. um das löschen der zugehörigen daten- und protokolldateien brauchen sie sich nicht zu kümmern, das erledigt sql server automatisch.

genauso einfach ist es, eine datenbank per transact-sql zu löschen:

```
drop database datenbankname
```

die im letzten beispiel umbenannte datenbank bingo löschen sie dann wie folgt:

```
drop database bingo
```

mit der transact-sql-anweisung drop database können sie auch mehrere datenbanken auf einmal löschen. geben sie dazu für *datenbankname* eine durch kommas getrennte liste der zu löschenden datenbanken an.

als datenbankname geben sie immer den namen an, den sie beim erstellen (oder umbenennen) der datenbank festgelegt haben, d.h. nicht den namen der daten- oder protokolldatei. die eigentlichen datenbankdateien können völlig anders lautende namen haben. sql server verwaltet diese namen und löscht die dateien automatisch.

6.6 datenbankdiagramme

vor allem bei größeren datenbankprojekten ist es wichtig, eine genaue übersicht über die gesamte datenbank zu haben. hierfür bieten sich diagramme an, die alle tabellen, die beziehungen zwischen den tabellen und die spalten in den tabellen in grafischer form darstellen.

wenn sie im index der online-dokumentation lediglich nach dem stichwort »diagramme« suchen, erscheint ein hilfethema zum windows-nt-systemmonitor. nähere informationen zum hier behandelten thema finden sie unter »datenbankdiagramme«.

darüber hinaus kann man mit datenbankdiagrammen auch datenbankobjekte erstellen, bearbeiten und löschen.

die beispieldatenbank northwind verfügt bereits über ein datenbankdiagramm. um dieses diagramm anzuzeigen, erweitern sie die konsolenstruktur im enterprise manager bis zum zweig *datenbanken*, klicken auf die datenbank *northwind*, doppelklicken im rechten fensterbereich auf *diagramme* und doppelklicken schließlich auf das element *relationships*.

der folgende abschnitt erläutert, wie sie für die beispieldatenbank pubs ein datenbankdiagramm erstellen.

6.6.1 datenbankdiagrammerstellungs-assistent

ein neues datenbankdiagramm läßt sich über den enterprise manager mit dem datenbankdiagrammerstellungs-assistenten erstellen. führen sie folgende schritte aus:

4. um den assistenten zu starten, wählen sie entweder aus dem menü **ansicht** den befehl **standarddatenbankdiagramm** oder klicken im rechten fensterbereich mit der rechten maustaste auf das element **diagramme** und wählen den befehl **neues datenbankdiagramm** aus dem kontextmenü (siehe abbildung 6.16).

bild

abbildung 6.16: den datenbankdiagrammerstellungs-assistenten im enterprise manager aufrufen

5. klicken sie im startdialogfeld des assistenten auf **weiter**. im zweiten dialogfeld lassen sich die tabellen auswählen, die im diagramm enthalten sein sollen (siehe abbildung 6.17). klicken sie im listenfeld **verfügbare tabellen** auf die gewünschte tabelle und dann auf die schaltfläche **hinzufügen**, um die tabelle in das diagramm zu übernehmen. wenn sie nur bestimmte tabellen aufnehmen wollen, können sie entscheiden, ob zusätzlich auch verknüpfte tabellen automatisch hinzuzufügen sind. schalten sie dazu das kontrollkästchen unter dem linken listenfeld ein. wenn sie alle tabellen hinzufügen, wird das kontrollkästchen deaktiviert.

einen zusammenhängenden tabellenbereich wählen sie durch niederhalten der **ë**-taste und klicken aus, mehrere einzeltabellen durch drücken der **ç**-taste und klicken.

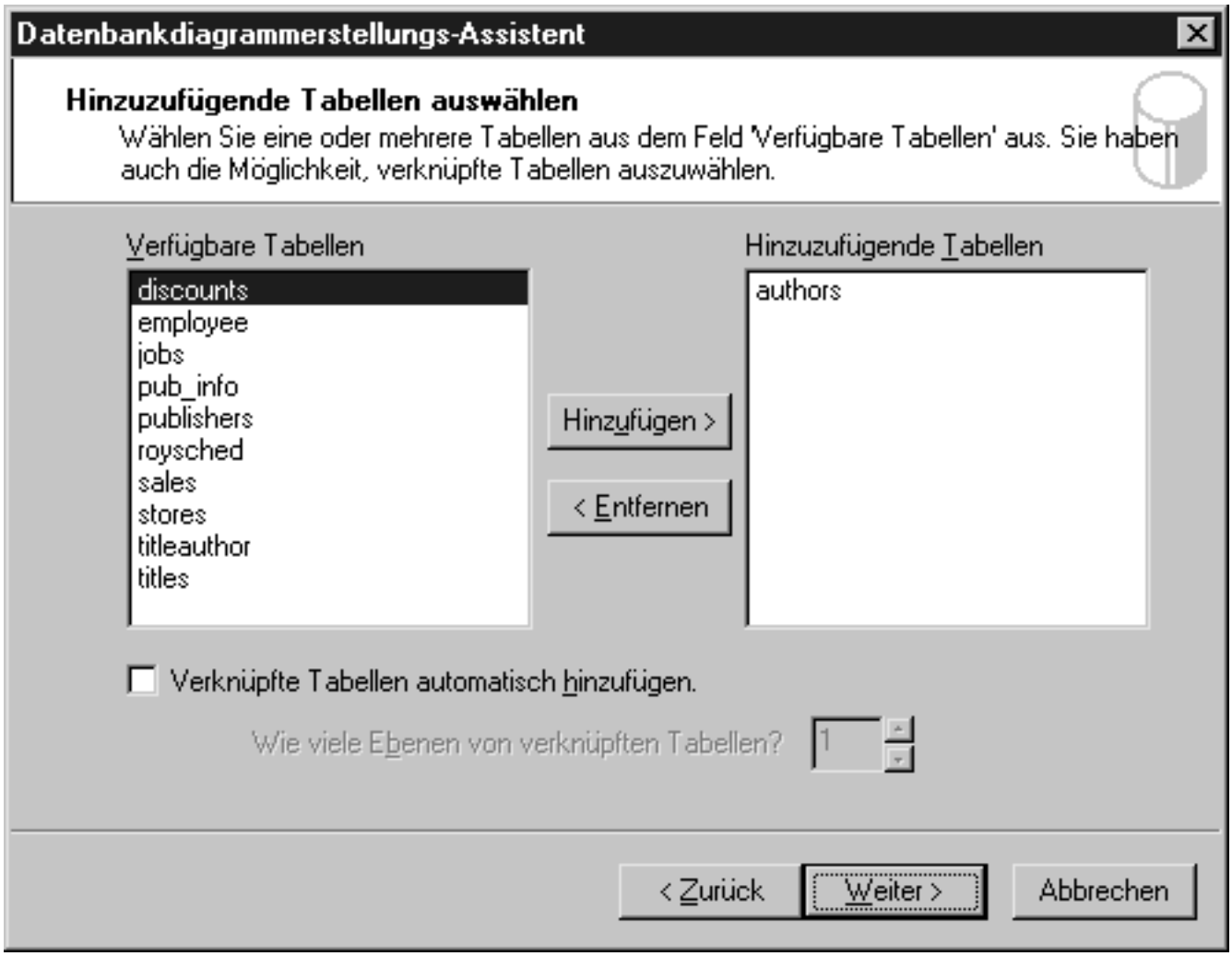


abbildung 6.17: im zweiten dialogfeld des datenbankdiagrammerstellungs-assistenten wählen sie die tabellen für das diagramm aus

6. wählen sie für das beispiel alle tabellen aus, und klicken sie dann auf **weiter**. im letzten dialogfeld des assistenten sind alle tabellen aufgeführt, die in das diagramm übernommen werden. klicken sie auf **zurück**, wenn sie änderungen vornehmen möchten, andernfalls auf **fertigstellen**. der assistent erstellt das diagramm und zeigt es im enterprise manager an (siehe abbildung 6.18).

bild

abbildung 6.18: das vom assistenten für die datenbank pubs erzeugte datenbankdiagramm

7. um das diagramm zu speichern, klicken sie in der symbolleiste auf die schaltfläche **speichern** (erste von links). geben sie einen passenden namen an, zum beispiel beziehungen.

6.6.2 datenbankdiagramm bearbeiten

die symbolleiste im fenster des datenbankdiagramms bietet verschiedene befehle, mit denen sie diagramme drucken und die datenbankstruktur auf grafischem weg bearbeiten können.

objekte auswählen

um eine einzelne tabelle auszuwählen, klicken sie einfach auf die tabelle. die titelleiste der ausgewählten tabelle ist hervorgehoben, die anderen titelleisten zeigen den deaktivierten zustand - genau wie bei aktivierten und deaktivierten fenstern. mehrere tabellen wählen sie aus, indem sie mit der maus einen markierungsrahmen um die betreffenden tabellen ziehen. die daraufhin ausgeführten aktionen (zum beispiel einstellen von anzeigeoptionen) beziehen sich dann nur auf die markierten tabellen. um alle objekte auszuwählen, klicken sie mit der rechten maustaste und wählen aus dem kontextmenü den befehl **alles markieren**.

anzeigeoptionen einstellen

nach dem erstellen eines datenbankdiagramms erscheinen die tabellennamen und die zugehörigen spaltennamen in der standardansicht wie in abbildung 6.18. über die schaltfläche **anzeigen** (vierte von rechts) können sie unter folgenden optionen wählen:

- *spalteneigenschaften*: erweitert die darstellung der ausgewählten tabelle(n) und zeigt die eigenschaften der spalten an (siehe abbildung 6.19). dieses format bietet die meisten informationen und eignet sich vor allem, wenn sie einen detaillierten überblick der datenbank ausdrucken wollen.

bild

abbildung 6.19: anzeige der spalteneigenschaften

- *spaltennamen*: bei dieser darstellung sind in der titelleiste die tabellennamen und darunter die spalten zu sehen (siehe dazu abbildung 6.18). diese form sollten sie drucken, wenn sie abfragen formulieren und dazu lediglich hinweise über den aufbau der datenbank brauchen. die datentypen lassen sich oftmals bereits aus der bezeichnung der spalten erkennen.
- *schlüsselspalten*: zeigt nur die spalten an, die in primär- oder fremdschlüsseln verwendet werden.
- *tabellennamen*: zeigt nur die tabellennamen an.
- *benutzerdefiniert*: diese option entspricht zunächst der ansicht *spalteneigenschaften*. allerdings haben sie jetzt die möglichkeit, bestimmte spalten von der anzeige auszublenden und die bildlaufleisten in den tabellen zu unterdrücken.

klicken sie mit der rechten maustaste auf die gewünschte tabelle (im beispiel titles). wählen sie den befehl **benutzerdefinierte ansicht ändern**. das dialogfeld **spaltenauswahl** (siehe abbildung 6.20) zeigt in der linken liste die verfügbaren spalten und in der rechten liste die für die anzeige ausgewählten spalten. indem sie die gewünschten spalten markieren und auf die schaltflächen zwischen beiden listen klicken, können sie zusätzliche spalten in die anzeige aufnehmen oder daraus entfernen. die reihenfolge

der spalten in der anzeige entspricht der reihenfolge im listenfeld **ausgewählte spalten**. mit den darunter befindlichen pfeiltasten ändern sie bei bedarf die position der spalten.

der begriff *spalten* bezieht sich hier nicht auf die spalten einer datenbanktabelle, sondern auf die in der diagrammansicht angezeigten spalten: die diagrammspalten zeigen die attribute der datenbankspalten an.

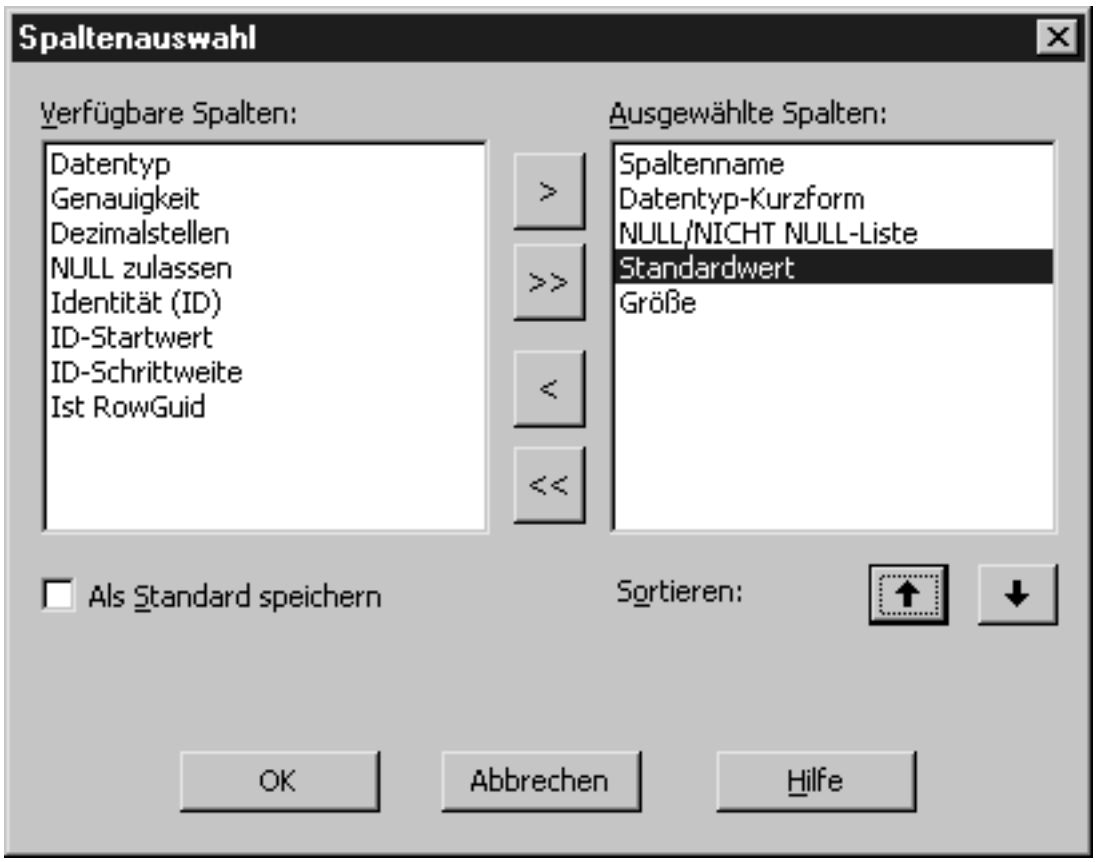


abbildung 6.20: im dialogfeld spaltenauswahl legen sie die anzuzeigenden attributspalten für die datenbanktabelle fest

wenn sie das kontrollkästchen **als standard speichern** einschalten, erscheint das datenbankdiagramm bei wahl der anzeigeoption *benutzerdefiniert* mit den von ihnen festgelegten spalten.

die größe der anzeige ändern sie bei bedarf über die schaltfläche **zoom** (fünfte von rechts in der symbolleiste).

layout anpassen

das vom datenbankdiagrammerstellungs-assistenten generierte diagramm können sie an ihre vorstellungen anpassen, wobei eine änderung des layouts keinen einfluß auf die definitionen in der datenbank hat. verschieben sie die objekte mit der maus, oder wählen sie den befehl **tabellen anordnen**, der die objekte automatisch arrangiert. die beziehungslinien wandern in jedem fall mit. nachdem sie das layout fertiggestellt haben, können sie das diagramm mit beschriftungen ergänzen. der befehl **neue textanmerkung** setzt in das diagramm ein textfeld, das sie nach belieben ausfüllen können.

die über den befehl **neue textanmerkung** eingefügten beschriftungen bleiben auf ihrer position, auch wenn sie das layout im nachhinein automatisch anordnen lassen. unter umständen sind dann einige beschriftungen nicht mehr zu sehen. nehmen sie den feinschliff ihres datenbankdiagramms deshalb erst

dann vor, wenn das layout feststeht.

diagramme drucken

bevor sie ein datenbankdiagramm drucken, können sie sich einen überblick über den seitenaufbau verschaffen. klicken sie dazu mit der rechten maustaste im diagramm, und wählen sie aus dem kontextmenü den befehl **seitenumbrüche anzeigen**. die eingeblendeten blauen markierungen geben die lage der blätter und die seitenzahlen für den ausdruck an. wie aus dem beispiel in abbildung 6.21 ersichtlich ist, liegt bei diesem diagramm eine tabelle genau zwischen zwei blättern. vor dem drucken können sie also noch korrekturen vornehmen. klicken sie dann mit der rechten maustaste, und wählen sie den befehl **seitenumbrüche neu berechnen**.

Bild

abbildung 6.21: datenbankdiagramm mit eingeblendeten seitenumbrüchen

wenn das diagramm schließlich ihren ästhetischen ansprüchen genügt, klicken sie auf die schaltfläche **drucken** in der symbolleiste (oder wählen sie den befehl **drucken** aus dem kontextmenü), um das diagramm auf papier zu bannen.

tabellen hinzufügen

falls sie beim erstellen des diagramms nicht alle tabellen ausgewählt haben, können sie über den befehl **tabellen hinzufügen** weitere tabellen der datenbank in das diagramm aufnehmen. dieser befehl hat keinen einfluß auf die zugrundeliegende definition der datenbank.

neue tabelle

mit dem befehl **neue tabelle** können sie der datenbank eine neue tabelle hinzufügen und die spalten mit den zugehörigen attributen festlegen. kapitel 8 widmet sich dem thema tabellen ausführlich. beim speichern des datenbankdiagramms erscheint die frage, ob sie die änderungen übernehmen wollen. wenn sie mit **ja** bestätigen, wird die neue tabelle in die datenbank aufgenommen. der befehl **neue tabelle** hat damit auswirkungen auf die definitionen der datenbank.

tabelle entfernen oder löschen

wenn sie eine tabelle in der diagrammansicht nicht benötigen, klicken sie mit der rechten maustaste auf die tabelle und wählen aus dem kontextmenü den befehl **tabelle aus datenbankdiagramm entfernen**. der befehl hat nur bedeutung für das datenbankdiagramm und ändert nichts an der definition der datenbank.

dagegen wirkt sich der befehl **tabelle aus datenbank löschen** auf die definition der datenbank aus. nehmen wir an, daß sie die tabelle titles aus der datenbank pubs löschen. zunächst erscheint die frage, ob sie die tabelle wirklich dauerhaft aus der datenbank entfernen wollen. auch wenn sie hier mit **ja** antworten, bleibt ihnen noch ein hintertürchen offen. sobald sie nämlich das datenbankdiagramm speichern, erscheint das in abbildung 6.22 wiedergegebene dialogfeld.

abbildung 6.22: sicherheitsabfrage vor dem endgültigen löschen

da das löschen der tabelle titles aufgrund der bestehenden beziehungen auch auf andere tabellen einfluß hat, sind diese tabellen in der sicherheitsabfrage (dialogfeld **speichern**) aufgeführt. beim löschen einer tabelle werden auch die beziehungen gelöscht. wenn sie im dialogfeld **speichern** auf **nein** klicken, bleibt alles beim alten.

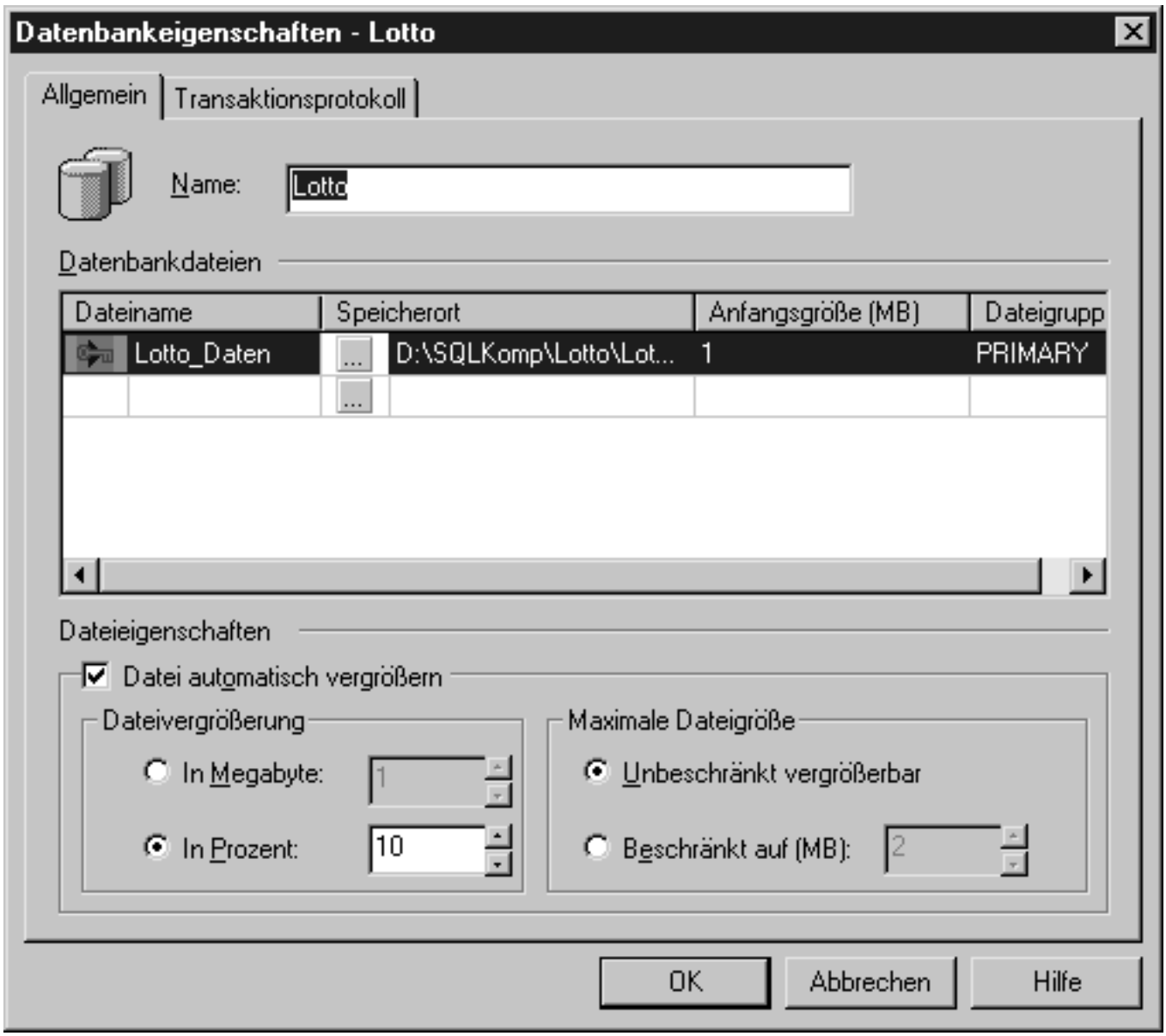
damit die - vormals zum löschen vorgesehene - tabelle wieder im datenbankdiagramm erscheint, schließen sie das dialogfeld durch klicken auf die schaltfläche **schliessen** in der titelleiste und antworten auf die frage, ob sie die änderungen im diagramm speichern wollen, mit **nein**. wenn sie das diagramm erneut öffnen, ist die tabelle wieder vorhanden.

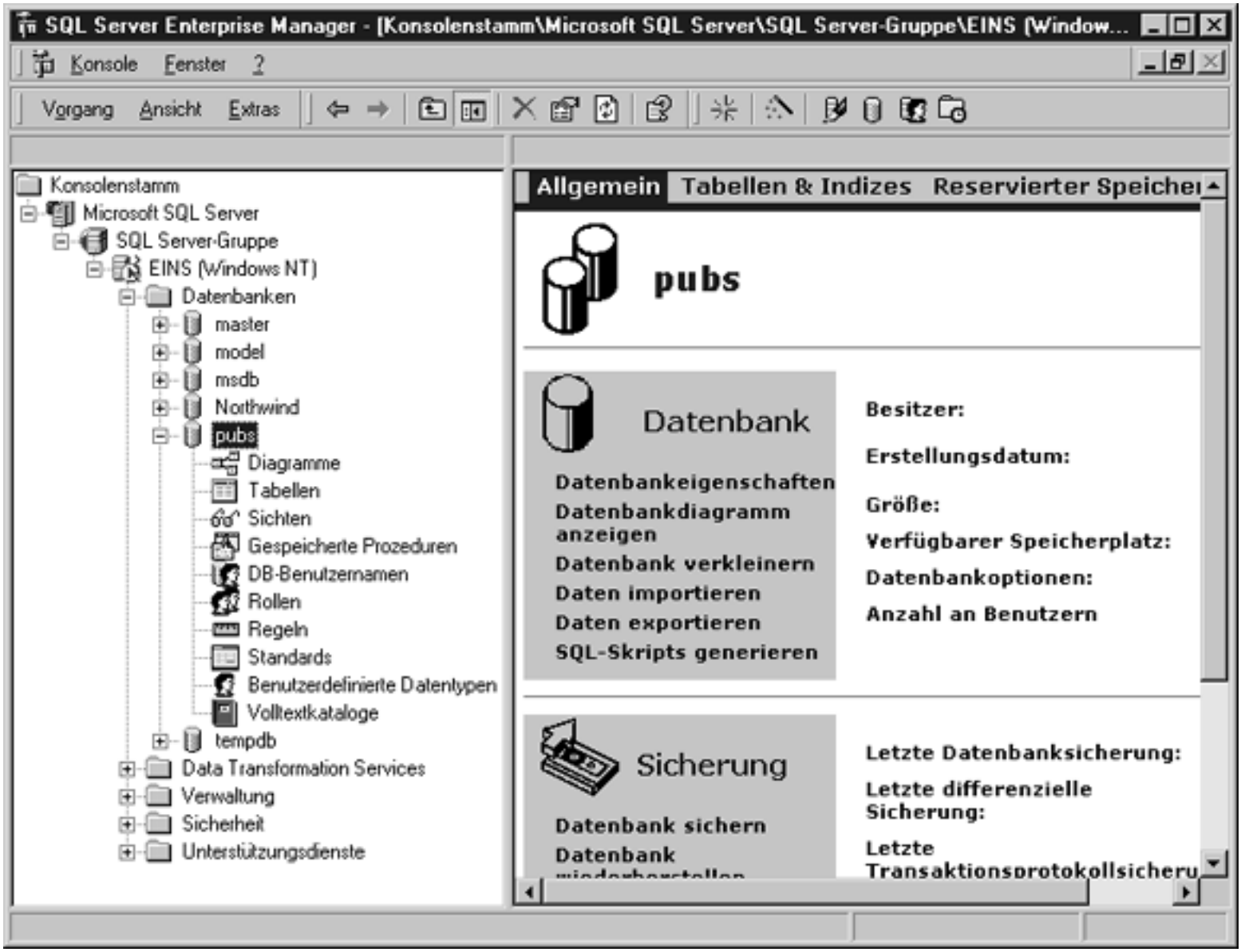
änderungsskript speichern

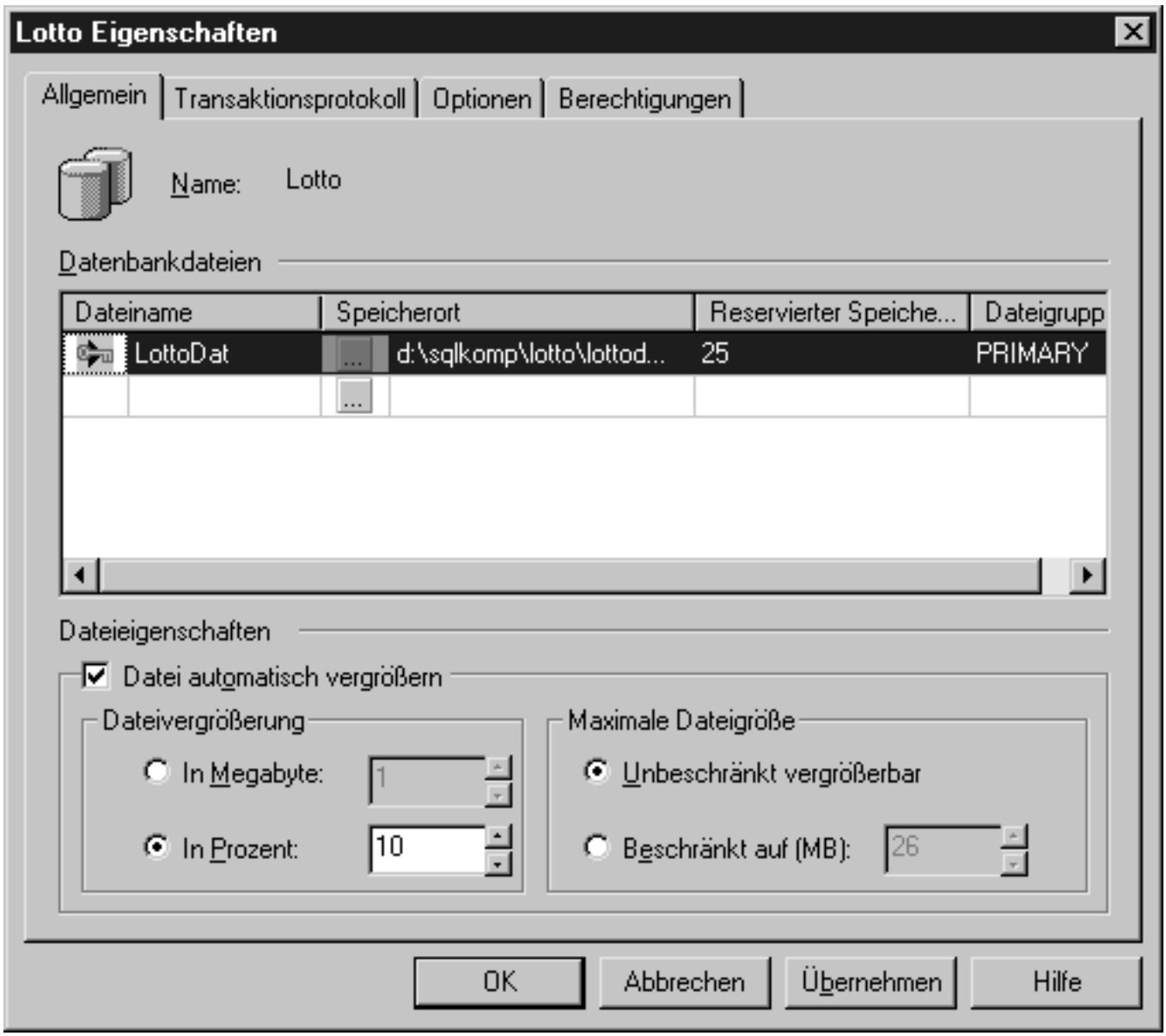
hier erscheinen die transact-sql-anweisungen, die bei änderungen am datenbankdiagramm automatisch erstellt wurden. diese anweisungen können sie speichern, um sie gegebenenfalls später beim wiederherstellen der datenbank zu verwenden.

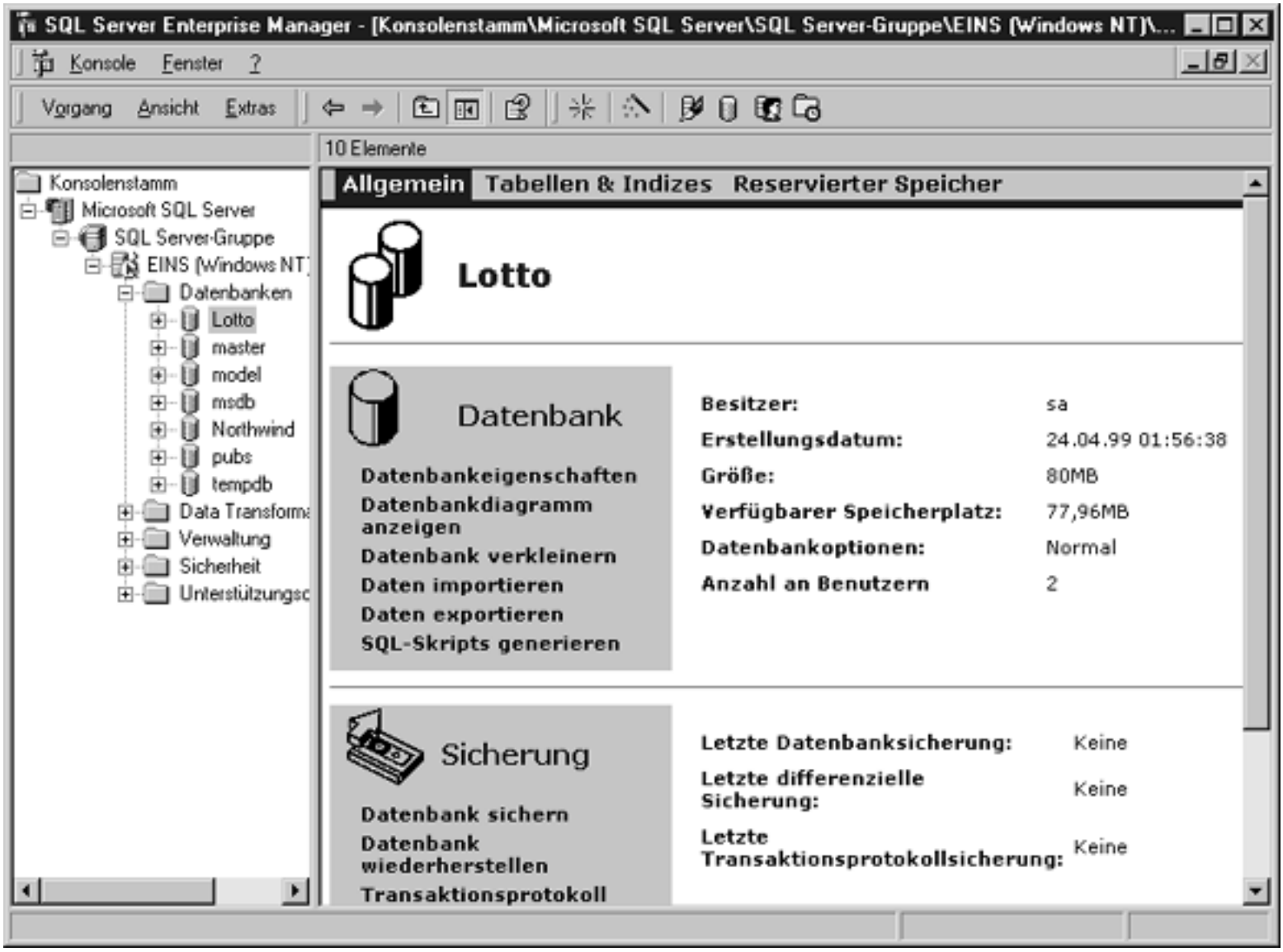
ein datenbankdiagramm bietet noch mehr funktionen. zum beispiel lassen sich schlüsselbeziehungen erstellen, auf die kapitel 16 eingeht. wenn sie die zeit dazu haben, sollten sie eine einfache testdatenbank erstellen und dann die verschiedenen möglichkeiten des diagramms ausprobieren.

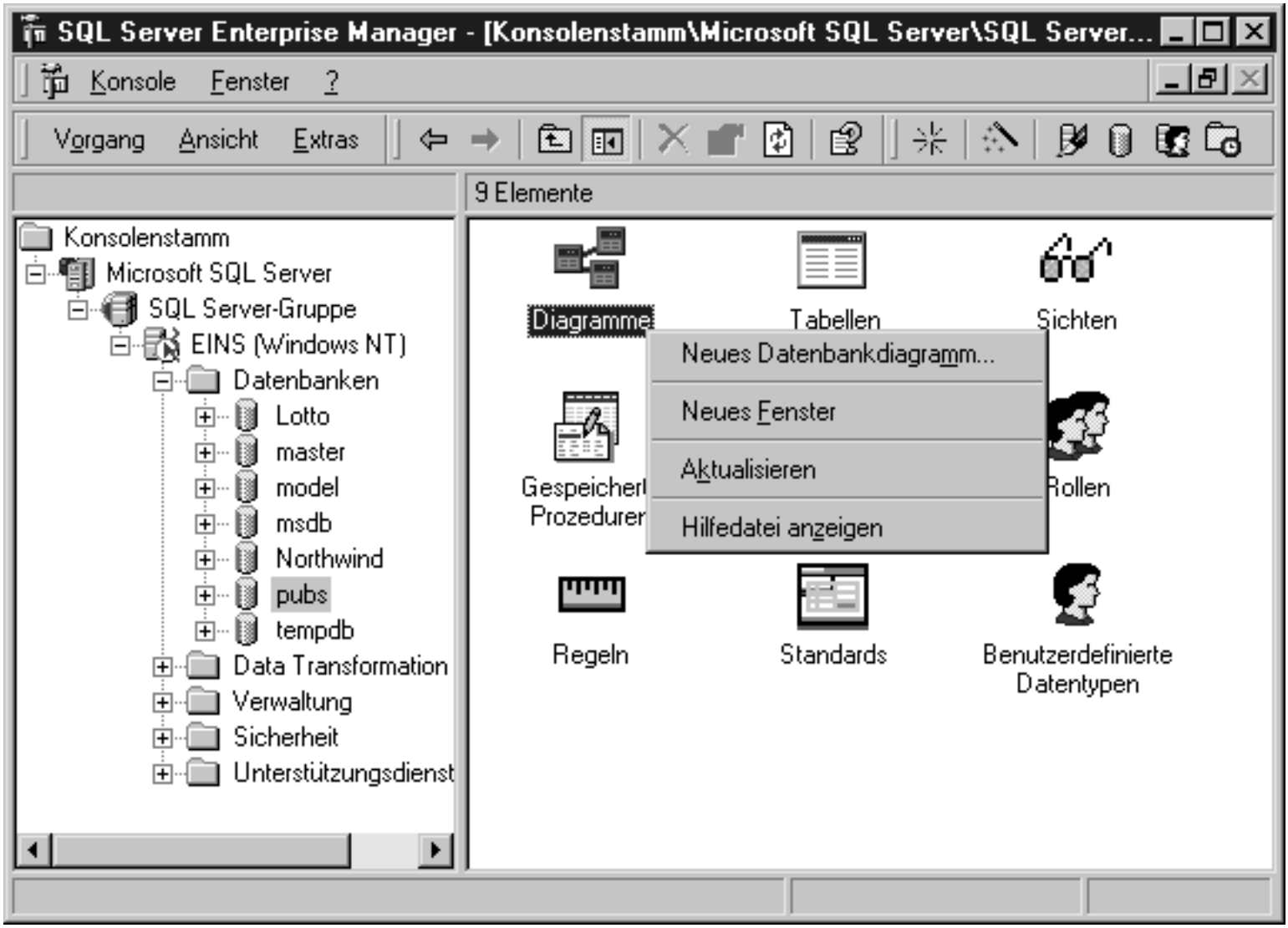

Datenbankerstellungs-Assistent - EINS
Datenbankerstellungs-Assistenten beenden
Sie haben die Schritte abgeschlossen, die zum Erstellen einer neuen Datenbank erforderlich sind. Die neu zu erstellende Datenbank ist wie folgt definiert.
Der Name der Datenbank lautet Lotto.
Die Datenbankdateien werden im Verzeichnis D:\SQLKomp\Lotto gespeichert.
Die Transaktionsprotokolldateien werden gespeichert in Verzeichnis D:\SQLKomp\Lotto gespeichert.
Die Datenbankdateien lauten: Lotto_Data, Größe 1
Die Datenbankdateien werden automatisch um 10 Prozent
< Zurück
Fertig stellen
Abbrechen

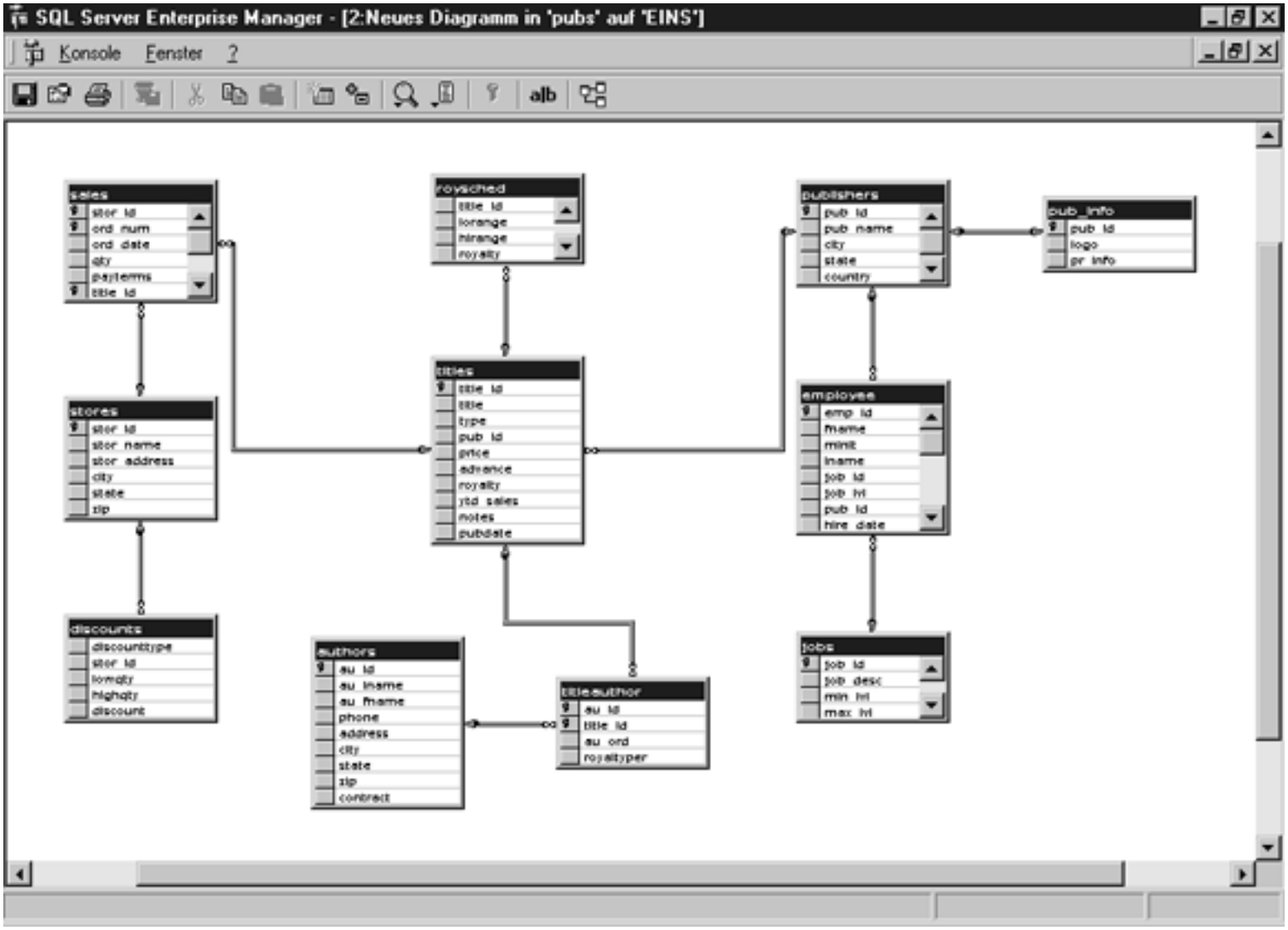

Datenbankeigenschaften - Lotto
Allgemein
Transaktionsprotokoll
Name:
Lotto
Datenbankdateien
Dateiname
Speicherort
Anfangsgröße (MB)
Dateigrupp
Lotto_Daten
D:\SQLKomp\Lotto\Lot...
1
PRIMARY
Dateieigenschaften
Datei automatisch vergrößern
Dateivergrößerung
In Megabyte:
1
In Prozent:
10
Maximale Dateigröße
Unbeschränkt vergrößerbar
Beschränkt auf (MB):
2
OK
Abbrechen
Hilfe

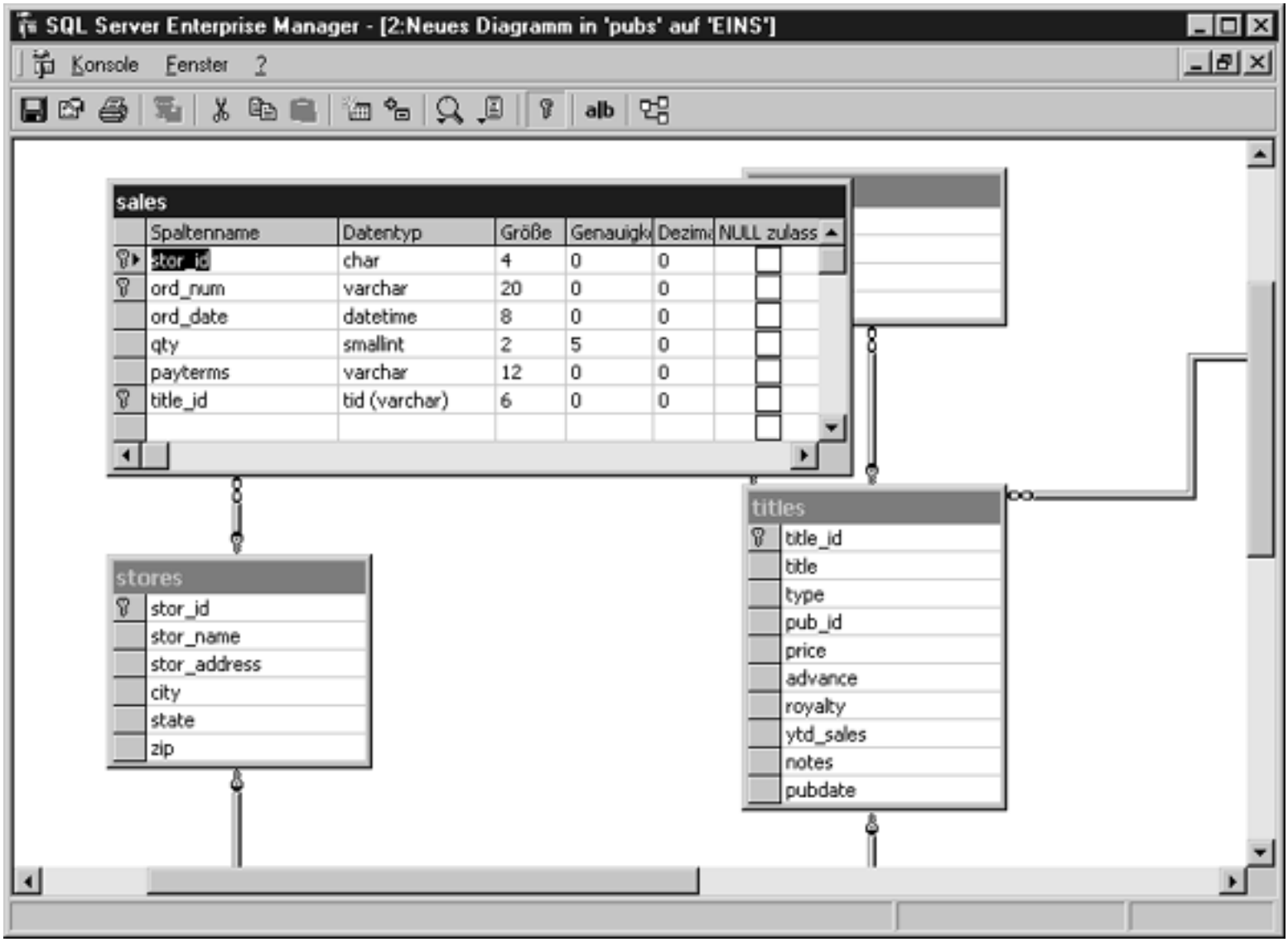

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Window...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
master
model
msdb
Northwind
pubs
Diagramme
Tabellen
Sichten
Gespeicherte Prozeduren
DB-Benutzernamen
Rollen
Regeln
Standards
Benutzerdefinierte Datentypen
Volltextkataloge
tempdb
Data Transformation Services
Verwaltung
Sicherheit
Unterstützungsdienste
Allgemein
Tabellen & Indizes
Reservierter Speicher
pubs
Datenbank
Datenbankeigenschaften
Datenbankdiagramm anzeigen
Datenbank verkleinern
Daten importieren
Daten exportieren
SQL-Skripts generieren
Besitzer:
Erstellungsdatum:
Größe:
Verfügbarer Speicherplatz:
Datenbankoptionen:
Anzahl an Benutzern
Sicherung
Datenbank sichern
Datenbank
Letzte Datenbanksicherung:
Letzte differenzielle Sicherung:
Letzte

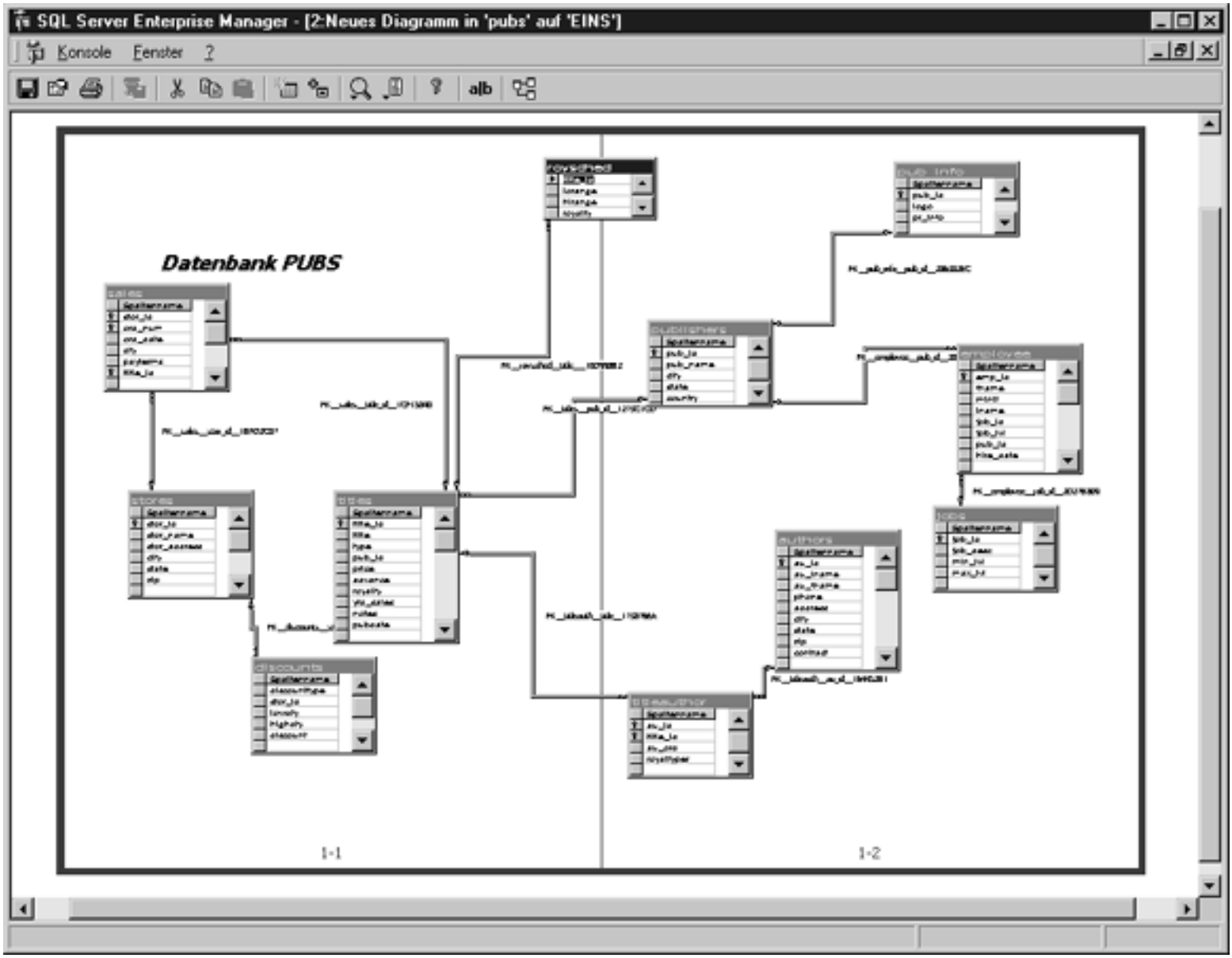

Lotto Eigenschaften
Allgemein
Transaktionsprotokoll
Optionen
Berechtigungen
Name: Lotto
Datenbankdateien
Dateiname
Speicherort
Reservierter Speiche...
Dateigrupp
LottoDat
d:\sqlkomp\lotto\lottod...
25
PRIMARY
Dateieigenschaften
Datei automatisch vergrößern
Dateivergrößerung
In Megabyte:
1
In Prozent:
10
Maximale Dateigröße
Unbeschränkt vergrößerbar
Beschränkt auf (MB):
26
OK
Abbrechen
Übernehmen
Hilfe


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
10 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Lotto
master
model
msdb
Northwind
pubs
tempdb
Data Transforma
Verwaltung
Sicherheit
Unterstützungsc
Allgemein
Tabellen & Indizes
Reservierter Speicher
Lotto
Datenbank
Datenbankeigenschaften
Datenbankdiagramm anzeigen
Datenbank verkleinern
Daten importieren
Daten exportieren
SQL-Skripts generieren
Besitzer:
sa
Erstellungsdatum:
24.04.99 01:56:38
Größe:
80MB
Verfügbarer Speicherplatz:
77,96MB
Datenbankoptionen:
Normal
Anzahl an Benutzern
2
Sicherung
Datenbank sichern
Datenbank wiederherstellen
Transaktionsprotokoll
Letzte Datenbanksicherung:
Keine
Letzte differenzielle Sicherung:
Keine
Letzte Transaktionsprotokollsicherung:
Keine


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
9 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Lotto
master
model
msdb
Northwind
pubs
tempdb
Data Transformation
Verwaltung
Sicherheit
Unterstützungsdienst
Diagramme
Tabellen
Sichten
Neues Datenbankdiagramm...
Neues Fenster
Aktualisieren
Hilfedatei anzeigen
Gespeicherl
Prozedurer
Rollen
Regeln
Standards
Benutzerdefinierte
Datentypen


SQL Server Enterprise Manager - [2:Neues Diagramm in 'pubs' auf 'EINS']
Konsole
Fenster
?
sales
stor_id
ord_num
ord_date
qty
payterms
title_id
roysched
title_id
lorange
hirange
royalty
publishers
pub_id
pub_name
city
state
country
pub_info
pub_id
logo
pr_info
stores
stor_id
stor_name
stor_address
city
state
zip
titles
title_id
title
type
pub_id
price
advance
royalty
ytd_sales
notes
pubdate
employee
emp_id
fname
minit
lname
job_id
job_lvl
pub_id
hire_date
discounts
discounttype
stor_id
lowqty
highqty
discount
authors
au_id
au_lname
au_fname
phone
address
city
state
zip
contract
titleauthor
au_id
title_id
au_ord
royaltyper
jobs
job_id
job_desc
min_lvl
max_lvl


SQL Server Enterprise Manager - [2:Neues Diagramm in 'pubs' auf 'EINS']
Konsole
Fenster
?
alb
sales
Spaltenname
Datentyp
Größe
Genauigk
Dezim
NULL zulass
stor_id
char
4
0
0
ord_num
varchar
20
0
0
ord_date
datetime
8
0
0
qty
smallint
2
5
0
payterms
varchar
12
0
0
title_id
tid (varchar)
6
0
0
stores
stor_id
stor_name
stor_address
city
state
zip
titles
title_id
title
type
pub_id
price
advance
royalty
ytd_sales
notes
pubdate


SQL Server Enterprise Manager - [2:Neues Diagramm in 'pubs' auf 'EINS']
Konsole
Fenster
?
Datenbank PUBS
1-1
1-2

kapitel 7 datenbanken sichern und wiederherstellen

7.1 murphys gesetz

oder »wenn etwas schiefgehen kann, dann wird es auch schiefgehen«. datenbanken sind von diesem gesetz nicht ausgenommen. störungen gehen hier nicht selten mit datenverlusten einher. inhalte von datenbanken unterliegen einem ständigen wechsel. hunderte oder tausende von benutzern können gleichzeitig auf die datenbanken zugreifen, die datenbestände abrufen, ändern und löschen. bei einem datenverlust kann die arbeit von tagen oder wochen im nirwana verschwinden - der aufwand für die wiederherstellung der datenbestände ist beträchtlich. zum teil ist es gar nicht mehr möglich, verlorengegangene daten wiederzubeschaffen.

die datensicherung gehört demnach zu den wichtigsten aufgaben des datenbankadministrators bei der verwaltung von datenbanken. mit hilfe von datensicherungen kann man aber auch datenbanken schnell und problemlos auf einen anderen server verschieben oder kopieren.

7.2 ursachen für datenverluste

störungen des datenbankbetriebs bzw. der datenbankintegrität können unter anderem folgende ursachen haben:

- hardware: zu dieser kategorie zählen ausfälle der festplatten oder festplatten-controller, defekte bausteine wie speicher oder prozessoren auf der hauptplatine und zerstörungen durch über-/unterspannungen.
- software: die mit der datenbank kommunizierenden (client-)programme können fehlerhafte anweisungen auslösen, anweisungen unvollständig ausführen oder datenbestände unbeabsichtigt verfälschen.
- benutzer: versehentliches ändern oder löschen von daten.
- äußere einflüsse: viren

7.3 sicherungsstrategien

spätestens mit der installation von sql server sollten sie auch die sicherungsstrategien für ihre datenbanken festlegen. in diesem zusammenhang sind unter anderem folgende fragen zu beantworten:

- wer ist für die sicherung verantwortlich?
- was ist zu sichern?
- wo wird gesichert?

- wie oft wird gesichert?
- wie überprüft man sicherungen?

7.3.1 wer ist für die sicherung verantwortlich?

diese frage ist eigentlich schnell beantwortet: der datenbankadministrator. doch wie sieht es aus, wenn es mehrere administratoren gibt? vertraut dann der eine auf den anderen? was passiert bei urlaub oder krankheit? hier ist also genau festzulegen, wer die sicherungen durchzuführen hat, wobei auch bei abwesenheit der vorgesehenen person ein ersatz gewährleistet sein muß.

7.3.2 was ist zu sichern?

kurz gesagt: alle system- und benutzerdatenbanken. zu den systemdatenbanken gehören:

- master
- msdb
- distribution (wenn der server als replikationsverteiler konfiguriert wurde)
- model

die datenbank tempdb brauchen sie nicht zu sichern, da sie beim start von sql server neu erstellt und beim herunterfahren ohnehin gelöscht wird.

7.3.3 sicherungsmodi

beim sichern einer datenbank wird eine kopie der datenbank angelegt. das betrifft den gesamten inhalt der datenbank sowie die erforderlichen abschnitte des transaktionsprotokolls. nicht gesichert werden unvollständige transaktionen oder transaktionen, die nach dem starten der datenbanksicherung ausgeführt werden.

sql server bietet die folgenden arten von sicherungen:

- datenbanksicherungen
- differentielle datenbanksicherungen
- transaktionsprotokollsicherungen
- datei-/dateigruppensicherungen

datenbanksicherungen

bei einer datenbanksicherung erstellt sql server eine vollständige sicherung - d.h. eine kopie - der datenbank. diese sicherungsart umfaßt sämtliche datenbankobjekte, einschließlich der benutzerdefinierten objekte, systemtabellen und daten.

differentielle datenbanksicherungen

die differentielle datenbanksicherung ist ein neues merkmal der version 7.0 von sql server. bei dieser sicherungsart zeichnet sql server nur die änderungen auf, die seit der letzten vollständigen datenbanksicherung ausgeführt wurden. dadurch läßt sich die zeit für das erstellen von sicherungen

verringern, was vor allem bei häufigen datenbanksicherungen von vorteil ist. allerdings läßt sich der zustand einer datenbank vor dem ausfall im gegensatz zu einer transaktionsprotokollsicherung (siehe nächster abschnitt) nicht wiederherstellen. man ergänzt daher differentielle datenbanksicherungen durch transaktionsprotokollsicherungen im anschluß an die differentielle datenbanksicherung.

transaktionsprotokollsicherungen

mit transaktionsprotokollsicherungen läßt sich der zustand eines systems bis zu dem punkt wiederherstellen, zu dem ein ausfall aufgetreten ist. das setzt voraus, daß man sowohl datenbank- als auch transaktionsprotokollsicherungen durchführt.

im transaktionsprotokoll stehen alle vollständigen transaktionen, die seit der letzten vollständigen, differentiellen oder transaktionsprotokollsicherung ausgeführt wurden. während aber eine differentielle datenbanksicherung alle änderungen an einer datenbank berücksichtigt, enthält eine transaktionsprotokollsicherung nur die protokollierten operationen.

bei einer transaktionsprotokollsicherung werden folgende operationen ausgeführt:

- der inaktive teil des transaktionsprotokolls wird auf das sicherungsgerät kopiert.
- der inaktive teil des transaktionsprotokolls wird abgeschnitten (löschen und speicher freigeben).

datei-/dateigruppensicherungen

bei sehr großen datenbanken kann das für eine sicherung zur verfügung stehende zeitfenster zu klein sein, um eine vollständige datenbanksicherung durchzuführen. für diese fälle bietet sql server die möglichkeit, einzelne dateien oder dateigruppen zu sichern. sind zum beispiel für sicherungsaufgaben die nachtstunden von 2 uhr bis 4 uhr vorgesehen und würde eine vollständige sicherung länger als 2 stunden in anspruch nehmen, läßt sich ein teil der datenbank sofort und der andere am nächsten tag sichern. datei- und dateigruppensicherungen setzen in der regel eine transaktionsprotokollsicherung voraus, damit bei einer wiederherstellung einzelner dateien oder dateigruppen der konsistente zustand der datenbank gewährleistet wird.

7.3.4 wo wird gesichert?

die sicherungen werden auf sogenannten *sicherungsmedien* physisch gespeichert. das können datenträger-, band- und named-pipe-medien sein. bei datenträgermedien erfolgt die sicherung in form von betriebssystemdateien. als datenträger kommen sowohl lokale festplatten als auch freigegebene remote-festplatten in frage. den standort der dateien geben sie bei remote-sicherungen als unc-name der form \\servername\freigabename\pfad\datei an. lokale sicherungen sollten nicht auf derselben festplatte erfolgen, die auch die zu sichernden datenbanken enthält. bei einem ausfall der platte gibt es keine möglichkeit, die datenbanken wiederherzustellen.

eine sicherung auf bandmedien kann nur auf geräte erfolgen, die physisch am sql-server-computer angeschlossen sind. remote-sicherungen auf bandgeräte werden nicht unterstützt.

named pipe-medien erlauben unter anderem, daß sie sicherungssoftware von drittanbietern einsetzen können.

sql server unterscheidet zwischen physischen und logischen medien. der physische name ist der im betriebssystem verwendete name (zum beispiel e:\mssql7\backup\pubsbak.bak). einfacher ist es, eine datenbanksicherung unter einem logischen namen wie pubsbak anzusprechen. das ist ein alias, den sql server zusammen mit den verweisen auf die physischen dateien in den systemtabellen verzeichnet.

sql server bietet die möglichkeit,

- sicherungsmedien zu verwenden, die vom sicherungsgerät erstmalig verwendet werden,
- sicherungen an das ende einer bereits auf dem medium vorhandenen sicherung anzuhängen und
- vorhandene sicherungen auf bereits verwendeten medien zu überschreiben.

neu in sql server 7.0 sind die konzepte des mediensatzes und der medienfamilie. damit lassen sich mehrere sicherungsmedien und sicherungsgeräte einsetzen.

7.3.5 wie oft wird gesichert?

die beantwortung dieser frage ist von vielen faktoren abhängig. bei datenbanken, in denen vorwiegend abfragen laufen und die daten nur selten geändert werden, sind sicherungen seltener erforderlich, als bei datenbanken mit starkem aufkommen an transaktionen.

für die systemdatenbanken master und msdb von sql server sollten sie wöchentliche sicherungen einplanen und diese datenbanken darüber hinaus sichern, wenn größere änderungen an anderen datenbanken, medien oder konfigurationseinstellungen vorgenommen wurden.

wie bereits erwähnt, brauchen sie die datenbank tempdb nicht zu sichern.

von der systemdatenbank model legen sie eine grundsätzliche sicherung an. weitere sicherungen sind in der regel nicht erforderlich.

für alle benutzerdatenbanken empfiehlt sich eine tägliche sicherung, und zwar zu einem zeitpunkt, wenn die nutzung der datenbanken am geringsten ist (normalerweise nachts). zumindest sollten sie das transaktionsprotokoll jeden tag sichern, falls die zeit für eine vollständige sicherung der datenbanken nicht ausreicht, und eine reguläre sicherung der datenbanken an jedem wochenende einplanen.

auf die sicherung der beispieldatenbanken northwnd und pubs können sie prinzipiell verzichten. diese datenbanken lassen sich ohne weiteres mit den installationsskripts instnwnd.sql bzw. instpubs.sql (im verzeichnis \mssql7\install) neu erstellen. falls sie diese datenbanken allerdings zum testen einsetzen und auf die vorgenommenen änderungen auch nach einem absturz nicht verzichten möchten, behandeln sie die betreffende datenbank wie jede benutzerdatenbank.

7.3.6 wie überprüft man sicherungen?

ein ausgefeilter sicherungsplan ist praktisch wertlos, wenn sie blind darauf vertrauen, daß die sicherungen im ernstfall auch funktionieren. deshalb sollten sie eine möglichkeit vorsehen, wie sie die sicherungen überprüfen können. hier bietet sich ein separater server an, auf dem sie die datenbanken wiederherstellen. das hat außerdem den vorteil, daß ein reservecomputer bereitsteht, auf den bei ausfall des hauptservers umgeschaltet werden kann.

7.4 sicherungen vorbereiten

bevor sie eine datenbank sichern, müssen sie zunächst ein sicherungsmedium erstellen. als sicherungsmedium kommen festplatten, bandlaufwerke oder netzlaufwerke in frage. die eingerichteten sicherungsmedien werden in der systemtabelle sysdevices verzeichnet.

7.4.1 sicherungsmedium mit enterprise manager erstellen

im enterprise manager erstellen sie ein sicherungsmedium in folgenden schritten:

1. erweitern sie die konsolenstruktur bis zum ordner **verwaltung**. im ordner **verwaltung** klicken sie mit der rechten maustaste auf den eintrag **sicherung** und wählen aus dem kontextmenü den befehl **neues sicherungsmedium**.
2. im dialogfeld **eigenschaften von sicherungsmedium - neues medium** (siehe abbildung 7.1) geben sie einen logischen namen für das medium in das feld **name** ein, im beispiel pubsbak. die option *name des bandlaufwerks* ist nur verfügbar, wenn ein bandlaufwerk an den server angeschlossen ist. die option *dateiname* gilt für eine sicherung auf einer festplatte. im zugehörigen bearbeitungsfeld können sie den vorgegebenen pfad überschreiben oder über die schaltfläche mit den drei punkten auswählen. im beispiel wird die datenbank pubs auf der zweiten festplatte (laufwerk g:) des servers eins gesichert.

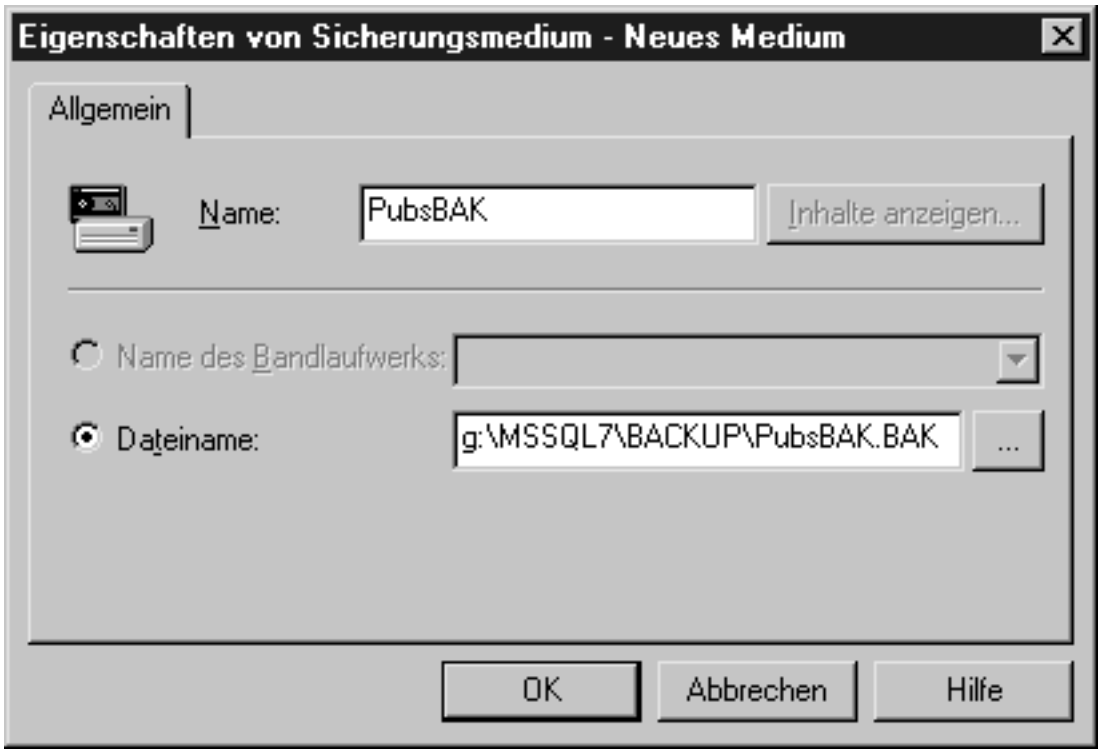


abbildung 7.1: neues sicherungsmedium erstellen

3. geben sie die erforderlichen informationen ein. klicken sie dann auf **ok**, um das sicherungsmedium erstellen zu lassen.

7.4.2 sicherungsmedium per transact-sql erstellen

ein sicherungsmedium erstellen sie mit der gespeicherten prozedur sp_addumpdevice, die folgende syntax aufweist:

```
sp_addumpdevice [@devtype =] 'medientyp',
[@logicalname =] 'logischername',
[@physicalname =] 'physischername'
[,
{ [@cntrltype =] controllertyp |
  [@devstatus = ] {'noskip' | 'skip'} }
]
```

als *medientyp* geben sie 'disk' für festplatten, 'pipe' für named pipe oder 'tape' für bandlaufwerke an. der parameter *logischername* ist erforderlich und bezeichnet den logischen namen des sicherungsmediums. der physische name ist ebenfalls erforderlich. hier spezifizieren sie den vollständigen pfad und den dateinamen der sicherungsdatei. der name muß den regeln des betriebssystems oder den unc-konventionen für netzwerkmedien entsprechen.

die parameter *controllertyp* und *@devstatus* sind zwar nicht erforderlich, können aber angegeben werden - allerdings nur jeweils ein parameter und nicht beide zusammen. für *controllertyp* sind die werte 2 für festplatten, 5 für bandlaufwerke und 6 für named pipe möglich. der parameter *@devstatus* gibt an, ob sql server ansi-bandbezeichnungen liest (noskip) oder ignoriert (skip).

die transact-sql-anweisung für das im letzten abschnitt mit dem enterprise manager erstellte sicherungsmedium lautet dann wie folgt:

```
sp_addumpdevice 'disk', 'pubsbak',
'g:\mssql7\backup\pubsbak.bak'
```

7.5 datenbanken sichern

eigentlich müßte diese überschrift »datenbanken vollständig und differentiell sowie transaktionsprotokolle und datei-/dateigruppen sichern« lauten, denn sql server verpackt die genannten sicherungsszenarien unter einer einheitlichen benutzeroberfläche. die schritte sind bei allen sicherungsformen gleich.

sql server erlaubt dynamische sicherungen. es ist also nicht erforderlich, die benutzung einer datenbank zu unterbrechen, um die sicherung durchzuführen. allerdings gibt es einige ausnahmen. eine datenbank kann man nicht sichern, wenn

- eine nicht protokollierte operation läuft,

- ein index erstellt wird,
- datenbankdateien erzeugt oder gelöscht werden,
- die datenbank (manuell oder automatisch) verkleinert wird.

die nächsten abschnitte zeigen jeweils eine sicherung mit dem sicherungs-assistenten, mit dem enterprise manager und per transact-sql.

7.5.1 sicherungs-assistent

gerade wenn sie nur gelegentlich die aufgaben eines datenbankadministrators wahrnehmen, bieten ihnen die assistenten von sql server wertvolle unterstützung für eine erfolgreiche arbeit. auch für die datenbanksicherung ist ein assistent vorhanden. das folgende beispiel zeigt, wie sie mit dem sicherungs-assistenten die beispieldatenbank pubs auf einer zweiten festplatte des servers eins sichern. führen sie die folgenden schritte aus:

1. um den sicherungs-assistenten zu starten, erweitern sie im enterprise manager die konsolenstruktur bis zum eintrag eins (windows nt), wählen dann im menü **extras** den befehl **assistenten** und erweitern im dialogfeld **assistent auswählen** den zweig **verwaltung** (siehe abbildung 7.2).

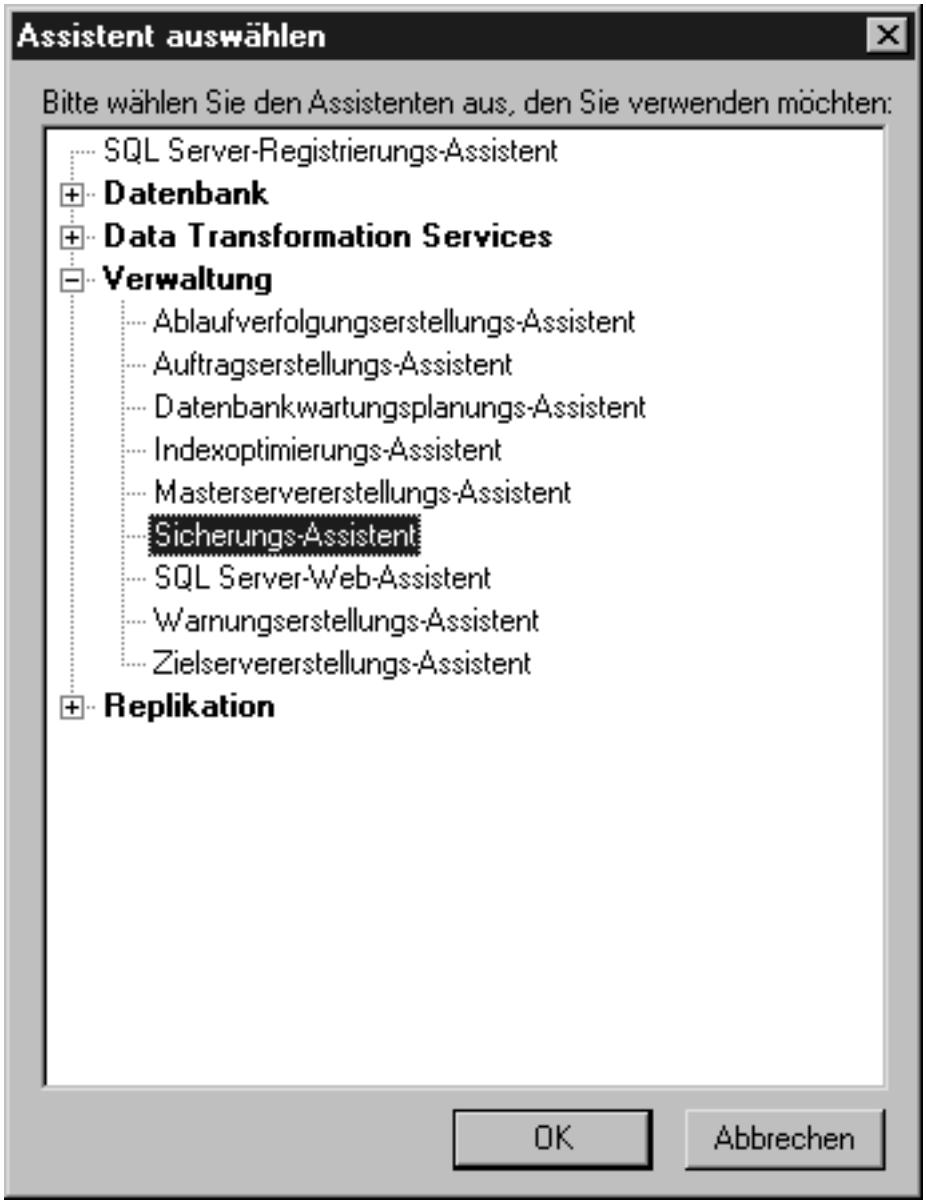


abbildung 7.2: im dialogfeld assistent auswählen markieren sie den sicherungs-assistenten

2. klicken sie im startdialogfeld des sicherungs-assistenten auf **weiter**. im zweiten dialogfeld wählen sie im listenfeld **datenbankname** die zu sichernde datenbank aus (im beispiel pubs).
3. das dritte dialogfeld gibt als name der sicherung pubs sicherung vor. bei bedarf überschreiben sie diesen namen einfach. optional können sie eine beschreibung für die sicherung eingeben. klicken sie auf **weiter**.
4. das dialogfeld **sicherungstyp auswählen** (siehe abbildung 7.3) stellt drei optionen für den sicherungstyp bereit. eine datei-/dateigruppensicherung können sie mit dem assistenten nicht ausführen. wählen sie für das beispiel die vollständige sicherung, und klicken sie auf **weiter**.

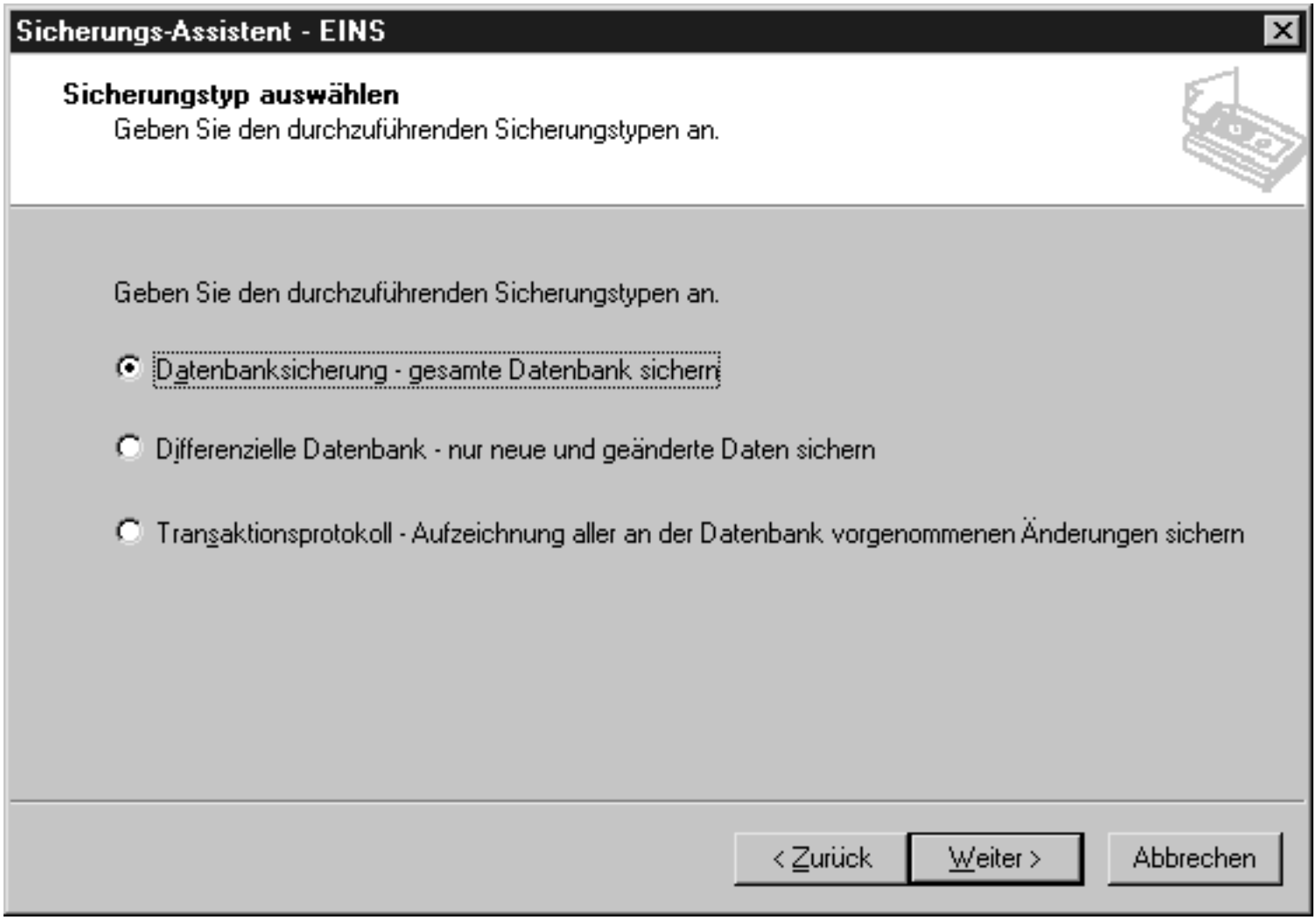


abbildung 7.3: das dialogfeld sicherungstyp auswählen

5. im dialogfeld **sicherungsziel und -aktionen auswählen** (siehe abbildung 7.4) legen sie fest, wo sie die sicherung speichern wollen. hier können sie den vollständigen pfad und den dateinamen angeben (option *datei*) oder ein bereits erstelltes sicherungsmedium wählen. im abschnitt **eigenschaften** haben sie die wahl, ob sie den sicherungssatz an bereits bestehende sicherungen anhängen (option *an medien anhängen*) oder vorhandene sicherungssätze löschen und mit einem neuen sicherungssatz beginnen wollen (option *medien überschreiben*).

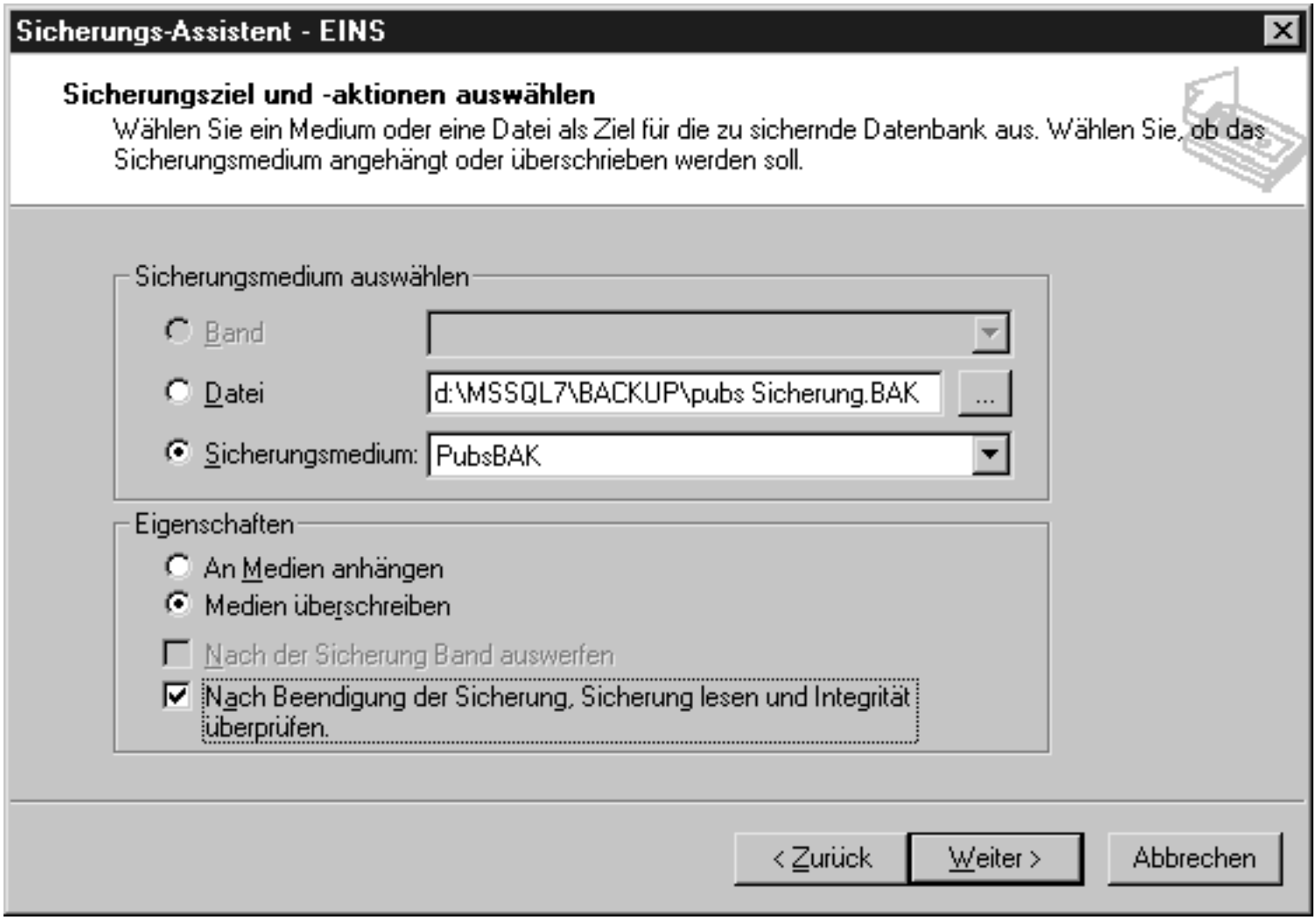


abbildung 7.4: hier legen sie das ziel der sicherung und weitere optionen fest

die option *an medien anhängen* bewirkt, daß der sicherungssatz an das ende der bereits erstellten sicherungen angefügt wird. es ist nicht möglich, bereits bestehende sicherungen zu überspringen und etwa die vierte von sechs sicherungen zu überschreiben.

in diesem abschnitt sollten sie auch das kontrollkästchen für die integritätsprüfung der sicherung einschalten, um eine überprüfung nach erfolgter sicherung vornehmen zu lassen.

bei der option *sicherungsmedium* haben die pfad- und dateinamen im bearbeitungsfeld der option *datei* sowie der im schritt 3 des assistenten festgelegte name der sicherung keine bedeutung. wenn sie das sicherungsmedium pubsbak wie im letzten abschnitt angegeben für den pfad g:\mssql7\backup\ und den dateinamen pubsbak.bak erstellt haben, schreibt sql server die sicherung unter dem namen pubsbak.bak (und nicht pubs sicherung.bak) in das verzeichnis g:\mssql7\backup\. es ist nicht erforderlich, daß sie das verzeichnis im dialogfeld ändern.

6. wenn sie im abschnitt **eigenschaften** die option *medien überschreiben* gewählt haben, gelangen sie zunächst zum dialogfeld **medien initialisieren** (siehe abbildung 7.5). hier können sie einen namen für den sicherungssatz festlegen. bei darauffolgenden sicherungen prüft sql den spezifizierten mediensatz, um ein versehentliches überschreiben zu verhindern.

auch wenn sie das kontrollkästchen **medien initialisieren und bezeichnen** nicht einschalten, werden die vorhandenen sicherungen überschrieben und durch die neue sicherung ersetzt.

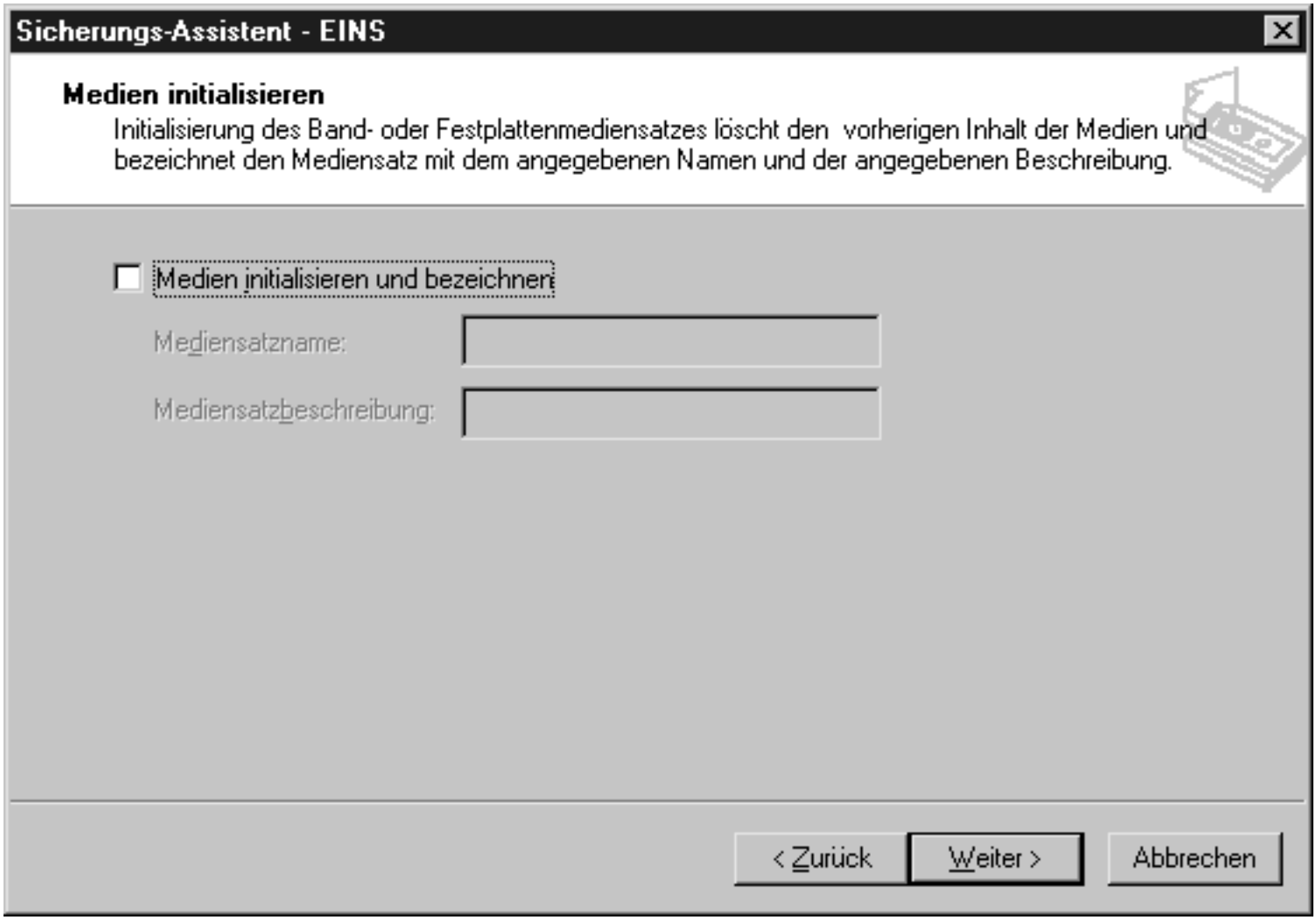


abbildung 7.5: das dialogfeld medien initialisieren

7. im dialogfeld **sicherungsüberprüfung und terminplanung** (siehe abbildung 7.6) können sie eine überprüfung spezifizieren, ob der ausgewählte mediensatz das verfallsdatum überschritten hat und überschrieben werden kann. im abschnitt **mediensatzüberprüfung** können sie den namen des zu prüfenden mediensatzes angeben, um ein versehentliches überschreiben zu verhindern. weiterhin bietet dieses dialogfeld die möglichkeit, einen terminplan für die sicherung einzurichten. nachdem sie alle gewünschten einstellungen vorgenommen haben, klicken sie auf **weiter**.

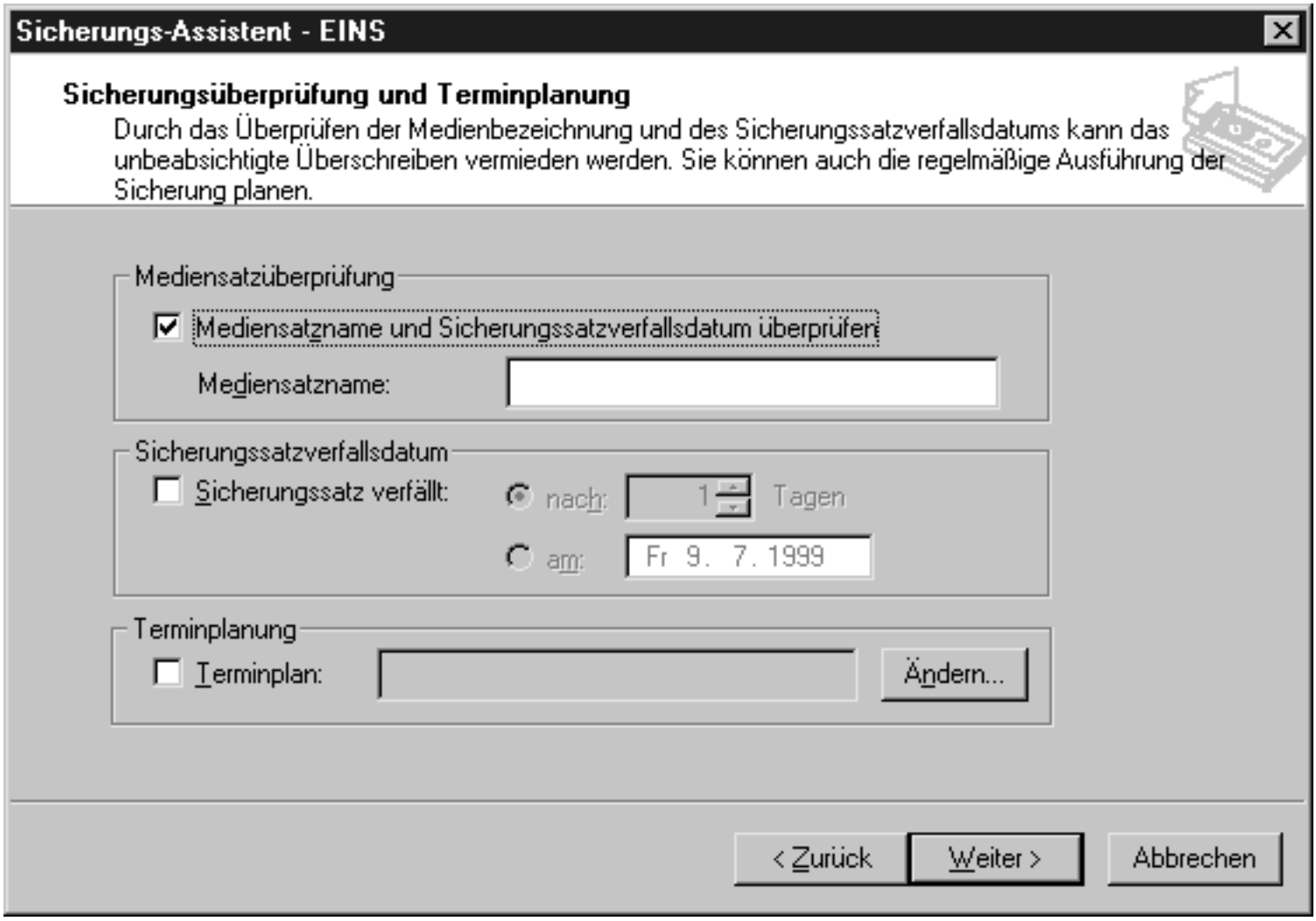


abbildung 7.6: das dialogfeld sicherungsüberprüfung und terminplanung

8. das letzte dialogfeld des sicherungs-assistenten (siehe abbildung 7.7) faßt die gewählten einstellungen zusammen. klicken sie auf **fertigstellen**, um die sicherung erstellen zu lassen. falls sie die überprüfung aktiviert haben, erscheint nach abschluß der sicherung eine entsprechende meldung.

bild

abbildung 7.7: letztes dialogfeld des sicherungs-assistenten

auch wenn sie die überprüfung eingeschaltet haben, sollten sie aus den zu beginn des kapitels im abschnitt »wie überprüft man sicherungen?« genannten gründen eine separate prüfung der sicherung durchführen.

7.5.2 enterprise manager

um eine datenbank über den enterprise manager zu sichern, erweitern sie zuerst in der konsolenstruktur die server-gruppe und den server, auf dem sich die zu sichernde datenbank befindet. die folgenden schritte zeigen wieder am beispiel der datenbank pubs, wie sie eine datenbank mit dem enterprise manager sichern:

1. klicken sie mit der rechten maustaste auf den ordner **datenbanken**. sie können den ordner auch

erweitern und mit der rechten maustaste auf die zu sichernde datenbank klicken.

2. im kontextmenü wählen sie **alle aufgaben** und im eingeblendeten untermenü **datenbank sichern**.
3. es erscheint das dialogfeld **sql server sicherung -** *datenbankname*. über das drop-down-listenfeld **datenbank** können sie gegebenenfalls eine andere datenbank zur sicherung auswählen.

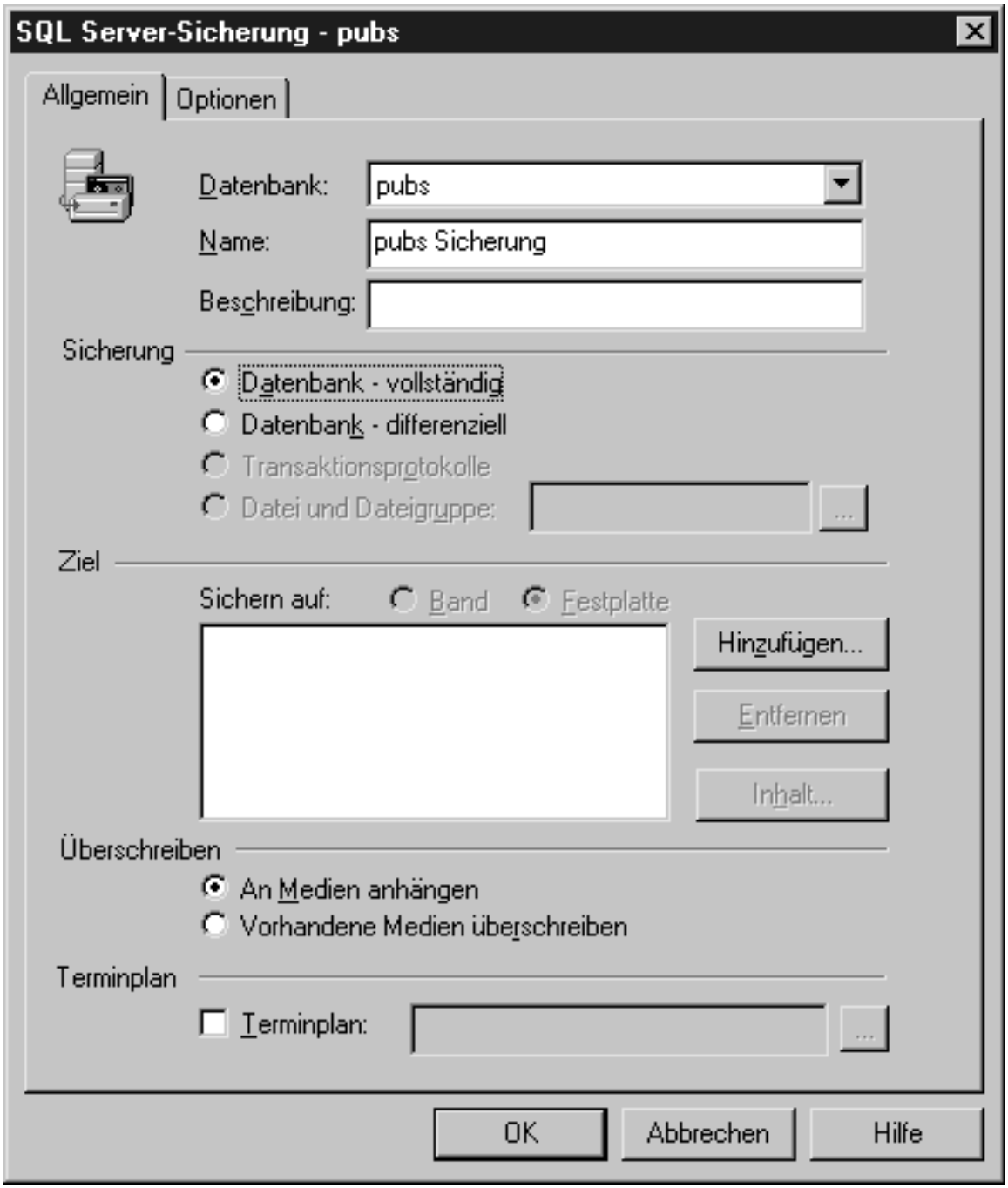


abbildung 7.8: die registerkarte allgemein des dialogfelds sql server-sicherung

4. das dialogfeld **sql server-sicherung** faßt auf zwei registerkarten alle schritte zusammen, die sie mit dem sicherungs-assistenten im letzten abschnitt durchlaufen haben. als erstes wählen sie auf der registerkarte **allgemein** den typ der sicherung aus. je nach struktur der datenbank ist hier auch die option *datei und dateigruppe* verfügbar.
5. im abschnitt **ziel** wählen sie das sicherungsmedium aus. wenn noch kein medium eingetragen ist oder sie ein neues medium verwenden wollen, klicken sie auf **hinzufügen**. im dialogfeld

sicherungsziel auswählen (siehe abbildung 7.9) legen sie entweder den dateinamen oder das sicherungsmedium fest. nachdem sie auf **ok** geklickt haben, erscheint das ziel auf der registerkarte **allgemein**.

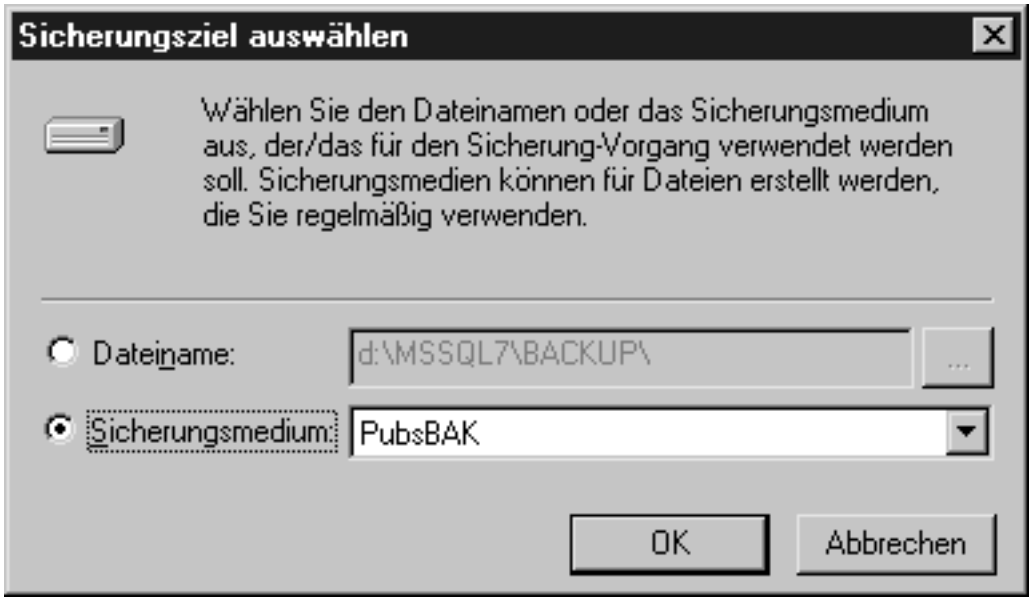


abbildung 7.9: das dialogfeld sicherungsziel auswählen

6. wie im sicherungs-assistenten können sie wählen, ob sie die sicherung an vorhandene medien anhängen oder die medien überschreiben wollen.
7. soll die sicherung nur einmalig erfolgen, lassen sie das kontrollkästchen **terminplan** ausgeschaltet und gehen zu schritt 12 weiter.
8. wenn die sicherung zu einem späteren zeitpunkt oder regelmäßig ausgeführt werden soll, schalten sie das kontrollkästchen **terminplan** ein und klicken auf die schaltfläche mit den drei punkten, um das dialogfeld **terminplan bearbeiten** zu öffnen. hier können sie einen namen für den terminplan und die details der ausführung spezifizieren (siehe abbildung 7.10)

abbildung 7.10: das dialogfeld terminplan bearbeiten

9. in der voreinstellung ist eine wöchentliche sicherung angegeben. klicken sie auf **ändern**, wenn sie das intervall und den zeitpunkt neu festlegen möchten.
10. nach der eingangs in diesem kapitel gegebenen empfehlung, die benutzerdatenbank täglich zu sichern, wählen sie im abschnitt **häufigkeit** die option *täglich* (siehe abbildung 7.11). prüfen sie allerdings im einzelfall, ob das wirklich notwendig ist. es kann durchaus passieren, daß sie bei einer täglichen sicherung mehrerer datenbanken den festplattenplatz allein mit sicherungen in den verschiedenen stadien verbrauchen.

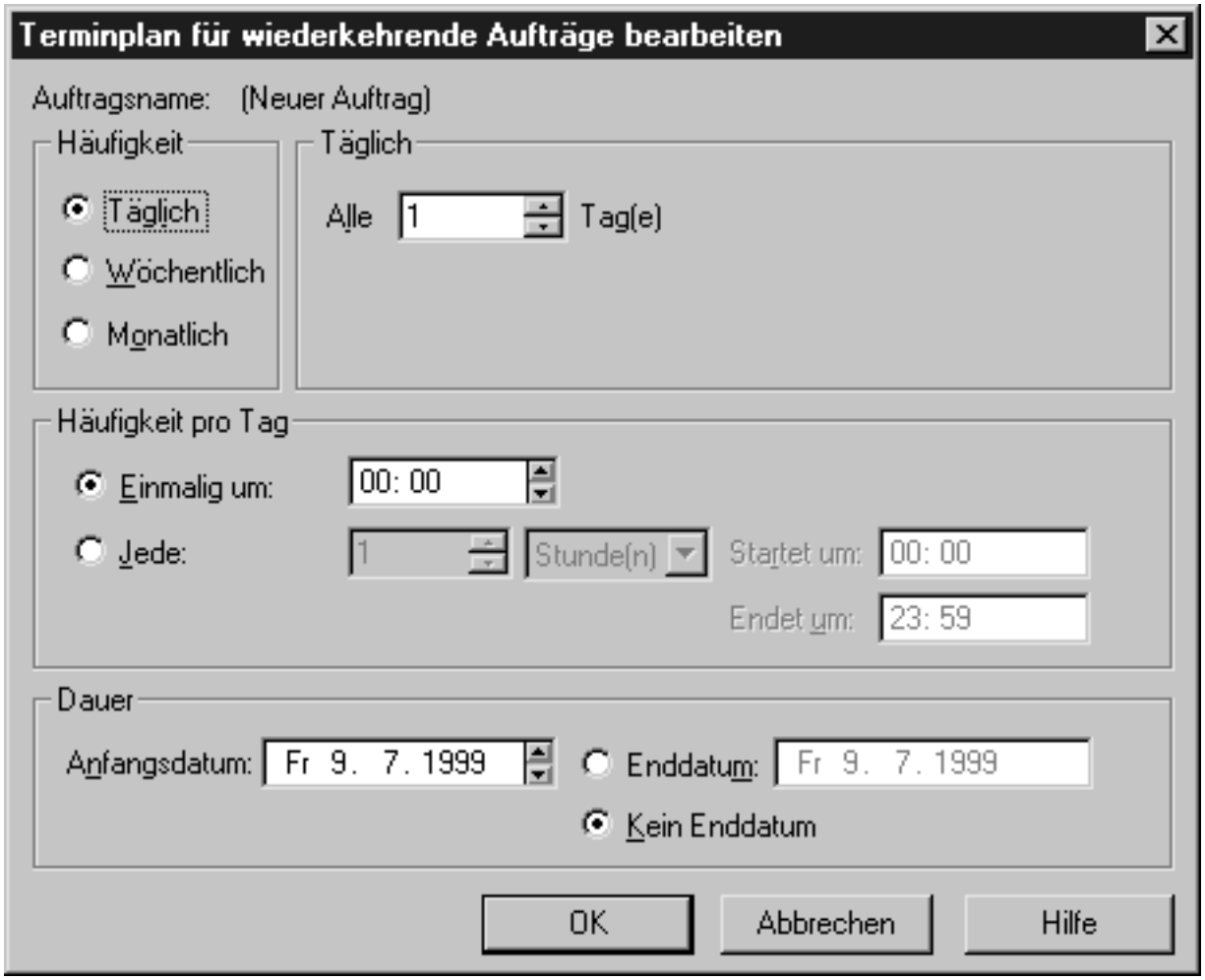


abbildung 7.11: in diesem dialogfeld legen sie das intervall für regelmäßige sicherungen fest

11. bestätigen sie die einstellungen mit **ok**, und klicken sie auch im dialogfeld **terminplan bearbeiten** auf **ok**, um zum dialogfeld **sql server-sicherung** zurückzukehren.
12. aktivieren sie die registerkarte **optionen**. schalten sie das kontrollkästchen **sicherung nach beendigung überprüfen** ein, damit sql server die integrität der sicherung testet. wenn sie auf der registerkarte **allgemein** die option *vorhandene medien überschreiben* gewählt haben, läßt sich das verfallsdatum für den sicherungssatz festlegen. bei der gewählten täglichen sicherung können sie den sicherungssatz beispielsweise nach 7 tagen zum überschreiben freigeben. schalten sie dazu das kontrollkästchen **sicherung verfällt** ein, wählen sie die option *nach tagen*, und tragen sie die 7 ein.

die verfallsoptionen verhindern lediglich, daß die sicherung versehentlich überschrieben wird. es ist also nicht so, daß beim gewählten täglichen intervall und einem verfallsdatum nach 7 tagen die 8. sicherung die 1. sicherung automatisch überschreibt, oder die 9. sicherung die zweite ersetzt usw. wie bereits weiter vorn erwähnt, können sie eine sicherung nur anhängen, oder sie müssen einen neuen mediensatz erstellen (und den alten löschen).

bild

abbildung 7.12: die registerkarte optionen des dialogfelds sql server-sicherung

13. klicken sie im dialogfeld **sql server-sicherung - pubs** auf **ok**.

bei einer geplanten sicherung wie im beispiel wird die sicherung nicht sofort, sondern erst zum

angegebenen zeitpunkt durchgeführt. wenn sie sich den zugehörigen auftrag ansehen möchten, erweitern sie die konsolenstruktur bis zum ordner **verwaltung**, öffnen diesen ordner, erweitern **sql server-agent** und klicken auf den eintrag **aufträge**.

7.5.3 sichern per transact-sql

für das sichern von datenbanken, transaktionsprotokollen und dateien/dateigruppen unterscheidet transact-sql drei formen der anweisung backup. die älteren dump-anweisungen stehen nur noch aus gründen der abwärtskompatibilität zur verfügung. die syntax der backup-anweisungen sieht wie folgt aus:

sichern der gesamten datenbank

```
backup database datenbankname
to <sicherungsmedium> [,...n]
[with
[blocksize = blockgröße]
[[,] description = text]
[[,] differential]
[[,] expiredate = datum
| retaindays = tage]
[[,] format | noformat]
[[,] {init | noinit}]
[[,] mediadescription = text]
[[,] medianame = medienname]
[[,] [name = sicherungssatz]
[[,] {noskip | skip}]
[[,] {nounload | unload}]
[[,] [restart]
[[,] stats [= prozentsatz]]
]
```

sichern von transaktionsprotokollen

```
backup log datenbankname
{[with { no_log | truncate_only }]}
|
{
to <sicherungsmedium> [,...n]
[with
[blocksize = blockgröße]
[[,] description = text]
```

```
[[,] expiredate = datum
| retaindays = tage]
[[,] format | noformat]
[[,] {init | noinit}]
[[,] mediadescription = text]
[[,] medianame = medienname]
[[,] [name = sicherungssatz]
[[,] no_truncate]
[[,] {noskip | skip}]
[[,] {nounload | unload}]
[[,] [restart]
[[,] stats [= prozentsatz]]
]
}
```

sichern von datei-/dateigruppen

```
backup database datenbankname
<datei/dateigruppe> [,...n]
to <sicherungsmedium> [,...n]
[with
[blocksize = blockgröße]
[[,] description = text]
[[,] expiredate = datum
| retaindays = tage]
[[,] format | noformat]
[[,] {init | noinit}]
[[,] mediadescription = text]
[[,] medianame = medienname]
[[,] [name = sicherungssatz]
[[,] {noskip | skip}]
[[,] {nounload | unload}]
[[,] [restart]
[[,] stats [= prozentsatz]]
]
```

die parameter *datenbankname*, *blockgröße*, *text*, *datum*, *tage*, *medienname* und *sicherungssatz* können auch als lokale variable angegeben werden.

der parameter *sicherungsmedium* hat für alle drei backup-anweisungen folgendes format:

```
{ sicherungsmedienname | {disk | tape | pipe} =
temporäressicherungsmedium }
```

für den parameter *datei/dateigruppe* gilt folgendes format:

```
{ file = logischerdateiname | filegroup =
logischerdateigruppenname }
```

sicherungsmedienname, *temporäressicherungsmedium*, *logischerdateiname* und *logischerdateigruppenname* können sie auch als lokale variable angeben.

7.5.4 optionen für protokollkürzung

ein transaktionsprotokoll setzt sich aus einem aktiven und einem inaktiven teil zusammen. der aktive teil muß immer zur verfügung stehen, damit sich eine datenbank beim start von sql server gegebenenfalls wiederherstellen läßt. im aktiven teil sind transaktionen enthalten, die noch nicht abgeschlossen sind, während der inaktive teil des protokolls nur abgeschlossene transaktionen enthält.

da jede änderung an einer datenbank zu einer vergrößerung des transaktionsprotokolls führt, muß das protokoll von zeit zu zeit bereinigt werden, damit es nicht irgendwann den gesamten platz auf der festplatte einnimmt. dieser vorgang heißt *protokollkürzung* oder *abschneiden des transaktionsprotokolls*.

in diesem zusammenhang verdienen die optionen für die protokollkürzung eine nähere betrachtung.

truncate_only

diese option löscht den inaktiven teil des transaktionsprotokolls, ohne das protokoll auf das sicherungsmedium zu kopieren. aus diesem grund erübrigt sich auch die angabe eines sicherungsmediums. eine anweisung zum sichern des transaktionsprotokolls mit dieser option sieht dann zum beispiel wie folgt aus:

```
backup log pubs with truncate_only
```

wenn sie ausschließlich mit vollständigen datenbanksicherungen und wiederherstellungen arbeiten, »sichern« sie das transaktionsprotokoll mit angabe der option truncate_only unmittelbar nach einer vollständigen datenbanksicherung, um den platz für den inaktiven teil des protokolls freizugeben.

generell sollten sie eine vollständige oder differentielle datenbanksicherung ausführen, *bevor* sie das protokoll mit der option truncate_only abschneiden. andernfalls können sie die vollständigen transaktionen im inaktiven teil des protokolls nicht wiederherstellen.

no_log

diese option entspricht im wesentlichen truncate_only, protokolliert aber nicht den befehl backup log. führen sie immer eine vollständige datenbanksicherung aus, nachdem sie das protokoll mit no_log abgeschnitten haben.

verwenden sie diese option nur als notbremse, wenn das transaktionsprotokoll rettungslos überfüllt ist. normalerweise können sie darauf verzichten, da sql server das transaktionsprotokoll automatisch vergrößert.

no_truncate

die sicherung des transaktionsprotokolls mit der option no_truncate sichert das protokoll, ohne es abzuschneiden. verwenden sie diese option, wenn die zugehörige datenbank beschädigt ist und sie die datenbank wiederherstellen wollen. allerdings muß sich das transaktionsprotokoll auf einem von der datenbank getrennten sicherungsmedium befinden, und die datenbank master darf nicht beschädigt sein. wenn sie die datenbank wiederherstellen, führen sie die wiederherstellung des mit no_truncate gesicherten transaktionsprotokolls als letzten schritt in diesem prozeß aus.

7.6 wiederherstellung

ihre datenbanken haben sie nun - hoffentlich noch rechtzeitig - gesichert. jetzt geht es darum, wie sie im ernstfall eine datenbank aus einer vorhandenen sicherung wiederherstellen können.

im gegensatz zur sicherung von datenbanken bietet sql server für die wiederherstellung keinen assistenten. zum glück ist aber die wiederherstellung einer datenbank nicht weiter kompliziert und läßt sich ohne weiteres mit dem enterprise manager oder per transact-sql abwickeln.

7.6.1 enterprise manager

das beispiel in diesem abschnitt geht davon aus, daß sie die datenbank pubs auf dem server eins im verzeichnis g:\mssql7\backup\ gesichert haben. da sich die datenbank pubs gegebenenfalls über das skript instpubs.sql neu aufbauen läßt, können sie die datenbank pubs auf dem server eins löschen und dann aus der vorhandenen sicherung wiederherstellen.

die datenbank pubs löschen sie zum beispiel mit der transact-sql-anweisung:

```
drop database pubs
```

als nächstes führen sie die folgenden schritte aus, um die datenbank pubs über den enterprise manager wiederherzustellen:

1. klicken sie in der konsolenstruktur mit der rechten maustaste auf den ordner **datenbanken**. aus dem kontextmenü wählen sie den befehl **alle aufgaben** und dann **datenbank wiederherstellen**. dieser befehl ist auch über das menü **extras** erreichbar. daraufhin wird das dialogfeld **datenbank wiederherstellen** (siehe abbildung 7.13) geöffnet.

bild

abbildung 7.13: die registerkarte allgemein des dialogfelds datenbank wiederherstellen

2. wenn eine datenbank beschädigt ist und noch nicht gelöscht wurde, können sie sie im drop-down-listenfeld **wiederherstellen als datenbank** auswählen. im beispiel haben sie die datenbank gelöscht, so daß sie den namen direkt eintragen müssen.
3. als nächstes wählen sie die option für die art der wiederherstellung. für das beispiel übernehmen sie die voreingestellte option *datenbank*.
4. über das drop-down-listenfeld **sicherungskopien der folgenden datenbank anzeigen** im abschnitt **parameter** wählen sie die vorhandene sicherung aus. daraufhin erscheint im drop-down-listenfeld **erste wiederherzustellende sicherung** die sicherung, die sie als letztes angelegt haben - d.h. die neueste. in der liste darunter finden sie detaillierte angaben zu dieser sicherung. wenn sie den eintrag markieren, wird die schaltfläche **eigenschaften** verfügbar. über diese schaltfläche gelangen sie zum dialogfeld **sicherungseigenschaften**, das alle angaben im überblick zeigt. bei bedarf können sie in diesem dialogfeld auch ein anderes wiederherstellungsmedium wählen.
5. wechseln sie nun zur registerkarte **optionen** (siehe abbildung 7.14).

Bild

abbildung 7.14: die registerkarte optionen des dialogfelds datenbank wiederherstellen

6. schalten sie die kontrollkästchen für die gewünschten optionen im oberen teil des dialogfelds ein. wenn sie mehrere datenbanken wiederherstellen und das kontrollkästchen **bestätigung vor wiederherstellen jeder einzelnen sicherung** einschalten, können sie nach jedem einzelnen wiederherstellungsvorgang entscheiden, ob sie mit der wiederherstellung der übrigen datenbanken fortfahren oder abbrechen wollen. bei einer beschädigten datenbank schalten sie das kontrollkästchen **wiederherstellung über vorhandene datenbanken erzwingen** ein, damit die beschädigte datenbank automatisch durch die einwandfreie sicherung überschrieben wird. die optionen im abschnitt **wiederherstellungsstatus** dürften ebenfalls selbsterklärend sein. voreingestellt ist die erste option. die dritte option können sie zum beispiel verwenden, um den zustand einer datenbank zu überprüfen, wenn sie zwar wissen, *daß*, aber nicht *wann* ein problem aufgetreten ist. nach jeder wiederherstellung des transaktionsprotokolls führen sie dann einen test auf integrität durch, bis sie die sicherung gefunden haben, die das problem verursacht. anschließend führen sie die wiederherstellung noch einmal durch, lassen aber das letzte transaktionsprotokoll weg.
7. klicken sie abschließend auf **ok**, um die wiederherstellung zu starten.

während der wiederherstellung erscheint eine fortschrittsanzeige, nach beendeter wiederherstellung eine fertigmeldung. die datenbank pubs ist nun wieder im enterprise manager aufgeführt und kann benutzt werden.

7.6.2 wiederherstellen per transact-sql

auch für das wiederherstellen existieren drei formen der transact-sql-anweisung restore. die älteren load-anweisungen stehen nur noch aus gründen der abwärtskompatibilität zur verfügung. die syntax der restore-anweisungen hat folgendes aussehen:

wiederherstellen einer ganzen datenbank

```
restore database datenbankname
[from <sicherungsmedium> [,...n]]
[with
[dbo_only]
[[,] file = dateinummer]
[[,] medianame = medienname]
[[,] move 'logischerdateiname' to 'betriebssystemdateiname']
[,...n]
[[,] {norecovery | recovery | standby = rückgängigdateiname}]
[[,] {nounload | unload}]
[[,] replace]
[[,] restart]
[[,] stats [ = prozentsatz]]]
```

wiederherstellen eines transaktionsprotokolls

```
restore log datenbankname
[from <sicherungsmedium> [,...n]]
[with
[dbo_only]
[[,] file = dateinummer]
[[,] medianame = medienname]
[[,] {norecovery | recovery | standby = rückgängigdateiname}]
[[,] {nounload | unload}]
[[,] restart]
[[,] stats [ = prozentsatz]]]
[[,] stopat = datumzeit]
]
```

wiederherstellen von dateien und dateigruppen

```
restore database datenbankname
```

```
<datei/dateigruppe> [,...n]
[from <sicherungsmedium> [,...n]]
[with
[dbo_only]
[[,] file = dateinummer]
[[,] medianame = medienname]
[[,] norecovery]
[[,] {nounload | unload}]
[[,] replace]
[[,] restart]
[[,] stats [ = prozentsatz]]]
```

die parameter *datenbankname*, *medienname*, *datumzeit* können auch als lokale variable angegeben werden.

die parameter *sicherungsmedium* und *datei/dateigruppe* haben das gleiche format wie bei der transact-sql-anweisung restore.

7.6.3 master-datenbank neu erstellen

wenn sich alles gegen sie verschworen hat und die master-datenbank nach einem systemabsturz nicht mehr verfügbar oder nicht mehr herzustellen ist, können sie als letztes mittel die master-datenbank neu erstellen.

in den versionen vor sql server 7.0 wurde die master-datenbank durch das ausführen von sql server setup neu erstellt. sql server 7.0 unterstützt diese funktion nicht mehr, bietet dafür aber das dienstprogramm rebuild master. wenn sie die master-datenbank neu erstellen, gehen allerdings vorher erstellte datenbanken verloren. darüber hinaus sind noch weitere wiederherstellungsmaßnahmen erforderlich.

führen sie die folgenden schritte aus, um die master-datenbank neu zu erstellen, wenn keine sicherung mehr vorhanden ist:

1. beenden sie sql server.
2. starten sie das dienstprogramm rebuild master an der eingabeaufforderung von ms-dos (rebuildm.exe). es erscheint das in bild 7.15 gezeigte dialogfeld **rebuild master**.

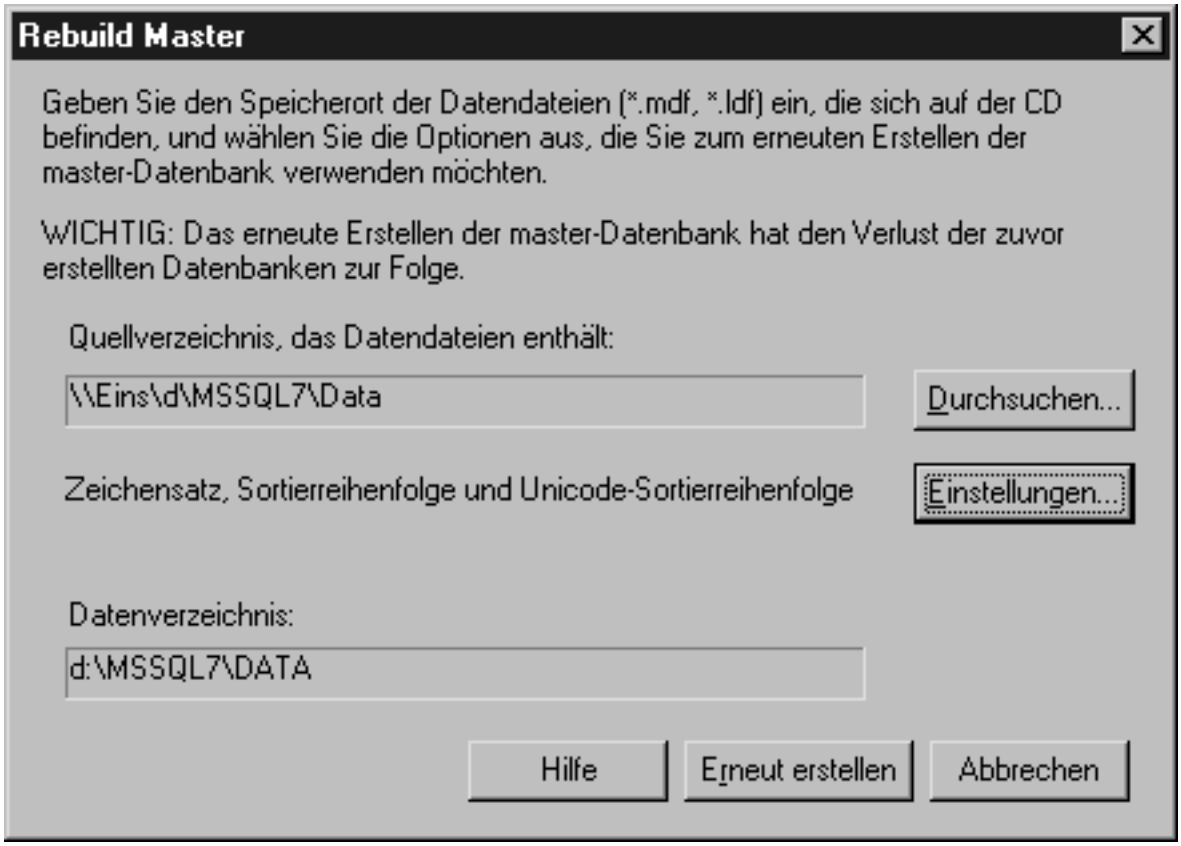


abbildung 7.15: das dialogfeld rebuild master

3. klicken sie auf **durchsuchen**, und wählen sie den ordner \data auf der sql-server-cd oder in dem freigegebenen netzwerkverzeichnis aus, von dem aus sql server installiert wurde.
4. klicken sie auf **einstellungen**. stellen sie sicher, daß der zeichensatz, die sortierreihenfolge und die unicode-sortierreihenfolge auf die einstellungen gesetzt sind, die sie bisher für die master-datenbank und alle anderen datenbanken festgelegt hatten. andernfalls lassen sich die datenbanken nicht wiederherstellen. dieser schritt ist auch erforderlich, wenn sie die einstellungen gar nicht ändern möchten, da sonst die schaltfläche **erneut erstellen** deaktiviert bleibt.
5. klicken sie auf **erneut erstellen**. das dienstprogramm rebuild master erstellt daraufhin die master-datenbank neu.

das war der einfachere teil der wiederherstellung. aufwendiger und komplizierter wird sich in der regel die wiederherstellung der anderen datenbanken gestalten.

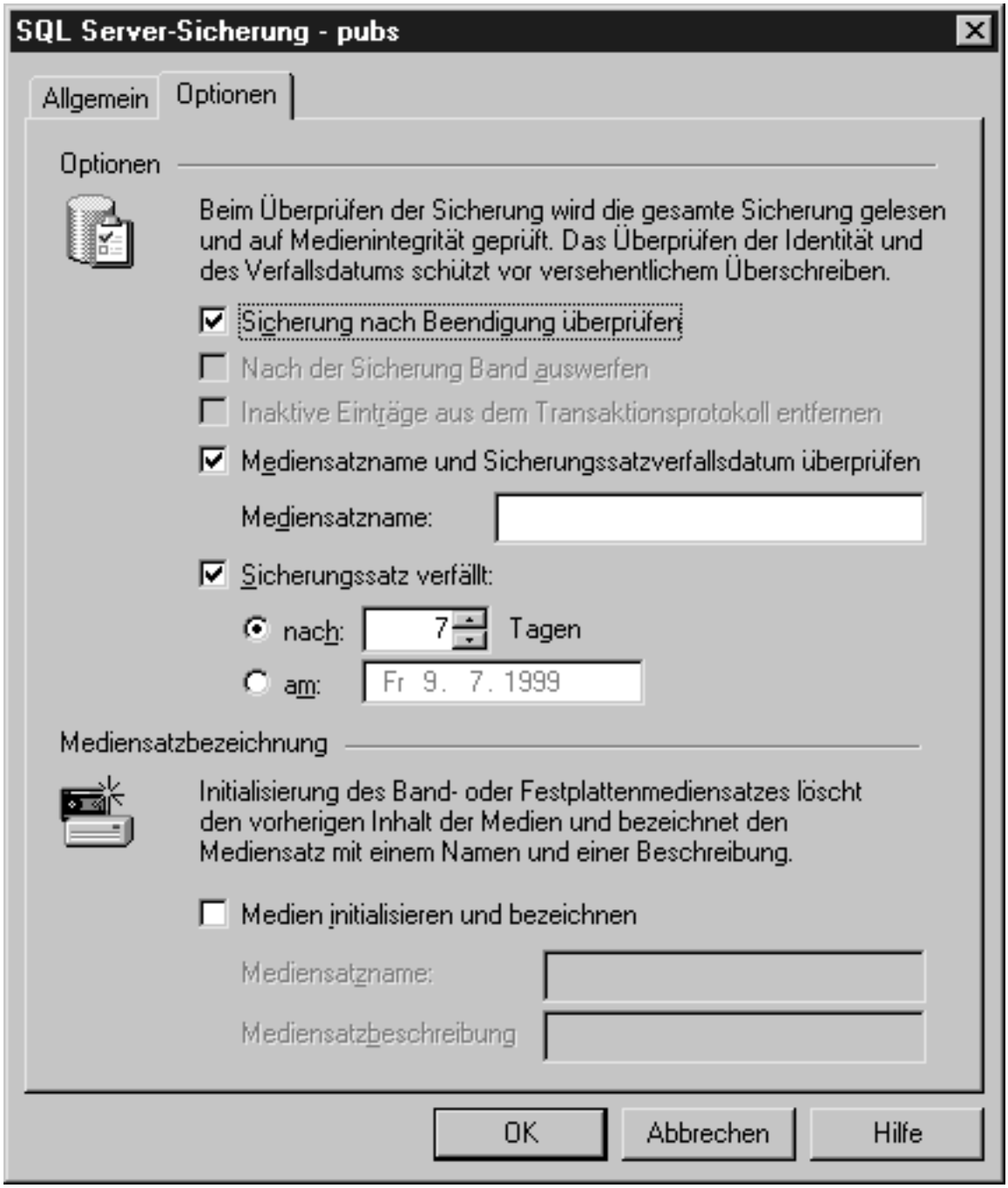
dazu sind unter umständen folgende schritte erforderlich:

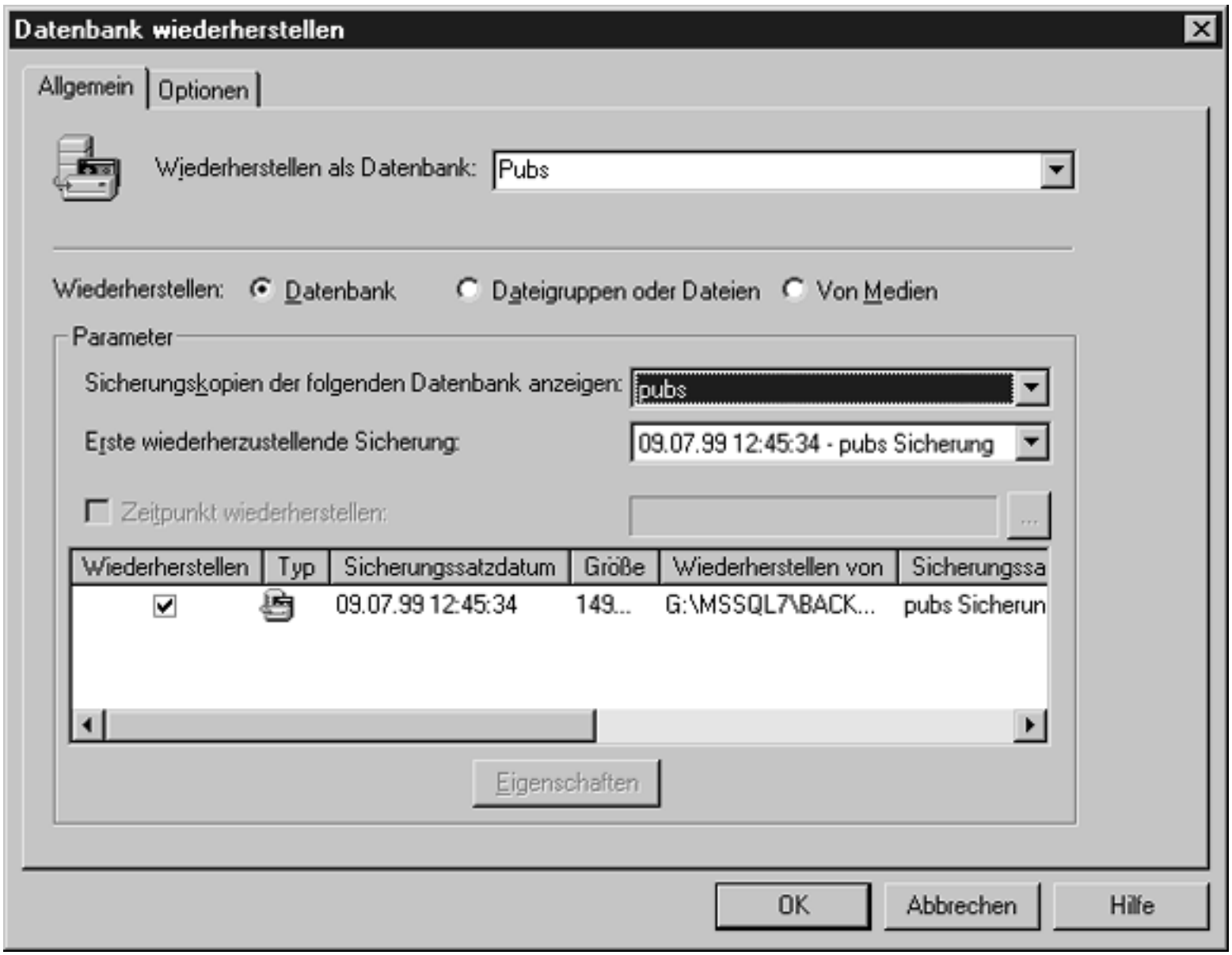
- notwendige sicherungsmedien erstellen
- sicherheitsvorgänge erneut implementieren
- die datenbank msdb wiederherstellen
- die datenbank model wiederherstellen
- die datenbank distribution wiederherstellen
- benutzerdatenbanken wiederherstellen oder anfügen

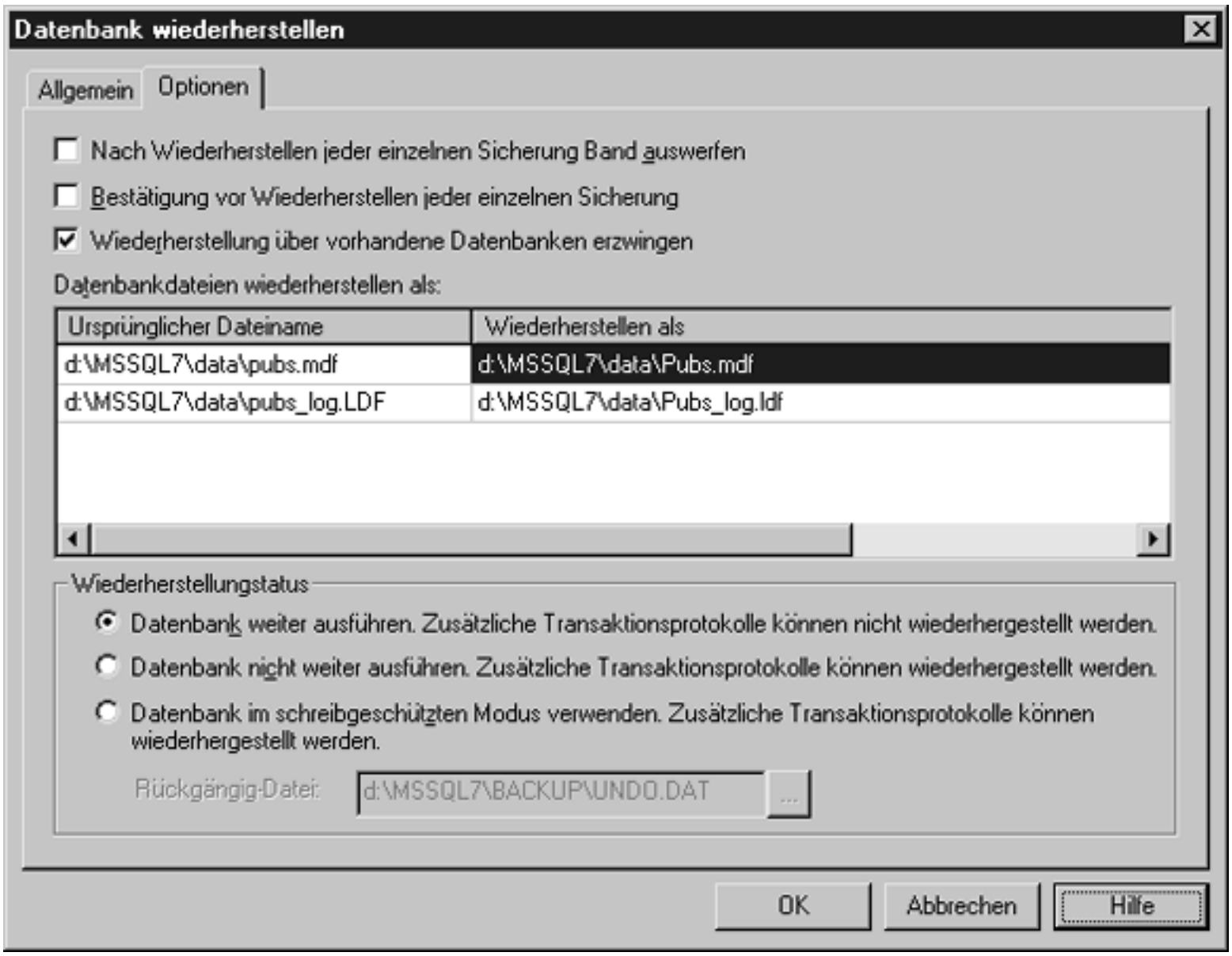
wenn sie alle schritte abgeschlossen haben und die datenbanken wieder zur zufriedenheit laufen, sollten sie zumindest eine erkenntnis gewonnen haben: sicherungen sind vielleicht nie erforderlich, aber wenn,

dann können sie eine unschätzbare hilfe sein, um geld und zeit zu sparen. erstellen sie daher sofort eine sicherung zumindest der master-datenbank.


Sicherungs-Assistent - EINS
Sicherungs-Assistenten beenden
Sie haben die Schritte beendet, die zum Erstellen einer Sicherung erforderlich sind. Sie haben die folgenden Optionen gewählt:
Name: pubs Sicherung
Beschreibung:
Typ: Datenbank
Sicherungsmedien: Device: [PubsBAK]
Sicherungsaktion: Sicherungsmedien überschreiben
Medien initialisieren und bezeichnen: Nein
Mediensatzname überprüfen: Ja
Sicherung überprüfen: Ja
Band auswerfen: Nein
< Zurück
Fertig stellen
Abbrechen


SQL Server-Sicherung - pubs
Allgemein
Optionen
Optionen
Beim Überprüfen der Sicherung wird die gesamte Sicherung gelesen und auf Medienintegrität geprüft. Das Überprüfen der Identität und des Verfallsdatums schützt vor versehentlichem Überschreiben.
Sicherung nach Beendigung überprüfen
Nach der Sicherung Band auswerfen
Inaktive Einträge aus dem Transaktionsprotokoll entfernen
Mediensatzname und Sicherungssatzverfallsdatum überprüfen
Mediensatzname:
Sicherungssatz verfällt:
nach:
7
Tagen
am:
Fr 9. 7. 1999
Mediensatzbezeichnung
Initialisierung des Band- oder Festplattenmediensatzes löscht den vorherigen Inhalt der Medien und bezeichnet den Mediensatz mit einem Namen und einer Beschreibung.
Medien initialisieren und bezeichnen
Mediensatzname:
Mediensatzbeschreibung
OK
Abbrechen
Hilfe


Datenbank wiederherstellen
Allgemein
Optionen
Wiederherstellen als Datenbank:
Pubs
Wiederherstellen:
Datenbank
Dateigruppen oder Dateien
Von Medien
Parameter
Sicherungskopien der folgenden Datenbank anzeigen:
pubs
Erste wiederherzustellende Sicherung:
09.07.99 12:45:34 - pubs Sicherung
Zeitpunkt wiederherstellen:
Wiederherstellen
Typ
Sicherungssatzdatum
Größe
Wiederherstellen von
Sicherungssa
09.07.99 12:45:34
149...
G:\MSSQL7\BACK...
pubs Sicherun
Eigenschaften
OK
Abbrechen
Hilfe


Datenbank wiederherstellen
Allgemein
Optionen
Nach Wiederherstellen jeder einzelnen Sicherung Band auswerfen
Bestätigung vor Wiederherstellen jeder einzelnen Sicherung
Wiederherstellung über vorhandene Datenbanken erzwingen
Datenbankdateien wiederherstellen als:
Ursprünglicher Dateiname
Wiederherstellen als
d:\MSSQL7\data\pubs.mdf
d:\MSSQL7\data\Pubs.mdf
d:\MSSQL7\data\pubs_log.LDF
d:\MSSQL7\data\Pubs_log.ldf
Wiederherstellungstatus
Datenbank weiter ausführen. Zusätzliche Transaktionsprotokolle können nicht wiederhergestellt werden.
Datenbank nicht weiter ausführen. Zusätzliche Transaktionsprotokolle können wiederhergestellt werden.
Datenbank im schreibgeschützten Modus verwenden. Zusätzliche Transaktionsprotokolle können wiederhergestellt werden.
Rückgängig-Datei:
d:\MSSQL7\BACKUP\UNDO.DAT
...
OK
Abbrechen
Hilfe

kapitel 8 tabellen

8.1 unter dach und fach

ein relationales datenbank-managementsystem speichert alle daten in tabellen. im sinne von sql server sind tabellen datenbankobjekte, die datenobjekte auf eine struktur von zeilen und spalten abbilden. in der regel umfaßt eine datenbank mehrere tabellen, die miteinander in beziehung stehen.

dieses kapitel zeigt, wie tabellen aufgebaut sind, wie man tabellen erstellt und ihre eigenschaften festlegt.

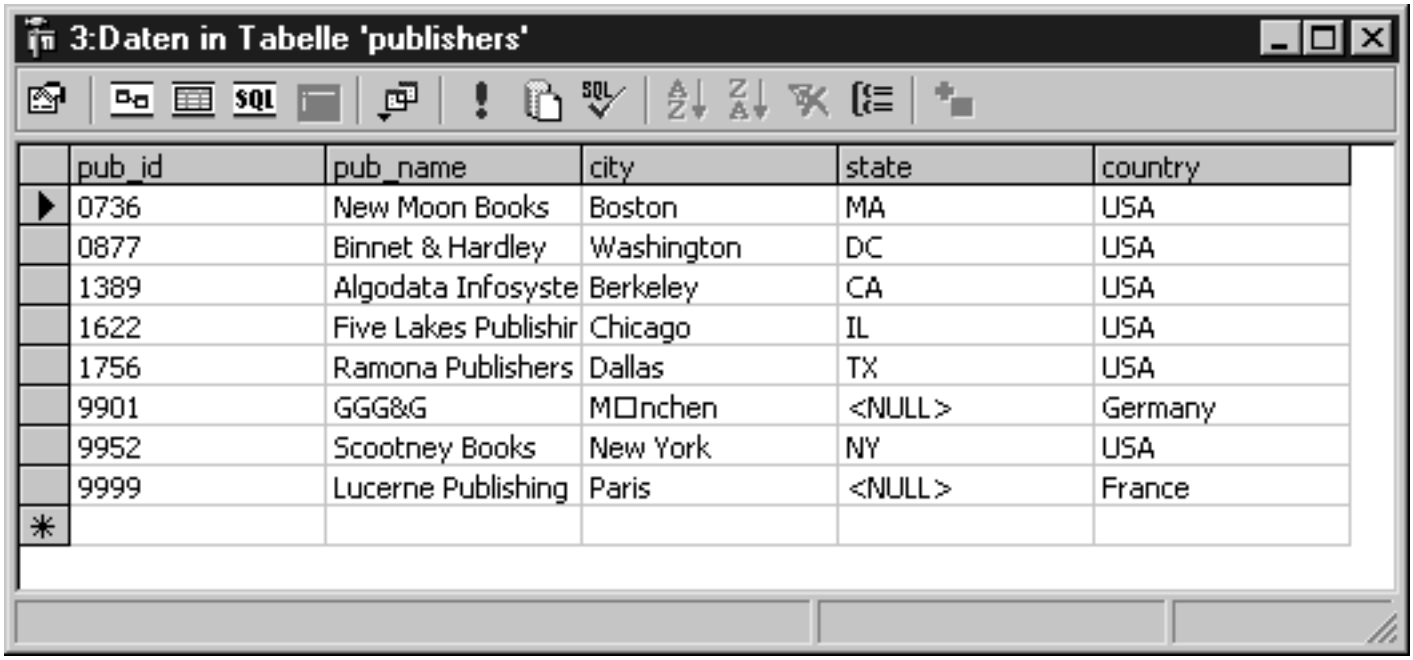
8.2 elemente von tabellen

tabellen speichern die daten ähnlich einem excel-tabellenblatt. allerdings ist jede tabelle in einer sql server-datenbank für eine bestimmte art von datenobjekten vorgesehen, die beim erstellen der tabelle festgelegt wird.

die spalten stellen die attribute der datenobjekte dar. abbildung 8.1 zeigt als beispiel die tabelle publishers der datenbank pubs. es ist zu erkennen, daß innerhalb einer spalte jeweils ähnliche daten stehen. zu den eigenschaften einer spalte gehört ihr *datentyp*, der die art der daten definiert. es ist nicht möglich, in ein und derselben spalte das eine mal numerische daten und ein anderes mal zeichendaten unterzubringen.

in einer zeile sind die attribute eines datenobjekts zusammengefaßt. jede zeile entspricht genau einem datenobjekt. die 1. normalform (siehe kapitel 4) besagt, daß in einer spalte je zeile nur ein wert stehen darf. deshalb ist ein datenobjekt in mehreren separaten zeilen zu speichern, falls für ein attribut mehrere werte möglich sind.

wenn eine spalte als zeichentyp definiert ist, kann sie natürlich mehrere einzelne wörter enthalten. die gesamtheit der wörter bildet die zeichenkette und damit einen einzelnen wert.



pub_id	pub_name	city	state	country
0736	New Moon Books	Boston	MA	USA
0877	Binnet & Hardley	Washington	DC	USA
1389	Algodata Infosyste	Berkeley	CA	USA
1622	Five Lakes Publishir	Chicago	IL	USA
1756	Ramona Publishers	Dallas	TX	USA
9901	GGG&G	M□nchen	<NULL>	Germany
9952	Scootney Books	New York	NY	USA
9999	Lucerne Publishing	Paris	<NULL>	France

abbildung 8.1: die tabelle publishers der datenbank pubs

8.3 tabellen entwerfen

nachdem das konzept einer datenbank festliegt und die schritte der normalisierung abgeschlossen sind, müssen die daten auf die einzelnen tabellen verteilt werden. beim entwurf einer tabelle sind folgende fragen zu klären:

- welche daten soll die tabelle aufnehmen?
- wie sind diese daten auf die spalten der tabelle aufzuteilen?
- welche datentypen sollen die spalten der tabelle haben?
- sind null-werte in den jeweiligen spalten zulässig?
- erhalten bestimmte spalten standardwerte?
- gelten einschränkungen (regeln)?
- ist die tabelle zu indizieren und wenn ja, wie und auf welchen spalten?

es ist natürlich ideal, wenn man von vornherein genau weiß, wie die datenbank und die tabellen aussehen sollen. in der praxis ergeben sich aber noch viele änderungen, die unter anderem aus den wünschen und rückmeldungen der benutzer resultieren. letztendlich lassen sich nicht alle fragen auf einmal beantworten. man kann auch mit einer basistabelle beginnen und später bei bedarf weitere tabellen hinzufügen, standardwerte festlegen und indizes erstellen.

eine grundlegende voraussetzung, um eine tabelle erstellen zu können, ist die kenntnis der datentypen, die man für die einzelnen spalten festlegen kann.

8.4 datentypen

datentypen bezeichnen den möglichen wertebereich, die mit den werten ausführbaren operationen und das speicherformat von datenelementen. sql server speichert alle daten in objekten - den sogenannten tabellen. mit jeder spalte in diesen tabellen sowie mit allen lokalen variablen, ausdrücken und parametern sind datentypen verbunden. die folgende übersicht erläutert die vom system sql server bereitgestellten datentypen. darüber hinaus lassen sich auch benutzerdefinierte datentypen erstellen. während man in den meisten programmiersprachen benutzerdefinierte datentypen aus mehreren einfachen datentypen bilden kann, so daß sich eine zusammengesetzte struktur mit mehreren elementen ergibt, sind benutzerdefinierte datentypen in sql server einfach aliasnamen für die vom system bereitgestellten datentypen. man verwendet benutzerdefinierte datentypen beispielsweise, um für bestimmte spalten eine spezielle länge festzulegen, die null-zulässigkeit zu definieren und/oder einen aussagekräftigen namen für den typ der spalte anzugeben.

8.4.1 systemdatentypen

in den folgenden darstellungen werden durchgängig kommas als tausendertrennzeichen und punkte als dezimalzeichen verwendet, da sql server die jeweiligen zahlenwerte in diesem format erwartet und ausgibt. die online-dokumentation ist in dieser hinsicht nicht konsequent und gibt bei manchen datentypen die deutsche schreibweise (mit punkt als tausendertrennzeichen und komma als dezimalzeichen), bei anderen datentypen die englische an.

ganzzahlen

bit

ganzzahliger datentyp, der die werte 1, 0 oder null annehmen kann. der für bit-werte belegte speicherplatz ergibt sich aus der anzahl der bit-spalten in einer tabelle. für jede angefangene gruppe von 8 bit wird ein byte reserviert.

int

ganzzahlige daten mit dem wertebereich -231 bis 231 - 1 (-2.147.483.648 bis 2.147.483.647). der typ int belegt 4 byte. in sql-92 heißt dieser datentyp integer.

smallint

ganzzahlige daten mit dem wertebereich -215 bis 215 - 1 (-32768 bis 32767). der typ smallint belegt 2 byte.

tinyint

ganzzahlige daten im bereich von 0 bis 255. der typ tinyint belegt 1 byte.

festkommazahlen

decimal[(p[, s])] und numeric[(p[, s])]

festkommazahlen, für die sich die genauigkeit *p* (d.h. die anzahl der signifikanten stellen) und die anzahl der dezimalstellen *s* optional festlegen läßt. der speicherbedarf ist von der genauigkeit abhängig. die maximale genauigkeit erlaubt werte im bereich von 1038 - 1 bis 1038 - 1. wird sql server mit sqlservr und dem schalter /p gestartet, ist s = 38, andernfalls ist s = 28. der wert für p muß zwischen 1 und der maximalen genauigkeit liegen. die anzahl der ziffern nach dem dezimalzeichen (d.h. die dezimalstellen) s darf zwischen 0 und p liegen, wobei der standardwert 0 gilt. tabelle 8.1 gibt den speicherbedarf in abhängigkeit von der genauigkeit an.

genauigkeit p	speicherbedarf in byte
1 bis 9	5
10 bis 19	9
20 bis 28	13
29 bis 38	17

tabelle 8.1: zusammenhang zwischen genauigkeit und speicherbedarf

sql-92-synonyme für decimal sind dec und dec(p, s).

währungsdaten

money

datentyp für währungswerte zwischen -263 und 263 - 1 (interpretiert als zahl im bereich -922.337.203.685.477,5808 bis 922.337.203.685.477,5807). die genauigkeit entspricht einem zehntausendstel der währungseinheit. der speicherbedarf beträgt 8 byte.

smallmoney

datentyp für währungswerte zwischen -214.748,3648 und 214.748,3647. die genauigkeit entspricht einem zehntausendstel der währungseinheit. der speicherbedarf beträgt 4 byte.

gleitkommazahlen

float[(n)]

gleitkommazahl im bereich zwischen -1,79e+308 und 1,79e+308. der optionale parameter *n* spezifiziert die anzahl der bits für die darstellung der mantisse. der wert von *n* muß zwischen 1 und 53 liegen. tabelle 8.2 gibt den zusammenhang zwischen *n*, genauigkeit und speicherbedarf bei diesem datentyp an.

n	genauigkeit (anzahl signifikanter stellen)	speicherbedarf (byte)
1 - 24	7	4
25 - 53	15	8

tabelle 8.2: zusammenhang zwischen n, genauigkeit und speicherbedarf

der sql-typ double precision entspricht float(53).

real

gleitkommazahl im bereich zwischen -3,40e+38 bis 3,40e+38. der speicherbedarf beträgt 4 byte. das synonym für real lautet in sql server float(24).

die online-dokumentation führt die gleitkommatypen in der kategorie »ungefähre numerische werte« auf. diese bezeichnung ist etwas verwirrend. man könnte dazu neigen, auf diese typen gänzlich zu verzichten, weil sie vielleicht nur im statistischen mittel stimmen, eigentlich aber unzuverlässige aussagen liefern.

offensichtlich soll zum ausdruck gebracht werden, daß durch die interne darstellung von dezimalzahlen mit binär kodierter mantisse und exponent unvermeidliche rundungsfehler entstehen. mit den gleitkommatypen von transact-sql können sie aber genauso umgehen wie mit den zahlenwerten, die ihr taschenrechner liefert. lediglich bei währungsdaten, wo es auf bruchteile von pfennigen ankommt, sollte man auf die speziellen datentypen für währungswerte zurückgreifen.

datums-/zeitwerte

datetime

datums- und zeitwerte zwischen dem 1. januar 1753 und dem 31. dezember 9999. die genauigkeit (auflösung) beträgt 3,33 millisekunden, wobei die werte auf vielfache von .000, .003 oder .007 gerundet werden. der speicherbedarf beträgt 8 byte. in den ersten 4 byte ist die anzahl der tage vor oder nach dem basisdatum 1. januar 1900 abgelegt, in den zweiten 4 byte steht die tageszeit als anzahl der millisekunden, die seit mitternacht vergangen sind.

smalldatetime

datums- und zeitwerte zwischen dem 1. januar 1900 und dem 6. juni 2079 mit einer genauigkeit von einer minute. der speicherbedarf beträgt 4 byte. in den ersten 2 byte ist die anzahl der tage nach dem 1. januar 1900 abgelegt, in den zweiten 2 byte steht die anzahl der minuten seit mitternacht.

spezielle numerische werte

cursor

verweis auf einen cursor.

timestamp

in der gesamten datenbank eindeutige zahl. der speicherbedarf beträgt 8 byte. in einer tabelle darf nur eine spalte vom typ timestamp vorhanden sein. derartige spalten eignen sich nicht für schlüssel, da sich beim aktualisieren einer zeile der betreffende wert in der timestamp-spalte ändert.

uniqueidentifier

global eindeutiger bezeichner (guid - globally unique identifier).

zeichenfolgen

char[(n)]

zeichentyp für zeichendaten fester länge mit n zeichen (maximal 8000, keine unicode-zeichen). der speicherbedarf beträgt *n* byte. fehlt die angabe von n, gilt 1 als standardwert (30 in der cast-funktion). in sql-92 entspricht char dem datentyp character.

varchar[(n)]

zeichentyp für zeichendaten variabler länge mit n zeichen (maximal 8000, keine unicode-zeichen). der speicherbedarf entspricht der länge der tatsächlichen daten und kann von 0 bis *n* reichen. fehlt die angabe von n, gilt 1 als standardwert (30 in der cast-funktion). in sql-92 entsprechen die synonyme char varying bzw. character varying dem typ varchar von sql server.

text

typ für zeichendaten mit variabler länge von maximal 231 - 1 (d.h. 2.147.483.647) zeichen (keine unicode-zeichen).

unicodezeichenfolgen

nchar[(n)]

unicodezeichendaten fester länge mit *n* zeichen, wobei *n* zwischen 1 und 4000 liegen muß. fehlt die angabe von *n*, gilt 1 als standardwert (30 in der cast-funktion). der speicherbedarf in byte beträgt das doppelte von *n*. die sql-92-synonyme für nchar lauten national char und national character.

nvarchar[(n)]

unicodezeichendaten variabler länge mit *n* zeichen, wobei *n* zwischen 1 und 4000 liegen muß. der speicherbedarf in byte beträgt das doppelte der eingegebenen anzahl von zeichen. die zeichenanzahl *n* muß ein wert zwischen 1 und 4000 sein. fehlt die angabe von *n*, gilt 1 als standardwert (30 in der cast-funktion). die speichergröße in byte beträgt das doppelte der anzahl der eingegebenen zeichen. die eingegebenen daten können 0 zeichen lang sein. die sql-92-synonyme für nvarchar sind national char varying und national character varying.

ntext

unicodedaten variabler länge mit maximal 230-1 (1.073.741.823) zeichen. der speicherbedarf in byte beträgt das zweifache der anzahl der eingegebenen zeichen. das sql-92-synonym für ntext ist national text.

binärzeichenfolgen

binary[(n)]

binärdaten mit fester länge von *n* byte, wobei *n* zwischen 1 und 8000 liegen muß. fehlt die angabe von *n*,

gilt 1 als standardwert (30 in der cast-funktion). der speicherbedarf beträgt n+4 byte.

varbinary[(n)]

binärdaten variabler länge von *n* byte, wobei n zwischen 1 und 8000 liegen muß. fehlt die angabe von n, gilt 1 als standardwert (30 in der cast-funktion). der speicherbedarf errechnet sich aus der tatsächlichen datenlänge (im bereich 0 bis n) plus 4 byte. das sql-92-synonym für varbinary ist binary varying.

image

binärdaten variabler länge von 0 bis 231-1 (2.147.483.647) byte.

8.4.2 null-werte

unter einem null-wert versteht man: gar kein wert. eine 0 ist *kein* null-wert, da es sich um einen konkreten zahlenwert handelt. das gleiche gilt bei einem zeichentyp für ein leerzeichen oder sogar eine leere zeichenfolge. null-werte bezeichnen unbekannte werte.

für eine spalte kann man festlegen, ob sie null-werte zuläßt. wenn für die spalte not null spezifiziert ist, muß man zum beispiel beim einfügen von daten in die tabelle für die betreffende spalte einen wert angeben, da sql server sonst einen fehler liefert.

8.4.3 true, false und unknown

logische operatoren und vergleichsoperatoren liefern normalerweise die ergebniswerte true (wahr) oder false (falsch) zurück. enthalten die daten allerdings null-werte, gibt es einen dritten möglichen wert: unknown (unbekannt). diese dreiwertige logik kann zu unerwarteten oder sogar fehlerhaften ergebnissen führen. mit is null oder is not null in der where-klausel können sie testen, ob null-werte vorhanden sind.

wenn die systemeinstellung ansi_nulls deaktiviert ist, wird ein null-wert in vergleichen wie false behandelt.

kapitel 10 geht auf null-werte und das ergebnis unknown in verbindung mit den logischen operatoren ein.

8.4.4 benutzerdefinierte datentypen

mit einem benutzerdefinierten datentyp lassen sich zugeschnittene, wiederverwendbare datentypen auf der basis eines vorhandenen sql-server-datentyps erzeugen. man kann damit die konsistenz von datentypen gewährleisten. will man zum beispiel für alle tabellen einer datenbank festlegen, daß die spalte kundennummer ganzzahlig ist und keine null-werte enthalten darf, kann man die tabellen mit einem benutzerdefinierten typ für die betreffende spalte erzeugen.

gültigkeitsbereiche von benutzerdefinierten datentypen

erstellt man einen benutzerdefinierten datentyp für die datenbank model, ist der datentyp für alle danach neu erstellten benutzerdatenbanken verfügbar. wird der datentyp für eine benutzerdatenbank definiert, ist er nur in dieser datenbank vorhanden.

der name des benutzerdefinierten datentyps muß in der datenbank eindeutig sein. allerdings dürfen unterschiedlich benannte datentypen die gleiche definition aufweisen. die namen müssen den regeln für bezeichner entsprechen.

erstellen benutzerdefinierter datentypen

um einen benutzerdefinierten datentyp zu erzeugen, sind folgende drei angaben erforderlich:

- name
- zugrundeliegender systemdatentyp
- null-zulässigkeit

die folgenden beispiele zeigen, wie man den benutzerdefinierten datentyp kdnr_typ erstellt. der typ soll ganzzahlig sein und keine null-werte aufweisen dürfen.

enterprise manager

mit dem enterprise manager führen sie folgende schritte aus:

1. erweitern sie die konsolenstruktur bis zu der datenbank, in der sie den benutzerdefinierten datentyp erzeugen möchten.
2. klicken sie mit der rechten maustaste auf die datenbank, wählen sie im kontextmenü **neu** und dann den untereintrag **benutzerdefinierter datentyp**. alternativ können sie in der konsolenstruktur den eintrag für die datenbank erweitern, mit rechts auf den eintrag **benutzerdefinierte datentypen** klicken und im kontextmenü **neuer benutzerdefinierter datentyp** wählen. es erscheint das dialogfeld **benutzerdefinierter datentyp - eigenschaften -** *servername* (siehe abbildung 8.2).

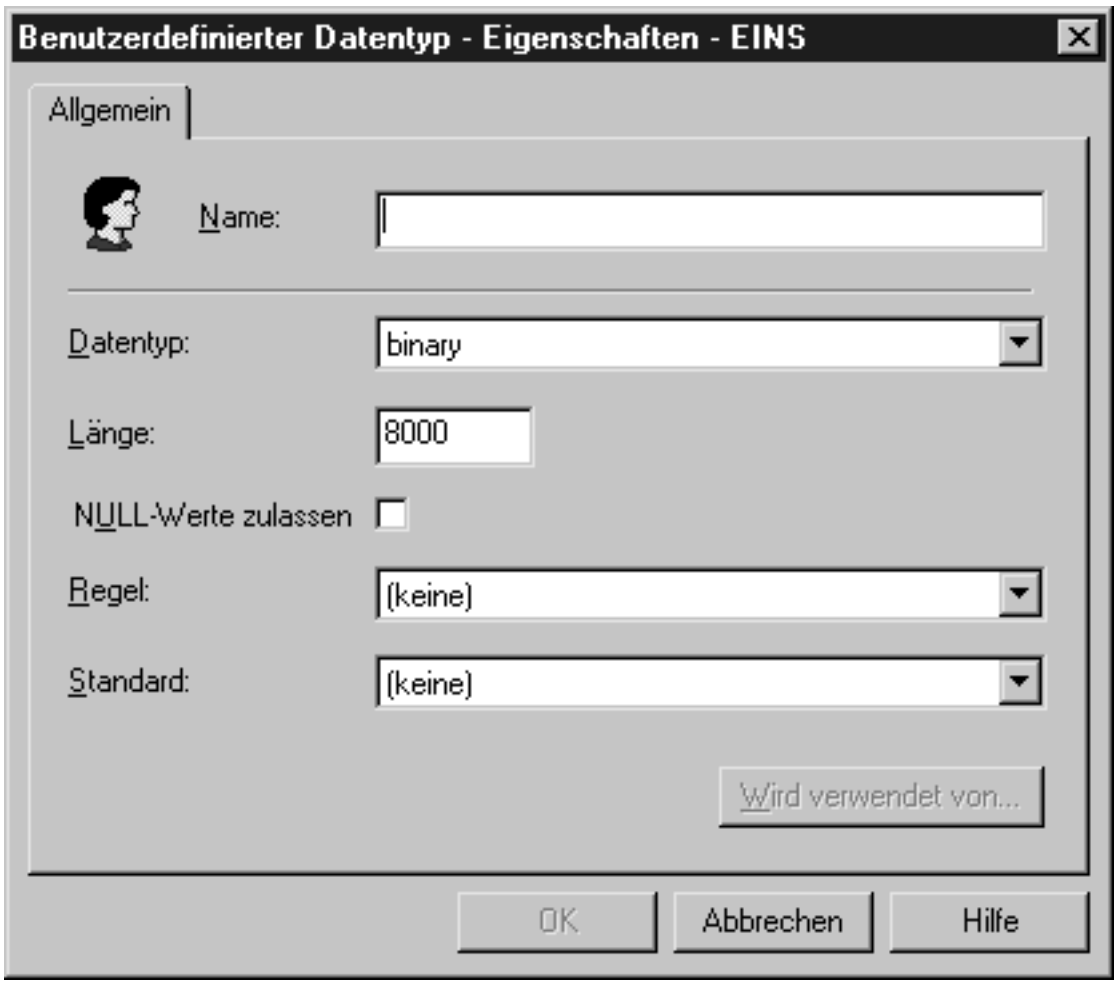


abbildung 8.2: mit diesem dialogfeld erstellen sie einen benutzerdefinierten datentyp

3. tragen sie in das feld **name** die bezeichnung des neuen datentyps ein, im beispiel *kdnr_typ*.
4. wählen sie aus dem drop-down-listenfeld *datentyp* den sql-server-basistyp aus (*int*).
5. da laut aufgabenstellung in der spalte kdnr keine null-werte vorkommen dürfen, lassen sie das kontrollkästchen **null-werte zulassen** ausgeschaltet.
6. falls für die datenbank bereits regeln oder standards definiert sind, können sie diese bei bedarf aus den betreffenden drop-down-listenfeldern auswählen und mit dem benutzerdefinierten datentyp verbinden.

transact-sql

in transact-sql erstellen sie einen benutzerdefinierten datentyp mit der gespeicherten prozedur sp_addtype gemäß folgender syntax:

```
sp_addtype [@typename =] typname,
[@phystype =] systemdatentyp
[, [@nulltype =] 'null-zulässigkeit']
```

parameter, die leerzeichen oder interpunktionszeichen enthalten, sind in einfache anführungszeichen zu

schreiben.

typname ist der bezeichner des benutzerdefinierten datentyps.

systemdatentyp gibt den sql-server-datentyp an, auf dem der benutzerdefinierte datentyp basiert. tabelle 8.3 listet die möglichen basistypen auf, die als systemdatentyp zulässig sind.

die null-zulässigkeit gibt die standardeinstellung für den neuen datentyp an. die möglichen werte null, not null bzw. nonull sind in einfache anführungszeichen zu schreiben. wenn sie eine tabelle erstellen und die null-zulässigkeit explizit angeben, hat diese vorrang gegenüber der standardfestlegung im benutzerdefinierten datentyp. es empfiehlt sich, die null-zulässigkeit immer bei der definition des datentyps anzugeben, um zweideutigkeiten zu vermeiden, die von der systemeinstellung von sql server abhängen.

'binary(n)'	image	smalldatetime
bit	int	smallint
'char(n)'	'nchar(n)'	text
datetime	ntext	tinyint
decimal	numeric	uniqueidentifier
'decimal[(p[, s])]'	'numeric[(p[, s])]'	'varbinary(n)'
float	'nvarchar(n)'	'varchar(n)'
'float(n)'	real	

tabelle 8.3: basistypen, die als systemdatentypen angegeben werden können

n bezeichnet die länge des datentyps

p steht für die maximale anzahl von dezimalstellen vor und nach dem dezimalzeichen.

s gibt die maximale zahl der dezimalstellen nach dem dezimalzeichen an.

die einfachen anführungszeichen in tabelle 8.3 sind erforderlich, wenn der bezeichner aus mehreren teilen besteht.

für die datenbank lotto ließe sich zum beispiel ein benutzerdefinierter datentyp ziehungstyp erstellen, der keine null-werte erlaubt und ganzzahlig ist:

```
sp_addtype ziehungstyp, 'int', 'not null'
```

konnte der typ erfolgreich erzeugt werden, erscheint die rückmeldung:

```
(1 row affected)
typ hinzugefügt.
```

statt int können sie auch den ansi-92-typ integer angeben.

8.4.5 rangfolge der datentypen

wenn man zwei ausdrücke mit arithmetischen, bitweisen oder zeichenfolgenoperatoren verknüpft, ist der resultierende datentyp vom operator abhängig. sind die datentypen unterschiedlich, wandelt sql server den datentyp mit der niedrigeren rangfolge in den datentyp mit der höheren um, sofern die konvertierung implizit unterstützt wird. andernfalls liefert das system einen fehler.

ausdrücke lassen sich mit einem operator verknüpfen, wenn

- der operator die datentypen beider ausdrücke unterstützt und
- beide ausdrücke den gleichen datentyp besitzen,
- der in der rangfolge niedrigere datentyp in den höheren datentyp konvertiert werden kann oder
- mit hilfe der cast-funktion (siehe kapitel 11) eine explizite umwandlung vom datentyp der niedrigeren rangfolge in den datentyp der höheren rangfolge möglich ist, wobei es sich beim höheren datentyp auch um einen dazwischenliegenden typ handeln kann, der sich implizit in den höheren zieltyp konvertieren läßt.

bei einer verknüpfung von ausdrücken mit vergleichs- oder logischen operatoren hat das ergebnis immer einen booleschen datentyp und nimmt einen der drei werte true, false oder unknown an.

haben die beiden operanden in einem vergleichsausdruck unterschiedliche datentypen, wird zuerst der eine datentyp entsprechend der rangfolge in den anderen umgewandelt und dann erst der vergleich durchgeführt.

die folgende übersicht listet die datentypen von der höchsten zur niedrigsten rangfolge auf:

- datetime
- smalldatetime
- float
- decimal
- money
- smallmoney
- int
- smallint
- tinyint
- bit
- ntext
- text
- image
- timestamp

- nvarchar
- nchar
- varchar
- char
- varbinary
- binary
- uniqueidentifier

8.4.6 datentypen konvertieren

oftmals haben ausdrücke, die sie miteinander vergleichen oder verknüpfen müssen, unterschiedliche datentypen. in diesem fall sind die datentypen umzuwandeln. sql server unterscheidet zwischen impliziter - d.h. automatischer - und expliziter konvertierung. in der online-dokumentation finden sie unter dem stichwort *cast* und wahl des themas *cast und convert* eine tabelle, in der alle möglichen impliziten konvertierungen dargestellt sind.

die explizite umwandlung zwischen datentypen nehmen sie mit den funktionen cast oder convert vor, auf die kapitel 11 eingeht.

8.4.7 berechnete spalten

die definition einer tabelle kann in sql server 7.0 auch berechnete spalten enthalten. derartige spalten (man spricht auch von virtuellen spalten) speichern nicht die eigentlichen - berechneten - daten, sondern lediglich den ausdruck, mit dem die berechnung der datenwerte erfolgt. enthält eine tabelle zum beispiel den preis eines artikels in der spalte preis, kann man die zugehörige umsatzsteuer als berechnete spalte erstellen:

```
ust as preis * .16
```

allerdings sind einige bedingungen zu beachten. berechnete spalten

- lassen sich nicht als schlüsselspalte in einem index oder als teil einer einschränkungsdefinition mit primary key, unique, foreign key oder default verwenden.
- können nicht das ziel einer insert- oder update-anweisung sein.

8.5 tabellen erstellen

die beispiele in diesem abschnitt beziehen sich auf die datenbank lotto, die lediglich über eine tabelle - ldaten - verfügt. in der tabelle sind folgende spalten vorgesehen:

- ziehung: gibt die fortlaufende ziehungsnummer von 1 aufwärts an. null-werte sind nicht erlaubt.
- datum: liefert das datum der ziehung, datentyp datetime.
- z1 bis z6: die sechs gezogenen zahlen, datentyp tinyint.

- zz: die zusatzzahl, datentyp tinyint.

die daten für die datenbank lotto finden sie auf der begleit-cd zum buch.

8.5.1 tabellen mit enterprise manager erstellen

einer vorhandenen datenbank fügen sie mit dem enterprise manager eine neue tabelle in folgenden schritten hinzu:

1. erweitern sie die konsolenstruktur bis zu der datenbank, für die sie eine neue tabelle einfügen möchten.
2. klicken sie mit der rechten maustaste auf den eintrag **tabellen**, und wählen sie aus dem kontextmenü den befehl **neue tabelle**.
3. daraufhin erscheint das eingabefeld **namen wählen** (siehe abbildung 8.3). geben sie hier den namen der tabelle ein (für das beispiel *ldaten*).

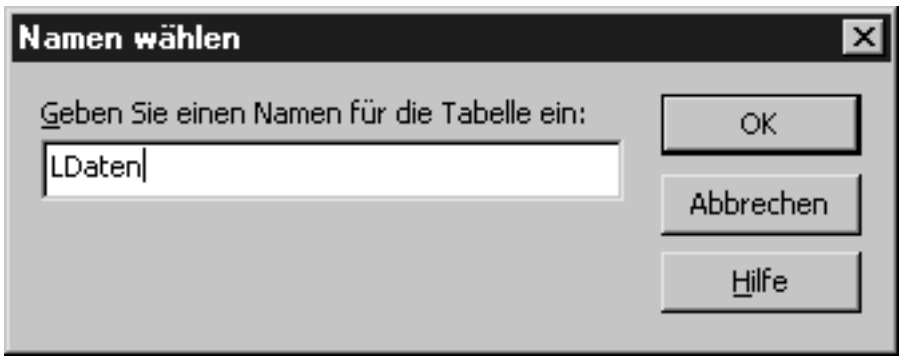


abbildung 8.3: dialogfeld namen festlegen

4. sql server zeigt nun eine leere tabelle an (siehe abbildung 8.4).

bild

abbildung 8.4: eine neue tabelle im enterprise manager

5. tragen sie für jede zeile spaltenname, datentyp und null-zulässigkeit entsprechend abbildung 8.5 ein. wenn sie sich in der spalte **datentyp** befinden, erscheint in der aktuellen zeile ein drop-down-pfeil. klicken sie auf diesen pfeil, um die liste der verfügbaren datentypen zu öffnen. wählen sie den gewünschten datentyp aus.

in der liste der verfügbaren datentypen sind auch die benutzerdefinierten datentypen aufgeführt. wenn sie einen benutzerdefinierten datentyp für eine spalte festlegen möchten, muß dieser typ bereits vor dem erstellen der tabelle existieren.

bild

abbildung 8.5: die definition der tabelle ldaten

6. klicken sie im dialogfeld **neue tabelle** auf die schaltfläche **speichern** (die erste schaltfläche von links in der symbolleiste), und schließen sie die tabelle. sql server nimmt nun die tabelle ldaten in die datenbank lotto auf.

8.5.2 create table

eine neue tabelle erstellen sie mit der transact-sql-anweisung create table. dieser anweisung begegnen sie auch, wenn sie mit dem dts-assistenten daten importieren, um beispielsweise die struktur der zu importierenden tabelle festzulegen (siehe dazu kapitel 9 zum importieren/exportieren).

die (vereinfachte) syntax der anweisung create table sieht folgendermaßen aus:

syntax

```
create table [datenbankname.[besitzer].| besitzer.]
tabellenname
(
  { spaltenname datentyp
  | spaltenname as berechneterspaltenausdruck
} [, ...n]
)
```

erweiterungen der create table-syntax lernen sie bei der behandlung von einschränkungen und standardwerten in kapitel 16 kennen.

argumente

der *datenbankname* bezeichnet die datenbank, in der sie die tabelle erstellen wollen. standardwert ist die aktuelle datenbank. als *besitzer* gilt in der voreinstellung der für die verbindung angemeldete benutzer. der *tabellenname* gibt den namen der neuen tabelle an und muß den regeln für bezeichner entsprechen.

im anschließenden syntaxabschnitt definieren sie die spalten. *spaltenname* steht für den namen einer spalte und muß den regeln für bezeichner entsprechen. der *datentyp* legt den typ der jeweiligen spalte fest. ein *berechneterspaltenausdruck* stellt eine virtuelle spalte dar, die physikalisch nicht in der tabelle enthalten ist, sondern durch einen ausdruck berechnet wird. der ausdruck kann den namen einer nicht berechneten (regulären) spalte, eine konstante, eine funktion, eine variable oder eine kombination aus diesen elementen enthalten. berechnete spalten können sie überall dort verwenden, wo auch reguläre ausdrücke zulässig sind, beispielsweise in select-listen, where-klauseln oder order by-klauseln. beachten sie aber die bedingungen, die der entsprechende abschnitt zu berechneten spalten weiter oben in diesem kapitel genannt hat.

beispiel:

die folgende anweisung erzeugt die tabelle ldaten für die datenbank lotto. das skript instlott.sql ist auf der begleit-cd enthalten.

```
create table ldaten (
ziehung int not null,
```

```
datum   datetime,
z1      tinyint,
z2      tinyint,
z3      tinyint,
z4      tinyint,
z5      tinyint,
z6      tinyint,
zz      tinyint )
```

in der spalte ziehung werden die fortlaufenden ziehungsnummern von 1 beginnend gespeichert. die spalte darf keine null-werte enthalten. die spalte datum ist vom typ datetime und nimmt das zugehörige ziehungsdatum auf. in den spalten z1 bis z6 stehen die sechs gezogenen zahlen, in der spalte zz die zusatzzahl.

8.6 tabellen ändern

wenn sich die definition aus irgendwelchen gründen ändern sollte, müssen sie die tabelle und die darin enthaltenen daten nicht löschen und in eine neue tabelle einfügen, sondern können die tabelle zum beispiel per enterprise manager oder mit der transact-sql-anweisung alter table ändern. diese unkomplizierte verfahrensweise ist erst in der version 7.0 von sql server möglich.

8.6.1 tabellen mit enterprise manager bearbeiten

die folgenden schritte zeigen anhand der tabelle ldaten der datenbank lotto, wie sie die definition einer tabelle mit dem enterprise manager ändern:

1. erweitern sie die konsolenstruktur bis zu den einträgen der gewünschten datenbank. klicken sie auf den eintrag **tabellen**, um im detailbereich des enterprise managers die vorhandenen tabellen anzuzeigen.
2. klicken sie mit der rechten maustaste auf die zu ändernde tabelle, im beispiel ldaten. wählen sie aus dem kontextmenü den befehl **tabelle bearbeiten**. daraufhin wird die tabelle zur bearbeitung geöffnet (siehe abbildung 8.6).
3. jetzt können sie die definition der tabelle ändern. im beispiel wurde die null-zulässigkeit für die spalten z1 bis z6 deaktiviert. dadurch lassen sich zum beispiel fehlerhafte eingaben erkennen, wenn sie an der entsprechenden position keine zahl bereitstellen. und ein standardwert ist in diesen spalten nicht sinnvoll. allerdings könnten sie einen standardwert für die spalte datum festlegen. hier bietet sich das aktuelle datum an, wenn sie jeden samstag die ziehungsergebnisse aktualisieren. tragen sie dazu in die spalte standardwert die funktion getdate() ein. abbildung 8.6 zeigt die bereits bearbeitete tabelle.

bild

abbildung 8.6: tabelle ldaten zur bearbeitung geöffnet

4. um die definition der tabelle zu übernehmen, klicken sie in der symbolleiste auf die schaltfläche

speichern (erste von links).

8.6.2 alter table

die vereinfachte syntax der transact-sql-anweisung alter table sieht folgendermaßen aus:

```
alter table tabellenname
{ [alter column spaltenname
  { neuerdatentyp [ (precision[, scale] )]
  [null | not null ]
  | {add | drop} rowguidcol
  } ]
| add
  { [ spaltenname datentyp ]
  | spaltenname as berechneterspaltenausdruck
  } [,...n]
}
```

an die stelle des schlüsselwortes create ist das schlüsselwort alter getreten. der *tabellenname* bezeichnet den namen der zu ändernden tabelle. die klausel alter column erlaubt es, die eigenschaften der in *spaltenname* angegebenen spalte zu ändern. allerdings gelten eine reihe von einschränkungen für die neue definition der spalte. beispielsweise muß sich der ursprüngliche datentyp in den neuen datentyp implizit konvertieren lassen, es darf sich nicht um eine berechnete spalte handeln, die spalte darf nicht in einer anderen berechneten spalte vorkommen, und die spalte darf nicht in einer einschränkung für primär- oder fremdschlüssel verwendet werden. weitere details dazu entnehmen sie bitte der online-dokumentation.

mit add bzw. drop rowguidcol können sie die spalte als global eindeutigen bezeichner festlegen bzw. diese festlegung aufheben.

interessant dürfte vor allem die klausel add für das hinzufügen einer neuen spalte sein. die argumente *spaltenname datentyp* und *berechneterspaltenausdruck* haben die gleiche bedeutung wie bei der anweisung create table.

8.7 tabellen löschen

von der verfahrensweise her gesehen ist es ein leichtes, eine tabelle zu löschen. allerdings sollten sie sich vorher genau darüber informieren, welche abhängigkeiten zwischen der zu löschenden tabelle und anderen datenbankobjekten bestehen. unter umständen können sie die gesamte datenbank durch löschen von nur einer tabelle unbrauchbar machen.

8.7.1 tabellen per enterprise manager löschen

erweitern sie die konsolenstruktur, bis die tabellen der datenbank, aus der sie eine tabelle löschen wollen, im detailbereich des enterprise managers zu sehen sind. im beispiel soll die tabelle titles der datenbank pubs gelöscht werden.

klicken sie dann mit der rechten maustaste auf die zu löschende tabelle (titles), und wählen sie aus dem kontextmenü den befehl **löschen**. daraufhin erscheint das dialogfeld **objekte löschen** (siehe abbildung 8.7).

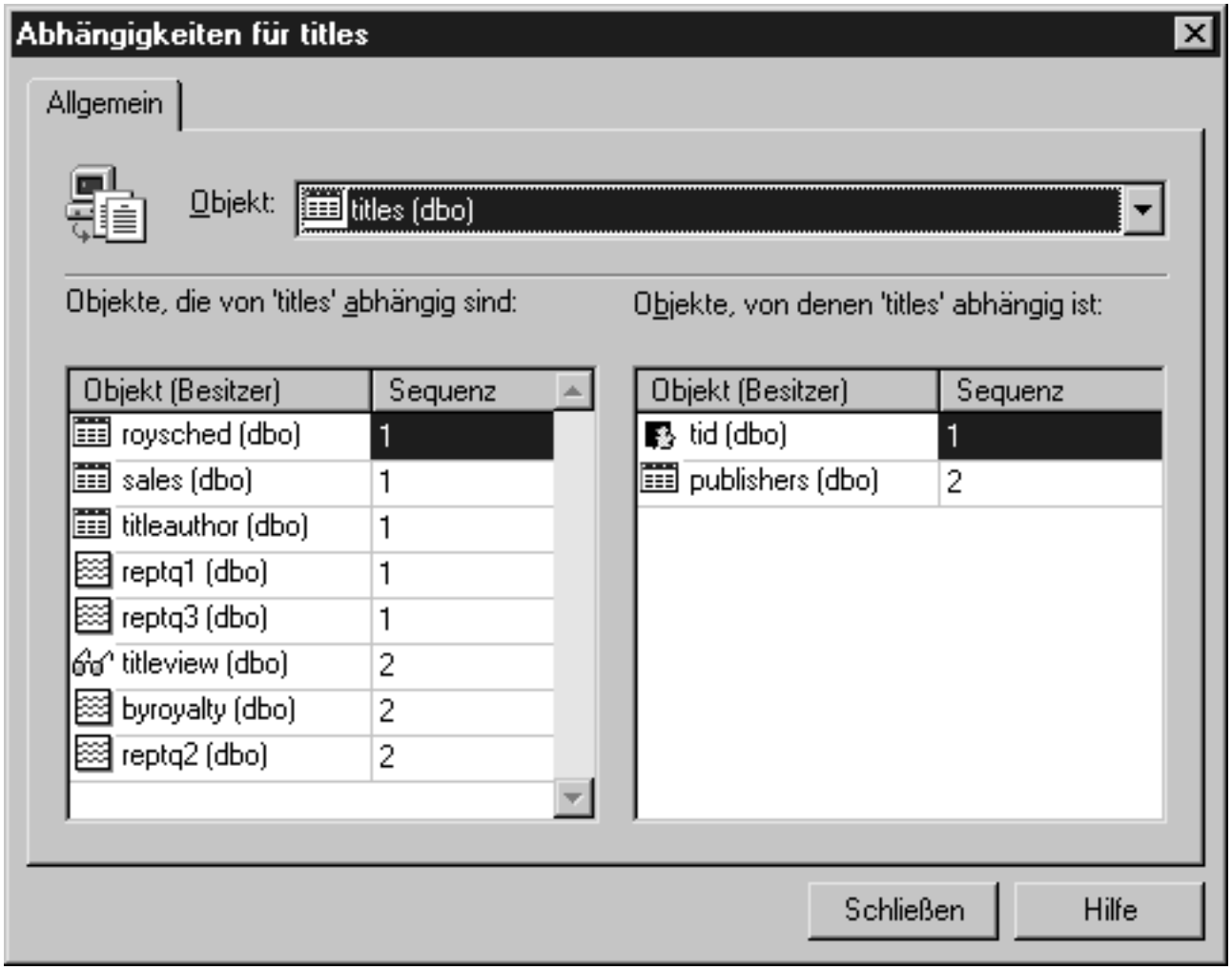


abbildung 8.7: das dialogfeld abhängigkeiten für die tabelle titles

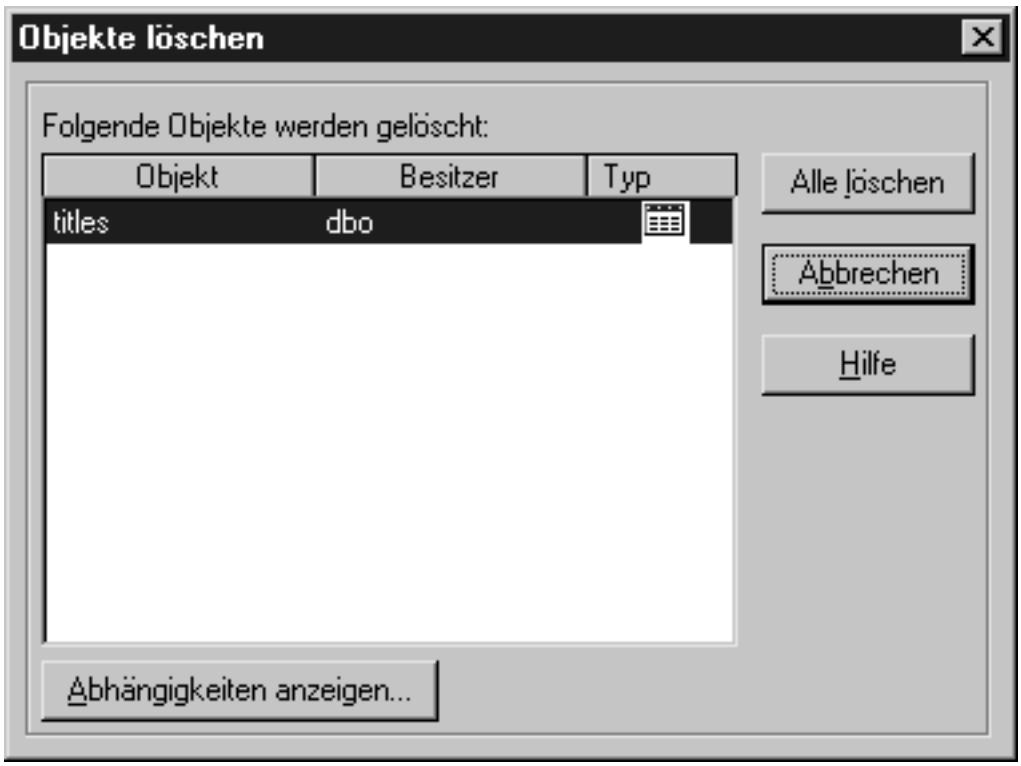


abbildung 8.8: das dialogfeld objekte löschen mit der zu löschenden tabelle

klicken sie auf die schaltfläche **abhängigkeiten anzeigen** am unteren rand des dialogfelds, um sich über die beziehungen zu anderen datenbankobjekten zu informieren. abbildung 8.8 zeigt das dialogfeld abhängigkeiten für die tabelle titles.

im dialogfeld **abhängigkeiten** erscheinen nicht nur andere tabellen, sondern zum beispiel auch benutzerdefinierte datentypen oder sichten.

nachdem sie sich davon überzeugt haben, daß sie die tabelle ohne auswirkungen auf andere teile der datenbank löschen können (was im hier dargestellten beispiel nicht der fall ist), schließen sie das dialogfeld und klicken im dialogfeld **objekte löschen** auf **alle löschen**. entweder verschwindet nun die tabelle problemlos, was ja auch ihr ziel war, oder es erscheint eine fehlermeldung, wie sie zum beispiel abbildung 8.9 zeigt.

Bild

abbildung 8.9: fehlermeldung, wenn tabelle nicht gelöscht werden konnte

in diesem fall müssen sie zuerst die fremdschlüsseltabelle oder die foreign key-einschränkung löschen. am einfachsten läßt sich das in einem datenbankdiagramm erledigen, da sie hier die »hinderlichen« beziehungen auf einen blick erfassen und auch unkompliziert löschen können.

8.7.2 drop table

die transact-sql-anweisung zum löschen einer tabelle sieht unscheinbar aus:

```
drop table tabellenname
```

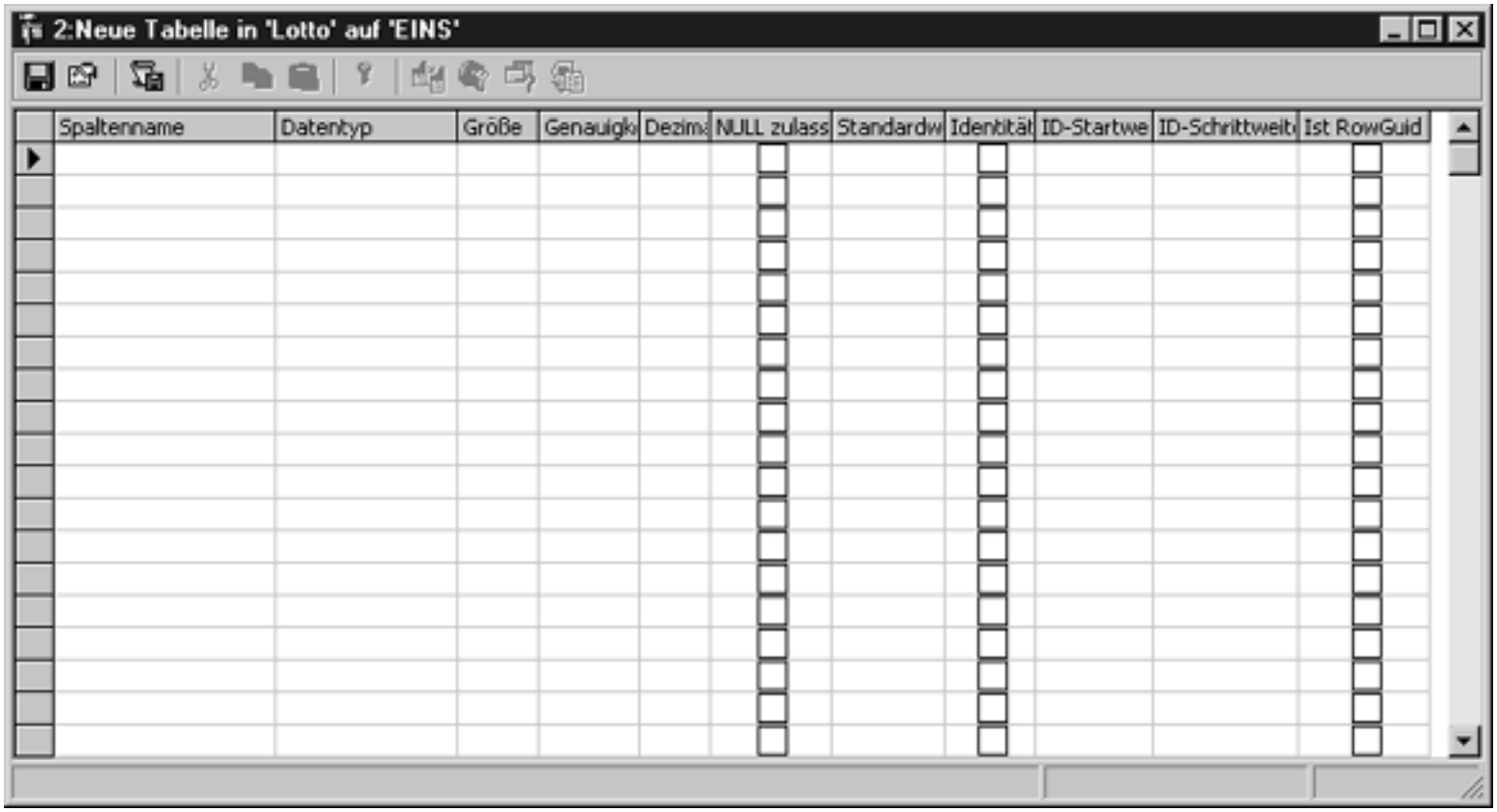
wie im letzten abschnitt zum löschen von tabellen mit dem enterprise manager erwähnt, müssen sie zuerst die entsprechenden fremdschlüsseltabellen oder die foreign key-einschränkungen löschen, bevor sie die tabelle selbst löschen können.

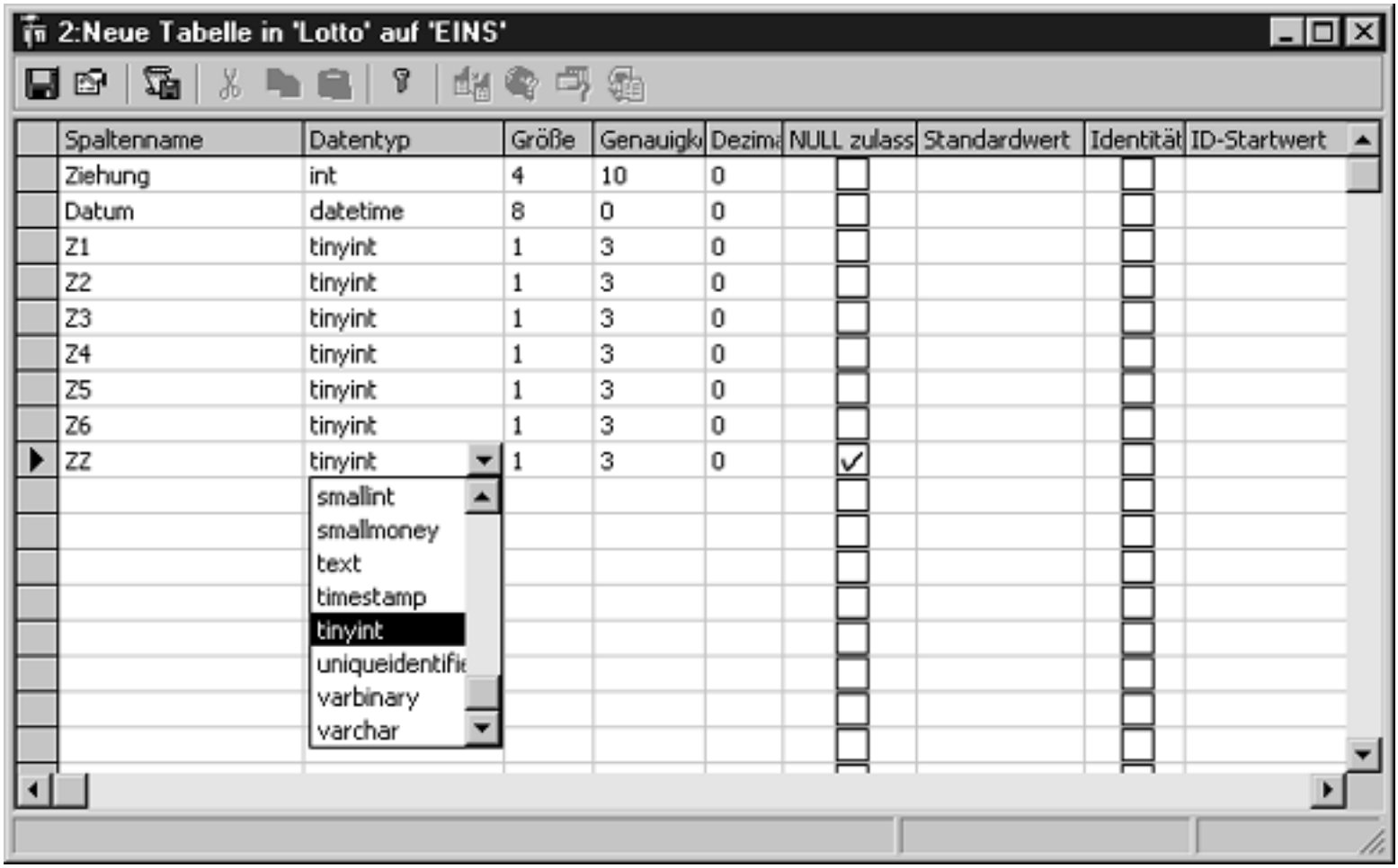
wenn sie zum beispiel die tabelle titles aus der datenbank pubs mit der anweisung

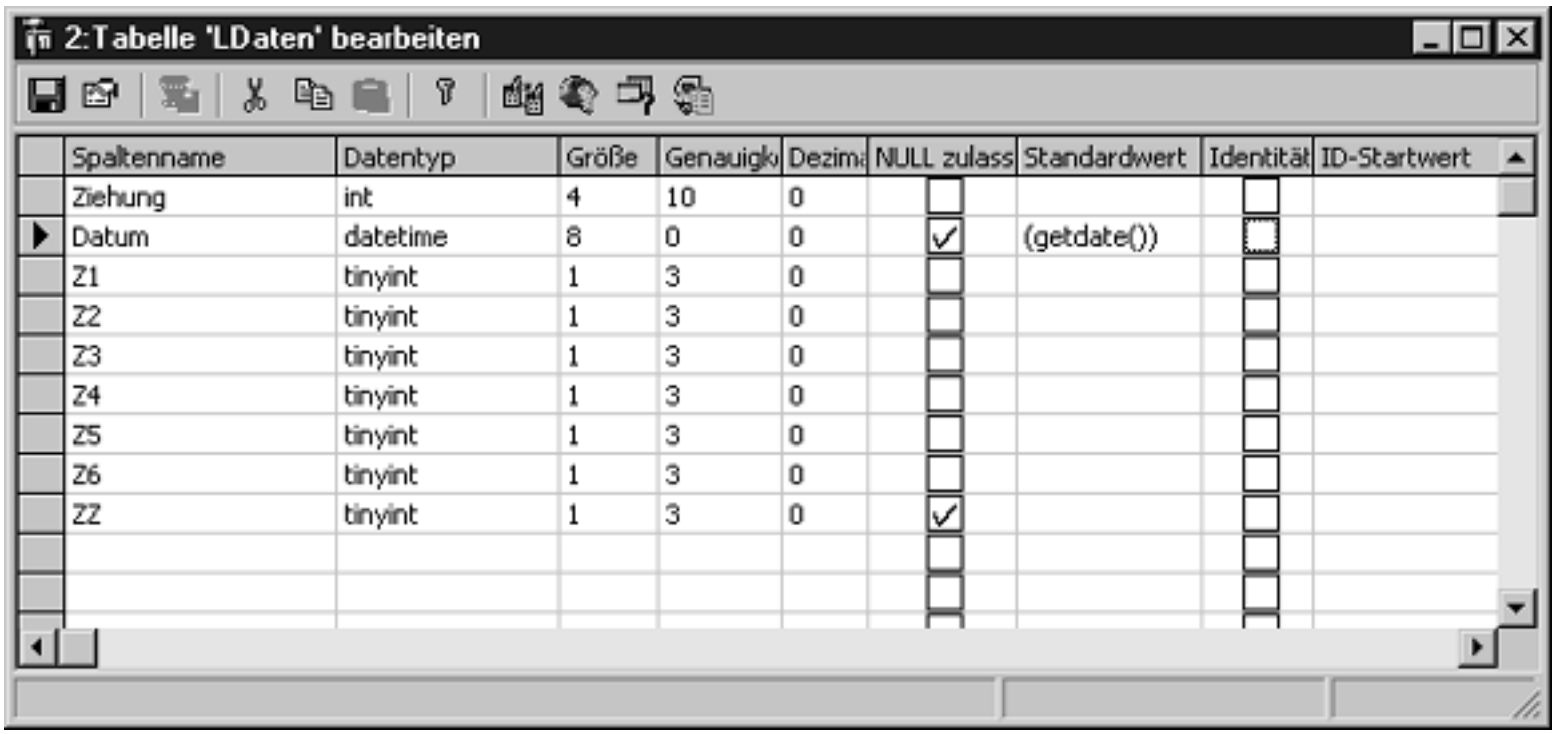
```
use pubs
drop table titles
```

zu löschen versuchen (ohne vorher die beziehungen zu entfernen), erhalten sie folgende fehlermeldung:

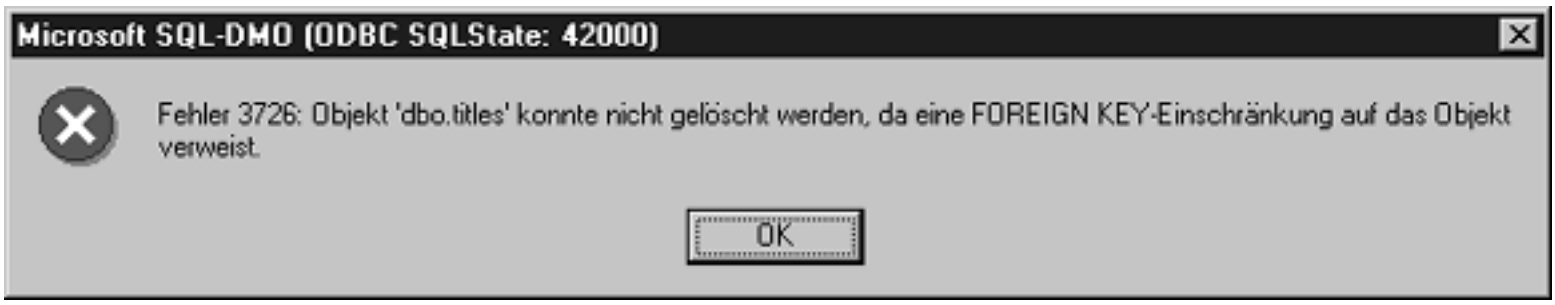
```
server: nachr.-nr. 3726, schweregrad 16, status 1, zeile 2
objekt 'titles' konnte nicht gelöscht werden, da eine foreign
key-einschränkung auf das objekt verweist.
```


2:Neue Tabelle in 'Lotto' auf 'EINS'
Spaltenname
Datentyp
Größe
Genauigk
Dezim
NULL zulass
Standardw
Identität
ID-Startwe
ID-Schrittweit
Ist RowGuid


2:Neue Tabelle in 'Lotto' auf 'EINS'
Spaltenname
Datentyp
Größe
Genauigk
Dezima
NULL zulass
Standardwert
Identität
ID-Startwert
Ziehung
int
4
10
0
Datum
datetime
8
0
0
Z1
tinyint
1
3
0
Z2
tinyint
1
3
0
Z3
tinyint
1
3
0
Z4
tinyint
1
3
0
Z5
tinyint
1
3
0
Z6
tinyint
1
3
0
ZZ
tinyint
1
3
0
smallint
smallmoney
text
timestamp
tinyint
uniqueidentifie
varbinary
varchar



Spaltenname	Datentyp	Größe	Genauigk	Dezim	NULL zulass	Standardwert	Identität	ID-Startwert
Ziehung	int	4	10	0	☐		☐	
Datum	datetime	8	0	0	☑	(getdate())	☐	
Z1	tinyint	1	3	0	☐		☐	
Z2	tinyint	1	3	0	☐		☐	
Z3	tinyint	1	3	0	☐		☐	
Z4	tinyint	1	3	0	☐		☐	
Z5	tinyint	1	3	0	☐		☐	
Z6	tinyint	1	3	0	☐		☐	
ZZ	tinyint	1	3	0	☑		☐	
					☐		☐	
					☐		☐	


Microsoft SQL-DMO (ODBC SQLState: 42000)
Fehler 3726: Objekt 'dbo.titles' konnte nicht gelöscht werden, da eine FOREIGN KEY-Einschränkung auf das Objekt verweist.
OK

kapitel 9 daten importieren und exportieren

9.1 quelle und ziel

wenn in einem unternehmen die entscheidung für den einsatz von sql server gefallen ist, liegen oftmals bereits datenbestände vor, die in datenbanken von sql server zu übertragen sind. umgekehrt kann es auch erforderlich sein, die in sql server gespeicherten daten anderen programmen zur verfügung zu stellen, ohne daß eine verbindung zu sql server bestehen muß.

das importieren und exportieren von daten gehört zu den aufgaben des datenbankadministrators. sql server unterstützt diese aufgaben mit verschiedenen werkzeugen und dienstprogrammen.

- data transformation services (dts)
- massenkopieren mit dem dienstprogramm bcp
- exportieren und importieren mit den verfahren zur sicherung/wiederherstellung von datenbanken
- replikation

dieses kapitel geht auf die beiden ersten verfahren ein. das sichern/wiederherstellen wurde bereits in kapitel 7 besprochen, der replikation widmet sich kapitel 22.

9.2 data transformation services

mit data transformation services (dts) lassen sich heterogene daten zwischen sql server und jedem ole-db-, odbc- oder textdateiformat übertragen. unter anderem sind die dts dafür vorgesehen, data warehouses und datamarts in sql server zu erstellen. dazu importiert und transformiert man daten aus verschiedenen heterogenen quellen entweder interaktiv oder nach einem terminplan ohne benutzereingriff.

dieses kapitel soll den einsatz der dts beim importieren und exportieren von daten nach/von sql server zeigen.

gegenüber dem dienstprogramm bcp (siehe weiter unten) sind die dts wesentlich einfacher einzusetzen. wenn es nicht auf abwärtskompatibilität oder maximale geschwindigkeit ankommt, sollten sie mit dts anstelle von bcp arbeiten.

9.2.1 dts-import-assistent

der dts-import-assistent hilft ihnen beim importieren von daten aus einer ole db-datenquelle in eine sql-server-datenbank.

9.2.2 dts-export-assistent

der dts-export-assistent hilft ihnen beim exportieren von daten aus einer sql-server-datenbank in eine ole-db-datenquelle.

9.2.3 daten mit dem dts-assistenten importieren

das folgende beispiel zeigt, wie man daten aus einer durch kommas getrennten textdatei mit hilfe des dts-assistenten in eine tabelle importiert und dabei gleichzeitig eine neue datenbank erstellt.

die im beispiel verwendete datei ldaten.txt ist auf der begleit-cd enthalten. kopieren sie diese datei in das verzeichnis lw:\sqlkomp\lotto\, wobei lw für das ziellaufwerk steht.

starten sie den dts-assistenten über **start / programme / microsoft sql server 7.0 / daten importieren und exportieren**. im ersten dialogfeld des dts-assistenten klicken sie auf **weiter**.

1. im zweiten dialogfeld des dts-assistenten wählen sie aus dem listenfeld **quelle** den eintrag *textdateien*. das dialogfeld zeigt daraufhin ein eingabefeld für den dateinamen an. klicken sie auf die schaltfläche mit den drei punkten, um die datei über die verzeichnisstruktur zu suchen. im beispiel befindet sich die datei im verzeichnis d:\sqlkomp\lotto\ldaten.txt. klicken sie auf **weiter**.
2. da die felder der datei ldaten.txt durch kommas getrennt sind, wählen sie im oberen teil des dialogfelds die erste option *getrennt. die spalten sind durch beliebige zeichen getrennt.* (siehe abbildung 9.1).

bild

abbildung 9.1: auswahl des zu importierenden dateiformats

3. als dateityp stehen ansi und unicode zur wahl. übernehmen sie hier die voreinstellung *ansi*. das zeilentrennzeichen bezieht sich auf die zeichen, mit denen in der textdatei die einzelnen zeilen (sprich datensätze) voneinander getrennt sind. übernehmen sie die voreinstellung *{cr}{lf}* für wagenrücklauf/zeilenschaltung. der textqualifizierer gibt das zeichen an, mit welchem textfelder eingeschlossen werden. in dieser zeichenfolge können dann auch sonderzeichen stehen, die andernfalls vielleicht als spaltentrennzeichen (siehe schritt 6) angesehen würden. die datei ldaten.txt enthält keine derartigen zeichen, so daß sie hier die voreinstellung übernehmen können.
4. im rechten teil dieses abschnitts legen sie die anzahl der zeilen fest, die am beginn der textdatei zu überspringen - d.h. nicht zu importieren - sind. dabei kann es sich zum beispiel um kommentare oder allgemeine überschriften handeln. für ldaten.txt übernehmen sie die vorgegebene 0.
5. wenn die zu importierende textdatei aus einer tabelle (beispielsweise einem excel-tabellenblatt) erzeugt wurde, kann die erste zeile die spaltenüberschriften enthalten, die nicht als werte in die zieltabelle zu importieren sind. schalten sie in diesem fall das kontrollkästchen **erste zeile enthält spaltennamen** ein. im unteren teil des dialogfelds erscheint eine dateivorschau, anhand der sie sich über die einzustellenden optionen informieren können. diese vorschau spiegelt nicht das format nach dem importieren wider. klicken sie auf **weiter**.
6. im dritten dialogfeld legen sie das zeichen fest, mit dem die einzelnen felder - die spalten - der datensätze in der textdatei getrennt sind. im beispiel sind die felder durch kommas getrennt. dieses format ist weit verbreitet und hat seinen niederschlag in der dateierweiterung *.csv für *comma*

separated values (= durch kommas getrennte werte) gefunden. übernehmen sie also die voreingestellte erste option. anhand der vorschau können sie sich davon überzeugen, ob das gewählte trennzeichen korrekt eingestellt ist. sollten die standardoptionen *komma*, *tabulator* oder *semikolon* nicht auf ihre datei zutreffen, wählen sie die option *andere* und tragen das in frage kommende trennzeichen manuell ein. das können auch mehrere zeichen sein. klicken sie auf **weiter**.

7. das ziel der zu importierenden textdatei geben sie im vierten dialogfeld (siehe abbildung 9.2) an. hier zeigt sich auch die flexibilität des dts-assistenten. im listenfeld **ziel** stehen zahlreiche formate zur wahl. der import ist also nicht ausschließlich auf sql server beschränkt. für das beispiel übernehmen sie natürlich die vorgegebene auswahl *microsoft ole db-provider für sql server*, da die textdatei in eine sql-server-datenbank importiert werden soll.

bild

abbildung 9.2: in diesem dialogfeld tragen sie die angaben für das ziel der neuen datenbank ein

8. im listenfeld **server** sind alle verfügbaren server aufgeführt. der import muß nicht in eine datenbank auf dem lokalen server erfolgen, sondern läßt sich auch über das netzwerk auf einen beliebigen anderen server abwickeln. im beispiel wählen sie die einstellung *(lokal)*.
9. die beiden optionen im mittleren teil des dialogfelds beziehen sich auf die anmeldung bei sql server. für das beispiel des unter windows nt laufenden lokalen servers wählen sie die zweite option.
10. benutzername und kennwort beziehen sich ebenfalls auf die anmeldung bei sql server. wenn sie noch keinen neuen benutzernamen oder ein kennwort festgelegt haben, geben sie als benutzernamen sa ein und lassen das kennwortfeld frei. das ist die voreinstellung nach installation von sql server. andernfalls müssen sie benutzername und kennwort für den server eintragen, auf den sie die textdatei importieren möchten.
11. im listenfeld **datenbank** wählen sie den eintrag <*neu*>, da die textdatei in eine noch nicht vorhandene datenbank importiert werden soll. daraufhin erscheint das dialogfeld **datenbank erstellen** (siehe abbildung 9.3).

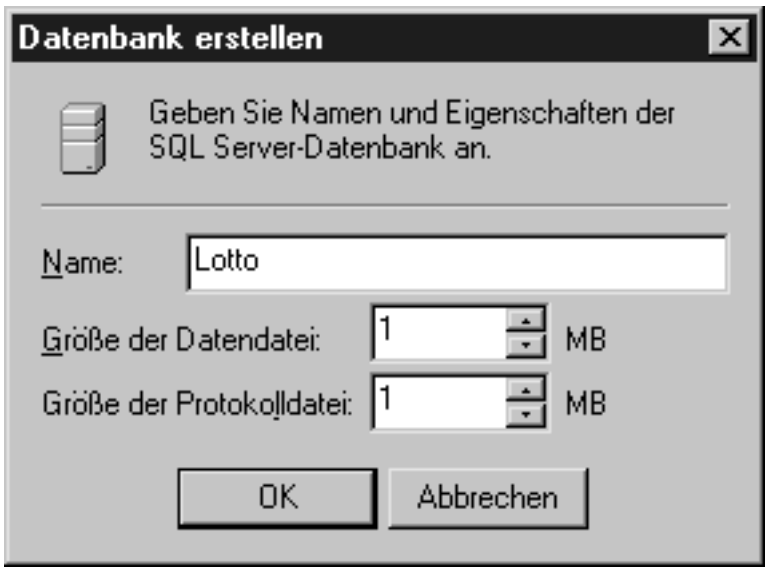


abbildung 9.3: im dialogfeld datenbank erstellen legen sie name und größe der datenbank fest

12. im dialogfeld **datenbank erstellen** tragen sie einen namen für die neue datenbank ein, zum beispiel lotto. die größen für datendatei und protokolldatei können sie mit 1 mbyte übernehmen. diese werte sind für das beispiel eigentlich zu groß, es ist aber die von sql server verwendete mindestgröße. in der regel brauchen sie diese werte auch bei größeren datenbeständen nicht zu verändern, da sie sql server automatisch an die aktuellen gegebenheiten anpaßt. klicken sie auf **ok**, um das dialogfeld zu schließen. sql server erstellt nun die neue datenbank.
13. das fünfte dialogfeld zeigt die verfügbaren quelltabellen an. falls mehrere einträge aufgeführt sind, schalten sie das in frage kommende kontrollkästchen ein. im feld **tabelle(n)** können sie den namen der zieltabelle ablesen.
14. über die schaltfläche mit den drei punkten in der spalte **transformation** gelangen sie zum dialogfeld **spaltenzuordnungen und transformationen**. hier legen sie für die zieltabelle die spaltennamen, die datentypen, die zulässigkeit von null-werten, die größe und die genauigkeit (abhängig vom jeweiligen datentyp) fest. nehmen sie die einstellungen gemäß abbildung 9.4 vor (die nicht mehr sichtbaren spalten z4, z5, z6 und zz sind alle vom typ tinyint). um einen namen oder einen typ zu ändern, klicken sie auf das betreffende feld und geben den namen ein bzw. wählen den typ aus dem listenfeld aus.

bild

abbildung 9.4: in diesem dialogfeld legen sie den typ der spalten fest

15. die bisherigen einstellungen haben sie interaktiv vorgenommen. wenn sie auf die schaltfläche **sql bearbeiten** klicken, erscheint die sql-anweisung für create table, die der dts-assistent automatisch aus ihren angaben erzeugt hat (siehe abbildung 9.5). diese anweisung können sie bearbeiten. die änderungen wirken nicht auf die bisher getroffenen spaltenzuordnungen zurück. umgekehrt führen spätere änderungen an den spaltenzuordnungen nicht zu einer änderung der sql-anweisung. falls sie die spaltenzuordnungen modifizieren (etwa einen anderen datentyp vorgeben) und die sql-anweisung auf den neuesten stand bringen wollen, klicken sie im dialogfeld **sql-anweisung** create table auf die schaltfläche **automatisch erzeugen**. danach greifen änderungen an der spaltenzuordnung wieder auf die sql-anweisung durch.

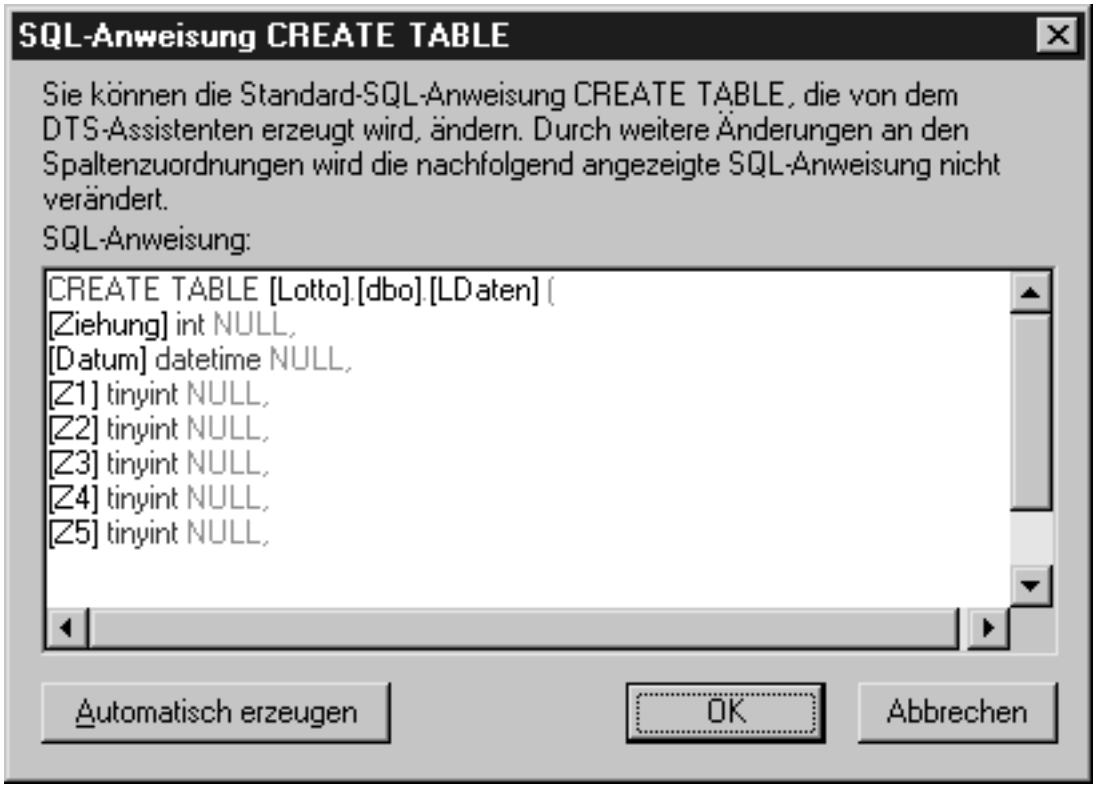


abbildung 9.5: die vom dts-assistenten automatisch erzeugte sql-anweisung können sie in diesem dialogfeld bearbeiten

16. klicken sie auf **ok**, um zum dts-assistenten zurückzukehren. über die schaltfläche **vorschau** können sie die ersten 100 zeilen der daten in der zieltabelle ansehen und kontrollieren. klicken sie gegebenenfalls auf die schaltfläche **zurück**, um die obigen schritte erneut zu durchlaufen.
17. im sechsten dialogfeld legen sie fest, wann der import durchzuführen ist und ob das dts-paket zu speichern bzw. zu replizieren ist. wenn sie das kontrollkästchen **speichern** einschalten, gelangen sie je nach gewählter option in ein weiteres dialogfeld, wo sie die entsprechenden angaben machen können.
18. im letzten dialogfeld **dts-assistenten beenden** erscheint eine zusammenfassung der getroffenen einstellungen. klicken sie auf **fertigstellen**, um die textdatei zu importieren (siehe abbildung 9.6).

bild

abbildung 9.6: im letzten dialogfeld des dts-assistenten erscheint eine übersicht der getroffenen einstellungen

19. es erscheint eine fortschrittsmeldung. nach abschluß der dateiübertragung meldet das dialogfeld **überträgt daten** entweder die erfolgreiche übertragung oder weist auf fehler hin. im letzten fall müssen sie zurückgehen und gegebenenfalls die einstellungen korrigieren. oftmals sind benutzername oder kennwort nicht richtig angegeben, oder die zieldatei ist schon vorhanden. klicken sie abschließend auf **fertig**.
20. im enterprise manager ist die neue datenbank lotto nun unter dem eintrag **datenbanken** aufgeführt (siehe abbildung 9.7). sollte die datenbank nicht zu sehen sein, markieren sie **datenbanken** in der strukturansicht und wählen aus dem menü **vorgang** den befehl **aktualisieren**.

Bild

abbildung 9.7: die neu erstellte datenbank erscheint im enterprise manager

um die symbole im rechten fensterbereich automatisch anordnen zu lassen, klicken sie mit der rechten maustaste in diesen bereich und wählen aus dem kontextmenü den befehl **symbole anordnen / automatisch anordnen**. dieser befehl steht nicht wie von anderen anwendungen gewohnt im menü **ansicht** zur verfügung. weiterhin ist der befehl **aktualisieren** nicht im menü **ansicht**, sondern im menü **vorgang** untergebracht.

9.3 das dienstprogramm dtswiz

mit dem dienstprogramm dtswiz lassen sich die dts-assistenten über optionen der befehlszeile starten. die syntax lautet:

```
dtswiz [{/? | {/n | [/u login_id] [/p kennwort]} [/f dateiname]
{/i | /x}
{/r providername | [/s servername][/d datenbankname] [/y]}}]
```

die optionen der befehlszeile lassen sich durch aufruf von

```
dtswiz /?
```

anzeigen (siehe abbildung 9.8).

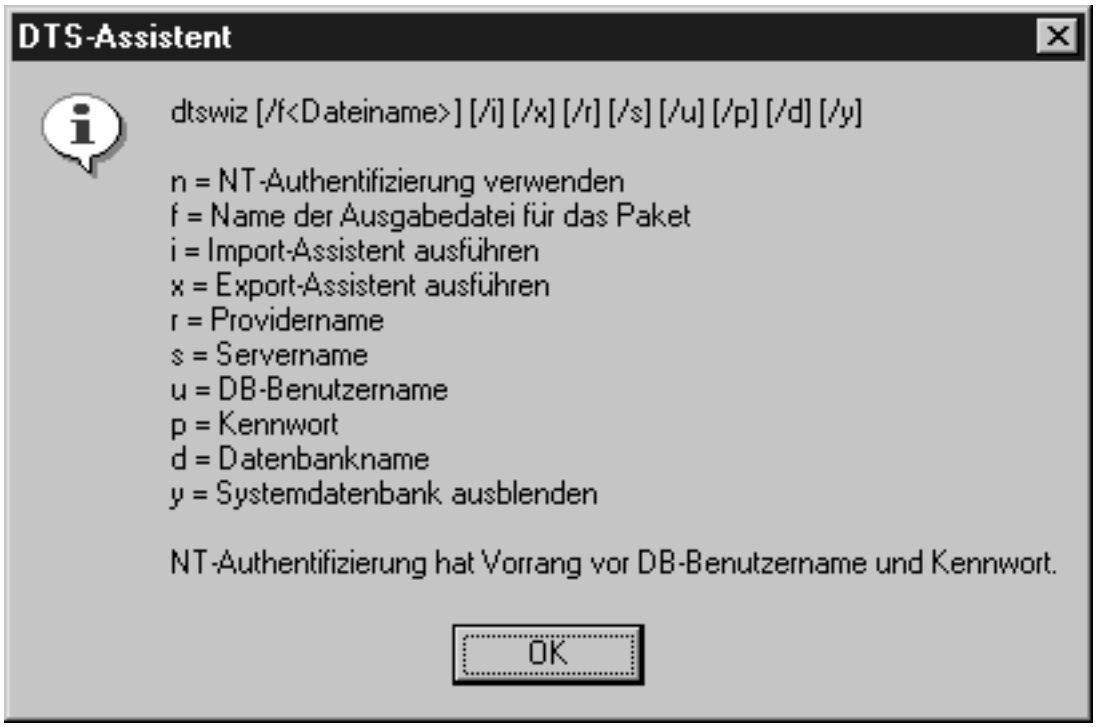


abbildung 9.8: befehlszeilenoptionen des dienstprogramms dtswiz

tabelle 9.1 erläutert die argumente.

argument	bedeutung
?	zeigt die optionen der befehlszeile an.
n	nt-authentifizierung verwenden. diese option hat vorrang gegenüber den optionen u und p.
f	das vom assistenten erzeugte paket wird unter diesem namen in einer com-strukturierten datei gespeichert.
i	dts-import-assistent ausführen.
x	dts-export-assistent ausführen.
r	providername. das ist der name des providers, zu dem eine verbindung für die datenquelle beim import der daten erfolgt oder der das ziel beim exportieren angibt. der microsoft-ole-db-provider für odbc lautet msdasql.
s	server-name. die netzwerkbezeichnung des servers, auf dem sql server läuft, wohin die daten exportiert bzw. woher die daten importiert werden. wenn die option /f nicht angegeben ist, fragt der assistent, ob das dts-paket in der sql-server-datenbank msdb gespeichert werden soll.
u	db-benutzername. die anmelde-id für die verbindung zu sql server.
p	kennwort. ein vom benutzer spezifiziertes kennwort, das in verbindung mit dem db-benutzernamen verwendet wird.
d	datenbankname. gibt die sql-server-datenbank an, aus der daten exportiert bzw. in die daten importiert werden.
y	systemdatenbank ausblenden. unterdrückt die anzeige der systemdatenbanken in der liste der quelldatenbanken beim importieren bzw. in der liste der zieldatenbanken beim exportieren der daten.

tabelle 9.1: argumente des dienstprogramms dtswiz

auf der befehlszeile lassen sich beliebig viele dieser optionen angeben. fehlende werte fragen die dts-assistenten bei bedarf ab.

das dts-paket kann in der sql-server-datenbank msdb, einer com-strukturierten datei oder im microsoft repository abgelegt werden.

9.4 datenbanken kopieren

eine datenbank läßt sich in einfacher weise kopieren, indem man zuerst eine sicherung der datenbank auf dem zielcomputer erzeugt und dann diese sicherung auf dem zielcomputer wiederherstellt. führen sie dazu die in kapitel 7 beschriebenen schritte zum sichern und wiederherstellen von datenbanken (beispielsweise mit dem enterprise manager) durch. zur wiederherstellung ist es nicht erforderlich, zuerst

die datenbank oder die dateien zu erzeugen.

die kopie einer datenbank kann man auch auf demselben computer - beispielsweise zu testzwecken - anlegen. in diesem fall existieren bereits die datenbankdateien für die originaldatenbank, so daß andere namen zu spezifizieren sind, wenn die datenbank während der wiederherstellung erzeugt wird.

9.5 massenkopieren mit bcp

die abkürzung bcp steht für bulk copy program - programm zum massenkopieren. mit diesem programm lassen sich daten importieren und exportieren. dabei können die daten im für sql server nativen modus, im textmodus oder als unicode-zeichen vorliegen.

der vorteil dieses dienstprogramms liegt zum einen in der ausführungsgeschwindigkeit und zum anderen im geringeren speicherbedarf verglichen mit dem werkzeug dts für die grafische benutzeroberfläche. weiterhin bietet sich bcp an, wenn es um die abwärtskompatibilität zu versionen vor sql server 7.0 geht. vor allem aber werden systemadministratoren, die sich schon jahrelang mit diesem programm, seinen eigentümlichkeiten und verzwickten einstellungen herumgeschlagen haben, auf bcp zurückgreifen.

zu den nachteilen von bcp gehört die komplexe syntax mit den zahlreichen schaltern, bei denen obendrein die groß-/kleinschreibung zu beachten ist. weiterhin ist die begrenzte auswahl von dateiformaten zu nennen. beispielsweise läßt sich kein excel-tabellenblatt direkt nach sql server importieren. ebenfalls zu bemängeln sind die spartanischen fehlermeldungen.

wenn die genannten vorteile nicht für sie relevant sind, sollten sie auf bcp verzichten und statt dessen mit den in sql server 7.0 eingeführten data transformation services (dts) arbeiten. dieses grafische werkzeug ist wesentlich einfacher zu handhaben. darüber hinaus bietet es die möglichkeit, daten beim importieren/exportieren zu transformieren, und es stellt eine größere auswahl von dateiformaten bereit.

9.5.1 syntax

```
bcp {[[datenbankname.][besitzer].]{tabellenname | sichtname}
| "abfrage"}
{in | out | queryout | format} datendatei
[schalter1 parameter1]
...
[schalter26 parameter26]
```

tabelle 9.2 erläutert die optionen und parameter des ersten syntaxteils von bcp. um die übersichtlichkeit zu wahren, sind die 26 möglichen schalter mit ihren parametern für den zweiten syntaxabschnitt nur symbolisch angegeben. die konkreten werte sind in tabelle 9.3 aufgeführt.

option/parameter	beschreibung

datenbankname	name der datenbank, auf die der zugriff erfolgen soll. der name ist optional. ist er nicht angegeben, gilt als voreinstellung die standarddatenbank des benutzers.
besitzer	besitzer der tabelle oder sicht. statt den besitzer konkret zu benennen, wie zum beispiel pubs.dbo.authors, können sie auch die allgemeinere syntax pubs..authors verwenden. das setzt allerdings voraus, daß der benutzer, der das massenkopieren vornimmt, auch besitzer der bezeichneten tabelle bzw. sicht ist.
tabellenname	name der sql-server-tabelle (quelle beim exportieren, ziel beim importieren).
sichtname	name der sql-server-sicht (quelle beim exportieren, ziel beim importieren). beim importieren müssen sich alle spalten der sicht auf dieselbe tabelle beziehen.
abfrage	eine transact-sql-anweisung, die eine ergebnismenge liefert. sollte die abfrage mehrere ergebnismengen liefern, übernimmt bcp nur die erste in die datendatei und ignoriert alle weiteren ergebnisse. der parameter queryout ist erforderlich. die abfrage ist in doppelte anführungszeichen einzuschließen, elemente innerhalb der abfrage in einfache anführungszeichen.
in | out | queryout | format	richtung der datenübertragung: in - aus datendatei in tabelle oder sicht out - aus tabelle oder sicht in datendatei beim exportieren ist queryout erforderlich, wenn die ausgabe aus einer transact-sql-abfrage oder einer gespeicherten prozedur stammt. format bewirkt das erstellen einer formatdatei nach den mit den schaltern -n, -c, -w oder -6 angegebenen einstellungen. wenn format angegeben ist, muß auch eine formatdatei mit dem schalter -f spezifiziert werden (siehe tabelle 9.3).
datendatei	name und pfad der datei, aus der die daten nach sql server zu importieren bzw. in die die daten aus sql server zu exportieren sind.

tabelle 9.2: optionen und parameter für den ersten syntaxabschnitt von bcp

schalter/parameter	beschreibung
-m maxfehler	maximalzahl der fehler, die auftreten dürfen, bevor die bcp-operation abbricht. jede fehlerhafte einfügeoperation zählt als ein fehler.
-f formatdatei	name und pfad der formatdatei, die beim exportieren/importieren zu verwenden ist. das kann zum beispiel eine formatdatei sein, die bei einer früheren bcp-operation für dieselbe tabelle oder sicht erzeugt wurde.
-e fehlerdatei	name und pfad einer fehlerdatei, in der bcp sowohl fehlermeldungen als auch nicht erfolgreich übertragene zeilen speichert. empfänger der fehlermeldungen ist die arbeitsstation des benutzers. die fehlerdatei ist vor allem bei nicht überwachten operationen nützlich, beispielsweise bei datenimporten, die in der nacht ablaufen.
`-f erste zeile`	erste zeile, die zu kopieren ist. der standardwert 1 entspricht der ersten zeile in der datendatei.
`-l letzte zeile`	letzte zeile, die zu kopieren ist. der standardwert 0 entspricht der letzten zeile der datendatei.
`-b stapelgröße`	die anzahl der pro stapel übertragenen zeilen. jeder stapel wird als eine transaktion übertragen. bei fehlerfreier übertragung führt sql server ein implizites commit aus, bei fehlerhafter übertragung ein rollback. per vorgabe ist die stapelgröße gleich der anzahl der zeilen in der datendatei. der schalter -b darf nicht zusammen mit der option -h "rows_per_batch = bb" verwendet werden.
`-n`	massenkopieren im nativen modus, der spezifisch für die datentypen von sql server ist. diese option fragt vom benutzer keine feldinformationen ab und verwendet systemeigene datentypen.
`-c`	massenkopieren im zeichenmodus. diese option fragt vom benutzer keine feldinformationen ab und verwendet die folgenden einstellungen: speichertyp char, keine präfixe, feldtrennzeichen \t und zeilenabschlußzeichen \n.

-w	massenkopieren im unicode-modus. diese option fragt vom benutzer keine feldinformationen ab und verwendet die folgenden einstellungen: speichertyp nchar, keine präfixe, feldtrennzeichen \t und zeilenabschlußzeichen \n. nicht für sql-server-versionen vor 7.0 verwendbar.
-n	export mit nativem modus für nichtzeichendaten und unicode-modus für zeichendaten. die option fragt vom benutzer keine feldinformationen ab und bietet sich an, wenn sie sowohl unicode-daten als auch systemeigene daten übertragen wollen, da das leistungsverhalten besser als bei der option -w ist. allerdings ist -n ebenfalls nicht für sql-server-versionen vor 7.0 verwendbar.
-6	massenkopieren mit den standarddatentypen von sql server version 6.x bei zeichenmodus oder nativem modus. verwenden sie diese option, wenn sie daten importieren wollen, die sie mit sql server version 6.x erzeugt haben. der schalter ist erforderlich, wenn sie datums- und währungsdaten (datentypen datetime/smalldatetime bzw. money/smallmoney) zwischen der version 7.0 und vorherigen versionen übertragen. beim export in eine datendatei erstellt bcp für datumswerte auch nicht mit dem schalter -6 die datumsformate von sql server 6.x, sondern schreibt sie immer im odbc-format. außerdem werden null-werte in bit-spalten als 0 ausgegeben, da die früheren versionen von sql server keine null-werte für bit-daten unterstützen.
-q	verwendet bezeichner in anführungszeichen.
-c codeseite	gibt die codeseite an, die von der importierten datei verwendet wird. diese angabe ist nur erforderlich, wenn die datei werte mit den datentypen char, varchar oder text enthält und werte größer als 127 oder kleiner als 32 vorkommen. tabelle 9.4 listet die gültigen codeseiten auf.
-t feldabschlußzeichen	zeichen zum trennen von feldern (spalten). tabelle 9.6 enthält die gültigen zeichen. standardzeichen ist \t.
-r zeilenabschlußzeichen	zeichen zum trennen von zeilen (datensätzen). tabelle 9.6 enthält die gültigen zeichen. standardzeichen ist \n.

`-i eingabedatei`	datei für die umleitung der eingabe im interaktiven modus von bcp, wenn die schalter -u, -c, -w, -6 oder -n nicht angegeben wurden. in dieser datei sind die antworten für die fragen von bcp nach den feldtypen enthalten.
`-o ausgabedatei`	datei für die umleitung der ausgabe. eine derartige datei empfiehlt sich bei nicht überwachter ausführung von bcp, da man im nachhinein die leistung und ausführung von bcp überwachen und diagnostizieren kann.
`-a paketgröße`	gibt die anzahl der bytes an, die in einem netzwerkpaket übertragen werden. der standardwert bei windows nt server und windows nt clients beträgt 4096. gültige werte liegen zwischen 512 und 65535. größere werte können die übertragungsleistung verbessern. die einstellungen können sie anhand der rückmeldungen von bcp überprüfen und gegebenenfalls anpassen.
`-e`	gibt an, daß man die werte in einer identitätsspalte der zieltabelle mit den werten aus der datendatei füllen will. ist dieser schalter nicht angegeben, füllt sql server die identitätsspalte automatisch und ignoriert die für das feld zutreffenden werte aus der datendatei.
`-s servername`	name des servers, der die datenbank und die tabelle enthält, die für die bcp-operation spezifiziert sind. als standardeinstellung gilt der lokale server, auf dem sql server ausgeführt wird. wenn sie bcp von einem remote-client des netzwerks ausführen, ist der schalter erforderlich.
`-u benutzername`	anmeldename bei sql server.
`-p kennwort`	kennwort für die anmeldung bei sql server. wenn kennwort nicht angegeben ist, fragt es bcp ab. wenn sie mit integrierter sicherheit arbeiten oder die anmeldung bei sql server kein kennwort erfordert (wie es in der standardeinstellung nach der installation der fall ist), fordert bcp dennoch ein kennwort an. verwenden sie den schalter -p ohne kennwort, um diese aufforderung zu umgehen.
`-t`	verwendet eine vertraute verbindung zu sql server. bei dieser option sind benutzername und kennwort nicht erforderlich.

`-v`	zeigt die verwendete version von bcp an.
`-r`	gibt an, daß währungs-, datums- und zeitwerte entsprechend den ländereinstellungen des client-computers importiert werden. die standardeinstellung ignoriert die ländereinstellungen.
`-k`	mit dieser option bleiben null-werte aus der datendatei beim importieren für die eingefügten spalten erhalten und werden nicht durch eventuell vorhandene standardwerte von sql server ersetzt.
`-h "ladehinweise [,...n]"`	benachrichtigt sql server, hinweise zum verbessern der importleistung zu verwenden. tabelle 9.5 enthält eine liste der gültigen hinweise.

tabelle 9.3: schalter und parameter für den zweiten syntaxabschnitt von bcp

parameter	beschreibung
acp	ansi/microsoft windows (iso 1252).
oem	standardcodeseite.
raw	keine konvertierung zwischen codeseiten (schnellste option).
<wert>	nummer der zu verwendenden codeseite, zum beispiel 850.

tabelle 9.4: einträge für den parameter codeseite

parameter	beschreibung
order{spalte [asc | desc][,n]}	sortierreihenfolge
rows_per_batch=bb	anzahl (bb) der datenzeilen pro stapel
kilobytes_per_batch=cc	ungefähre größe der daten pro stapel (cc) in kbyte.
tablock	sperrung auf tabellenebene während des massenkopierens
check_constraints	prüfen der für die zieltabelle festgelegten einschränkungen

tabelle 9.5: einträge für den parameter ladehinweise

zeichentyp	syntax
tabulator	\t
neue zeile	\n
wagenrücklauf	\r
backslash	\\

null-abschlußzeichen	\0
benutzerdefinierte abschlußzeichen	zeichen (^, %, * usw.)

tabelle 9.6: von bcp akzeptierte abschlußzeichen

9.5.2 beispiele

textdatei importieren

die folgende anweisung importiert die ziehungsdaten aus der tabelle ldaten.txt in die tabelle ldaten der datenbank lotto. die daten liegen in der datei ldaten.txt im textformat vor, wobei die einzelnen felder durch kommas getrennt und die zeilen mit wagenrücklauf/zeilenvorschub abgeschlossen sind. das zeilenabschlußzeichen brauchen sie demnach nicht anzugeben. allerdings müssen sie das feldabschlußzeichen mit -t, spezifizieren. die option -f2250 bewirkt, daß der import ab zeile (sprich ziehung) 2250 erfolgt. die anmeldung läuft unter dem standardnamen (sa) ohne kennwort. die option -c gibt an, daß die daten im zeichenformat vorliegen. die korrekte umwandlung der datumswerte stellt die option -r sicher.

```
bcp lotto..ldaten in ldaten.txt -t, -usa -p -c -r -f2250
```

textdatei exportieren

die nachstehende anweisung kopiert die daten aus der tabelle ldaten der datenbank lotto im durch komma getrennten textformat mit dem standardzeilenabschlußzeichen in die datei ld.txt. die parameter haben die gleiche wirkung wie im beispiel zum importieren der textdatei. es fehlt lediglich die option -f, so daß bcp alle vorhandenen zeilen der tabelle exportiert.

```
bcp lotto..ldaten out ld.txt -t, -usa -p -c -r
```

im nativen format importieren/exportieren

wenn sie keine lesbare textdatei brauchen, können sie auf die umwandlung der datentypen in das textformat verzichten und die daten im systemeigenen format (option -n) importieren:

```
bcp lotto..ldaten in ldaten.dat -usa -p -n
```

bzw. exportieren:

```
bcp lotto..ldaten out ldaten.dat -usa -p -n
```

9.5.3 interaktives bcp

das massenkopieren starten sie von der befehlszeile mit allen erforderlichen angaben. wenn die schalter -n, -n, -c, -f und/oder -w nicht angegeben sind, wechselt bcp automatisch in den interaktiven modus und fragt die erforderlichen einstellungen ab.

die folgende beispielsitzung zeigt, wie sie interaktiv mit bcp die textdatei ldaten.txt in die datenbank lotto importieren und dabei eine formatdatei erstellen. genau diese formatdatei verwenden sie im übernächsten abschnitt, um die daten aus der datenbank in eine ausgabedatei zu exportieren.

führen sie die folgenden schritte aus:

1. öffnen sie ein ms-dos-fenster. wechseln sie in das verzeichnis \sqlkomp\lotto (oder in das verzeichnis, in das sie die datei ldaten.txt kopiert haben).
2. geben sie den folgenden befehl an der eingabeaufforderung ein:

```
bcp lotto..ldaten in ldaten.txt -usa -p
```

die anmeldung erfolgt hierbei mit sa ohne kennwort. da die oben genannten schalter nicht angegeben sind, wechselt bcp in den interaktiven modus und zeigt folgende aufforderung an:

```
geben sie den speichertyp des felds ziehung ein [int-
null]:
```

bcp hat bereits eine verbindung zur datenbank hergestellt, kennt also die spalten, die datentypen und die null-zulässigkeit. allerdings nützt ihnen das bei einer textdatei nicht viel. da die daten im textformat und nicht im systemeigenen (nativen) format von sql server vorliegen, müssen sie den typ char (abgekürzt durch c) eingeben.

3. als nächstes erscheint die aufforderung:

```
geben sie die präfixlänge des felds ziehung ein [1]:
```

das sogenannte längenpräfixzeichen weist auf die länge eines feldes hin. für den da5tentyp int mit null-zulässigkeit ist ein präfixzeichen erforderlich. damit sie die präfixlänge nicht selbst bestimmen müssen, geben sie hier 0 ein.

in der online-dokumentation finden sie eine tabelle der präfixlängen. suchen sie im index nach dem stichwort *präfixlänge*, und wählen sie das gleichnamige thema.

4. als nächstes will bcp wissen, wie lang das jeweilige feld in der datendatei maximal sein kann:

```
geben sie die feldlänge für ziehung ein [12]:
```

die ziehungsnummer ist höchstens vierstellig, also geben sie hier eine 4 ein. allerdings schadet es nichts, den vorgegebenen wert (12) mit **è** zu übernehmen.

5. der vierte teil der aufforderung für die spalte ziehung lautet:

```
geben sie das feldabschlusszeichen ein [none]:
```

hier geht es um das zeichen, mit dem die einzelnen felder (und späteren spalten) in der textdatei getrennt sind. geben sie also ein simples komma ein, und drücken sie **è**.

6. für die spalten datum, z1 bis z6 führen sie die gleichen schritte aus. der speichertyp ist immer c (für char), die präfixlänge legen sie mit 0 fest, als feldlänge wählen sie für die spalte datum den wert 10 und für die übrigen spalten den wert 2. schließlich tragen sie für das feldabschlußzeichen ein komma ein.
7. achtung bei der letzten spalte (zz): da die ziehungen in der datei ldaten.txt zeilenweise gespeichert sind, kommt nach der letzten spalte kein komma als trennzeichen, sondern das zeichen für »neue zeile«. deshalb müssen sie als feldabschlußzeichen für die spalte zz die zeichenfolge \n eingeben.
8. nachdem sie alle angaben für die spalten bestätigt bzw. eingegeben haben, erscheint die frage:

```
möchten sie diese informationen in einer datei speichern
[j/n]:
```

wenn sie jetzt ein »j« eingeben, warten sie vergeblich auf die abfrage eines dateinamens. bcp gibt zwar deutsche meldungen aus, erwartet aber ein »y« für »yes« zur bestätigung. tippen sie also ein y ein, oder drücken sie einfach **è**. daraufhin kommt die aufforderung:

```
hostdateiname [bcp.fmt]:
```

9. tippen sie jetzt den namen der formatdatei ein. hier bietet sich natürlich lotto.fmt wie von selbst an. die dateierweiterung ist frei wählbar und wird auch nicht automatisch ergänzt.
10. nach einem abschließenden **è** kommt der spannende moment: es erscheint die meldung, daß der kopiervorgang startet. schließlich zeigt bcp an, wie viele zeilen kopiert wurden, wie groß die netzwerkpakete sind und wie lange das ganze gedauert hat.

9.5.4 formatdatei

wenn sie öfter massenkopieren mit daten gleicher struktur ausführen, bietet es sich an, eine formatdatei bereitzustellen. eine derartige datei können sie auch anlegen lassen, wenn sie die datensätze das erste mal interaktiv importieren oder exportieren - so geschehen im letzten abschnitt. die erzeugte formatdatei lotto.fmt hat folgendes aussehen:

```
7.0
9
1       sqlchar       0    12   ","              1     ziehung
2       sqlchar       0    10   ","              2     datum
3       sqlchar       0    2    ","              3     z1
4       sqlchar       0    2    ","              4     z2
5       sqlchar       0    2    ","              5     z3
6       sqlchar       0    2    ","              6     z4
7       sqlchar       0    2    ","              7     z5
8       sqlchar       0    2    ","              8     z6
9       sqlchar       0    2    "\r\n"           9     zz
```

in der ersten zeile steht die versionsnummer (für sql server 7.0). die zweite zeile gibt die anzahl der spalten an. die nachfolgenden zeilen enthalten die angaben:

- reihenfolge des feldes in der quelldatei
- datentyp in der quelldatei. bei ascii-dateien verwenden sie sqlchar (für den datentyp char), bei datendateien im systemeigenen format die standarddatentypen.
- präfixlänge
- maximale datenlänge in der quelldatei
- abschlußzeichen, d.h. feldtrennzeichen zwischen den einzelnen feldern und zeilentrennzeichen beim letzten feld, wenn die quelldatei zeilenweise aufgebaut ist
- reihenfolge der spalten in der sql-server-tabelle
- name der spalte in der sql-server-tabelle

weitere einsatzmöglichkeiten der formatdatei entnehmen sie bitte der online-dokumentation. beispielsweise ist es möglich, die spaltenzuordnung zu ändern oder spalten gänzlich aus dem kopiervorgang auszublenden.

9.5.5 kopieren mit formatdatei

wie versprochen, setzen sie jetzt die oben erstellte formatdatei ein, um die - vielleicht inzwischen aktualisierten - ziehungsergebnisse in eine textdatei lneu.txt zu exportieren. das geschieht ganz einfach mit der folgenden anweisung:

```
bcp lotto..ldaten out lneu.txt -usa -p -flotto.fmt
```

wenn sie den parameter out durch in ersetzen, geht alles wieder in die andere richtung.

wenn sie per bcp textdateien mit datumsfeldern importieren, müssen sie darauf achten, daß die datumsfelder in einem von sql server akzeptierten format vorliegen.

9.6 massenkopieren mit transact-sql (bulk insert)

die transact-sql-anweisung bulk insert ist neu in sql server 7.0. damit können sie ein massenkopieren von daten in eine tabelle oder sicht einer sql-server-datenbank durchführen. im gegensatz zum programm bcp erlaubt es bulk insert nicht, daten aus einer sql-server-datenbank zu exportieren.

die anweisung bulk insert bietet sich zum beispiel an, wenn sie eine datenbank neu erstellen und sofort mit umfangreichen daten füllen möchten.

syntax:

```
bulk insert [['datenbank'.]['besitzer'].]
{'tabellenname' from datendatei}
[with
(
[ batchsize[=stapelgröße]]
[[,] check_constraints]
[[,] codepage[='acp' | 'oem' | 'raw' | 'codeseite']]
[[,] datafiletype[={'char'|'native'|'widechar'|'widenative'}]]
[[,] fieldterminator[='feldabschlußzeichen']]
[[,] firstrow[=erstezeile]]
[[,] formatfile[='formatdatei']]
[[,] keepidentity]
[[,] keepnulls]
[[,] kilobytes_per_batch[=stapelgröße]]
[[,] lastrow[=letztezeile]]
[[,] maxerrors[=maxfehler]]
[[,] order ({spalte[asc | desc]} [,...n])]
[[,] rows_per_batch[=zeilenprostapel]]
[[,] rowterminator[='zeilenabschlußzeichen']]
[[,] tablock]
)
]
```

tabelle 9.7 erläutert die optionen und parameter für die anweisung bulk insert.

optionen und parameter	beschreibung
'datenbank'	name der datenbank, in der die zieltabelle für das massenkopieren enthalten ist.
'besitzer'	name des besitzers der tabelle oder sicht. dieser parameter ist optional, wenn der ausführende von bulk insert auch besitzer der betreffenden tabelle oder sicht ist.
'tabellenname'	name der tabelle oder sicht, in die die daten massenkopiert werden.
datendatei	dateiname und pfad zur datei, in der die zu importierenden daten enthalten sind.
batchsize[=stapelgröße]	die anzahl der zeilen, die in einem stapel als transaktion nach sql server kopiert werden. läuft die übertragung eines stapels fehlerfrei ab, führt sql server ein commit, ansonsten ein rollback aus. in der voreinstellung nimmt sql server die gesamte datendatei als einen stapel an.
check_constraints	mit dieser option überprüft sql server während der übertragung alle einschränkungen, die für tabellenname definiert sind. in der standardeinstellung werden keine einschränkungen beachtet.
codepage[='acp' | 'oem' | 'raw' | 'codeseite']	die codeseite der daten in der datendatei. diese angabe ist nur von bedeutung, wenn in spalten vom typ char, varchar oder text zeichen mit einem wert kleiner 32 oder größer 127 vorkommen. die parameter entsprechen den in tabelle 9.4 genannten werten für das programm bcp.
datafiletype[={'char'|'native'|'widechar'|'widenative'}]	führt den kopiervorgang entsprechend der angegebenen standardeinstellung aus.

fieldterminator[='feldabschlußzeichen']	spezifiziert das zeichen, mit dem die einzelnen felder (spalten) in der datendatei getrennt sind. die voreinstellung nimmt das tabulatorzeichen (\t) an.
firstrow[=erstezeile]	legt die erste zeile in der datendatei für den kopiervorgang fest. standardeinstellung ist 1.
formatfile[='formatdatei']	spezifiziert name und pfad zu einer formatdatei, die den inhalt der datendatei beschreibt. die antworten wurden mit dem programm bcp für die gleiche tabelle oder sicht erstellt.
keepidentity	gibt an, daß die datendatei werte für eine identitätsspalte enthält. ohne diesen parameter ignoriert sql server die identitätswerte und weist automatisch der zieltabelle eindeutige werte entsprechend der tabellendefinition zu.
keepnulls	bei dieser option behalten leere spalten einen null-wert in der zieltabelle, andernfalls weist sql server standardwerte zu.
kilobytes_per_batch[=stapelgröße]	geschätzte größe der in einem stapel zu übertragenden datenmenge. in der standardeinstellung überträgt bulk insert die daten in einem datenblock (chunk).
lastrow[=letztezeile]	letzte zu kopierende zeile. beim standardwert 0 kopiert die anweisung bis zur letzten in der datendatei vorhandenen zeile.
maxerrors[=maxfehler]	größte zahl von fehlern, die auftreten dürfen, bevor bulk insert abgebrochen wird. standardwert ist 10. jede nicht importierte zeile zählt als ein fehler.
order ({spalte[asc | desc]} [,...n])	gibt die sortierreihenfolge der jeweiligen spalte in der datendatei an. diese option kann eine leistungsverbesserung bewirken.

rows_per_batch[=zeilenprostapel]	wenn batchsize nicht angegeben ist, können sie mit dieser option die anzahl der pro stapel zu übertragenden datenzeilen festlegen.
rowterminator[='zeilenabschlußzeichen']	spezifiziert das zeichen, mit dem die einzelnen zeilen in der datendatei getrennt sind. als voreinstellung gilt das zeichen für neue zeile (\n).
tablock	legt eine sperre auf tabellenebene für die dauer des massenkopierens fest. diese option bringt eine merkliche leistungsverbesserung, da weniger sperrkonflikte für die tabelle auftreten.

tabelle 9.7: optionen und parameter für die anweisung bulk insert

beispiel:

auf der begleit-cd ist das skript instlott.sql enthalten, das die datenbank lotto löscht, neu erstellt, die tabelle ldaten anlegt und mit den ziehungsdaten aus der textdatei ldaten.txt füllt. in der datei ldaten.txt sind die spalten durch kommas getrennt. das skript verwendet die anweisung bulk insert, um die tabelle ldaten der datenbank lotto mit den immerhin über 2200 zeilen zu füllen:

```
bulk insert ldaten from 'd:\sqlkomp\lotto\ldaten.txt'
with (
fieldterminator=','
     )
```

ändern sie gegebenenfalls das quellverzeichnis, falls sie ldaten.txt in einem anderen verzeichnis installiert haben. als einzige option ist das komma als feldabschlußzeichen anzugeben, da in der standardeinstellung das tabulatorzeichen gilt.

9.7 text und bilder importieren und exportieren (textcopy)

das dienstprogramm textcopy erlaubt es, text- und bilddateien zu importieren und zu exportieren. beispiele hierfür liefern die dateien pubimage.bat und pubtext.bat, die sie im verzeichnis \mssql7\install finden. diese stapeldateien importieren bilder bzw. texte in die beispieldatenbank pubs.

mit textcopy können sie einen einzelnen text- oder bildwert zwischen einer datei und sql server übertragen (importieren/exportieren). der wert ist eine angegebene text- oder bildspalte einer einzelnen zeile (die durch die where-klausel festgelegt ist) der angegebenen tabelle.

die parameter lassen sich mit

```
textcopy /?
```

anzeigen.

syntax:

die syntax des dienstprogramms textcopy sieht folgendermaßen aus:

```
textcopy [/s [server]] [/u [anmeldename]] [/p [kennwort]]
[/d [datenbank]] [/t tabelle] [/c spalte] [/w"where-klausel"]
[/f datei] [{/i | /o}] [/k chunk-größe] [/z] [/?]
```

tabelle 9.8 erläutert die parameter.

parameter	beschreibung
/s servername	name des servers, zu dem eine verbindung herzustellen ist. wenn dieser parameter nicht angegeben ist, wird der lokale sql server verwendet.
/u anmeldename	der anmeldename des benutzers. wenn dieser parameter nicht angegeben ist, wird eine vertraute verbindung verwendet.
/p kennwort	das zum anmeldenamen gehörende kennwort. ist dieser parameter nicht angegeben, wird das standardkennwort (null) verwendet.
/d datenbank	die datenbank, die die tabelle mit den text- oder bilddaten enthält. ist *datenbank* nicht angegeben, wird die zum anmeldenamen gehörende standarddatenbank verwendet.
/t tabelle	die tabelle, die den text- oder den bildwert enthält.
/c spalte	die text- oder bildspalte der tabelle.
/w "where-klausel"	eine vollständige where-klausel (einschließlich des schlüsselwortes where), die eine einzelne zeile der tabelle spezifiziert.
/f datei	der name der datendatei.
/i	steht für input - eingabe, d.h. importieren. kopiert den text- oder bildwert aus der datendatei nach sql server.
/o	steht für output - ausgabe, d.h. exportieren. kopiert den text- oder bildwert aus sql server in die datendatei.

/k chunk-größe	größe des puffers für die datenübertragung in bytes. der minimale wert ist 1024, der standardwert 4096 byte.
/z	zeigt während der ausführung debug-informationen an.
/?	zeigt versionsnummer und parameter an.

tabelle 9.8: befehlszeilenparameter des dienstprogramms textcopy

wenn sie bestimmte optionen nicht angeben, fordert das dienstprogramm textcopy zur eingabe auf.

die parameter /i bzw. /o bezeichnen die übertragungsrichtung aus sicht von sql server, d.h. nicht vom standpunkt des benutzers aus. beim importieren wird der wert in der zieltabelle durch den wert aus der datei ersetzt. einen exportierten wert schreibt sql server in die angegebene datendatei, wobei eine eventuell vorhandene datei gleichen namens überschrieben wird.

das beispiel zeigt die bereits erwähnte datei pubimage.bat. diese stapeldatei kopiert mit dem befehl textcopy die im verzeichnis \mssql7\install\ installierten bilddateien (*.gif) in die spalte logo der tabelle pub_info der datenbank pubs. als befehlszeilenparameter sind der name des zielservers und optional das kennwort anzugeben.

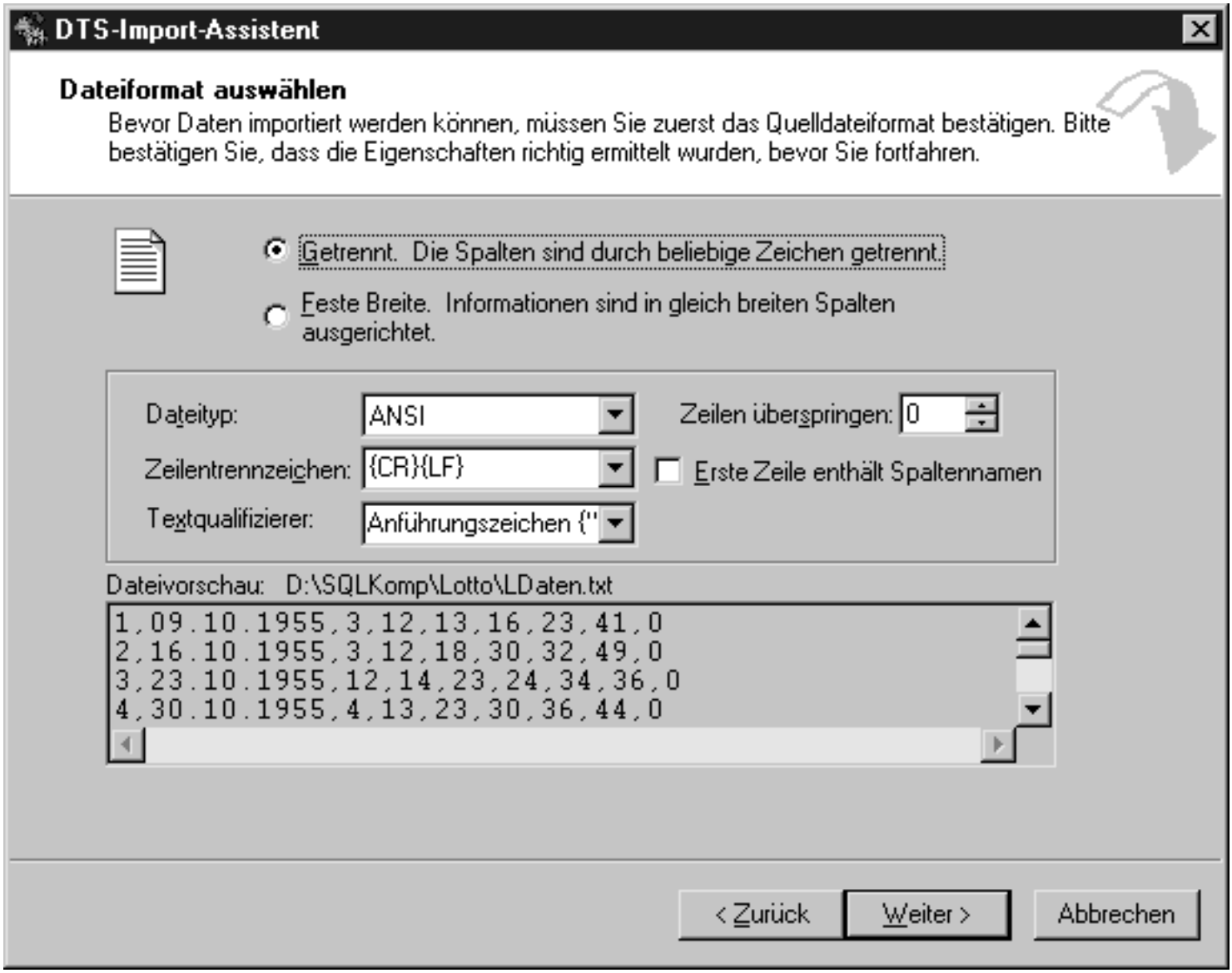
```
@echo off
if "%1" == "" goto usage
echo.
echo inserting images into pubs database on server %1
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '1389'" -falgodata.gif
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '0877'" -fbinnet.gif
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '9901'" -fgggg.gif
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '1622'" -f5lakes.gif
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '0736'" -fnewmoon.gif
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '9999'" -flucerne.gif
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '1756'" -framona.gif
textcopy -i -usa -p%2 -s%1 -dpubs -tpub_info -clogo -w"where
pub_id = '9952'" -fscootney.gif
echo image update complete!
echo.
goto done
:usage
echo.
echo usage: pubimage servername [sapassword]
```

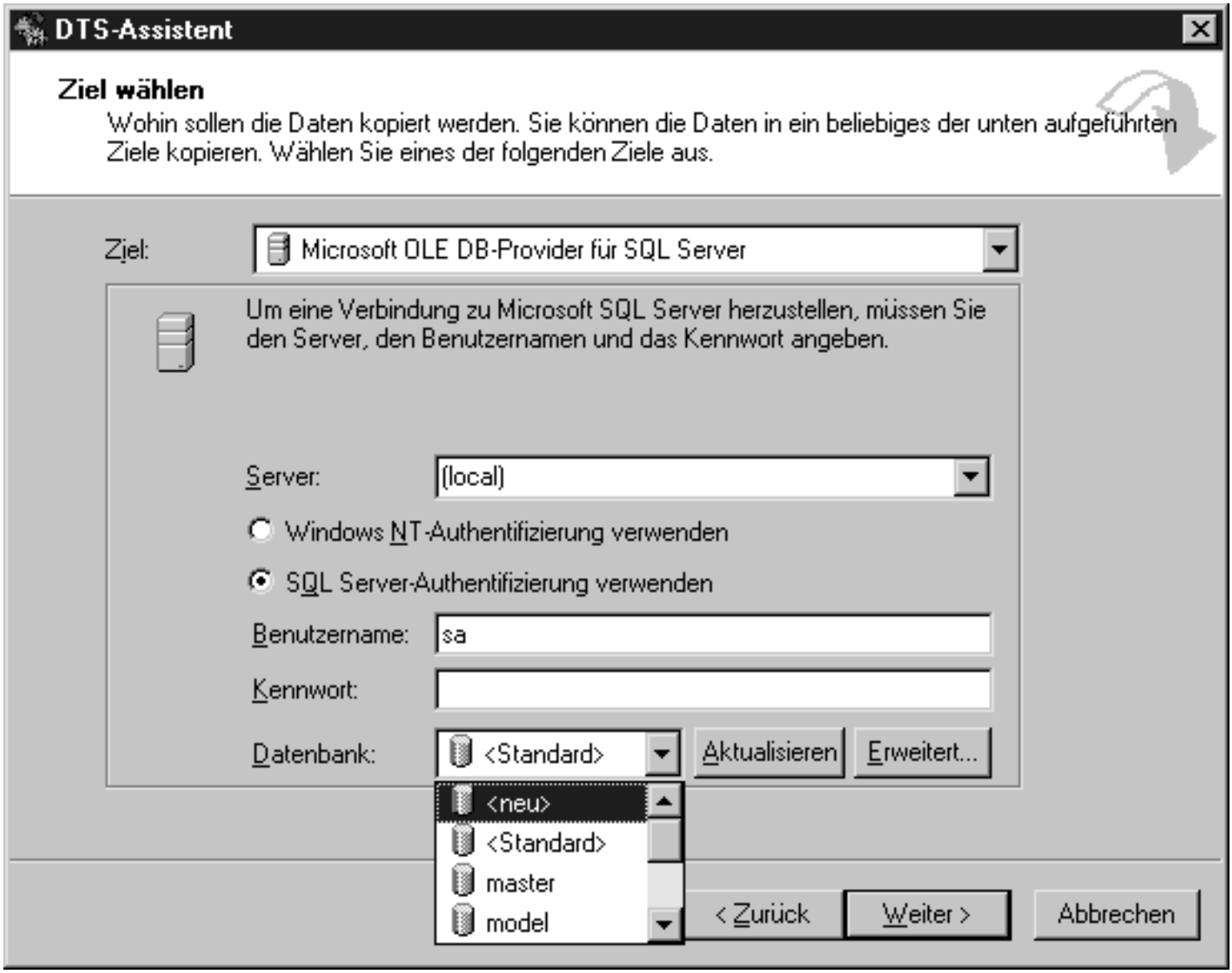
```
echo.
:done
```

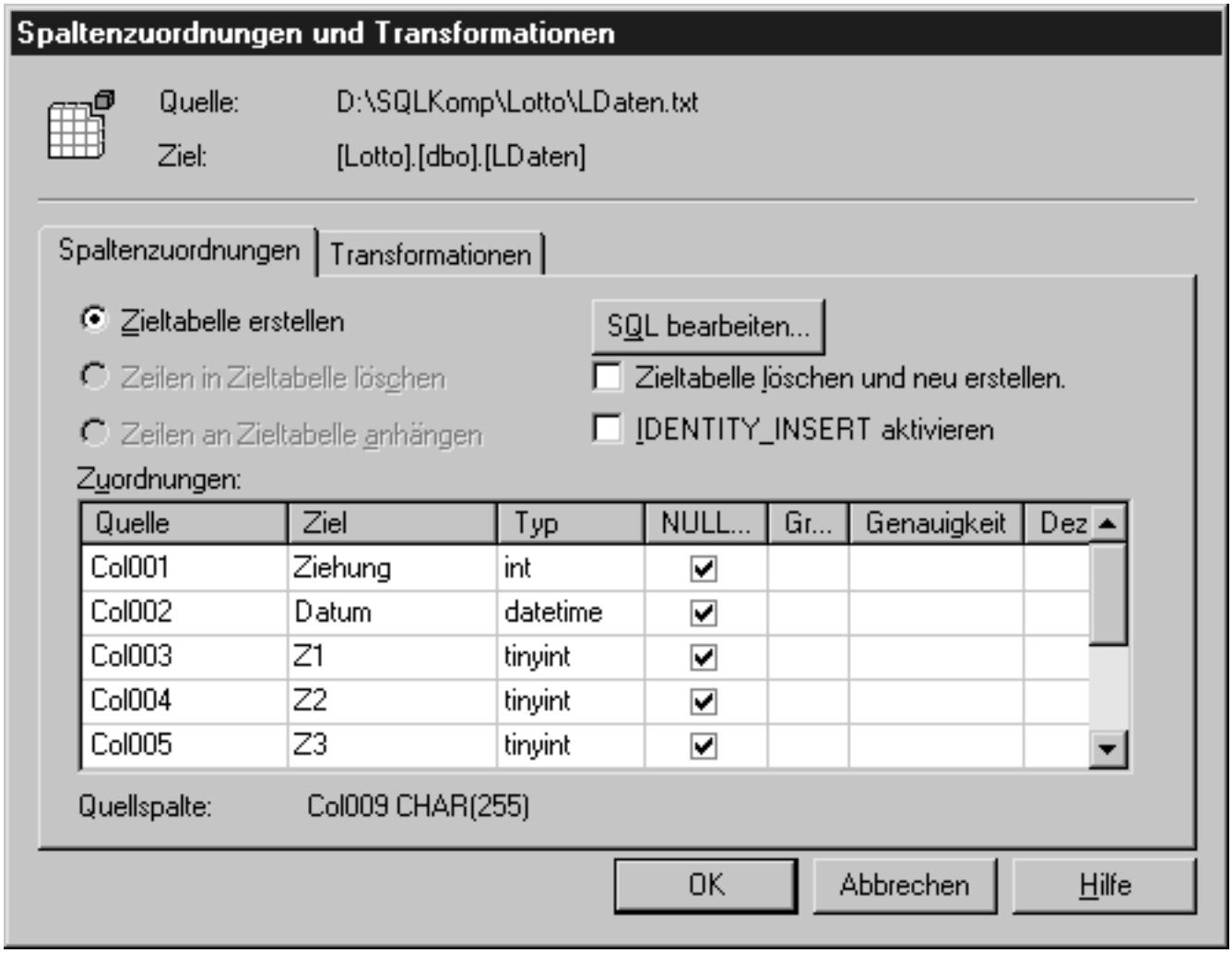
öffnen sie ein ms-dos-fenster, wechseln sie ins verzeichnis \mssql7\install (bzw. das von ihnen bei der installation festgelegte verzeichnis), und geben sie an der ms-dos-eingabeaufforderung den befehl

```
pubimage eins
```

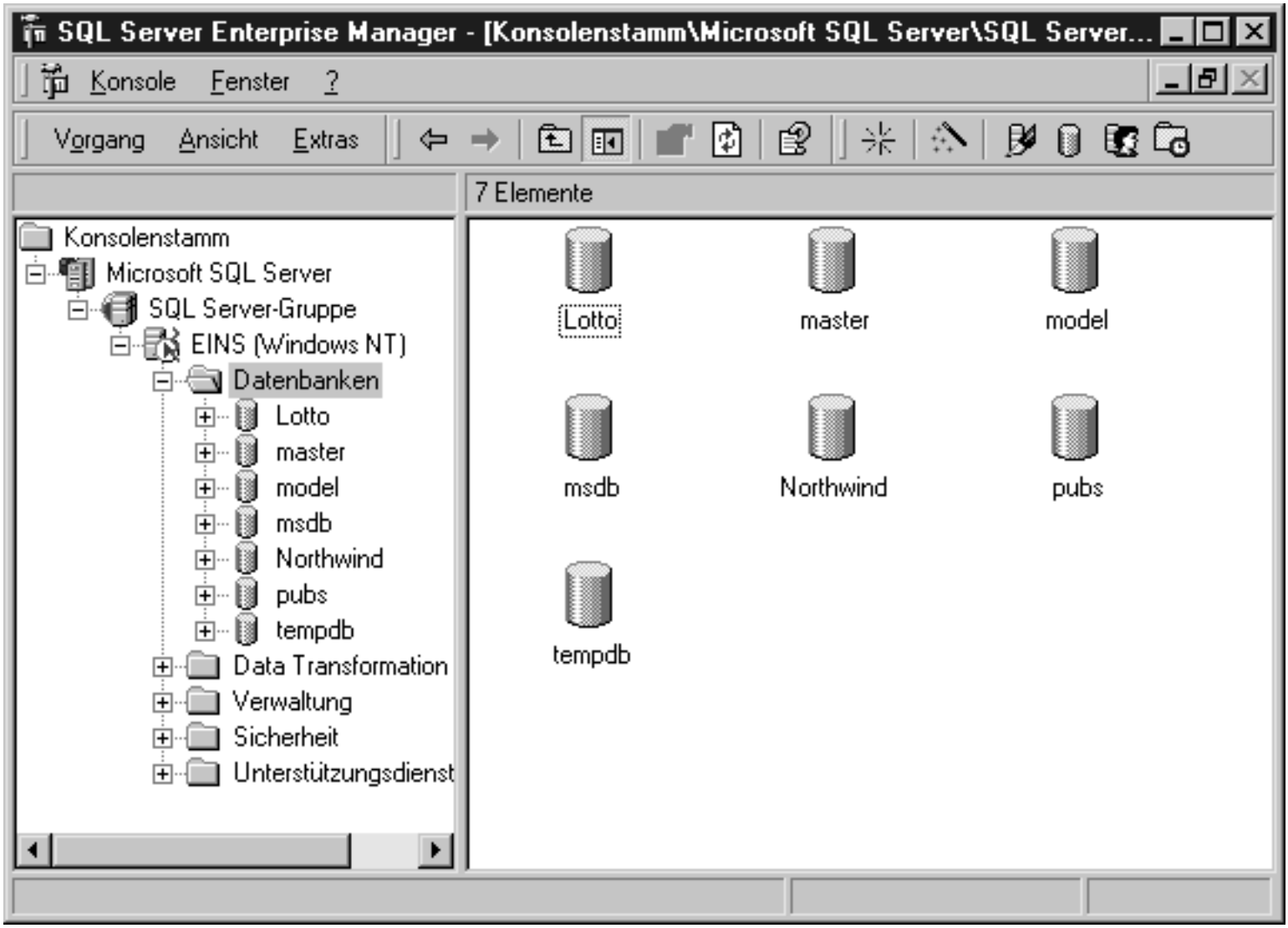
ein. den befehl können sie auch von einem remote-server ausführen.


DTS-Import-Assistent
Dateiformat auswählen
Bevor Daten importiert werden können, müssen Sie zuerst das Quelldateiformat bestätigen. Bitte bestätigen Sie, dass die Eigenschaften richtig ermittelt wurden, bevor Sie fortfahren.
Getrennt. Die Spalten sind durch beliebige Zeichen getrennt.
Feste Breite. Informationen sind in gleich breiten Spalten ausgerichtet.
Dateityp:
ANSI
Zeilen überspringen:
0
Zeilentrennzeichen:
{CR}{LF}
Erste Zeile enthält Spaltennamen
Textqualifizierer:
Anführungszeichen {"
Dateivorschau: D:\SQLKomp\Lotto\LDaten.txt
1,09.10.1955,3,12,13,16,23,41,0
2,16.10.1955,3,12,18,30,32,49,0
3,23.10.1955,12,14,23,24,34,36,0
4,30.10.1955,4,13,23,30,36,44,0
< Zurück
Weiter >
Abbrechen


DTS-Assistent
Ziel wählen
Wohin sollen die Daten kopiert werden. Sie können die Daten in ein beliebiges der unten aufgeführten Ziele kopieren. Wählen Sie eines der folgenden Ziele aus.
Ziel:
Microsoft OLE DB-Provider für SQL Server
Um eine Verbindung zu Microsoft SQL Server herzustellen, müssen Sie den Server, den Benutzernamen und das Kennwort angeben.
Server:
(local)
Windows NT-Authentifizierung verwenden
SQL Server-Authentifizierung verwenden
Benutzername:
sa
Kennwort:
Datenbank:
<Standard>
Aktualisieren
Erweitert...
<neu>
<Standard>
master
model
< Zurück
Weiter >
Abbrechen


Spaltenzuordnungen und Transformationen
Quelle: D:\SQLKomp\Lotto\LDaten.txt
Ziel: [Lotto].[dbo].[LDaten]
Spaltenzuordnungen
Transformationen
Zieltabelle erstellen
Zeilen in Zieltabelle löschen
Zeilen an Zieltabelle anhängen
SQL bearbeiten...
Zieltabelle löschen und neu erstellen.
IDENTITY_INSERT aktivieren
Zuordnungen:
Quelle
Ziel
Typ
NULL...
Gr...
Genauigkeit
Dez
Col001
Ziehung
int
Col002
Datum
datetime
Col003
Z1
tinyint
Col004
Z2
tinyint
Col005
Z3
tinyint
Quellspalte: Col009 CHAR(255)
OK
Abbrechen
Hilfe


DTS-Import-Assistent
DTS-Assistenten beenden
Sie haben alle Informationen angegeben, die erforderlich sind, um die Daten zu kopieren und zu transformieren. Überprüfen Sie die Auswahl, und klicken Sie dann auf 'Fertig stellen', um die Transformation zu starten.
Zusammenfassung:
Quelle: Flat File
Verwendet SQL Server DTS Flat File OLE DB Provider
Speicherort: D:\SQLKomp\Lotto\LDaten.txt
Ziel: Microsoft SQL Server
Verwendet Microsoft OLE DB Provider for SQL Server
Speicherort: (local)
Datenbank: Lotto
< Zurück
Fertig stellen
Abbrechen


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
7 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Lotto
master
model
msdb
Northwind
pubs
tempdb
Data Transformation
Verwaltung
Sicherheit
Unterstützungsdienst
Lotto
master
model
msdb
Northwind
pubs
tempdb

kapitel 10 daten abrufen und modifizieren

10.1 19, 20, 24, 29, 43, 45

wenn das ihr tip für die letzte samstagsziehung war und sie wissen wollen, wann sie damit etwas gewonnen hätten, können sie die ergebnisse aus der zum buch gehörenden beispieldatenbank lotto ermitteln. dazu müssen sie natürlich wissen, mit welchen befehlen sie auf die datenbank zugreifen und daten abrufen können. außerdem ist es erforderlich, daten einzufügen, zu aktualisieren und gegebenenfalls auch zu löschen. dieses kapitel zeigt, wie sie diese aufgaben mit den anweisungen von transact-sql lösen.

10.2 datenmanipulation

zu den häufigsten operationen in datenbanken gehört das abrufen, aktualisieren, einfügen und löschen von daten. transact-sql stellt dafür die anweisungen

- select
- update
- insert
- delete

bereit. das abrufen von daten bezeichnet man als *abfrage*. die anweisungen, mit denen sich daten abrufen, aktualisieren, einfügen und löschen lassen, gehören zur kategorie der datenmanipulationssprache (dml - data manipulation language).

10.3 daten abrufen (select)

die select-anweisung ist die wichtigste und am häufigsten gebrauchte anweisung überhaupt. die anweisung ruft zeilen aus der datenbank ab und erlaubt die auswahl einer oder mehrerer zeilen oder spalten aus einer oder mehreren tabellen. eine derart wichtige anweisung hat natürlich auch eine komplexe syntax. deshalb konzentrieren wir uns hier auf die wichtigsten klauseln der select-anweisung. die vollständige syntaxbeschreibung entnehmen sie bitte der online-dokumentation.

die syntax mit den wichtigsten elementen sieht folgendermaßen aus:

```
select auswahlliste
from tabellenliste
[where suchbedingungen]
```

```
[group by gruppenliste]
[having suchbedingungen]
[order by sortierliste [asc | desc]]
```

der logische aufbau einer select-anweisung ist nahezu intuitiv zu erfassen. select heißt auswählen. schon drängt sich die frage auf, *was* ausgewählt werden soll. die antwort darauf gibt man in der *auswahlliste* an.

wenn die auswahlliste lediglich konstanten, variablen oder arithmetische ausdrücke enthält und sich auf keine spalten einer tabelle bezieht, kann man sie fast wie eine print-anweisung von visual basic einsetzen:

```
select 15*3
```

liefert das ergebnis:

```
-----------
45
```

die groß-/kleinschreibung spielt bei transact-sql-anweisungen keine rolle. die anweisung select 15*3 können sie auch als select 15*3 oder select 15*3 (oder in komplett gemischter schreibung) eingeben.

wenn sie die beispiele mit dem dienstprogramm osql nachvollziehen, müssen sie nach den angegebenen anweisungen am beginn einer neuen zeile den ausführungsbefehl go eingeben. zusammen mit der eingabeaufforderung sehen die obigen anweisungszeilen dann wie folgt aus:

```
>1 select 15*3
>2 go
```

natürlich ist die select-anweisung nicht nur für derartig triviale aufgaben vorgesehen. normalerweise führt die auswahlliste die gewünschten spalten für die ergebnismenge auf. die nächste frage lautet also: *woher* sollen die spalten kommen? diese antwort gibt man in der *tabellenliste* an.

aktuelle datenbank festlegen

normalerweise geben sie nicht die datenbank an, auf die sich die anweisungen der datenmanipulation beziehen. wenn sie den namen der datenbank nicht angeben, gilt die aktuelle datenbank. nach der anmeldung eines benutzers bei sql server ist das in der regel die standarddatenbank. mit der transact-sql-anweisung use können sie die aktuelle datenbank jederzeit ändern:

```
use datenbankname
```

die folgenden beispiele setzen voraus, daß die verlegerdatenbank pubs als aktuelle datenbank aktiviert ist. führen sie dazu als erstes den befehl

```
use pubs
go
```

aus. die einstellung für die aktuelle datenbank bleibt so lange gültig, bis sie eine neue datenbank festlegen. im sql server query analyzer können sie die aktuelle datenbank auch über das drop-down-listenfeld **db** in der symbolleiste des abfragefensters auswählen (siehe abbildung 10.1).

bild

abbildung 10.1: auswahl der aktuellen datenbank in der symbolleiste des abfragefensters von query analyzer

einzelne spalten abrufen

die anweisung

```
select title_id from titles
```

ruft alle einträge der spalte title_id aus der tabelle titles ab und liefert die ergebnismenge:

```
title_id
--------
pc1035
ps1372
bu1111
...
mc3026
bu2075

(18 row(s) affected)
```

die anzeige für die anzahl der betroffenen zeilen (rows affected) am ende der ergebnismenge läßt sich mit der anweisung set nocount on unterdrücken.

mehrere spalten abrufen

um mehrere spalten abzurufen, gibt man sie durch komma getrennt in der auswahlliste an:

```
select title_id, type, price from titles
```

diese anweisung ruft die spalten title_id, type und price aus der tabelle titles ab. die ergebnismenge lautet:

```
title_id type         price
-------- ------------ ---------------------
bu1032   business     19.9900
bu1111   business     11.9500
bu2075   business     2.9900
bu7832   business     19.9900
...
ps3333   psychology   19.9900
ps7777   psychology   7.9900
tc3218   trad_cook    20.9500
tc4203   trad_cook    11.9500
tc7777   trad_cook    14.9900
```

alle spalten abrufen

mit einem sternchen in der auswahlliste rufen sie alle spalten einer tabelle ab:

```
select * from titleauthor
```

diese anweisung liefert alle spalten der tabelle titleauthor:

```
au_id       title_id au_ord royaltyper
----------- -------- ------ -----------
172-32-1176 ps3333   1      100
213-46-8915 bu1032   2      40
213-46-8915 bu2075   1      100
...
899-46-2035 mc3021   2      25
899-46-2035 ps2091   2      50
998-72-3567 ps2091   1      50
```

```
998-72-3567 ps2106   1      100
```

10.3.1 spaltenüberschriften ändern

in der ergebnismenge einer abfrage entsprechen die spaltenüberschriften den namen, die sie in der auswahlliste angegeben haben - d.h. den tatsächlichen spaltennamen, wie sie für die tabelle definiert sind. allerdings können sie auch eigene überschriften festlegen, indem sie aliasnamen für die spalten angeben. dabei haben sie die wahl zwischen der ansi-syntax und der transact-sql-syntax. nach der ansi-syntax hängt man den aliasnamen mit as an den spaltennamen an:

```
spaltenname as aliasname
```

die transact-sql-syntax sieht die form

```
aliasname = ausdruck
```

vor, wobei *ausdruck* ein spaltenname, eine konstante, eine funktion oder eine beliebige, durch operatoren verknüpfte kombination von spaltennamen, konstanten und funktionen oder eine unterabfrage sein kann.

beispiele:

die folgende abfrage verwendet die ansi-syntax, um die überschriften der spalten au_lname und au_fname mit deutschen bezeichnungen auszugeben:

```
use pubs
select au_lname as nachname, au_fname as vorname
from authors
```

das ergebnis lautet:

```
nachname                                 vorname
---------------------------------------- --------------------
bennet                                   abraham
blotchet-halls                           reginald
carson                                   cheryl
...                                      ...
straight                                 dean
stringer                                 dirk
```

```
white                                    johnson
yokomoto                                 akiko
```

die gleiche abfrage läßt sich mit der transact-sql-syntax folgendermaßen formulieren:

```
select nachname = au_lname, vorname = au_fname
from authors
```

beachten sie, daß hier die aliasnamen vor den spaltennamen stehen.

10.3.2 spaltenbreite anpassen

vor allem textspalten sind oftmals so ausgelegt, daß sie für die längste zu erwartende zeichenfolge mehr als ausreichend sind. ruft man mehrere spalten ab, führt der automatische zeilenumbruch in der ausgabe zu fast unleserlichen ergebnissen - zumindest, wenn man mit dem dienstprogramm osql arbeitet oder die ergebnisse mit sql server query im textformat ausgibt.

mit der funktion convert lassen sich die ausgaben in einer übersichtlicheren form darstellen. auf die funktion convert geht kapitel 11 näher ein.

```
select convert(varchar(15),au_lname) as nachname,
       convert(varchar(12),au_fname) as vorname,
       convert(varchar(15),phone) as telefon
from authors


nachname        vorname      telefon
--------------- ------------ ---------------
white           johnson      408 496-7223
green           marjorie     415 986-7020
carson          cheryl       415 548-7723
...             ...          ...
ringer          anne         801 826-0752
ringer          albert       801 826-0752
```

10.3.3 bedingte auswahl (where)

die ergebnismenge in den obigen beispielen hat immer alle werte der abgerufenen spalten - d.h. alle zeilen - geliefert. bei kleinen tabellen ist das noch problemlos durchführbar. wenn in den tabellen aber mehrere tausend datensätze gespeichert sind, muß man die zeilen gezielt abrufen können. die

select-anweisung bietet zu diesem zweck die where-klausel, in der sich suchbedingungen formulieren lassen. damit wirkt die where-klausel als filter für die abzurufenden zeilen.

syntax:

die syntax der select-anweisung mit where-klausel sieht folgendermaßen aus:

```
select auswahlliste
from tabellenliste
where suchbedingungen
```

die suchbedingungen haben den aufbau

```
[not] ausdruck1 operator ausdruck2
[{ and | or } [not] ausdruck3 operator ausdruck4]
```

in den meisten fällen stellt *ausdruck1* einen spaltennamen dar. man kann aber auch arithmetische ausdrücke angeben.

vergleichsoperatoren

die beiden ausdrücke einer suchbedingung verknüpft man mit einem vergleichsoperator nach tabelle 10.1.

operator	beschreibung
=	gleich
<>	ungleich
!=	ungleich
>	größer als
>=	größer oder gleich
!>	nicht größer als
<	kleiner als
<=	kleiner oder gleich
!<	nicht kleiner als

tabelle 10.1: vergleichsoperatoren in einer where-klausel

abhängig von der einstellung ansi_nulls ergeben die vergleichsoperatoren das ergebnis true, false und

unknown, wenn im vergleich ein null-wert beteiligt ist. diese dreiwertige logik kann zu schwer auffindbaren fehlern führen. hinweise zu ansi_nulls finden sie weiter unten in diesem kapitel bei der behandlung der null-werte sowie in kapitel 5 im abschnitt zu den vergleichsoperatoren.

beispiele:

das erste beispiel wählt die spalten title_id und price aus der tabelle titles der datenbank pubs für diejenigen titel aus, die weniger als $ 10 kosten (10 nicht eingeschlossen):

```
select title_id, price from titles
where price<10
```

das ergebnis lautet:

```
title_id price
-------- ---------------------
bu2075   2.99
mc3021   2.99
ps2106   7.00
ps7777   7.99
```

logische operatoren (not, and, or)

mit den logischen operatoren not, and und or lassen sich mehrere suchbedingungen miteinander verknüpfen:

- not negiert den ausdruck und liefert false, wenn der ausdruck true liefert, und true, wenn der ausdruck false ergibt.
- and verknüpft zwei ausdrücke und liefert true, wenn beide ausdrücke true ergeben.
- or verknüpft zwei ausdrücke und gibt true zurück, wenn mindestens einer der ausdrücke das ergebnis true liefert.

beispiele:

die anweisung

```
select au_id, royaltyper from titleauthor
where royaltyper > 30 and royaltyper < 75
```

ermittelt die autoren-id und den prozentualen anteil der tantiemen aus der tabelle titleauthor der datenbank pubs. die where-klausel enthält die bedingung, daß die tantiemen größer als 30 prozent *und*

kleiner als 75 prozent sein sollen. das ergebnis lautet:

```
au_id       royaltyper
----------- -----------
213-46-8915 40
267-41-2394 40
409-56-7008 60
427-17-2319 50
672-71-3249 40
724-80-9391 60
846-92-7186 50
899-46-2035 50
998-72-3567 50
```

wollen sie die anteile der tantiemen ermitteln, die kleiner gleich 30 prozent oder größer gleich 75 prozent sind, nehmen sie die gegenstücke der vergleichsoperatoren und formulieren die verknüpfung mit dem or-operator:

```
select au_id, royaltyper from titleauthor
where royaltyper <= 30 or royaltyper >= 75
```

die ergebnismenge lautet:

```
au_id       royaltyper
----------- -----------
172-32-1176 100
213-46-8915 100
238-95-7766 100
267-41-2394 30
274-80-9391 100
472-27-2349 30
486-29-1786 100
486-29-1786 100
648-92-1872 100
712-45-1867 100
722-51-5454 75
724-80-9391 25
756-30-7391 75
807-91-6654 100
899-46-2035 25
998-72-3567 100
```

bereiche (between)

mit einer bereichsbedingung in der where-klausel lassen sich alle zeilen abrufen, deren spaltenwerte in einem bestimmten bereich liegen.

syntax:

```
[not] between untergrenze and obergrenze
```

diese bedingung testet, ob ein wert im bereich zwischen *untergrenze* oder *obergrenze* (jeweils einschließlich) liegt. and ist hier kein logischer operator, sondern ein platzhalter, der die beiden grenzwerte voneinander trennt. not between testet, ob der wert kleiner als *untergrenze* oder größer als *obergrenze* ist, d.h. nicht im bereich liegt.

das nächste beispiel sucht mit einer bereichsbedingung in der where-klausel die autoren-ids und tantiemen mit einem prozentsatz zwischen 30 und 75 prozent für den autor heraus:

```
select au_id, royaltyper from titleauthor
where royaltyper between 30 and 75
```

als ergebnismenge erhält man:

```
au_id       royaltyper
----------- -----------
213-46-8915 40
267-41-2394 40
267-41-2394 30
409-56-7008 60
427-17-2319 50
472-27-2349 30
672-71-3249 40
722-51-5454 75
724-80-9391 60
756-30-7391 75
846-92-7186 50
899-46-2035 50
998-72-3567 50
```

beachten sie den unterschied zum letzten beispiel, das die vergleichsoperatoren verwendet hat. es ist zu erkennen, daß die bereichsgrenzen in einer between-bedingung eingeschlossen sind. das nächste beispiel zeigt die anteile der tantiemen an, die außerhalb des bereichs 30 bis 75 prozent liegen:

```
select au_id, royaltyper from titleauthor
where royaltyper not between 30 and 75
```

die bedingung not between schließt die bereichsgrenzen nicht ein, die prozentwerte 30 und 75 erscheinen demnach nicht in der ergebnismenge:

```
au_id       royaltyper
----------- -----------
172-32-1176 100
213-46-8915 100
238-95-7766 100
274-80-9391 100
486-29-1786 100
486-29-1786 100
648-92-1872 100
712-45-1867 100
724-80-9391 25
807-91-6654 100
899-46-2035 25
998-72-3567 100
```

wenn die ergebnismenge den bereich zwischen untergrenze und obergrenze einschließen, aber nicht die werte der grenzen enthalten soll, müssen sie mit den vergleichsoperatoren arbeiten und die bedingungen mit and verknüpfen.

listen (in)

die bedingung in der where-klausel läßt sich auch in form einer liste formulieren.

syntax:

```
testausdruck [not] in {ausdruck | unterabfrage}
```

die bedingung ist erfüllt, wenn der *testausdruck* im *ausdruck* bzw. im ergebnis der *unterabfrage* enthalten ist.

wenn der ausdruck zeichentypen oder datumstypen enthält, sind diese in einfache anführungszeichen einzuschließen.

beispiele:

die folgende anweisung ruft aus der tabelle sales der datenbank pubs die zeilen ab, die in der spalte stor_id den wert 6380 oder 7066 enthalten. die spalte stor_id ist vom typ char (mit einer länge von 4 zeichen) definiert. deshalb sind die werte in der liste in einfachen anführungszeichen anzugeben:

```
select stor_id, title_id from sales
where stor_id in ('6380', '7066')
```

das ergebnis lautet:

```
stor_id title_id
------- --------
6380    bu1032
6380    ps2091
7066    pc8888
7066    ps2091
```

das nächste beispiel zeigt, wie sich kombinationen von werten in verschiedenen spalten auf bequeme weise mit listen in der where-klausel ermitteln lassen:

```
use lotto
select ziehung, z1,z2,z3,z4,z5,z6,zz from ldaten
where 6 in (z1,z2,z3,z4,z5,z6) and
      10 in (z1,z2,z3,z4,z5,z6) and
      19 in (z1,z2,z3,z4,z5,z6) and
      43 in (z1,z2,z3,z4,z5,z6)
```

da nicht bekannt ist, in welcher spalte die zu vergleichenden zahlen stehen, läßt man sie in allen spalten suchen und verknüpft die einzelergebnisse mit dem logischen operator and. die anweisung ermittelt also alle ziehungen, in denen die zahlenkombination 6, 10, 19, 43 gezogen wurde. das ergebnis lautet:

```
ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
254         6    10   11   12   19   43   20
1425        6    10   19   30   43   47   14
```

null-werte

einen test auf null-werte kann man in der where-klausel nach zwei methoden formulieren:

- mit dem schlüsselwort is:

```
select price from titles
where price is not null
```

- mit vergleichsoperatoren:

```
select price from titles
where price <> null
```

die erste methode ist immer gültig, während die zweite von der einstellung ansi_nulls abhängig ist. in der voreinstellung ist ansi_nulls auf on gesetzt. ein vergleich liefert damit nicht das ergebnis true oder false, sondern unknown. das obige beispiel führt mit dieser einstellung bei verwendung der vergleichsoperatoren zu einer leeren ergebnismenge.

damit auch die vergleichsoperatoren beim test auf null-werte die logischen ergebnisse true bzw. false zurückgeben, ist ansi_nulls auf off zu setzen:

```
set ansi_nulls off
```

mustervergleich (like)

will man nur diejenigen zeilen auswählen, in denen eine zeichenfolge ein bestimmtes muster enthält, kann man in der where-klausel eine bedingung mit dem operator like formulieren. eine vergleichbare funktionalität ließe sich auch mit den operatoren = und != erreichen, like bietet aber den vorteil, daß das muster platzhalterzeichen enthalten kann.

die allgemeine syntax lautet:

```
vergleichsausdruck [not] like muster [escape escapezeichen]
```

der *vergleichsausdruck* muß einen zeichenfolgentyp liefern. es handelt sich um die zeichenfolge, in der nach dem muster zu suchen ist. das *muster* ist die zeichenfolge, die im *vergleichsausdruck* entsprechend der bedingung enthalten (like) oder nicht enthalten (not like) sein soll. tabelle 10.2 zeigt die

platzhalterzeichen, die man im *muster* angeben kann.

zeichen	beschreibung
%	eine zeichenfolge mit null oder mehr zeichen
_	ein einzelnes zeichen
[]	beliebiges einzelnes zeichen im angegebenen bereich ([a-e]) oder in der angegebenen zeichenmenge ([abcde])
[^]	beliebiges einzelzeichen, das nicht im angegebenen bereich ([^a-e]) oder der angegebenen zeichenmenge ([^abcde]) enthalten sein darf

tabelle 10.2: platzhalterzeichen in like-bedingungen

mit der escape-klausel läßt sich ein platzhalterzeichen als normales suchzeichen spezifizieren. dazu definieren sie nach dem schlüsselwort escape ein sogenanntes *escapezeichen* - ein zeichen, das beim vergleich als nicht existent behandelt wird - und schreiben es im muster vor das umfunktionierte platzhalterzeichen.

beispiele:

die erste anweisung sucht in der datenbank pubs alle titel heraus, die die zeichenfolge comp (für computer) an beliebiger stelle enthalten:

```
use pubs
select title_id, title from titles
where title like '%comp%'
```

das ergebnis lautet:

```
title_id title
-------- --------------------------------------------------
ps1372   computer phobic and non-phobic individuals: behavior v
bu1111   cooking with computers: surreptitious balance sheets
bu7832   straight talk about computers
mc3026   the psychology of computer cooking
bu2075   you can combat computer stress!

(5 row(s) affected)
```

wer nichts von computern wissen will, dreht die bedingung in der where-klausel einfach um:

```
use pubs
select title_id, title from titles
where title not like '%comp%'
```

und erhält damit alle anderen titel:

```
title_id title
-------- ----------------------------------------------------
pc1035   but is it user friendly?
ps7777   emotional security: a new algorithm
tc4203   fifty years in buckingham palace kitchens
ps2091   is anger the enemy?
ps2106   life without fear
pc9999   net etiquette
tc3218   onions, leeks, and garlic: cooking secrets of the medi
ps3333   prolonged data deprivation: four case studies
pc8888   secrets of silicon valley
mc2222   silicon valley gastronomic treats
tc7777   sushi, anyone?
bu1032   the busy executive's database guide
mc3021   the gourmet microwave

(13 row(s) affected)
```

das nächste beispiel sucht alle titel heraus, die mit den buchstaben a, b, c, d oder e beginnen:

```
use pubs
select title_id, title from titles
where title like '[a-e]%'
```

als ergebnis erhält man:

```
title_id title
-------- ----------------------------------------------------
pc1035   but is it user friendly?
ps1372   computer phobic and non-phobic individuals: behavior v
bu1111   cooking with computers: surreptitious balance sheets
ps7777   emotional security: a new algorithm

(4 row(s) affected)
```

das folgende beispiel zeigt die verwendung der escape-klausel. das escapezeichen ist ein ausrufezeichen. im muster steht das ausrufezeichen vor einem prozentzeichen, das damit nicht mehr als platzhalter fungiert, sondern als reales zeichen (nämlich als %) in den mustervergleich eingeht:

```
use master
select error, description from sysmessages
where error<104 and description like '%!%%' escape '!'
```

diese anweisung sucht in der systemtabelle sysmessages alle fehlermeldungen mit einer fehlernummer kleiner 104, die ein prozentzeichen enthalten.

das prozentzeichen dient in den fehlermeldungen der systemtabelle sysmessages selbst als platzhalterzeichen (siehe dazu kapitel 19). das beispiel sucht also nach den fehlermeldungen, die platzhalterzeichen enthalten.

die ergebnismenge sieht wie folgt aus (aus platzgründen rechts abgeschnitten):

```
error       description
----------- --------------------------------------------------
21          warnung: fataler fehler %1! am %2! aufgetreten. not
21          warning: fatal error %d occurred at %s_date. note t
102         falsche syntax in der nähe von '%1!'.
102         incorrect syntax near '%.*ls'.
103         die mit %1! beginnende '%2!' ist zu lang. die maxim
103         the %s_msg that starts with '%.*ls' is too long. ma

(6 row(s) affected)
```

kommentieren sie nun die escape-klausel aus

```
use master
select error, description from sysmessages
where error<104 and description like '%!%%' --escape '!'
```

und führen sie die abfrage erneut aus. das ergebnis lautet jetzt:

```
error       description
----------- --------------------------------------------------
```

```
21          warnung: fataler fehler %1! am %2! aufgetreten. not
102         falsche syntax in der nähe von '%1!'.
103         die mit %1! beginnende '%2!' ist zu lang. die maxim

(3 row(s) affected)
```

beachten sie den feinen unterschied. die anweisung im ersten beispiel sucht ein prozentzeichen, das an beliebiger stelle in der spalte description vorkommt. im zweiten beispiel wird dagegen nach einem ausrufezeichen gesucht, das an beliebiger stelle steht, weil das ausrufezeichen für die musterzeichenfolge nicht als escapezeichen deklariert ist. das dritte prozentzeichen ist in diesem fall sogar überflüssig.

10.3.4 duplikate ausblenden (distinct)

normalerweise liefert eine abfrage alle zeilen, die einer angegebenen bedingung entsprechen - also auch mehrfach vorkommende spaltenwerte. wenn sie zum beispiel die anweisung

```
use pubs
select pub_id from employee
```

ausführen, enthält die ergebnismenge 43 zeilen, in denen mehrere werte mehrfach vorkommen. hier ein ausschnitt:

```
pub_id
------
0877
1389
0877
0877
0877
9952
9952
1389
...
0736
0736
0877

(43 row(s) affected)
```

wenn sie lediglich an einer liste bereits vergebener verleger-ids interessiert sind, können sie die mehrfach

vorkommenden werte mit dem schlüsselwort distinct in der select-anweisung ausblenden. der betreffende teil der select-syntax sieht folgendermaßen aus:

```
select distinct auswahlliste
from tabellenliste
where suchbedingungen
```

die eingangs gezeigte anweisung formulieren sie mit dem schlüsselwort distinct wie folgt:

```
use pubs
select distinct pub_id from employee
```

als ergebnismenge erhalten sie:

```
pub_id
------
0736
0877
1389
1622
1756
9901
9952
9999

(8 row(s) affected)
```

obendrein ist die liste noch sortiert (abhängig von der installierten sortierreihenfolge und einer eventuell angegebenen order by-klausel). wenn die ausgewählte spalte null-werte enthält, liefert die ergebnismenge genau einen null-wert zurück.

in der auswahlliste können sie auch mehrere spalten angeben. das schlüsselwort distinct bezieht sich dann auf die (zeilenweise) kombination der spaltenwerte, liefert für mehrfache kombinationen nur eine zeile zurück und sortiert nach der zuerst genannten spalte.

die folgende anweisung liefert für alle eindeutigen kombinationen der spalten pub_id und job_lvl genau eine zeile zurück:

```
use pubs
select distinct pub_id, job_lvl from employee
```

das ergebnis lautet:

```
pub_id job_lvl
------ -------
0736   100
0736   150
0736   165
0736   170
...
9999   152
9999   170
9999   211

(36 row(s) affected)
```

kombination bedeutet hier, daß für jede zeile aus den werten der angegebenen spalten *ein* zusammengefaßter wert gebildet wird. es handelt sich also nicht um das kartesische produkt der spalten, das jede spalte mit jeder spalte kombiniert. auf das kartesische produkt geht der abschnitt zur verknüpfung von tabellen weiter unten in diesem kapitel ein.

10.3.5 daten sortieren (order by)

wenn sie die ergebnismenge sortieren wollen, können sie das mit einer order by-klausel realisieren. der relevante teil der select-syntax sieht folgendermaßen aus:

```
select auswahlliste
from tabellenliste
order by sortierliste [asc | desc]
```

in der order by-klausel können sie die sortierrichtung mit den schlüsselwörtern asc für aufsteigend (englisch: ascending) oder desc für absteigend (englisch: descending) spezifizieren. die standardeinstellung ist aufsteigend.

wenn die spalten der sortierliste auch in der auswahlliste erscheinen, können sie in der sortierliste die nummern der spalten entsprechend der reihenfolge in der auswahlliste angeben.

die anweisung im folgenden beispiel ruft die spalten pub_id, fname und lname aus der tabelle employee der datenbank pubs ab und sortiert die ergebnisse nach den spalten pub_id und lname:

```
use pubs
```

```
select pub_id, fname, lname from employee
order by pub_id, lname
```

als ergebnis erhalten sie:

```
pub_id fname                lname
------ -------------------- ------------------------------
0736   palle                ibsen
0736   karin                josephs
0736   matti                karttunen
0736   laurence             lebihan
0736   roland               mendel
0736   timothy              o'rourke
0736   mary                 saveley
...
9952   francisco            chang
9952   philip               cramer
9952   ann                  devon
9999   carlos               hernadez
9999   karla                jablonski
9999   pirkko               koskitalo
9999   patricia             mckenna
9999   helvetius            nagy
9999   manuel               pereira
9999   annette              roulet

(43 row(s) affected)
```

die sortierung erfolgt in aufsteigender richtung zuerst nach pub_id und innerhalb der zu einer pub_id gehörenden zeilen nach der spalte lname (verschachteltes sortieren).

da die in der order by-klausel aufgeführten spalten in der auswahlliste enthalten sind, können sie die anweisung auch wie folgt formulieren:

```
use pubs
select pub_id, fname, lname from employee
order by 1, 3
```

das ergebnis ist das gleiche. nachteilig ist allerdings, daß sie für die nummern in der sortierliste der order by-klausel keine variablen verwenden können.

diese nummern haben nichts mit der reihenfolge der spalten in der definition der tabelle zu tun.

10.3.6 gruppieren (group by und having)

mit der klausel group by lassen sich die ergebnisse einer abfrage in gruppen einteilen. oftmals interessieren nur summen, maximal- oder minimalwerte. der betreffende syntaxabschnitt der select-anweisung lautet:

```
select auswahlliste
from tabellenliste
where suchbedingungen
[group by gruppenliste]
[having suchbedingungen]
```

die having-klausel erfüllt auf gruppenebene die gleiche aufgabe wie die where-klausel für die gesamte auswahlliste der select-anweisung und legt eine suchbedingung für eine gruppe oder ein aggregat fest. normalerweise verwendet man die having-klausel in verbindung mit einer group by-klausel. ist group by nicht vorhanden, verhält sich having wie eine where-klausel.

das folgende beispiel verwendet die tabelle sales der datenbank pubs. damit sie die funktionsweise von group by nachvollziehen können, rufen sie zunächst die mengen der einzelnen titel für jede buchhandlung aus der tabelle ab:

```
use pubs
select stor_id as buchhdlg, qty as menge, title_id from sales
```

das ergebnis lautet:

```
buchhdlg menge  title_id
-------- ------ --------
6380     5      bu1032
6380     3      ps2091
7066     50     pc8888
7066     75     ps2091
7067     10     ps2091
7067     40     tc3218
7067     20     tc4203
7067     20     tc7777
7131     20     ps2091
7131     25     mc3021
7131     20     ps1372
7131     25     ps2106
```

```
7131     15     ps3333
7131     25     ps7777
7896     15     bu7832
7896     10     mc2222
7896     35     bu2075
8042     15     mc3021
8042     10     bu1032
8042     25     bu1111
8042     30     pc1035
```

wenn sie jetzt lediglich die gesamtmengen der einzelnen buchhandlungen wissen wollen, gruppieren sie die abfrage nach der spalte stor_id (kennummer der buchhandlung) und geben in der auswahlliste nicht einfach die spalte qty (menge), sondern die summe dieser spalte an:

```
use pubs
select stor_id as buchhandlung, sum(qty) as gesamt from sales
group by stor_id
```

die aggregatfunktion sum und andere funktionen behandelt kapitel 11.

das ergebnis lautet:

```
buchhandlung gesamt
------------ -----------
6380         8
7066         125
7067         90
7131         130
7896         60
8042         80
```

mit einer having-klausel können sie die ergebnismenge einschränken und zum beispiel nur die buchhandlungen mit einer gesamtmenge > 10 anzeigen:

```
use pubs
select stor_id as buchhandlung, sum(qty) as gesamt  from sales
group by stor_id
having sum(qty) > 10
```

als ergebnis erhalten sie:

```
buchhandlung gesamt
------------ -----------
7066         125
7067         90
7131         130
7896         60
8042         80
```

10.3.7 zusammenfassen (compute by)

die klausel compute by erlaubt es, zusätzliche ergebniszeilen zu bilden, die zum beispiel teilsummen oder gesamtsummen anzeigen. im gegensatz zur group by-klausel, die ebenfalls zusammenfassungen bildet, gibt die compute by-klausel untergruppen mit zusammenfassungswerten zurück.

die syntax der select-anweisung unter berücksichtigung der compute by-klausel sieht folgendermaßen aus:

```
select auswahlliste
from tabellenliste
[where suchbedingungen]
[order by sortierliste [asc | desc]]
[compute aggregatausdrücke
[by] spaltenliste]
```

wenn sie die compute by-klausel verwenden, müssen sie auch die order by-klausel angeben. in der *spaltenliste* der compute by-klausel sind nur spalten zulässig, die auch in der *sortierliste* der order by-klausel erscheinen. außerdem muß die reihenfolge der spalten in der order by- und der compute by-klausel übereinstimmen, und spalten dürfen nicht übersprungen werden.

die klausel compute by ist ein exot unter den klauseln der select-anweisung, da sie zusätzliche zeilen liefert und damit nicht dem relationalen modell entspricht. deshalb können sie diese klausel nicht in verbindung mit select into (siehe weiter unten in diesem kapitel) verwenden, um neue tabellen zu erzeugen.

das folgende beispiel ist dem ersten beispiel für die group by-klausel ähnlich, führt aber die mengen für die titel einzeln auf und liefert jeweils eine zusätzliche zeile für die teilsumme der mengen pro buchhandlung:

```
use pubs
select stor_id as buchhandlung, qty as menge, title_id as titel
```

```
from sales
order by stor_id
compute sum(qty) by stor_id
```

das ergebnis sieht folgendermaßen aus:

```
buchhandlung menge  titel
------------ ------ ------
6380         5      bu1032
6380         3      ps2091

             sum
             ==========
             8


7066         50     pc8888
7066         75     ps2091

             sum
             ==========
             125


7067         10     ps2091
7067         40     tc3218
7067         20     tc4203
7067         20     tc7777

             sum
             ==========
             90


7131         20     ps2091
7131         25     mc3021
7131         20     ps1372
7131         25     ps2106
7131         15     ps3333
7131         25     ps7777

             sum
             ==========
             130
```

```
7896          15     bu7832
7896          10     mc2222
7896          35     bu2075

              sum
              ==========
              60


8042          15     mc3021
8042          10     bu1032
8042          25     bu1111
8042          30     pc1035

              sum
              ==========
              80
```

10.4 tabellen verknüpfen

in den seltensten fällen besteht eine datenbank aus nur einer tabelle. schon aus gründen der normalisierung teilt man eine datenbank in mehrere tabellen auf. zu den grundlegenden eigenschaften des relationalen modells gehört es, mehrere tabellen miteinander verknüpfen zu können. das ergebnis einer verknüpfung ist wiederum eine tabelle, die alle spalten der in der select-anweisung aufgeführten auswahlliste enthält und deren zeilen den spezifizierten suchbedingungen entsprechen.

es gibt folgende arten von verknüpfungen (englisch: joins):

- innere verknüpfungen: vergleichen die werte in den verknüpften spalten mit vergleichsoperatoren. es werden alle zeilen *ausgeschlossen*, die nicht mit einer zeile aus der anderen tabelle übereinstimmen - daher auch die bezeichnung *exklusionsverknüpfung*. zu den inneren verknüpfungen gehören gleichheitsverknüpfungen und natürliche verknüpfungen.
- äußere verknüpfungen: liefern alle zeilen der linken (rechten) tabelle bei einer linken (rechten) äußeren verknüpfung und nur die zeilen der rechten (linken) tabelle, die der bedingung entsprechen. bei einer vollständigen äußeren verknüpfung enthält die ergebnismenge alle zeilen der linken und rechten tabellen. äußere verknüpfungen schließen zeilen ein - daher die bezeichnung *inklusionsverknüpfung*.
- kreuzverknüpfungen: kombinieren alle zeilen der linken tabelle mit allen zeilen der rechten tabelle und geben das sogenannte *kartesische produkt* aus beiden tabellen zurück.

verknüpfungen lassen sich auch als filter auffassen. in dieser filterkonstruktion wird zuerst die from-klausel, dann die where-klausel und schließlich die having-klausel abgearbeitet.

aliasnamen für tabellen

da verknüpfungen immer auf spalten in zwei tabellen verweisen, müssen diese spalten entweder eindeutige namen tragen oder durch einen präfix unterschieden werden. als präfix können sie einen aliasnamen (bei selbstverknüpfungen zwingend erforderlich) oder den vollständigen namen der tabelle verwenden. aliasnamen legen sie in der from-klausel wie folgt fest:

```
from tabellenname as aliasname
```

das schlüsselwort as ist optional. den bezug auf eine spalte in der select-anweisung formulieren sie dann beispielsweise so:

```
select s.stor_id, t.stor_name, s.qty
from sales s, stores t
where s.stor_id = t.stor_id
```

dabei gilt die regel: einmal alias - immer alias. es ist nicht möglich, an der einen stelle einen vollständigen tabellennamen und in derselben anweisung an einer anderen stelle einen alias für die tabelle zu verwenden.

10.4.1 syntaxformen für verknüpfungen

sql server unterscheidet zwei syntaxformen: die ansi-syntax (sql-92) und die sql-server-syntax. die syntax der select-anweisung für verknüpfungen sieht folgendermaßen aus:

sql-server-syntax:

```
select tabellenname.spaltenname [,...n]
from tabellenliste
where { tabellenname.spaltenname
        verknüpfungsoperator
        tabellenname.spaltenname } [,...n]
```

in der from-klausel sind die tabellen aufgeführt, die in die verknüpfung einzubeziehen sind. die verknüpfung steht in der where-klausel. der *verknüpfungsoperator* kann =, >, <, >=, <=, <>, !> und !< sein. außerdem können in der where-klausel zusätzliche bedingungen für die zurückzugebenden zeilen angegeben werden.

ansi-syntax:

```
select tabellenname.spaltenname [,...n]
from {tabellenname [verknüpfungstyp] join tabellenname
on suchkriterium} [,...n]
where suchkriterium
```

die verknüpfung steht in der from-klausel, und die where-klausel spezifiziert die zeilen, die aus den verknüpften tabellen zurückzugeben sind. als *verknüpfungstyp* sind inner, outer und cross möglich.

10.4.2 innere verknüpfungen

die werte in den verknüpften spalten werden bei einer inneren verknüpfung mit den vergleichsoperatoren verglichen. in der allgemeinen schreibweise der verknüpfungsbedingung setzt man oftmals stellvertretend für die operatoren den griechischen buchstaben theta und spricht deshalb auch von *theta-verknüpfungen*. der sonderfall einer theta-verknüpfung ist die gleichheitsverknüpfung, bei der für das theta ein gleichheitszeichen steht.

die ergebnismenge bei einer *gleichheitsverknüpfung* liefert alle spalten der beiden tabellen und enthält alle zeilen, die in den verknüpfungsspalten gleiche werte aufweisen:

```
/* sql server-syntax: */
select *
from sales, stores
where sales.stor_id = stores.stor_id

/* ansi-syntax: */
select *
from sales inner join stores
on sales.stor_id = stores.stor_id
```

diese anweisung ruft alle spalten aus der tabelle sales und alle spalten aus der tabelle stores ab, die übereinstimmende werte in der spalte stor_id aufweisen. beide tabellen haben sechs spalten, die ergebnismenge enthält demnach zwölf spalten, wobei die spalte stor_id doppelt erscheint.

da die verknüpfungsspalte sowohl in der linken als auch der rechten tabelle vorhanden ist, taucht sie in der ergebnismenge zweimal auf. mit einer *natürlichen verknüpfung* läßt sich die überflüssige doppelte spalte aus der ergebnismenge ausblenden. eine spezielle kennzeichnung für natürliche verknüpfungen gibt es nicht. man gibt einfach die auszuwählenden spalten an:

```
/* sql server-syntax: */
select s.stor_id, s.ord_num, t.stor_name
from sales s, stores t
```

```
where s.stor_id = t.stor_id

/* ansi-syntax: */
select s.stor_id, s.ord_num, t.stor_name
from sales as s inner join stores as t
on s.stor_id = t.stor_id
```

diese anweisung ruft die spalten stor_id und ord_num aus der tabelle sales und den namen der buchhandlung (stor_name) aus der tabelle stores ab. das ergebnis lautet:

```
stor_id ord_num              stor_name
------- -------------------- ----------------------------------
6380    6871                 eric the read books
6380    722a                 eric the read books
7066    a2976                barnum's
7066    qa7442.3             barnum's
7067    d4482                news & brews
7067    p2121                news & brews
...
7896    x999                 fricative bookshop
8042    423ll922             bookbeat
8042    423ll930             bookbeat
8042    p723                 bookbeat
8042    qa879.1              bookbeat

(21 row(s) affected)
```

10.4.3 äußere verknüpfungen

mit äußeren verknüpfungen kann man alle zeilen aus der einen tabelle übernehmen, während man für die andere tabelle bedingungen spezifiziert und so die ergebnismenge aus dieser tabelle einschränkt.

zu den äußeren verknüpfungen gehören:

- left join oder left outer join (linke äußere verknüpfung)
- right join oder right outer join (rechte äußere verknüpfung)
- full join oder full outer join (vollständige äußere verknüpfung)

die angegebenen schlüsselwörter sind für die ansi-syntax gültig. die - ältere - sql-server-syntax sieht die operatoren *= für die linke äußere verknüpfung und =* für die rechte äußere verknüpfung in der where-klausel vor. da sich diese syntax in einigen fällen verschiedenartig interpretieren läßt und somit zu nicht eindeutigen abfragen führt, rät microsoft von der alten sql-server-syntax ab und empfiehlt die ansi-syntax.

wann setzt man nun äußere verknüpfungen ein? sehen sie sich dazu das folgende beispiel an. es soll alle titelnummern und titel aus der tabelle titles und dazu die verkauften mengen aus der tabelle sales anzeigen. die beiden tabellen sind über die spalte title_id verbunden. bei einer gleichheitsverknüpfung oder einer natürlichen verknüpfung erhalten sie nur die titel, die auch verkauft wurden und folglich in der tabelle sales aufgeführt sind. wenn sie aber alle titel unabhängig von der verkauften anzahl auflisten wollen, wenden sie eine linke äußere verknüpfung (auf die tabelle titles) an:

```
-- linke äußere verknüpfung --
/* sql server-syntax: */
select t.title_id, t.title, s.qty
from titles t, sales s
where t.title_id *= s.title_id

/* ansi-syntax: */
select t.title_id, t.title, s.qty
from titles t left outer join sales s
on t.title_id = s.title_id
```

zum vergleich können sie die beiden abfragen - nach sql-server-syntax und nach ansi-syntax - im sql server query analyzer so wie oben angegeben unmittelbar nacheinander ausführen. aktivieren sie am besten im menü **abfrage** die option **ergebnisse in gitternetz**. dann können sie zwischen den ergebnismengen der ersten abfrage nach sql-server-syntax und der zweiten abfrage nach ansi-syntax wechseln (siehe abbildung 10.2).

bild

abbildung 10.2: zwei abfragen zum vergleich nebeneinander im ergebnisbereich

die ergebnismenge hat folgendes aussehen:

```
title_id title                                          qty
-------- ---------------------------------------------- ------
pc1035   but is it user friendly?                       30
ps1372   computer phobic and non-phobic individuals: beh 20
bu1111   cooking with computers: surreptitious balance s 25
ps7777   emotional security: a new algorithm            25
tc4203   fifty years in buckingham palace kitchens      20
ps2091   is anger the enemy?                            3
ps2091   is anger the enemy?                            75
ps2091   is anger the enemy?                            10
ps2091   is anger the enemy?                            20
ps2106   life without fear                              25
pc9999   net etiquette                                  null
tc3218   onions, leeks, and garlic: cooking secrets of t 40
```

```
ps3333   prolonged data deprivation: four case studies   15
pc8888   secrets of silicon valley                       50
mc2222   silicon valley gastronomic treats               10
bu7832   straight talk about computers                   15
tc7777   sushi, anyone?                                  20
bu1032   the busy executive's database guide             5
bu1032   the busy executive's database guide             10
mc3021   the gourmet microwave                           25
mc3021   the gourmet microwave                           15
mc3026   the psychology of computer cooking              null
bu2075   you can combat computer stress!                 35

(23 row(s) affected)
```

in der ergebnismenge sind auch titel enthalten, die nicht verkauft wurden und in der spalte qty den wert null zeigen. eine gleichheitsverknüpfung hätte nur 21 zeilen geliefert.

10.4.4 kreuzverknüpfungen

wenn sie in der sql-server-syntax die where-klausel weglassen bzw. in der ansi-syntax das schlüsselwort cross join angeben und die on-klausel nicht spezifizieren, entsteht eine kreuzverknüpfung, die das kartesische produkt der beiden tabellen zurückgibt. dabei werden alle spalten der linken tabelle mit allen spalten der rechten tabelle kombiniert:

```
/* sql server-syntax: */
select s.stor_id, t.stor_name
from sales s, stores t

/* ansi-syntax: */
select s.stor_id, t.stor_name
from sales as s cross join stores as t
```

die tabelle sales enthält 21 zeilen, die tabelle stores 6 zeilen, so daß die ergebnismenge aus 21 * 6 = 126 zeilen besteht:

```
stor_id stor_name
------- ----------------------------------------
6380    eric the read books
8042    eric the read books
8042    eric the read books
...
7067    bookbeat
```

```
7067    bookbeat
7067    bookbeat

(126 row(s) affected)
```

10.4.5 selbstverknüpfungen

selbst wenn es im ersten moment etwas eigenartig klingt: eine tabelle läßt sich auch mit sich selbst verknüpfen. in der regel führt man *selbstverknüpfungen* - ein sonderfall der inneren verknüpfung - durch, um auf gleiche informationen zu prüfen. das folgende beispiel listet alle autoren auf, die in derselben stadt mit derselben postleitzahl wohnen:

```
/* sql server-syntax: */
select a1.au_fname, a1.au_lname,
       a2.au_fname, a2.au_lname,
       a1.zip, a1.city
from authors a1, authors a2
where a1.zip = a2.zip
  and a1.city = a2.city
  and a1.au_id < a2.au_id
order by a1.city

/* ansi-syntax: */
select a1.au_fname, a1.au_lname,
       a2.au_fname, a2.au_lname,
       a1.zip, a1.city
from authors a1 inner join authors a2
on  a1.zip  = a2.zip
and a1.city = a2.city
where a1.au_id < a2.au_id
order by a1.city
```

das ergebnis lautet:

```
au_fname au_lname   au_fname au_lname   zip   city
-------- ---------- -------- ---------- ----- ----------------
cheryl   carson     abraham  bennet     94705 berkeley
dean     straight   dirk     stringer   94609 oakland
dean     straight   livia    karsen     94609 oakland
dirk     stringer   livia    karsen     94609 oakland
ann      dull       sheryl   hunter     94301 palo alto
```

```
anne     ringer     albert   ringer     84152 salt lake city

(6 row(s) affected)
```

10.5 unterabfragen

eine unterabfrage ist praktisch nichts weiter als eine select-anweisung, die in einer anderen select-anweisung verschachtelt ist. man spricht auch von äußerer und innerer abfrage. unterabfragen lassen sich an verschiedenen stellen angeben:

- in verbindung mit aliasnamen
- in listen mit in bzw. not in
- in update-, insert- und delete-anweisungen
- mit vergleichsoperatoren
- mit den operatoren any, some und all
- in testausdrücken mit exists bzw. not exists
- an stellen, an denen auch ausdrücke erlaubt sind

wenn eine unterabfrage einen einzelnen wert zurückgibt, kann man sie überall dort einsetzen, wo auch einzelne werte stehen können. handelt es sich dagegen beim rückgabewert um eine liste, etwa eine ganze spalte, die mehrere werte enthält, kann man die unterabfrage nur in einer where-klausel verwenden.

auf den charakter einer inneren abfrage weisen die klammern hin, in die eine unterabfrage einzuschließen ist. genau wie die auswertung von ausdrücken in klammern zuerst erfolgt, wird eine unterabfrage erst komplett abgearbeitet, damit die äußere abfrage die ergebnisse verwenden und verarbeiten kann.

anstelle einer unterabfrage kann man oftmals auch eine verknüpfung verwenden. manchmal ist jedoch eine unterabfrage die einzig mögliche lösung. im hinblick auf die leistung besteht kaum ein unterschied zwischen einer verknüpfung und einer unterabfrage. wenn allerdings tests auf bestimmte daten zu realisieren sind, ist eine verknüpfung schneller, weil die unterabfrage für jedes ergebnis der äußeren abfrage auszuführen ist, um duplikate beseitigen zu können.

die (vereinfachte) syntax einer verschachtelten select-anweisung entspricht bis auf die klammern weitgehend einer normalen select-anweisung:

```
(select auswahlliste
from tabellenliste
[where suchbedingungen]
[group by gruppenliste]
[having suchbedingungen])
```

nehmen wir an, sie wollen eine buchlesung veranstalten und suchen alle autoren, die im selben land wie

die buchhandlung ansässig sind. das können sie zum beispiel mit der folgenden anweisung ermitteln:

```
use pubs
select au_id, au_lname, state
from authors
where state in
(select state from stores)
order by au_lname
```

beachten sie, daß sich die spalte state in der where-klausel auf die tabelle authors bezieht, während die gleichnamige spalte in der unterabfrage für die tabelle stores gilt. als ergebnis erhalten sie:

```
au_id       au_lname                                 state
----------- ---------------------------------------- -----
409-56-7008 bennet                                   ca
648-92-1872 blotchet-halls                           or
238-95-7766 carson                                   ca
427-17-2319 dull                                     ca
213-46-8915 green                                    ca
472-27-2349 gringlesby                               ca
846-92-7186 hunter                                   ca
756-30-7391 karsen                                   ca
486-29-1786 locksley                                 ca
724-80-9391 macfeather                               ca
893-72-1158 mcbadden                                 ca
267-41-2394 o'leary                                  ca
274-80-9391 straight                                 ca
724-08-9931 stringer                                 ca
172-32-1176 white                                    ca
672-71-3249 yokomoto                                 ca

(16 row(s) affected)
```

die unterabfrage können sie auch separat testen, bevor sie sie in einer äußeren abfrage einsetzen. damit erhalten sie einen überblick über art und umfang der von der unterabfrage zurückgegebenen informationen.

auch wenn es sich bei unterabfragen grundsätzlich um select-anweisungen handelt, sind gewisse regeln zu beachten:

- wenn man eine unterabfrage in einem ausdruck verwendet, darf sie nur einen wert zurückgeben.
- in der order by-klausel sind unterabfragen nicht zulässig.

- in der auswahlliste der unterabfrage sind keine text- und bilddatentypen zulässig.
- wenn die where-klausel der äußeren abfrage mit in formuliert ist, darf die unterabfrage nur eine einzige spalte zurückgeben.
- ist die where-klausel der äußeren abfrage mit exists formuliert, muß die auswahlliste der unterabfrage mit dem sternchen (*) spezifiziert sein.
- die datentypen der in der äußeren where-klausel angegebenen spalte und der in der unterabfrage verwendeten spalte müssen kompatibel sein.

die gleiche abfrage wie im letzten beispiel können sie mit exists wie folgt formulieren:

```
select au_id, au_lname, state
from authors a
where exists
(select * from stores
 where state = a.state)
order by au_lname
```

beachten sie das sternchen in der auswahlliste der unterabfrage.

10.6 korrelierte unterabfragen

eine reguläre unterabfrage ist eine vollkommen selbständige einheit und wird unabhängig von den werten der äußeren abfrage ausgeführt. dagegen bezieht sich eine korrelierte unterabfrage auf eine tabelle in der äußeren abfrage. dabei wird die ergebnismenge einer unterabfrage in der where-klausel der äußeren abfrage eingesetzt. die unterabfrage hängt von den werten der äußeren abfrage ab. wenn die äußere abfrage mehrere zeilen auswählt, wird die unterabfrage einmal für jede dieser zeilen ausgeführt. das bedeutet aber auch, daß sie eine korrelierte unterabfrage nicht wie bei einer regulären unterabfrage selbständig testen können.

die folgende anweisung zeigt alle titel an, für die tantiemen an zwei autoren gezahlt werden:

```
select title_id, title
from titles t
where 2 in (select au_ord from titleauthor
       where title_id = t.title_id)
```

in der spalte au_ord der tabelle titleauthor steht eine art ordnungsnummer der mitautoren. erhält ein autor 100 prozent der tantiemen, ist der betreffende titel nur einmal eingetragen und weist in der spalte au_ord den wert 1 auf. bei mehreren autoren sind dementsprechend mehrere einträge in titleauthor zu finden, und die spalte au_ord enthält eine nummer im bereich von 1 bis zur anzahl der autoren. die where-klausel der obigen anweisung testet also auf die titel, für die ein zweiter autor angegeben ist.

die unterabfrage ist nicht selbstständig, sondern hängt vom wert der spalte title_id der äußeren abfrage ab. bei der auswertung wird jede zeile der tabelle titles in die unterabfrage eingesetzt. trifft die bedingung der where-klausel zu, wird die zeile in die ergebnismenge übernommen:

```
title_id title
-------- ----------------------------------------------------
bu1032   the busy executive's database guide
bu1111   cooking with computers: surreptitious balance sheets
mc3021   the gourmet microwave
pc8888   secrets of silicon valley
ps1372   computer phobic and non-phobic individuals: behavior
ps2091   is anger the enemy?
tc7777   sushi, anyone?

(7 row(s) affected)
```

10.7 select into

mit der transact-sql-anweisung select into fragen sie eine tabelle wie bei einer normalen select-anweisung ab, erstellen aber gleichzeitig eine neue tabelle, in die die ergebnismenge geschrieben wird. die um das schlüsselwort into erweiterte syntax der select-anweisung sieht wie folgt aus:

```
select auswahlliste
into neuetabelle
from tabellenliste
[where suchbedingungen]
[order by sortierliste [asc | desc]]
```

die *neuetabelle* kann eine permanente oder eine temporäre tabelle sein. für eine permanente tabelle müssen sie die datenbankoption *auswahl / massenkopieren* (bzw. *select into/bulkcopy*) einschalten:

```
exec sp_dboption pubs, sel, true
```

auf datenbankoptionen und die gespeicherte prozedur sp_dboption wurde in kapitel 6 eingegangen.

die anweisung select into erzeugt die neue tabelle automatisch, so daß sie die tabelle nicht explizit definieren müssen. der name der neuen tabelle muß in der datenbank eindeutig sein und den regeln für bezeichner entsprechen.

wenn in der auswahlliste eine berechnete spalte enthalten ist, wird sie in der neuen tabelle nicht wieder

zur berechneten spalte, sondern spiegelt die berechneten ergebnisse der originaltabelle zum zeitpunkt der ausführung der select into-anweisung wider.

die folgende anweisung ruft die spalten title_id und price aus der tabelle titles ab, berechnet den preis in euro aus der spalte price und schreibt die resultierenden zeilen in die temporäre tabelle #tmpneuetabelle. eine temporäre tabelle läßt sich genauso abfragen wie eine reguläre tabelle, was die select-anweisung in zeile 6 zeigt:

```
1:  use pubs
2:  select title_id, price as preis, price/1.95583 as euro
3:  into #tmpneuetabelle
4:  from titles
5:
6:  select * from #tmpneuetabelle
```

die ergebnismenge hat folgendes aussehen:

```
(18 row(s) affected)

title_id preis                 euro
-------- --------------------- --------------------------------
bu1032   19.9900               10.22072470511
bu1111   11.9500               6.10993798029
bu2075   2.9900                1.52876272477
bu7832   19.9900               10.22072470511
mc2222   19.9900               10.22072470511
mc3021   2.9900                1.52876272477
mc3026   null                  null
pc1035   22.9500               11.73414867345
pc8888   20.0000               10.22583762392
pc9999   null                  null
ps1372   21.5900               11.03879171502
ps2091   10.9500               5.59864609909
ps2106   7.0000                3.57904316837
ps3333   19.9900               10.22072470511
ps7777   7.9900                4.08522213075
tc3218   20.9500               10.71156491106
tc4203   11.9500               6.10993798029
tc7777   14.9900               7.66426529913

(18 row(s) affected)
```

die erste meldung zur anzahl der betroffenen zeilen (rows affected) bezieht sich auf das abrufen mit der select into-anweisung.

die obige select into-anweisung überträgt eine berechnete spalte in die neue tabelle. dabei müssen sie die spalte eindeutig benennen. wenn sie einfach den namen der spalte weglassen und zeile 2 des beispiels als

```
select title_id, price as preis, price/1.95583
```

formulieren, erhalten sie die fehlermeldung:

```
server: nachr.-nr. 8155, schweregrad 16, status 1, zeile 2
keine spalte wurde für spalte 3 von '#tmpneuetabelle' angegeben.
```

10.8 der operator union

mit dem operator union lassen sich zwei oder mehr abfragen zu einer einzigen ergebnismenge kombinieren. in der voreinstellung werden dabei duplikate eliminiert. mit dem schlüsselwort all können sie alle zeilen einschließlich der doppelten zurückgeben.

die syntax einer union-operation hat folgendes aussehen:

```
<abfragespezifikation>
union [all]
<abfragespezifikation>
[union [all]
<abfragespezifikation>
[...n] ]
```

die *abfragespezifikation* ist eine abfrage oder ein abfrageausdruck. die damit zurückgegebenen daten sind mit den daten der anderen abfragespezifikationen zu kombinieren. dabei gelten folgende regeln:

- anzahl und reihenfolge der spalten müssen in beiden abfragen übereinstimmen.
- die definitionen der spalten brauchen nicht identisch zu sein, jedoch müssen sich die datentypen implizit konvertieren lassen.

wenn sie eine into-klausel verwenden, muß diese in der ersten abfrage stehen. die klauseln group by und having können sie nur in den einzelnen teilabfragen angeben. dagegen sind die klauseln order by und compute nur am ende der union-anweisung erlaubt, um die reihenfolge der ergebnismenge festzulegen oder zusammenfassungswerte zu berechnen.

das folgende beispiel soll ladenhüter ermitteln, die keine buchhandlung verkauft hat:

```
use pubs
select t.title_id,
       convert(char(25), title),
       convert(char(20),stor_name),
       qty
from   titles t, stores b, sales a
where  t.title_id=a.title_id
  and  a.stor_id=b.stor_id
union
select title_id,
       convert(char(25), title),
       'ladenhüter',
       null
from   titles
where  title_id not in
      (select title_id from sales)
order by qty desc
```

die erste abfrage ruft titelnummer, titel, name der buchhandlung und menge der verkauften titel aus den tabellen titles, stores und sales ab. aufgrund der bedingungen in der where-klausel erscheinen nicht alle titel in dieser abfrage, sondern nur diejenigen, die auch in den buchhandlungen verkauft wurden.

die zweite abfrage ruft alle titel aus der tabelle titles ab und prüft, ob die title_id in der ergebnismenge der unterabfrage enthalten ist. wenn das nicht der fall ist, handelt es sich um einen nicht verkauften titel. und genau diese ladenhüter bilden die ergebnismenge der zweiten abfrage.

durch kombination beider abfragen erhält man sowohl alle verkauften als auch alle nicht verkauften titel. weil in einer union-verknüpfung die anzahl und reihenfolge der spalten übereinstimmen müssen, steht in der ergebnisspalte für die buchhandlungen die zeichenfolge »ladenhüter« und in der spalte für menge (qty) ein null-wert. die convert-funktionen dienen hier nur kosmetischen zwecken, um die breite der ausgabespalten zu verringern. auf die convert-funktion geht kapitel 11 ein. das ergebnis der union-abfrage lautet:

```
title_id                                                qty
-------- ------------------------- -------------------- ------
ps2091   is anger the enemy?       barnum's             75
pc8888   secrets of silicon valley barnum's             50
tc3218   onions, leeks, and garlic news & brews         40
bu2075   you can combat computer s fricative bookshop   35
pc1035   but is it user friendly?  bookbeat             30
mc3021   the gourmet microwave     doc-u-mat: quality l 25
ps7777   emotional security: a new doc-u-mat: quality l 25
```

```
bu1111   cooking with computers: s bookbeat             25
ps2106   life without fear         doc-u-mat: quality l 25
tc7777   sushi, anyone?            news & brews         20
ps2091   is anger the enemy?       doc-u-mat: quality l 20
tc4203   fifty years in buckingham news & brews         20
ps1372   computer phobic and non-p doc-u-mat: quality l 20
ps3333   prolonged data deprivatio doc-u-mat: quality l 15
mc3021   the gourmet microwave     bookbeat             15
bu7832   straight talk about compu fricative bookshop   15
ps2091   is anger the enemy?       news & brews         10
mc2222   silicon valley gastronomi fricative bookshop   10
bu1032   the busy executive's data bookbeat             10
bu1032   the busy executive's data eric the read books  5
ps2091   is anger the enemy?       eric the read books  3
pc9999   net etiquette            ladenhüter           null
mc3026   the psychology of compute ladenhüter           null

(23 row(s) affected)
```

die sortierung der ergebnismenge erfolgt mit hilfe der order by-klausel in fallender richtung (schlüsselwort desc) nach der menge der verkauften bücher, so daß die ladenhüter am ende der ergebnismenge zu finden sind.

10.9 daten modifizieren

in kapitel 9 haben sie erfahren, wie man daten aus fremden quellen in eine datenbank importiert. im vorliegenden kapitel ging es bis jetzt ausschließlich darum, daten abzurufen - wenn man einmal von der speziellen anweisung select into absieht. in einer datenbank wollen sie die daten sicherlich nicht nur importieren, sondern auch einfügen, ändern und löschen können. für diesen zweck bietet transact-sql die anweisungen

- insert
- update
- delete

diese anweisungen gehören wie select zur datenmanipulationssprache. alle änderungen von daten zeichnet sql server im transaktionsprotokoll (siehe kapitel 18) auf.

die folgenden abschnitte beziehen sich auf die datenbanken lotto und pubs. wenn sie die datenbank pubs im derzeitigen zustand beibehalten möchten, legen sie am besten eine sicherung dieser datenbank an und stellen sie unter einem neuen namen wieder her. die datenbank lotto läßt sich am einfachsten mit dem auf der begleit-cd befindlichen skript instlott.sql in den anfangszustand versetzen.

10.9.1 einfügen (insert)

mit der anweisung insert lassen sich daten in eine datenbank einfügen. es gibt zwei grundsätzliche formen der insert-anweisung:

- insert values: fügt werte für jeweils eine zeile in eine tabelle ein.
- insert select: kann mehrere zeilen in eine tabelle einfügen, wobei die einzufügenden daten aus einer oder mehreren tabellen mit select abgerufen werden.

die vereinfachte syntax der insert-anweisung hat folgendes aussehen:

```
insert [into] tabellenname
{ [(spaltenliste)]
  { values
     ( { default | null | ausdruck }[,...n] )
     | auswahlanweisung
     | ausführungsanweisung
  }
}
| default values
```

tabellenname gibt die zieltabelle an, in die die daten einzufügen sind. die optionale *spaltenliste* spezifiziert die spalten, für die in der darauffolgenden values-liste werte angegeben sind. wenn eine spalte nicht in der spaltenliste aufgeführt ist, muß sql server in der lage sein, automatisch einen wert für diese spalte - basierend auf ihrer definition - einzutragen. dabei gelten folgende regeln:

- bei einer identity-spalte wird der nächste identitätswert verwendet.
- ist für die spalte ein standardwert definiert, trägt sql server den standardwert ein.
- ist die spalte vom typ timestamp, wird der aktuelle timestamp-wert eingetragen.
- läßt die spalte null-werte zu, fügt sql server einen null-wert ein.

die spalten in der spaltenliste und die werte in der values-liste werden durch kommas getrennt und müssen in anzahl und reihenfolge übereinstimmen.

in der values-liste erzwingt default das einfügen eines standardwertes und null das einfügen eines null-wertes. der *ausdruck* stellt eine konstante, eine variable oder einen regulären ausdruck dar.

die *auswahlanweisung* ist eine select-anweisung, die daten aus einer oder mehreren tabellen abruft. die ergebnismenge dieser select-anweisung wird durch insert in die mit *tabellenname* spezifizierte tabelle eingefügt. die *ausführungsanweisung* bezeichnet eine execute-anweisung, die daten per select oder readtext liefert.

wenn sie professionell mit einer datenbank arbeiten, haben sie sicherlich eine detaillierte dokumentation zur verfügung, die einen genauen überblick über die struktur der tabellen, die datentypen der spalten, eventuell definierte standardwerte und weitere angaben bietet. im testbetrieb erstellen sie aber häufig tabellen, die nur für die dauer der tests von interesse sind. in diesem fall können sie wahrscheinlich nicht

auf eine dokumentation zurückgreifen. für das einfügen von daten ist aber eine genaue kenntnis der tabellenstruktur unumgänglich. diese informationen können sie sich bei bedarf leicht beschaffen.

einen guten überblick über die struktur der datenbank und die tabellen erhalten sie mit einem datenbankdiagramm, wie es in kapitel 6 beschrieben wurde. drucken sie das diagramm am besten mit der anzeigeoption *spalteneigenschaften* aus.

informationen zu einzelnen tabellen lassen sich mit der gespeicherten prozedur sp_help abrufen. führen sie zum beispiel die anweisung

```
use pubs
exec sp_help titles
```

im sql server query analyzer aus, um sich über die tabelle titles der datenbank pubs zu informieren. es empfiehlt sich, die abfrageergebnisse im gitternetz anzuzeigen (siehe abbildung 10.3). wählen sie dazu **abfrage / ergebnisse in gitternetz**.

Bild

abbildung 10.3: eigenschaften einer tabelle im query analyzer anzeigen

im enterprise manager lassen sich ebenfalls alle benötigten informationen über tabellen anzeigen. erweitern sie dazu die konsolenstruktur bis zum eintrag **tabellen** der jeweiligen datenbank, so daß im detailbereich eine liste der tabellen erscheint. klicken sie mit der rechten maustaste auf die gewünschte tabelle, und wählen sie aus dem kontextmenü den befehl **eigenschaften**. daraufhin zeigt das dialogfeld **tabelleneigenschaften** die definition der tabelle an (siehe abbildung 10.4).

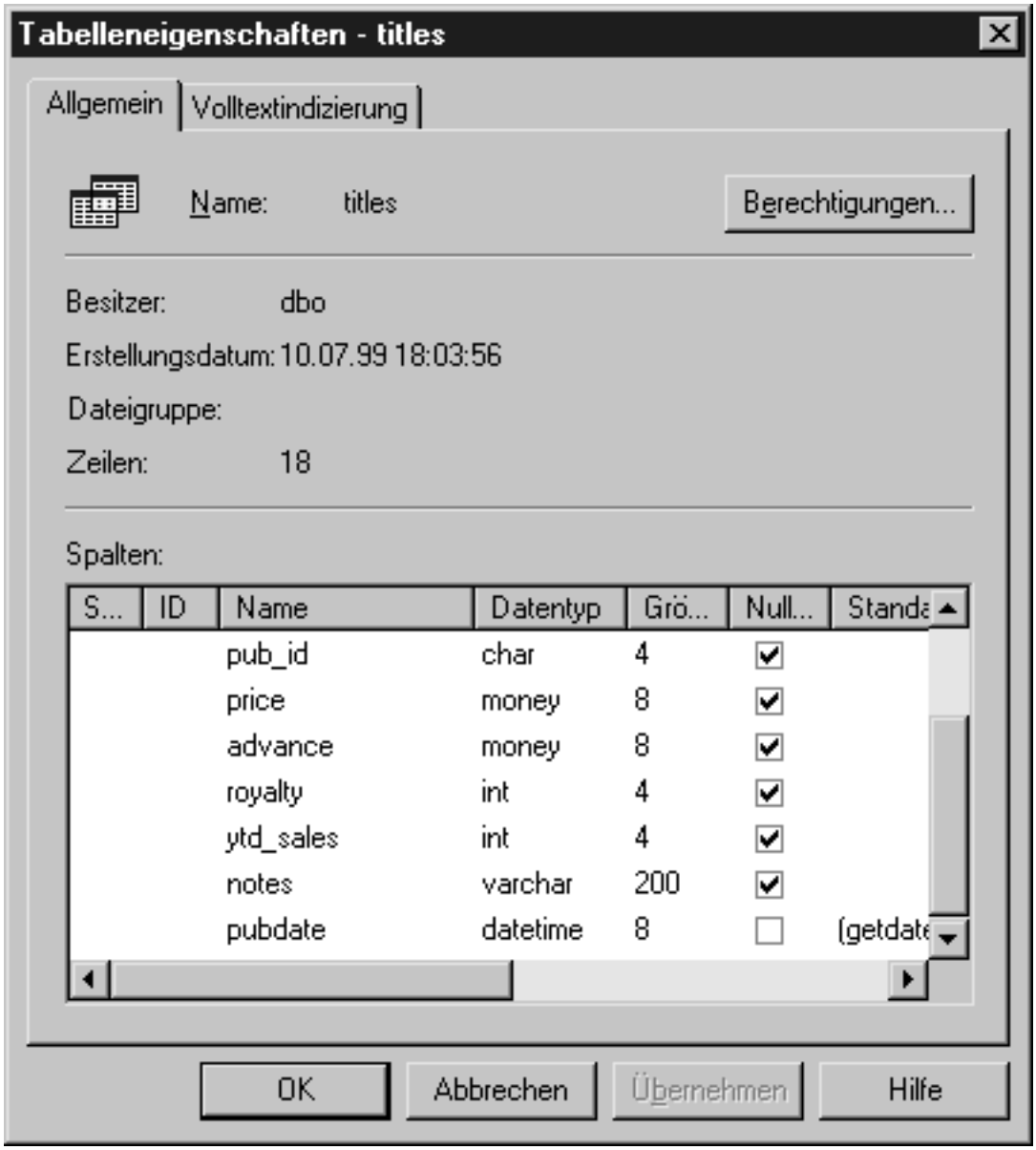


abbildung 10.4: eigenschaften einer tabelle im enterprise manager anzeigen

insert values

in der einfachsten form der insert-anweisung brauchen sie lediglich den tabellennamen und die einzufügenden werte anzugeben:

```
use lotto
insert into ldaten values (2261,'6.2.99',13,29,37,38,45,48,33)
```

die werte sind in der richtigen reihenfolge der spalten anzugeben. sql server zeigt zur bestätigung die folgende meldung an:

```
(1 row(s) affected)
```

fragen sie die tabelle ldaten zur kontrolle ab:

```
select * from ldaten
where ziehung >2259
```

das ergebnis lautet:

```
ziehung     datum           z1   z2   z3   z4   z5   z6   zz
----------- --------------- ---- ---- ---- ---- ---- ---- ----
2260        1999-01-30      4    5    18   20   22   41   40
2261        1999-02-06      13   29   37   38   45   48   33

(2 row(s) affected)
```

in der spalte datum erscheint auch die uhrzeit, die hier aus platzgründen (breite der druckseite) entfernt wurde.

wenn sie nicht alle werte in der values-liste bereitstellen, müssen sie die spalten explizit in der spaltenliste angeben:

```
insert into ldaten (ziehung,z1,z2,z3,z4,z5,z6,zz)
            values (2262,    3,29,35,38,43,49,37)
```

diese anweisung fügt keinen wert für die spalte datum ein. mit der obigen abfrage für ziehungen größer 2259 sieht die ergebnismenge jetzt wie folgt aus:

```
ziehung     datum           z1   z2   z3   z4   z5   z6   zz
----------- --------------- ---- ---- ---- ---- ---- ---- ----
2260        1999-01-30      4    5    18   20   22   41   40
2261        1999-02-06      13   29   37   38   45   48   33
2262        null            3    29   35   38   43   49   37
```

wenn eine spalte nicht in der spaltenliste aufgeführt ist, muß sql server automatisch einen wert bereitstellen können. das ist dann möglich, wenn

- es sich um eine identity-spalte handelt. in diesem fall nimmt sql server den nächsten identitätswert.

- für die spalte ein standardwert oder ein timestamp-wert definiert ist.
- die spalte null-werte zuläßt. sql server fügt dann einen null-wert ein.

im obigen beispiel wurde für die nicht angegebene spalte datum ein null-wert eingefügt. kapitel 16 zeigt, wie sie diese spalte mit einem standardwert - dem aktuellen datum - versehen.

für die spalte ziehung ist laut definition der tabelle ldaten kein null-wert zulässig. in der anweisung

```
insert into ldaten (z1,z2,z3,z4,z5,z6,zz)
            values (1,9,13,23,46,48,36)
```

ist kein wert für die spalte ziehung angegeben. sql server bringt deshalb die fehlermeldung:

```
server: nachr.-nr. 515, schweregrad 16, status 2, zeile 1
der wert null kann in spalte 'ziehung', tabelle
'lotto.dbo.ldaten' nicht eingefügt werden. die spalte lässt
null-werte nicht zu. insert schlägt fehl.
die anweisung wurde beendet.
```

in der tabelle publishers der datenbank pubs ist für die spalte country der standardwert 'usa' definiert. die anweisung

```
use pubs
insert into publishers (pub_id, pub_name, city, state)
            values     ('9998', 'mcp', 'indianapolis', 'in')
```

fügt eine vollständige zeile in die tabelle publishers ein und ersetzt den fehlenden wert für die spalte country durch 'usa':

```
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
9998   mcp                   indianapolis         in    usa
9999   lucerne publishing    paris                null  france
```

die spaltenliste können sie auch weglassen, wenn sie für den fehlenden wert das schlüsselwort default angeben:

```
insert into publishers
       values ('9998', 'mcp', 'indianapolis', 'in', default)
```

schließlich können sie auch noch die values-liste weglassen und dafür die schlüsselwörter default values angeben, wenn für alle spalten einer tabelle standardwerte definiert sind:

```
insert into tabelle default values
```

diese bedingung trifft aber für keine der beispieltabellen zu.

insert select

die im letzten abschnitt vorgestellte insert-anweisung fügt jeweils eine zeile in eine datenbank ein. das massenkopieren mit dem dienstprogramm bcp ist auch nichts anderes als eine wiederholte ausführung derartiger insert-anweisungen. es ist aber auch möglich, mehrere zeilen auf einmal einzufügen, wenn man diese in einer unterabfrage mit select bereitstellt.

die auswahlliste der unterabfrage muß mit der spaltenliste der insert-anweisung übereinstimmen. daß die datentypen kompatibel sein müssen, haben sie sicherlich schon vermutet. im gegensatz zu den weiter vorn in diesem kapitel behandelten unterabfragen ist die unterabfrage in einer insert select-anweisung nicht in klammern eingeschlossen.

das folgende beispiel ruft in der unterabfrage alle titel ab, die in der tabelle titles für das gebiet (spalte type) psychologie gespeichert sind, und fügt die resultierenden zeilen in eine temporäre tabelle #tmppsycho ein:

```
1:  use pubs
2:  select * into #tmppsycho from titles where type='xxx'
3:  go
4:  insert into #tmppsycho
5:    select * from titles
6:    where type = 'psychology'
```

um die bedingung nach kompatiblen spalten mit gleicher anzahl zu erfüllen, zeigt dieses beispiel einen kleinen trick: zeile 2 erstellt mit der select into-anweisung eine temporäre tabelle #tmppsycho, deren

definition genau mit der tabelle titles übereinstimmt. die bedingung in der where-klausel stellt dabei sicher, daß die temporäre tabelle leer ist.

die unterabfrage in den zeilen 5 und 6 ruft alle titel für das gebiet psychologie ab. die insert-anweisung ab zeile 4 fügt diese ergebnismenge in die tabelle #tmppsycho ein.

fragen sie die temporäre tabelle #tmppsycho zur kontrolle mit der folgenden anweisung ab:

```
select title_id, convert(char(40),title) as titel,
       type as gebiet
  from #tmppsycho
```

das ergebnis lautet:

```
title_id titel                                    gebiet
-------- ---------------------------------------- ------------
ps1372   computer phobic and non-phobic individua psychology
ps2091   is anger the enemy?                      psychology
ps2106   life without fear                        psychology
ps3333   prolonged data deprivation: four case st psychology
ps7777   emotional security: a new algorithm      psychology

(5 row(s) affected)
```

10.9.2 aktualisieren (update)

die werte in vorhandenen zeilen einer tabelle lassen sich mit der transact-sql-anweisung update aktualisieren - d.h. modifizieren. die vereinfachte syntax der update-anweisung lautet:

```
update tabellenname
set { spaltenname = {ausdruck | default | null} } [,...n]
[from tabellenliste]
[where suchbedingung]
```

mit der set-klausel legen sie den neuen wert für die betreffende spalte fest. in der from-klausel spezifizieren sie bei bedarf mehrere tabellen, die als quelle der daten für die update-anweisung in frage kommen. ist die optionale where-klausel nicht angegeben, führt update die aktualisierung für alle werte der spalte durch:

```
use pubs
```

```
update titles set price = price * 10
```

diese anweisung multipliziert die werte in der preisspalte mit 10. da keine where-klausel spezifiziert ist, wirkt diese anweisung auf die gesamte spalte, was auch die meldung nach ausführung der anweisung bestätigt:

```
(18 row(s) affected)
```

die aktualisierte spalte price hat dann folgende werte:

```
title_id price
-------- ---------------------
bu1032   199.9000
bu1111   119.5000
...
tc3218   209.5000
tc4203   119.5000
tc7777   149.9000

(18 row(s) affected)
```

die nächste anweisung aktualisiert die im beispiel zur insert-anweisung eingefügt zeile für die ziehung 2262 der datenbank lotto, um den automatisch eingetragenen null-wert durch das aktuelle datum zu ersetzen:

```
use lotto
update ldaten
set datum=getdate()
where ziehung=2262
```

vorausgesetzt, daß diese anweisung am 20.02.1999 ausgeführt wurde, sehen die letzten drei zeilen jetzt folgendermaßen aus:

```
ziehung     datum           z1   z2   z3   z4   z5   z6   zz
----------- --------------- ---- ---- ---- ---- ---- ---- ----
2260        1999-01-30      4    5    18   20   22   41   40
2261        1999-02-06      13   29   37   38   45   48   33
2262        1999-02-20      3    29   35   38   43   49   37
```

wie aus der syntax hervorgeht, lassen sich auch mehrere spalten auf einmal aktualisieren. auf das schlüsselwort set folgen dann die entsprechenden ausdrücke durch komma getrennt. die folgende anweisung zeigt diese konstruktion am beispiel der datenbank pubs:

```
update titles
set price = price * 1.2, notes = 'neuer preis! ' + notes
where type = 'business'

select title_id, type, price,
       convert(char(13), notes) from titles

(4 row(s) affected)

title_id type         price
-------- ------------ --------------------- -------------
bu1032   business     23.9880               neuer preis!
bu1111   business     14.3400               neuer preis!
bu2075   business     3.5880                neuer preis!
bu7832   business     23.9880               neuer preis!
mc2222   mod_cook     19.9900               favorite reci
mc3021   mod_cook     2.9900                traditional f
...
tc4203   trad_cook    11.9500               more anecdote
tc7777   trad_cook    14.9900               detailed inst

(18 row(s) affected)
```

die zeilen einer tabelle können sie auch auf den werten einer verknüpften tabelle basierend aktualisieren. nehmen wir an, daß sie die verleger-id (pub_id) in der tabelle titles für einen bestimmten autor aktualisieren müssen. in der tabelle titles sind allerdings keine informationen zu den autoren zu finden. um die geforderte aufgabe zu realisieren, verknüpfen sie die tabelle titles mit titleauthor und geben die beiden tabellen in der from-klausel der update-anweisung an. die anweisung im folgenden beispiel setzt die verleger-id für den autor charlene locksley (mit der autoren-id 486-29-1786) auf 1389:

```
use pubs
update titles
set pub_id = '1389'
from titles t, titleauthor a
where t.title_id = a.title_id and
      au_id = '486-29-1786'
```

10.9.3 löschen (delete)

mit der transact-sql-anweisung delete löschen sie eine komplette zeile aus einer tabelle. die syntax der anweisung delete lautet:

```
delete [from] tabellenname
[where suchbedingung]
```

ohne die where-klausel löscht die delete-anweisung alle zeilen in der angegebenen tabelle. zum beispiel beseitigt die anweisung

```
delete pub_info
```

radikal alle zeilen aus der tabelle pub_info. falls sie den befehl gleich ausprobiert haben - die tabelle pub_info ist für die nächsten beispiele entbehrlich.

in der regel sind nur einzelne zeilen zu löschen. dazu legen sie in der where-klausel eine bedingung fest, genau wie sie es von der select-anweisung her kennen. die folgende anweisung löscht alle buchtitel mit der titel-id pc8888 aus der tabelle titleauthor:

```
use pubs
delete titleauthor
where title_id = 'pc8888'
```

der bedingung entsprechen 2 zeilen. die tabelle titleauthor hat im ursprünglichen zustand 25 zeilen. fragen sie die tabelle zur kontrolle mit der folgenden anweisung ab:

```
select * from titleauthor
order by title_id
```

das nach der spalte title_id sortierte ergebnis zeigt, daß die zeilen mit der titel-id pc8888 nicht mehr vorhanden sind:

```
au_id       title_id au_ord royaltyper
----------- -------- ------ -----------
213-46-8915 bu1032   2      40
```

```
409-56-7008 bu1032   1      60
267-41-2394 bu1111   2      40
724-80-9391 bu1111   1      60
213-46-8915 bu2075   1      100
274-80-9391 bu7832   1      100
712-45-1867 mc2222   1      100
722-51-5454 mc3021   1      75
899-46-2035 mc3021   2      25
238-95-7766 pc1035   1      100
486-29-1786 pc9999   1      100
724-80-9391 ps1372   2      25
756-30-7391 ps1372   1      75
...
472-27-2349 tc7777   3      30
672-71-3249 tc7777   1      40

(23 row(s) affected)
```

schließlich können sie auch zeilen in einer tabelle löschen, indem sie kriterien einer anderen tabelle in form einer unterabfrage heranziehen. in der tabelle titles ist zum beispiel ein titel enthalten, für den in verschiedenen spalten null-werte auftauchen. offensichtlich gibt es für diesen titel noch keinen autor. nehmen wir nun an, daß sie alle titel löschen wollen, denen kein autor zugeordnet ist. in der tabelle titles sind allerdings keine angaben zu den autoren enthalten.

als lösung bietet sich hier eine unterabfrage an, die die werte der spalte title_id aus der tabelle titleauthor liefert. die äußere abfrage kann dann für jede zeile der tabelle titles prüfen, ob die title_id nicht in titleauthor vorkommt und somit kein autor für den betreffenden titel nominiert ist. das ergebnis dieses tests dient als kriterium für die delete-anweisung zum löschen der betreffenden zeilen.

aus der tabelle titles können sie auf diese weise keinen titel löschen, da es zu schlüsselverletzungen kommen würde. deshalb kopiert das folgende beispiel zunächst die tabelle titles in die temporäre tabelle #tmptitel, um das oben beschriebene verfahren ohne weitere eingriffe in die bestehenden beziehungen der datenbank pubs demonstrieren zu können:

```
1:  use pubs
2:  select * into #tmptitel from titles
3:
4:  delete from #tmptitel
5:  where title_id not in
6:    (select title_id from titleauthor)
7:
8:  select title_id, convert(char(40),title) from #tmptitel
```

zeile 2 kopiert alle spalten der tabelle titles mit hilfe der select into-anweisung in die temporäre tabelle #tmptitel. in zeile 6 ist die unterabfrage zu finden, die alle titel-ids aus der tabelle titleauthor zurückgibt. die where-klausel in zeile 5 prüft, ob die aktuelle title_id nicht in dieser ergebnismenge enthalten ist. die unterabfrage wird dabei einmal für jede zeile der tabelle #tmptitel ausgeführt. wenn die bedingung der where-klausel erfüllt ist, löscht die delete-anweisung die betreffende zeile aus der tabelle #tmptitel.

das ergebnis der kontrollabfrage von zeile 8 lautet:

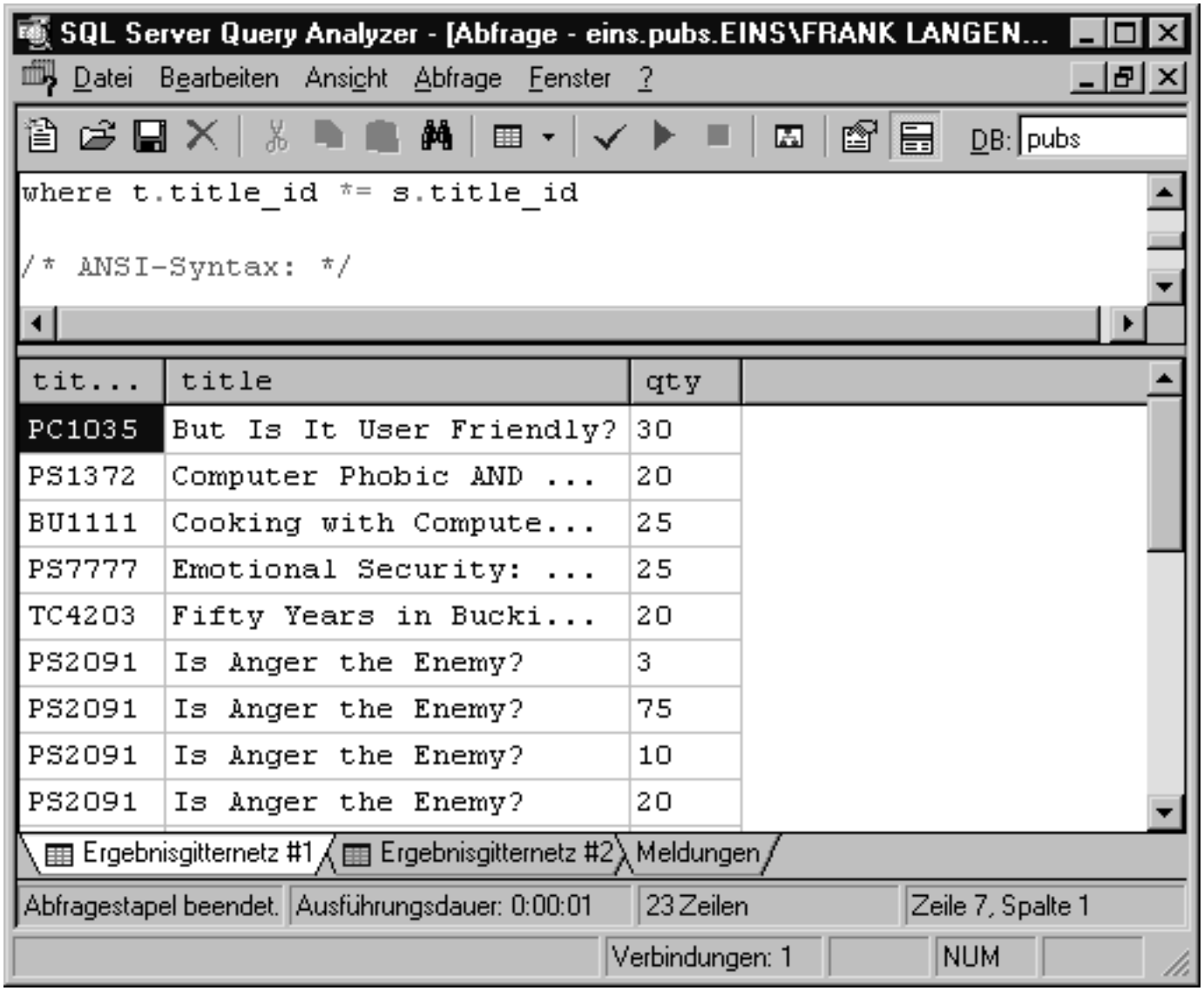
```
title_id
-------- ----------------------------------------
bu1032   the busy executive's database guide
bu1111   cooking with computers: surreptitious ba
bu2075   you can combat computer stress!
bu7832   straight talk about computers
mc2222   silicon valley gastronomic treats
mc3021   the gourmet microwave
mc3026   the psychology of computer cooking
pc1035   but is it user friendly?
pc8888   secrets of silicon valley
pc9999   net etiquette
ps1372   computer phobic and non-phobic individua
ps2091   is anger the enemy?
ps2106   life without fear
ps3333   prolonged data deprivation: four case st
ps7777   emotional security: a new algorithm
tc3218   onions, leeks, and garlic: cooking secre
tc4203   fifty years in buckingham palace kitchen
tc7777   sushi, anyone?

(18 row(s) affected)


(1 row(s) affected)

title_id
-------- ----------------------------------------
bu1032   the busy executive's database guide
bu1111   cooking with computers: surreptitious ba
bu2075   you can combat computer stress!
bu7832   straight talk about computers
mc2222   silicon valley gastronomic treats
mc3021   the gourmet microwave
pc1035   but is it user friendly?
...
tc4203   fifty years in buckingham palace kitchen
tc7777   sushi, anyone?
```

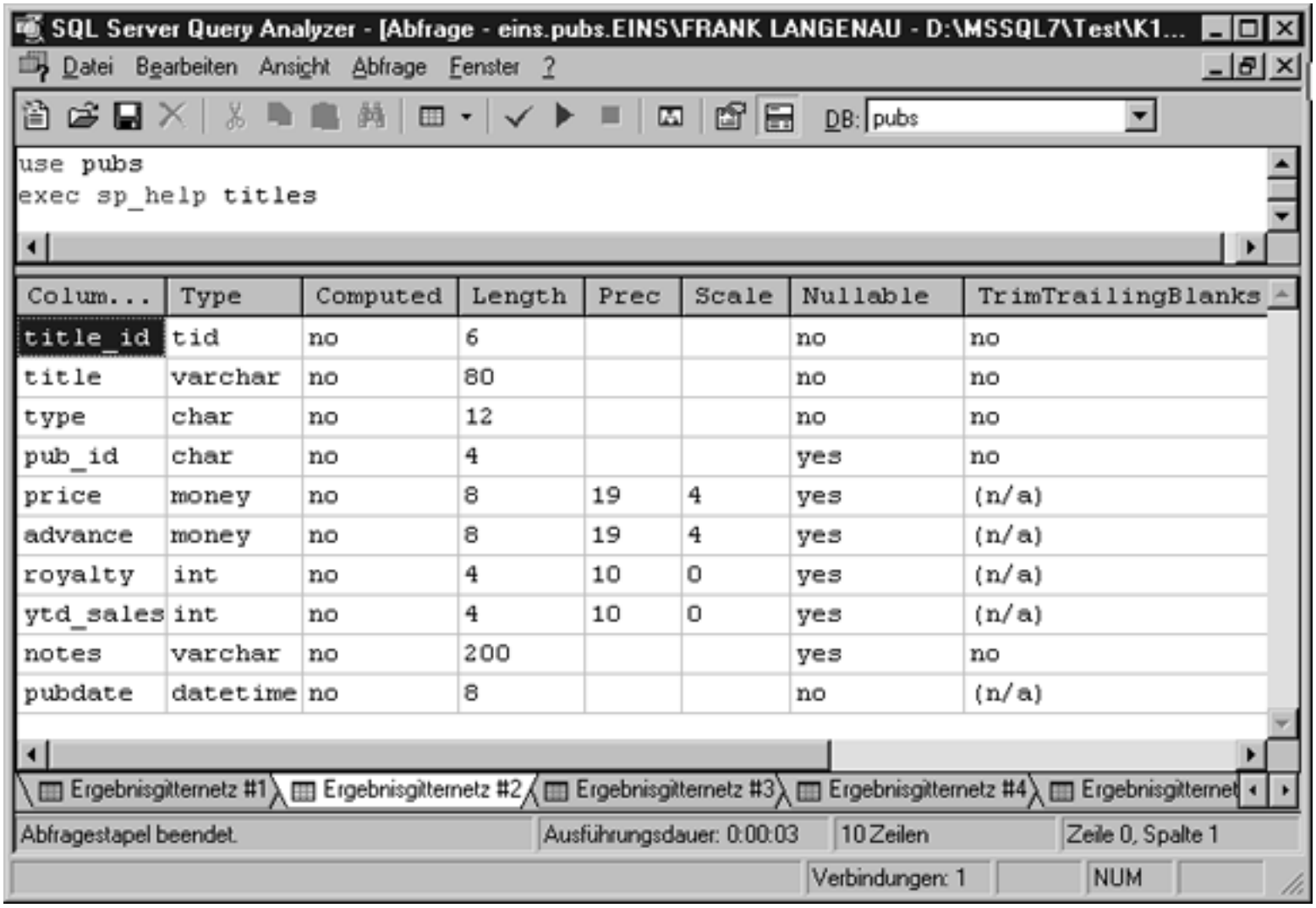
```
(17 row(s) affected)
```

von den ursprünglichen 18 zeilen bleiben noch 17 übrig. der fragliche titel hatte die titel-id mc3026.

wenn sie die beispiele in diesem kapitel mit der originaldatenbank pubs nachvollzogen haben, sollten sie das skript instpubs.sql ausführen, um die datenbank pubs wieder in den ausgangszustand zu bringen. trennen sie vorher alle von ihnen hergestellten verbindungen zu sql server, um gegebenenfalls temporäre tabellen zu löschen. melden sie sich anschließend neu an, und starten sie das skript.


Abfrage - eins.pubs.EINS\FRANK LANGENAU - (unbenannt) - select title_id...*
DB: pubs
select title_id from titles
Lotto
master
Mist
model
msdb
Northwind
pubs
tempdb
test
<Aktualisieren>
title_id
PC1035
PS1372
BU1111
PS7777
TC4203
Ergebnisse
Abfragestapel beendet.
Ausführungsdauer: 0:00:00
18 Zeilen
Zeile 1, Spalte 28


SQL Server Query Analyzer - [Abfrage - eins.pubs.EINS\FRANK LANGEN...
Datei Bearbeiten Ansicht Abfrage Fenster ?
DB: pubs
where t.title_id *= s.title_id
/* ANSI-Syntax: */
tit... title qty
PC1035 But Is It User Friendly? 30
PS1372 Computer Phobic AND ... 20
BU1111 Cooking with Compute... 25
PS7777 Emotional Security: ... 25
TC4203 Fifty Years in Bucki... 20
PS2091 Is Anger the Enemy? 3
PS2091 Is Anger the Enemy? 75
PS2091 Is Anger the Enemy? 10
PS2091 Is Anger the Enemy? 20
Ergebnisgitternetz #1
Ergebnisgitternetz #2
Meldungen
Abfragestapel beendet.
Ausführungsdauer: 0:00:01
23 Zeilen
Zeile 7, Spalte 1
Verbindungen: 1
NUM



Colum...	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks
title_id	tid	no	6			no	no
title	varchar	no	80			no	no
type	char	no	12			no	no
pub_id	char	no	4			yes	no
price	money	no	8	19	4	yes	(n/a)
advance	money	no	8	19	4	yes	(n/a)
royalty	int	no	4	10	0	yes	(n/a)
ytd_sales	int	no	4	10	0	yes	(n/a)
notes	varchar	no	200			yes	no
pubdate	datetime	no	8			no	(n/a)

Kapitel 11 Funktionen

11.1 Wer hat den längsten ...

... Urlaub von allen Mitarbeitern? Diese Frage läßt sich leicht beantworten, indem Sie aus der entsprechenden Tabelle den Datensatz mit dem größten Wert in der Urlaubsspalte abrufen. Dabei müssen Sie aber nicht jeden Wert mit jedem vergleichen, sondern können auf eine Funktion MAX zurückgreifen, die den größten Wert einer Spalte liefert. Damit sind die Möglichkeiten bei weitem noch nicht erschöpft. Transact-SQL definiert einen umfangreichen Satz an Funktionen, die das Thema dieses Kapitels sind.

11.2 Funktionen in Transact-SQL

Funktionen sind benannte Befehlsfolgen, die eine logische Einheit bilden und einen Rückgabewert liefern. Um die spezifizierte Operation auszuführen, ruft man die Funktion unter ihrem Namen auf und übergibt die zu verarbeitenden Daten in Form einer Parameterliste. Grundsätzlich lassen sich Funktionen auch durch einzelne Anweisungen darstellen. Allerdings bieten Funktionen eine Reihe von Vorteilen:

- Sie liegen in kompilierter Form vor und laufen damit schneller ab als einzeln ausgeführte Anweisungen.
- Die Ausführung der Funktionen erfolgt auf dem Server. Damit verringert sich die Bandbreite der Datenübertragung im Netzwerk. Außerdem können die Funktionen ohne Umwege auf die zugrundeliegenden Daten zugreifen, was gegenüber der Ausführung auf dem lokalen System ebenfalls zu einer höheren Geschwindigkeit beiträgt.

Die Sprache Transact-SQL unterscheidet drei Arten von Funktionen:

- *Aggregatfunktionen* fassen mehrere Werte zu einem Rückgabewert zusammen.
- *Skalare Funktionen* verarbeiten einen Wert und geben einen Wert zurück.
- *Rowset-Funktionen* lassen sich wie Tabellenverweise verwenden.

Im ANSI-SQL-Standard sind lediglich fünf Funktionen verankert: die Aggregatfunktionen AVG, SUM, MIN, MAX und COUNT. Alle anderen Funktionen sind herstellerspezifische Erweiterungen im SQL-Dialekt des jeweiligen Datenbank-Managementsystems. Dennoch haben viele Funktionen den Status eines De-facto-Standards und sind - zum Teil mit etwas abweichenden Namen - in den gängigsten Implementierungen von SQL zu finden.

Die folgenden Abschnitte behandeln die wichtigsten Transact-SQL-Funktionen im Detail und geben Beispiele für ihren Einsatz an. Weniger wichtige Funktionen oder Funktionen, deren Anwendung sich ohne weiteres erkennen läßt, sind im tabellarischen Überblick wiedergegeben.

11.3 Aggregatfunktionen

Aggregatfunktionen fassen mehrere Werte zu einem Wert zusammen. Damit lassen sich zum Beispiel Teilsummen und Gesamtsummen berechnen oder der Mittelwert einer Zahlenkolonne bestimmen.

AVG

Liefert das arithmetische Mittel der Werte für die angegebene Gruppe.

Syntax:

```
AVG([ALL | DISTINCT] Ausdruck)
```

Argumente:

ALL wendet die Aggregatfunktion auf alle Werte an (Standardeinstellung).

DISTINCT wendet die Funktion nur auf eindeutig unterscheidbare Werte an, d.h. bei mehreren gleichen Werten nur für eine Instanz dieser Werte.

Der *Ausdruck* kann einen Wert eines numerischen Datentyps (außer bit) liefern. Aggregatfunktionen und Unterabfragen sind im Ausdruck nicht zulässig.

Rückgabetyp:

Der Typ des zurückgegebenen Wertes wird durch den Typ des Ausdrucks bestimmt (siehe Tabelle 11.1).

Ausdrucksergebnis	Rückgabetyp
Ganzzahltypen	int
decimal(p,s)	decimal(38, 2) geteilt durch decimal(10, 0)
Währungstypen	money
Gleitkommatypen	float

Tabelle 11.1: Rückgabetypen der Funktion AVG

Beispiele:

```
use pubs
select pub_id, title_id, price from titles
order by pub_id
compute avg(price) by pub_id
```

Diese Anweisung ruft die Spalten Verleger-ID (pub_id), Titelnummer (title_id) und Preis (price) aus der Tabelle titles der Datenbank pubs ab, gruppiert die Ergebnisse nach der Verleger-ID und berechnet den Mittelwert der Preise für die einzelnen Gruppen. Die Ausgabe sieht folgendermaßen aus:

```
pub_id title_id price
------ -------- ---------------------
0736   BU2075   2.9900
0736   PS2091   10.9500
0736   PS2106   7.0000
0736   PS3333   19.9900
0736   PS7777   7.9900

               avg
               ==================
               9.7840


0877   MC2222   19.9900
0877   MC3021   2.9900
0877   MC3026   NULL
0877   PS1372   21.5900
0877   TC3218   20.9500
0877   TC4203   11.9500
0877   TC7777   14.9900

               avg
               ==================
               15.4100


1389   BU1032   19.9900
1389   BU1111   11.9500
1389   BU7832   19.9900
1389   PC1035   22.9500
1389   PC8888   20.0000
1389   PC9999   NULL

               avg
               ==================
               18.9760


(21 row(s) affected)
```

```
Warnung: NULL-Wert aus dem Aggregat gelöscht.
```

Um die NULL-Zeilen von vornherein auszublenden, kann man eine WHERE-Klausel vorsehen:

```
use pubs
select pub_id, title_id, price from titles
where price is not null
order by pub_id
compute avg(price) by pub_id
```

Damit erhält man folgendes Ergebnis:

```
pub_id title_id price
------ -------- ---------------------
0736   BU2075   2.9900
0736   PS2091   10.9500
0736   PS2106   7.0000
0736   PS3333   19.9900
0736   PS7777   7.9900

               avg
               ==================
               9.7840


0877   MC2222   19.9900
0877   MC3021   2.9900
0877   PS1372   21.5900
0877   TC3218   20.9500
0877   TC4203   11.9500
0877   TC7777   14.9900

               avg
               ==================
               15.4100


1389   BU1032   19.9900
1389   BU1111   11.9500
1389   BU7832   19.9900
1389   PC1035   22.9500
1389   PC8888   20.0000
```

```
                avg
                ==================
                18.9760


(19 row(s) affected)
```

Die Anzahl der von einer Transact-SQL-Anweisung betroffenen Zeilen (rows affected) ergibt sich aus der Anzahl der zurückgegebenen Werte. Im ersten Beispiel liefert die Abfrage in den einzelnen Gruppen 5 + 7 + 6 = 18 Zeilen. Dazu kommen noch 3 Zeilen für die Mittelwerte. Im zweiten Beispiel wurden die NULL-Werte mit der WHERE-Klausel ausgeblendet. Es ergeben sich demnach nur 19 Zeilen.

In den folgenden Beispielen wurde die Rückgabe der betroffenen Zeilen mit der Anweisung SET NOCOUNT ON unterdrückt.

Die folgenden Beispiele zeigen den Unterschied zwischen den Optionen ALL und DISTINCT. Die erste Anweisung berechnet den Mittelwert über alle Preise der Spalte:

```
use pubs
select avg(price) from titles
where price is not null
```

Das Ergebnis lautet:

```
----------------------
14.7662
```

Die nächste Anweisung bildet den Mittelwert nur über die eindeutigen Werte. Wie Sie dem Beispiel weiter oben entnehmen können, kommt der Wert 19.99 viermal vor.

```
use pubs
select avg(distinct price) from titles
where price is not null
```

Damit erhält man:

```
---------------------
14.6681
```

COUNT

Gibt die Anzahl von Einträgen in einer Gruppe zurück.

Syntax:

```
COUNT({[ALL | DISTINCT] Ausdruck] | *})
```

Argumente:

ALL wendet die Aggregatfunktion auf alle Werte an (Standardeinstellung).

DISTINCT liefert die Anzahl der eindeutigen Werte, die nicht NULL sind. Mehrere gleiche Werte tragen nur eine 1 zum Ergebnis bei.

Der Ausdruck kann einen beliebigen Typ außer uniqueidentifier, text, image oder ntext zurückgeben. Aggregatfunktionen und Unterabfragen sind nicht zulässig.

Rückgabetyp:

Der zurückgegebene Wert ist vom Typ int.

Beispiele:

```
use pubs
select count(price) from titles
```

Gibt die Anzahl der Einträge in der Spalte price der Tabelle titles zurück. Als Ausgabe erhält man:

```
-----------
16
```

Die Anweisung

```
use pubs
select count(*) from titles
```

liefert die Gesamtzahl der Zeilen in der Tabelle titles:

```
-----------
18
```

Mit einer zusätzlichen WHERE-Klausel kann man die Anzahl der Zeilen ermitteln, die einer bestimmten Bedingung genügen.

```
use pubs
select count(price) from titles
where price = 19.99
```

Wie aus den vorherigen Beispielen zu erwarten ist, lautet das Ergebnis 4.

SUM

Gibt die Summe der spezifizierten Werte von numerischen Spalten zurück. NULL-Werte werden nicht berücksichtigt.

Syntax:

```
SUM([ALL | DISTINCT] Ausdruck)
```

Argumente:

ALL wendet die Aggregatfunktion auf alle Werte an (Standardeinstellung).

DISTINCT summiert nur die eindeutigen Werte. Von mehreren gleichen Werten geht also nur ein Wert in die Summe ein.

Der *Ausdruck* kann eine Konstante, Spalte, Funktion oder Kombination aus arithmetischen, bitweisen und Zeichenfolgenoperatoren darstellen und muß einen numerischen Typ (mit Ausnahme von bit) liefern. Aggregatfunktionen und Unterabfragen sind nicht zulässig.

Rückgabetyp:

Gibt die Summe aller im Ausdruck aufgeführten Werte zurück. Als Datentyp wird der genaueste Typ der Ausdrucksliste verwendet.

Beispiele:

```
use pubs
select sum(price) from titles
where price is not null
```

Liefert die Summe aller Preise in der Spalte price. Das Ergebnis lautet:

```
----------------------
236.2600
```

Die Anweisung

```
use pubs
select sum(distinct price) from titles
where price is not null
```

liefert dagegen nur 161.35, weil 19.99 viermal, 2.99 zweimal und 11.95 zweimal vorkommen, aber aufgrund des Schlüsselwortes DISTINCT nur jeweils einmal in die Summe einfließen.

MAX

Gibt den größten Wert der mit *Ausdruck* spezifizierten Werte zurück.

Syntax:

```
MAX([ALL | DISTINCT] Ausdruck)
```

Argumente:

ALL wendet die Aggregatfunktion auf alle Werte an (Standardeinstellung).

DISTINCT ist nur aus Gründen der Kompatibilität zu SQL-92 vorgesehen und hat keine praktische Bedeutung, weil es bei mehreren gleichen Werten im eigentlichen Sinne keinen größten Wert gibt.

Der *Ausdruck* kann eine Konstante, Spalte, Funktion oder Kombination aus arithmetischen, bitweisen und Zeichenfolgenoperatoren darstellen. Aggregatfunktionen und Unterabfragen sind nicht zulässig.

Die Funktion läßt sich nicht nur für numerische Spalten, sondern auch für Spalten mit Zeichen- und Datumstypen (jedoch nicht auf bit-Spalten) anwenden.

Rückgabetyp:

Der zurückgegebene Wert ist vom Typ des Ausdrucks.

Beispiele:

Das folgende Beispiel verwendet die Datenbank Lotto (die Sie im Verlauf der Kapitel 6 bzw. 9 erstellt haben) und ermittelt die Ziehung, in der die erste Zahl in numerischer Reihenfolge am größten ist:

```
use lotto
select Ziehung, z1, z2, z3, z4, z5, z6, zz from LDaten
where z1=(select max(z1) from LDaten)
```

Das Ergebnis lautet:

```
Ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
465         34   36   38   41   43   45   4
1732        34   35   39   47   48   49   32
```

MIN

Gibt den kleinsten der mit *Ausdruck* spezifizierten Werte zurück.

Syntax:

```
MIN([ALL | DISTINCT] Ausdruck)
```

Argumente:

ALL wendet die Aggregatfunktion auf alle Werte an (Standardeinstellung).

DISTINCT ist nur aus Gründen der Kompatibilität zu SQL-92 vorgesehen und hat wie bei MAX keine praktische Bedeutung.

Der *Ausdruck* kann eine Konstante, Spalte, Funktion oder Kombination aus arithmetischen, bitweisen und Zeichenfolgenoperatoren darstellen. Aggregatfunktionen und Unterabfragen sind nicht zulässig.

Die Funktion läßt sich nicht nur für numerische Spalten, sondern auch für Spalten mit Zeichen- und Datumstypen (jedoch nicht auf bit-Spalten) anwenden.

Rückgabetyp:

Der zurückgegebene Wert ist vom Typ des Ausdrucks.

Beispiele:

Die Anweisung bestimmt die Ziehung, in der die größte der geordneten Zahlen (ohne Beachtung der Zusatzzahl) am kleinsten ist:

```
use lotto
select Ziehung, z1, z2, z3, z4, z5, z6, zz from LDaten
where z6=(select min(z6) from LDaten)
```

Das Ergebnis lautet:

```
Ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
1841        2    5    6    8    10   16   41
```

VAR, VARP, STDEV, STDEVP

VAR gibt die Varianz, VARP die Varianz für die Auffüllung, STDEV die Standardabweichung und STDEVP die Standardabweichung für die Auffüllung aller Werte des angegebenen Ausdrucks zurück.

Die Varianz oder Streuung ist eine statistische Maßzahl für die Größe der Abweichung vom Mittelwert einer Zufallsgröße. Um die Varianz zu berechnen, bildet man die Quadrate der Abweichungen vom Mittelwert, multipliziert diese mit der zugehörigen Wahrscheinlichkeit und addiert diese Produkte. Die Standardabweichung oder mittlere quadratische Abweichung ist die Wurzel aus der Varianz.

Syntax:

```
VAR(Ausdruck)
VARP(Ausdruck)
STDEV(Ausdruck)
STDEVP(Ausdruck)
```

Argumente:

Der *Ausdruck* muß einen numerischen Datentyp (bit-Werte ausgenommen) liefern. Aggregatfunktionen und Unterabfragen sind nicht zulässig.

Rückgabewert:

Der zurückgegebene Wert ist vom Typ float.

Beispiele:

Die folgenden Beispiele berechnen die statistischen Maßzahlen für die ersten Zahlen der Ziehungen 2200 bis 2209: 1, 3, 14, 4, 12, 2, 15, 14, 5 und 12.

```
use lotto
select var(z1) from ldaten
where ziehung>=2200 and ziehung<2210
select varp(z1) from ldaten
where ziehung>=2200 and ziehung<2210
select stdev(z1) from ldaten
where ziehung>=2200 and ziehung<2210
select stdevp(z1) from ldaten
where ziehung>=2200 and ziehung<2210
```

Die Ergebnisse lauten:

```
-----------------------------------------------------
31.955555555555559


-----------------------------------------------------
28.760000000000002


-----------------------------------------------------
5.6529245135200208


-----------------------------------------------------
5.3628350711167689
```

11.4 Skalare Funktionen

Die umfangreichste Funktionsgruppe in Transact-SQL wird unter der Bezeichnung *skalare Funktionen* geführt. Die Funktionen dieser Gruppe lassen sich weiter in folgende Kategorien einordnen:

- Konfigurationsfunktionen
- Cursorfunktionen
- Datums- und Zeitfunktionen
- Mathematische Funktionen
- Metadatenfunktionen
- Sicherheitsfunktionen
- Zeichenfolgenfunktionen
- Systemfunktionen
- Statistische Systemfunktionen
- Text- und Image-Funktionen

11.4.1 Konfigurationsfunktionen

Die Funktionen dieser Gruppe geben Informationen über die aktuellen Konfigurationseinstellungen zurück. Die Namen der Funktionen beginnen mit zwei At-Zeichen. Ältere Versionen von SQL Server bezeichnen damit globale Variablen. Verschiedentlich findet man auch in der neueren Literatur zu SQL Server und selbst in der Online-Dokumentation die Bezeichnung *globale Variable* für die mit @@ beginnenden Bezeichner. Die Online-Dokumentation weist aber darauf hin, daß die globalen Variablen von Transact-SQL in der Version 7.0 den Charakter von Funktionen tragen. Daß es sich tatsächlich um Funktionen und nicht um Variablen handelt, soll das folgende Beispiel verdeutlichen:

Die Anweisung

```
print @@servername
```

zeigt den Namen des lokalen Servers an (siehe Funktionsbeschreibung weiter unten in diesem Abschnitt).

Probieren Sie nun folgende Anweisung aus:

```
set @@servername='VIER'
print @@servername
```

SQL Server führt diese Anweisungen gar nicht erst aus, sondern liefert folgende Fehlermeldung:

```
Server: Nachr.-Nr. 190, Schweregrad 15, Status 1, Zeile 1
```

```
Funktion '@@servername' kann nicht aktualisiert werden.
```

Natürlich kann man in SQL Server auch Variablen verwenden. Lokale Variablen erhalten dabei ein einzelnes At-Zeichen vor dem Variablennamen. Auf die Deklaration und Verwendung von Variablen geht Kapitel 12 näher ein.

Tabelle 11.2 faßt die Konfigurationsfunktionen zusammen.

Funktion	Rückgabewert / Beschreibung
@@CONNECTIONS	Anzahl der Verbindungen oder versuchten Verbindungen seit dem letzten Start von SQL Server
@@DATEFIRST	aktueller Wert des SET DATEFIRST-Parameters
@@DBTS	aktueller Wert vom Typ timestamp für die aktuelle Datenbank
@@LANGID	Lokalsprachen-ID der momentan verwendeten Sprache
@@LANGUAGE	Name der momentan verwendeten Sprache
@@LOCK_TIMEOUT	maximale Zeitspanne, die für eine Blockierung von Ressourcen eingestellt ist
@@MAX_CONNECTIONS	Anzahl der zulässigen gleichzeitigen Benutzerverbindungen zu SQL Server
@@MAX_PRECISION	Grad der Genauigkeit für die Datentypen decimal und numeric
@@NESTLEVEL	Schachtelungsebene der aktuellen Ausführung einer gespeicherten Prozedur
@@OPTIONS	Informationen über aktuelle Optionen von SET
@@REMSERVER	Name des Remotedatenbankservers
@@SPID	Serverprozeß-ID des aktuellen Benutzerprozesses
@@SERVERNAME	Name des lokalen Servers
@@SERVICENAME	Name des Registrierungsschlüssels
@@TEXTSIZE	aktueller Wert der SET TEXTSIZE-Option
@@VERSION	Datum, Versionsnummer und Prozessortyp der aktuellen Installation von SQL Server

Tabelle 11.2: Konfigurationsfunktionen

@@CONNECTIONS

Gibt die Anzahl der Verbindungen oder versuchten Verbindungen seit dem letzten Start von SQL Server

zurück.

Die Anzahl der Verbindungen ist nicht gleich der Anzahl der Benutzer. Ein und derselbe Benutzer kann - zum Beispiel über eine Client-Anwendung - mehrere Verbindungen zu SQL Server aufbauen (siehe auch @@MAX_ CONNECTIONS weiter unten in diesem Kapitel).

Beispiel:

```
select @@connections as Anmeldungen
```

liefert das Ergebnis

```
Anmeldungen
-----------
229
```

Es bestehen oder bestanden 229 Anmeldungen zu SQL Server. Einen Bericht mit mehreren Statistiken zu SQL Server erhalten Sie mit der gespeicherten Prozedur sp_monitor.

@@DATEFIRST

Gibt die aktuelle Einstellung für den ersten Tag einer Woche an, wobei die Zuordnung 1 = Montag, 2 = Dienstag, ... , 7 = Sonntag gilt.

Beispiel:

Wenn als Ländereinstellung Deutschland gilt, liefert die Anweisung

```
select @@datefirst as 'Erster Wochentag'
```

das Ergebnis

```
Erster Wochentag
----------------
1
```

Die Woche fängt in Deutschland also mit Montag an. Diese Einstellung ist von Bedeutung, wenn man zum Beispiel in einem Bericht den soundsovielten Wochentag für eine Auswertung heranzieht.

@@DBTS

Gibt den zuletzt verwendeten Wert zurück, den eine Spalte vom Datentyp timestamp beim Einfügen oder Aktualisieren einer Zeile erhalten hat. Dieser Wert ist für eine Datenbank eindeutig.

Beispiel:

Die folgende Anweisung gibt den zuletzt in der Datenbank pubs verwendeten Timestamp-Wert zurück:

```
use pubs
select @@dbts as 'TimeStamp'
```

Das Ergebnis lautet:

```
TimeStamp
------------------
0x000000000000012C
```

@@LANGID

Gibt die Lokalsprachen-ID der derzeit verwendeten Sprache zurück.

Beispiel:

```
select @@langid as 'Sprachen-ID'
```

```
Sprachen-ID
-----------
1
```

Eine Tabelle der Sprachen-IDs finden Sie in Kapitel 19 bei der Behandlung der Systemtabelle syslanguages.

@@LANGUAGE

Gibt den Namen der momentan verwendeten Sprache zurück.

Beispiel:

Die Anweisung

```
select @@language as Sprache
```

liefert das Ergebnis:

```
Sprache
--------
Deutsch
```

@@LOCK_TIMEOUT

Mit dieser Funktion läßt sich die maximale Zeitspanne ermitteln, die für eine Blockierung von Ressourcen eingestellt ist. Eine Anwendung kann die Zeitdauer mit SET LOCK_TIMEOUT festlegen. Wenn die Anwendung länger warten muß, bricht SQL Server die blockierte Anweisung ab und gibt eine Fehlermeldung zurück.

Beispiel:

Die Anweisung

```
select @@lock_timeout as Sperrzeit
```

liefert zu Beginn einer Verbindung das Ergebnis:

```
Sperrzeit
-----------
-1
```

Das gilt unter der Voraussetzung, daß noch kein Wert für LOCK_TIMEOUT festgelegt wurde.

@@MAX_CONNECTIONS

Gibt die Anzahl der gleichzeitig zulässigen Benutzerverbindungen zu SQL Server zurück.

Beispiel:

Die Anweisung

```
select @@max_connections as 'Maximalzahl Benutzerverbindungen'
```

liefert das Ergebnis:

```
Maximalzahl Benutzerverbindungen
--------------------------------
32767
```

Die tatsächliche Anzahl der zulässigen Benutzerverbindungen muß nicht unbedingt der derzeit konfigurierten Anzahl entsprechen und ist außerdem von der eingesetzten SQL-Server-Version abhängig.

@@MAX_PRECISION

Die Funktion liefert den Grad der Genauigkeit, der für die Datentypen decimal und numeric (siehe Kapitel 8) eingestellt ist. In der Voreinstellung beträgt der Genauigkeitsgrad 28 und läßt sich bis auf höchstens 38 erweitern.

Beispiel:

Mit der Anweisung

```
select @@max_precision as 'Genauigkeit'
```

erhält man das Ergebnis:

```
Genauigkeit
-----------
28
```

@@NESTLEVEL

Gibt die Schachtelungsebene der aktuellen Ausführung einer gespeicherten Prozedur zurück.

Beispiel:

```
select @@nestlevel as Prozedurebene
```

liefert das Ergebnis:

```
Prozedurebene
-------------
0
```

Die anfängliche Schachtelungsebene ist 0. Weitere Hinweise zum Verschachteln von gespeicherten Prozeduren finden Sie in Kapitel 12.

@@OPTIONS

Liefert Informationen über aktuelle Optionen von SET zurück. Jeder Option ist eine Bitposition zugeordnet (siehe Tabelle 11.3).

Wert	Name	Beschreibung
1	DISABLE_DEF_ CNST_CHK	steuert die Einschränkungsüberprüfung
2	IMPLICIT_TRANSACTIONS	steuert den impliziten Start einer Transaktion bei Ausführung einer Anweisung
4	CURSOR_CLOSE_ ON_COMMIT	steuert das Verhalten eines Cursors, wenn ein Commit-Befehl ausgelöst wird
8	ANSI_WARNINGS	steuert das Abschneiden und NULL in Aggregatwarnungen
16	ANSI_PADDING	steuert das Auffüllen mit Leerzeichen oder Nullen bei Variablen fester Länge
32	ANSI_NULLS	steuert die Verarbeitung von NULL in Vergleichen
64	ARITHABORT	bricht eine Abfrage ab, wenn während der Ausführung ein Fehler durch Überlauf oder Division durch Null aufgetreten ist
128	ARITHIGNORE	gibt bei einer Abfrage, die zu einem Fehler durch Überlauf oder Division durch Null geführt hat, den Wert NULL zurück
256	QUOTED_IDENTIFIER	unterscheidet zwischen einfachen und doppelten Anführungszeichen bei der Auswertung von Ausdrücken
512	NOCOUNT	unterdrückt die Ausgabe der betroffenen Zeilen am Ende einer ausgeführten Anweisung

1024	ANSI_NULL_DFLT_ON	stellt ANSI-Kompatibilität hinsichtlich NULL-Zulässigkeit ein (für neue Spalten gilt NULL-Zulässigkeit, auch wenn diese nicht ausdrücklich festgelegt ist)
2048	ANSI_NULL_DFLT_OFF	schaltet ANSI-Kompatibilität hinsichtlich NULL-Zulässigkeit ab (für neue Spalten gilt keine NULL-Zulässigkeit, wenn sie nicht ausdrücklich definiert ist)

Tabelle 11.3: Bitposition der Benutzeroptionen

Beispiele:

Das erste Beispiel schaltet die Anzeige der betroffenen Zeilen ein (falls nicht ohnehin die Standardeinstellung wirksam ist) und zeigt den Wert für user_options an:

```
set nocount off
select @@options as Optionen
```

Wenn Sie die Anzeige *ein*schalten, ist der Parameter off (*aus*) anzugeben. Genaugenommen schalten Sie also die Nichtanzeige aus.

Die Ergebnismenge hat folgendes Aussehen:

```
Optionen
-----------
1080

(1 row(s) affected)
```

Das zweite Beispiel schaltet die Anzeige der betroffenen Zeilen ab und zeigt den neuen Wert der user_options an:

```
set nocount on
select @@options as Optionen
```

Zum bisherigen Wert (1080) wird der Wert für die Bitposition NOCOUNT (512) addiert. Das Ergebnis lautet:

```
Optionen
```

```
-----------
1592
```

11.4.2 Cursorfunktionen

Zu den Cursorfunktionen gehören

- @@CURSOR_ROWS
- CURSOR_STATUS
- @@FETCH_STATUS

Eine Beschreibung dieser Funktionen finden Sie im Abschnitt »Cursorfunktionen« von Kapitel 15.

11.4.3 Datums- und Zeitfunktionen

Die Funktionen dieser Kategorie erlauben Manipulationen mit Datumswerten. Einige Funktionen verwenden die in Tabelle 11.4 gezeigten Datumseinheiten.

Datumseinheit (Abkürzung)	Beschreibung (Bereich)
year (yy, yyyy)	Jahr (1753 bis 9999)
quarter (qq, q)	Quartal (1 bis 4)
month (mm, m)	Monat (1 bis 12)
dayofyear (dy, y)	Tag des Jahres (1 bis 366)
day (dd, d)	Tag im Monat (1 bis 31)
week (wk, ww)	Kalenderwoche (1 bis 53)
weekday (dw)	Wochentag (1 bis 7)
hour (hh)	Stunde (0 bis 23)
minute (mi, n)	Minute (0 bis 59)
second (ss, s)	Sekunde (0 bis 59)
millisecond (ms)	Millisekunde (0 bis 999)

Tabelle 11.4: Datumseinheiten

DATEADD

Die Funktion DATEADD addiert für ein bestimmtes Datum zur spezifizierten Datumseinheit die angegebene Zahl von Einheiten und liefert einen neuen datetime-Wert zurück:

```
DATEADD (Datumseinheit, Anzahl, Datum)
```

Die folgende Anweisung addiert zum 28. Februar 1999 einen Tag:

```
select dateadd(dd,1,'28.2.1999')
```

Das Ergebnis ist wie erwartet der 1.3.1999.

Mit dieser Funktion können Sie auch die Tauglichkeit für das Jahr 2000 testen.

DATEDIFF

Die Funktion DATEDIFF gibt die Anzahl der überschrittenen Datumseinheiten zwischen zwei Daten zurück:

```
DATEDIFF (Datumseinheit, Startdatum, Endedatum)
```

Die folgende Anweisung berechnet die Anzahl der Tage zwischen dem 24. Dezember 1999 und dem 6. Januar 2000:

```
select datediff(dd,'24.12.1999','6.1.2000')
```

Das Ergebnis lautet 13.

DATENAME

Die Funktion DATENAME gibt eine Zeichenfolge für die angegebene Datumseinheit zurück:

```
DATENAME (Datumseinheit, Datum)
```

Das Quartal für das Datum 20.04.1999 läßt sich wie folgt anzeigen:

```
select datename(quarter,'20.4.99')
```

Das Ergebnis lautet:

```
------------------------------
2
```

Wenn Sie wissen wollen, der wievielte Tag des Jahres der 6. September ist, verwenden Sie die Anweisung:

```
select datename(y,'6.9.99')
```

Als Ergebnis erhalten Sie 249.

Und auf welchen Wochentag fällt der 1.1.2000?

```
select datename(dw,'1.1.2000')

------------------------------
Samstag
```

DATEPART

Während die Funktion DATENAME eine Zeichenfolge zurückgibt, liefert die Funktion DATEPART eine ganze Zahl für die angegebene Datumseinheit des übergebenen Datums zurück. Die übrige Funktionalität entspricht der Funktion DATENAME.

DAY, MONTH, YEAR

Diese Funktionen geben eine ganze Zahl für die jeweilige Datumseinheit (Tag, Monat bzw. Jahr) des angegebenen Datums zurück:

```
DAY (Datum)
MONTH (Datum)
YEAR (Datum)
```

Zum Beispiel gibt die Anweisung

```
select month('17.7.1999')
```

den Wert 7 zurück.

GETDATE

Die Funktion GETDATE liefert das aktuelle Systemdatum und die aktuelle Uhrzeit in Form eines Wertes vom Typ datetime zurück.

Die Anweisung

```
select getdate()
```

liefert zum Beispiel:

```
---------------------------
1999-03-14 21:32:47.403
```

Die Funktion GETDATE (inklusive der Klammern) können Sie auch in den bisher behandelten Datumsfunktionen als Parameter *Datum* einsetzen. Beispielsweise ermittelt die folgende Anweisung die Anzahl der Tage vom aktuellen Datum bis zum 1.1.2000:

```
select datediff(dd,getdate(),'1.1.2000')
```

11.4.4 Mathematische Funktionen

Die mathematischen Funktionen

Funktion	Parameter	Rückgabewert / Beschreibung
ABS	(numerischer_Ausdruck)	Absolutwert des Ausdrucks
ACOS	(float_Ausdruck)	Winkel im Bogenmaß, dessen Cosinus der angegebene Ausdruck ist (Arkuscosinus)
ASIN	(float_Ausdruck)	Winkel im Bogenmaß, dessen Sinus der angegebene Ausdruck ist (Arkussinus)
ATAN	(float_Ausdruck)	Winkel im Bogenmaß, dessen Tangens der angegebene Ausdruck ist (Arkustangens)

ATN2	(float_Ausdr1, float_Ausdr2)	Winkel im Bogenmaß, dessen Tangens gleich float_Ausdr1 / float_Ausdr2 ist (Arkustangens)
CEILING	(numerischer_Ausdruck)	Kleinste Ganzzahl, die größer oder gleich dem angegebenen Ausdruck ist
COS	(float_Ausdruck)	Cosinus des im Ausdruck angegebenen Winkels im Bogenmaß
COT	(float_Ausdruck)	Cotangens des im Ausdruck angegebenen Winkels im Bogenmaß
DEGREES	(numerischer_Ausdruck)	wandelt Bogenmaß in Grad um
EXP	(float_Ausdruck)	die im Ausdruck angegebene Potenz zur Basis e (2,71828...)
FLOOR	(numerischer_Ausdruck)	Größte Ganzzahl, die kleiner oder gleich dem angegebenen Ausdruck ist
LOG	(float_Ausdruck)	natürlicher Logarithmus zum angegebenen Ausdruck
LOG10	(float_Ausdruck)	dekadischer Logarithmus zum angegebenen Ausdruck
PI	()	konstanter Wert Pi (3,14159265358979)
POWER	(numerischer_Ausdruck, y)	Wert des Ausdrucks zur Potenz y; der Datentyp des Rückgabewertes entspricht dem Datentyp des Ausdrucks
RADIANS	(numerischer_Ausdruck)	wandelt Grad in Bogenmaß um
RAND	([Startwert])	zufälliger float-Wert zwischen 0 und 1; der Startwert ist ganzzahlig (tinyint, smallint oder int)
ROUND	(numerischer_Ausdruck, Länge [, Operation])	rundet den Ausdruck auf die angegebene Länge oder Genauigkeit, wenn Operation nicht spezifiziert oder 0 ist; bei Operation ungleich 0 wird der Ausdruck abgeschnitten

SIGN	(numerischer_Ausdruck)	Vorzeichen des Ausdrucks: 1, wenn Ausdruck positiv 0, wenn Ausdruck null -1, wenn Ausdruck negativ
SIN	(float_Ausdruck)	Sinus des angegebenen Winkels im Bogenmaß
SQUARE	(float_Ausdruck)	Quadrat des Ausdrucks
SQRT	(float_Ausdruck)	Quadratwurzel des Ausdrucks
TAN	(float_Ausdruck)	Tangens des Ausdrucks

Tabelle 11.5: Mathematische Funktionen

Die als *numerischer_Ausdruck* bezeichneten Parameter können vom Datentyp INTEGER, MONEY, REAL und FLOAT sein, wenn nichts anderes angegeben ist.

Zufallszahlen

Mit der Funktion RAND lassen sich Zufallszahlen nur im Bereich zwischen 0 und 1 erzeugen. Wenn Sie Zufallszahlen für einen beliebigen Bereich brauchen, verwenden Sie den folgenden Ausdruck:

```
Zufallszahl = (Obergrenze - Untergrenze) * RAND() + Untergrenze
```

Der Bereich der Zufallszahlen läuft von *Untergrenze* bis *Obergrenze*. Zum Beispiel lassen sich ganzzahlige Zufallszahlen im Bereich von 1 bis 49 mit folgender Anweisung erzeugen:

```
select round((49-1)*rand()+1,0,0)
```

11.4.5 Metadatenfunktionen

Die Metadatenfunktionen geben Informationen über die Datenbank und Datenbankobjekte zurück. Tabelle 11.6 faßt die Metadatenfunktionen zusammen.

Funktion	Parameter	Rückgabewert/Beschreibung
COL_LENGTH	('Tabelle', 'Spalte')	Länge der Spalte
COL_NAME	(Tabellen-ID, Spalten-ID)	Name der Datenbankspalte
COLUMNSPROPERTY	(ID, Spalte, Eigenschaft)	Informationen über einen Spalten- oder Prozedurparameter

DATABASEPROPERTY	(Datenbank, Eigenschaft)	Wert der benannten Datenbankeigenschaft für den angegebenen Datenbank- und Eigenschaftsnamen
DB_ID	('Datenbankname')	Datenbank-ID
DB_NAME	(Datenbank-ID)	Datenbankname
FILE_ID	('Dateiname')	Datei-ID für den logischen Dateinamen
FILE_NAME	(Datei-ID)	logischer Dateiname für die angegebene Datei-ID
FILEGROUP_ID	('Dateigruppenname')	Dateigruppen-ID
FILEGROUP_NAME	(Dateigruppen-ID)	Dateigruppenname
FILEGROUPPROPERTY	(Dateigruppenname, Eigenschaft)	Eigenschaftswert für den angegebenen Dateigruppen- und Eigenschaftsnamen
FILEPROPERTY	(Dateiname, Eigenschaft)	Eigenschaftswert für den angegebenen Dateinamen
FULLTEXTCATALOGPROPERTY	(Katalogname, Eigenschaft)	Informationen über die Eigenschaften des Volltextkatalogs
FULLTEXTSERVICEPROPERTY	(Eigenschaft)	Informationen über die Eigenschaften der Volltextdienstebene
INDEX_COL	('Tabelle', Index-ID, Schlüssel-ID)	Name einer indizierten Spalte
INDEXPROPERTY	(Tabellen-ID, Index, Eigenschaft)	benannter Indexeigenschaftswert
OBJECT_ID	('Objekt')	ID des Datenbankobjekts
OBJECT_NAME	(Objekt-ID)	Datenbankobjektname
OBJECTPROPERTY	(ID, 'Eigenschaft')	Informationen zu Objekten der aktuellen Datenbank
@@PROCID		ID der gespeicherten Prozedur für die aktuelle Prozedur
TYPEPROPERTY	(Datentyp, Eigenschaft)	Informationen zum Datentyp

Tabelle 11.6: Metadatenfunktionen

Beispiele:

Die Länge der Spalte title_id der Tabelle titles in der aktuellen Datenbank (pubs) liefert die folgende Anweisung:

```
select col_length('titles', 'title_id')
```

Als Ergebnis erhalten Sie:

```
------
6
```

Die nächste Anweisung ruft die eingestellte Genauigkeit für den Datentyp float ab:

```
select typeproperty('float', 'precision')
```

Das Ergebnis lautet:

```
-----------
53
```

11.4.6 Sicherheitsfunktionen

Die Funktionen dieser Kategorie geben Informationen über Benutzer und Rollen zurück. Tabelle 11.7 faßt die Sicherheitsfunktionen zusammen.

Funktion	Parameter	Rückgabewert/Beschreibung
IS_MEMBER	({'Gruppe' | 'Rolle'})	zeigt an, ob der aktuelle Benutzer Mitglied der angegebenen Windows-NT-Gruppe oder der SQL-Server-Rolle ist
IS_SRVROLEMEMBER	('Rolle' [, 'Anmeldung'])	zeigt an, ob der aktuelle Benutzername zur angegebenen Server-Rolle gehört
SUSER_ID (*)	(['Anmeldung'])	Benutzername-ID des Benutzers -> SUSER_SID

SUSER_NAME (*)	([Benutzernamen-ID])	Benutzername des Benutzers -> SUSER_SNAME
SUSER_SID	(['Anmeldung'])	Sicherheits-ID für den Benutzernamen des Benutzers
SUSER_SNAME	([Sicherheits-ID])	Benutzername für die Sicherheits-ID eines Benutzers
USER_ID	(['Benutzer'])	Datenbank-ID des Benutzers

Tabelle 11.7: Sicherheitsfunktionen

Ein Sternchen nach dem Funktionsnamen gibt an, daß die Funktion nur aus Gründen der Abwärtskompatibilität bereitgestellt wird. In der Spalte *Beschreibung* ist ein Verweis auf die in SQL Server 7.0 zu bevorzugende Funktion angegeben.

11.4.7 Zeichenfolgenfunktionen

Die Funktionen dieser Kategorie sind vorrangig für Operationen auf Zeichendaten vorgesehen. Zum Beispiel kann man Zeichenfolgen verketten, Teile aus Zeichenfolgen heraustrennen oder Steuerzeichen in eine auszugebende Meldung einfügen.

ASCII

Gibt den ASCII-Wert des ersten Zeichens im Zeichenausdruck zurück.

Syntax:

```
ASCII(Zeichenausdruck)
```

Der *Zeichenausdruck* ist ein Ausdruck vom Typ char oder varchar. Die Funktion gibt einen Wert vom Typ int zurück.

Beispiel:

```
use pubs
select type, ascii(type)as 'ASCII' from titles
```

Diese Anweisung ermittelt den ASCII-Wert für das erste Zeichen der Einträge in der Spalte type und gibt die Werte in der berechneten Spalte ASCII aus. Der Spaltenname ASCII ist in einfache Anführungszeichen einzuschließen, da es sich um ein reserviertes Wort handelt. Die ersten Zeilen der Ergebnismenge lauten:

```
type         ASCII
------------ -----------
business     98
business     98
business     98
business     98
mod_cook     109
mod_cook     109
UNDECIDED    85
popular_comp 112
```

CHAR

Die Funktion CHAR stellt das Gegenstück zu ASCII dar. Sie konvertiert einen ASCII-Wert in ein Zeichen. Mit dieser Funktion kann man zum Beispiel Steuerzeichen in eine Zeichenfolge einfügen, um das Druckbild zu beeinflussen.

Syntax:

```
CHAR(Ausdruck)
```

Der *Ausdruck* muß einen ganzzahligen Wert im Bereich 0 bis 255 ergeben. Andernfalls liefert die Funktion den Wert NULL zurück. Normalerweise ist der Rückgabetyp der Funktion ein Wert vom Typ char(1).

Beispiel:

Die folgende Anweisung gibt eine Meldung in zwei Zeilen aus. Nach der ersten Zeile bewirkt die Funktion char(10) die Ausgabe des Steuerzeichens für Zeilenvorschub.

```
print 'Erste Zeile' + char(10) + 'Zweite Zeile'
```

Das Ergebnis lautet:

```
Erste Zeile
Zweite Zeile
```

CHARINDEX

Die Funktion CHARINDEX gibt die Anfangsposition eines Ausdrucks in einer Zeichenfolge zurück. Die Arbeitsweise ist mit den Funktionen InStr in Basic oder Pos in Pascal vergleichbar. Die Syntax lautet:

```
CHARINDEX (Ausdruck1, Ausdruck2 [, Startposition])
```

Der *Ausdruck2* ist gewöhnlich eine Spalte, die nach *Ausdruck1* zu durchsuchen ist. Optional können Sie eine *Startposition* angeben, ab der die Suche in *Ausdruck2* beginnen soll.

Das folgende Beispiel sucht in der Spalte title der Tabelle titles (Datenbank pubs) das erste Auftreten der Zeichenfolge 'Computer':

```
use pubs
select convert(char(40),title) as Titel,
       charindex('Computer',title) as Position
from titles
```

Das Ergebnis lautet:

```
Titel                                    Position
---------------------------------------- -----------
But Is It User Friendly?                 0
Computer Phobic AND Non-Phobic Individua 1
Cooking with Computers: Surreptitious Ba 14
Emotional Security: A New Algorithm      0
Fifty Years in Buckingham Palace Kitchen 0
Is Anger the Enemy?                      0
Life Without Fear                        0
Net Etiquette                            0
Onions, Leeks, and Garlic: Cooking Secre 0
Prolonged Data Deprivation: Four Case St 0
Secrets of Silicon Valley                0
Silicon Valley Gastronomic Treats        0
Straight Talk About Computers            21
Sushi, Anyone?                           0
The Busy Executive's Database Guide      0
The Gourmet Microwave                    0
The Psychology of Computer Cooking       19
You Can Combat Computer Stress!          16

(18 row(s) affected)
```

Den von CHARINDEX zurückgegebenen Wert können Sie auch in anderen Funktionen verwenden.

PATINDEX

Die Funktion liefert die Startposition für das erste Auftreten eines Musters (pattern) im angegebenen Ausdruck. Ist das Muster nicht im Ausdruck enthalten, gibt die Funktion NULL zurück. Als Argumente sind alle gültigen Text- und Zeichentypen möglich.

Syntax:

```
PATINDEX('%Muster%', Ausdruck)
```

Muster ist eine Literalzeichenfolge, die Platzhalterzeichen enthalten kann, oder ein Ausdruck, der einen Zeichendatentyp (kurze Zeichen) zurückgibt. Die Prozentzeichen sind erforderlich, außer wenn Sie nach den ersten oder letzten Zeichen suchen. Bei der Suche nach den ersten Zeichen kann das erste Prozentzeichen entfallen und bei der Suche nach den letzten Zeichen das zweite.

Die möglichen Platzhalterzeichen finden Sie in Tabelle 10.2 von Kapitel 10.

Ausdruck bezeichnet normalerweise eine Spalte, die nach dem Muster zu durchsuchen ist. Der Ausdruck muß einen Zeichenfolgentyp liefern.

Das obige Beispiel zur Funktion CHARINDEX läßt sich mit der Funktion PATINDEX wie folgt formulieren:

```
select convert(char(40),title) as Titel,
       patindex('%Computer%',title) as Position
```

Die Ergebnismenge entspricht der bei CHARINDEX gezeigten.

Gegenüber CHARINDEX ist die Funktion PATINDEX flexibler einsetzbar, weil sie Platzhalterzeichen erlaubt und auch mit text-Datentypen umgehen kann.

LEFT, RIGHT

Die Funktion LEFT liefert den linken Teil, die Funktion RIGHT den rechten Teil einer Zeichenfolge zurück:

```
LEFT (Zeichenausdruck, Ganzzahlausdruck)
RIGHT (Zeichenausdruck, Ganzzahlausdruck)
```

Der *Ganzzahlausdruck* gibt die Anzahl der Zeichen an, die von links bzw. rechts beginnend aus dem *Zeichenausdruck* zurückzugeben sind.

Die folgende Anweisung gibt die Vornamen aller Autoren sowie die ersten drei und die letzten drei Zeichen der Vornamen aus:

```
use pubs
select au_fname as Vorname,
       left(au_fname,3) as Links,
       right(au_fname,3) as Rechts
from authors
```

Das Ergebnis finden Sie weiter unten bei der Beschreibung der Funktionen LOWER und UPPER.

LEN

Die Funktion LEN gibt die Anzahl der Zeichen (nicht die Anzahl der Bytes) eines Zeichenfolgenausdrucks zurück. Nachgestellte Leerzeichen werden nicht mitgezählt.

```
LEN (Zeichenfolgenausdruck)
```

Die Anweisung

```
select len(au_fname) from authors
```

gibt die Länge der Vornamen in der Tabelle authors an. Das Ergebnis ist zusammen mit anderen Ergebnissen bei der Beschreibung der Funktionen LOWER und UPPER zu finden.

LOWER, UPPER

Die Funktion LOWER gibt eine Zeichenfolge zurück, die durchgängig in Kleinbuchstaben umgewandelt ist. Die Funktion UPPER wandelt alle Zeichen in Großbuchstaben um:

```
LOWER (Zeichenfolgenausdruck)
UPPER (Zeichenfolgenausdruck)
```

Zum Beispiel wandelt die Anweisung

```
select lower(au_fname) from authors
```

alle Zeichen in der Spalte au_fname der Tabelle authors in Kleinbuchstaben um.

Die folgende Anweisung faßt die Funktionen LEFT, RIGHT, LEN, LOWER und UPPER in einem Beispiel zusammen:

```
use pubs
select au_fname as Vorname,
       left(au_fname,3) as Links,
       right(au_fname,3) as Rechts,
       len(au_fname) as Länge,
       lower(au_fname) as Klein,
       upper(au_fname) as Groß
from authors
```

Die Ergebnismenge sieht wie folgt aus:

```
Vorname       Links Rechts Länge Klein                Groß
------------- ----- ------ ----- -------------------- ---------
Abraham       Abr   ham    7     abraham              ABRAHAM
Reginald      Reg   ald    8     reginald             REGINALD
Cheryl        Che   ryl    6     cheryl               CHERYL
Michel        Mic   hel    6     michel               MICHEL
Innes         Inn   nes    5     innes                INNES
Ann           Ann   Ann    3     ann                  ANN
...
Anne          Ann   nne    4     anne                 ANNE
Meander       Mea   der    7     meander              MEANDER
Dean          Dea   ean    4     dean                 DEAN
Dirk          Dir   irk    4     dirk                 DIRK
Johnson       Joh   son    7     johnson              JOHNSON
Akiko         Aki   iko    5     akiko                AKIKO

(23 row(s) affected)
```

LTRIM, RTRIM

Die Funktionen LTRIM und RTRIM entfernen führende bzw. nachgestellte Leerzeichen aus einer

Zeichenfolge:

```
LTRIM (Zeichenfolgenausdruck)
RTRIM (Zeichenfolgenausdruck)
```

Die Anweisung

```
select '->' + '   ABC   ' + '<-' as Leerzeichen,
       '->' + ltrim('   ABC   ') + '<-' as 'LTRIM',
       '->' + rtrim('   ABC   ') + '<-' as 'RTRIM',
       '->' + ltrim(rtrim('   ABC   ')) + '<-' as 'LTRIM und
RTRIM'
```

demonstriert, wie sich die Funktionen LTRIM und RTRIM auf die zurückgegebene Zeichenfolge auswirken:

```
Leerzeichen   LTRIM         RTRIM         LTRIM und RTRIM
------------- ------------- ------------- ---------------
->   ABC   <- ->ABC   <-    ->   ABC<-    ->ABC<-
```

NCHAR, UNICODE

Die Funktion NCHAR gibt ein Unicode-Zeichen entsprechend dem übergebenen *Ganzzahlausdruck* zurück:

```
NCHAR (Ganzzahlausdruck)
```

Die Anweisung

```
select nchar(65)
```

liefert das Zeichen A.

Die Funktion UNICODE ist das Gegenstück zu NCHAR und gibt für ein Unicode-Zeichen die zugehörige Ganzzahl zurück:

```
UNICODE ('Unicode-Ausdruck')
```

Die Anweisung

```
select unicode('A')
```

gibt den Wert 65 zurück.

REPLACE

Die Zeichenfolgenfunktion REPLACE mit der Syntax

```
REPLACE ('Zeichenfolgenausdruck1',
         'Zeichenfolgenausdruck2',
         'Zeichenfolgenausdruck3')
```

ersetzt alle Vorkommen von *Zeichenfolgenausdruck2* in *Zeichenfolgenausdruck1* durch *Zeichenfolgenausdruck3*.

REPLICATE

Die Funktion REPLICATE wiederholt einen Zeichenfolgenausdruck entsprechend der angegebenen Anzahl:

```
REPLICATE (Zeichenfolgenausdruck, Anzahl)
```

Die Anweisung

```
select replicate('ha ',10)
```

liefert das Ergebnis:

```
------------------------------
ha ha ha ha ha ha ha ha ha ha
```

REVERSE

Mit der Funktion REVERSE läßt sich ein Zeichenfolgenausdruck in umgekehrter Reihenfolge der Zeichen zurückgeben:

```
REVERSE (Zeichenfolgenausdruck)
```

Die Anweisung

```
select reverse('ABCDEFGHIJKLMN')
```

gibt die Buchstaben in umgekehrter Reihenfolge aus:

```
--------------
NMLKJIHGFEDCBA
```

SOUNDEX

Diese Funktion gibt einen vierstelligen Code zurück, der nach dem sogenannten Soundex-Algorithmus gebildet wird. Mit dieser Funktion lassen sich Zeichenfolgen auf Ähnlichkeit untersuchen. Das ist zum Beispiel bei einer phonetischen Suche nach Namen wie Meier, Meyer, Maier oder Mayer nützlich.

Die Syntax der Funktion lautet:

```
SOUNDEX (Zeichenausdruck)
```

Für die oben angegebenen Namen ermitteln Sie die Soundex-Werte wie folgt:

```
select soundex('Meier') as Meier,
       soundex('Meyer') as Meyer,
       soundex('Maier') as Maier,
       soundex('Mayer') as Mayer
```

Das Ergebnis lautet:

```
Meier Meyer Maier Mayer
```

```
----- ----- ----- -----
M600  M600  M600  M600
```

Dagegen liefert die Anweisung

```
select soundex('Meister') as Meister,
       soundex('Meisser') as Meisser,
       soundex('Meissner') as Meissner,
       soundex('Messner') as Messner
```

folgende Werte:

```
Meister Meisser Meissner Messner
------- ------- -------- -------
M236    M260    M256     M256
```

Die Auswertung von Soundex-Werten können Sie mit der nachstehend beschriebenen Funktion DIFFERENCE vornehmen.

DIFFERENCE

Die Funktion DIFFERENCE liefert den Unterschied zwischen zwei Werten, die nach dem Soundex-Algorithmus ermittelt wurden:

```
DIFFERENCE (Zeichenausdruck1, Zeichenausdruck2)
```

Da ein Soundex-Wert aus vier Zeichen besteht und die Funktion DIFFERENCE die Anzahl der in beiden Ausdrücken übereinstimmenden Zeichen ermittelt, liegt der Rückgabewert im Bereich zwischen 0 und 4. Dabei bedeutet 4 die größte Übereinstimmung.

Die bei der Funktion SOUNDEX ermittelten Werte für Meier & Co. von durchweg M600 deuten auf eine große Übereinstimmung hin, wie es das folgende Beispiel mit Hilfe der Funktion DIFFERENCE zeigt:

```
select difference('Meier', 'Mayer')
```

Das Ergebnis lautet 4, also größte Übereinstimmung - jedenfalls nach dem Soundex-Algorithmus. Zum

Vergleich noch Meister und Meissner:

```
select difference('Meister', 'Meisser')
```

Hier ergibt sich nur ein Wert von 3.

Wie Sie die Unterschiede interpretieren, hängt zum großen Teil von der konkreten Anwendung ab. Zur Erkennung von Eingabefehlern sind diese Funktionen allerdings weniger geeignet, zur fehlertoleranten Suche in einem Namensverzeichnis dagegen hervorragend.

SPACE

Die Funktion SPACE liefert eine Zeichenfolge mit der angegebenen Anzahl von Leerzeichen:

```
SPACE (Anzahl)
```

Die folgende Anweisung fügt zwischen 'A' und 'B' einen Abstand von zehn Leerzeichen ein:

```
select 'A' + space(10) + 'B'

------------
A          B
```

STR

Die Funktion STR wandelt numerische Daten in Zeichenfolgen um:

```
STR (float_Ausdruck [, Länge [, Dezimalstellen]])
```

Die *Länge* bezieht sich auf die gesamte Länge der Zeichenfolge. Darin sind Dezimalzeichen, Vorzeichen, alle Ziffern und Leerstellen eingeschlossen. *Dezimalstellen* gibt die Anzahl der Stellen nach dem Dezimalzeichen an.

Die folgende Anweisung gibt die Zahl Pi mit vier Nachkommastellen in einem Feld von zwölf Zeichen aus:

```
select str(pi(),12,4)

------------
```

```
      3.1416
```

STUFF

Die Funktion STUFF mit der Syntax

```
STUFF (Zeichenausdruck1, Start, Anzahl, Zeichenausdruck2)
```

löscht aus *Zeichenausdruck1* ab der Position *Start* die angegebene *Anzahl* Zeichen und fügt in *Zeichenausdruck1* ab *Start*+1 den *Zeichenausdruck2* ein.

Die folgende Anweisung macht aus der Umleitung eine Umgebung:

```
select stuff('Umleitung',3,4,'geb')

--------
Umgebung
```

SUBSTRING

Die Funktion SUBSTRING gibt einen Teil des *Ausdrucks* zurück. Es lassen sich nicht nur Zeichenfolgen, sondern auch Binär-, Text- und Bildausdrücke verwenden:

```
SUBSTRING (Ausdruck, Start, Länge)
```

Die Anweisung

```
select substring('Das ist ein Test',9,3)
```

extrahiert die Zeichenfolge `'ein'` aus der übergebenen Zeichenfolge

```
 'Das ist ein Test':

----
ein
```

11.4.8 Systemfunktionen

Mit den Systemfunktionen lassen sich Informationen über das Computersystem, Benutzer, Datenbanken und Datenbankobjekte ermitteln. Tabelle 11.8 gibt einen Überblick über die Systemfunktionen von SQL Server.

Funktion	Parameter	Rückgabewert/Beschreibung
APP_NAME	()	Anwendungsname der aktuellen Sitzung
CAST	(Ausdruck AS Datentyp)	konvertiert einen Ausdruck von einem Datentyp in einen anderen
CONVERT	(Datentyp [(Länge)], Ausdruck [, Stil])	konvertiert einen Ausdruck von einem Datentyp in einen anderen
COALESCE	(Ausdruck [,...n])	erster Ausdruck, der ungleich NULL ist
CURRENT_ TIMESTAMP		aktuelles Datum und aktuelle Uhrzeit entspricht GETDATE()
CURRENT_USER		aktueller Benutzer entspricht USER_NAME()
DATALENGTH	(Ausdruck)	Anzahl der erforderlichen Bytes für das Speichern des Ausdrucks
@@ERROR		Fehlernummer für die zuletzt ausgeführte Transact-SQL-Anweisung
FORMATMESSAGE	(Meldungsnummer, Parameterwert [,...n])	erstellt eine Meldung aus einer in sysmessages gespeicherten Meldung
GETANSINULL	(['Datenbankname'])	Standardeinstellung der NULL-Zulässigkeit für die Datenbank
HOST_ID	()	Kennummer des Servers
HOST_NAME	()	Name des Servers
IDENT_INCR	('Tabelle_oder_Sicht')	Schrittweite für das Erstellen einer Identitätsspalte
IDENT_SEED	('Tabelle_oder_Sicht')	Ausgangswert für das Erstellen einer Identitätsspalte
@@IDENTITY		zuletzt eingefügter Identitätswert

IDENTITY	(Datentyp, [, Ausgangswert, Schrittweite]) AS Spaltenname	fügt Identitätsspalte in neue Tabelle ein (nur in SELECT-Anweisung mit Klausel INTO *Tabelle* verwendet)
ISDATE	(Ausdruck)	prüft Ausdruck auf gültiges Datum
ISNULL	(Testausdruck, Ersatzwert)	ersetzt NULL durch den Ersatzwert
ISNUMERIC	(Ausdruck)	prüft Ausdruck auf numerischen Datentyp
NEWID	()	erstellt eindeutigen Wert vom Typ uniqueidentifier
NULLIF	(Ausdruck1, Ausdruck2)	NULL, wenn beide Ausdrücke äquivalent sind
PARSENAME	('Objektname', Objektteil)	Teil des Objektnamens (Objektname, Besitzername, Datenbankname, Servername)
PERMISSIONS	([ObjektID [, 'Spalte']])	Bitmuster für Anweisungs-, Objekt- oder Spaltenberechtigungen des aktuellen Benutzers
@@ROWCOUNT		Anzahl der von der letzten Anweisung betroffenen Zeilen
SESSION_USER		läßt das Einfügen eines systemunterstützten Wertes für den Benutzernamen der aktuellen Sitzung in eine Tabelle zu, wenn kein Standardwert angegeben ist
STATS_DATE	(TabellenID, IndexID)	Datum der letzten Statistikaktualisierung für den angegebenen Index
SYSTEM_USER		ermöglicht das Einfügen eines vom System gelieferten Systembenutzernamens in eine Tabelle, wenn kein Standardwert angegeben ist
@@TRANCOUNT		Anzahl der aktiven Transaktionen für die aktuelle Verbindung

USER		ermöglicht das Einfügen eines vom System gelieferten Benutzernamens in eine Tabelle, wenn kein Standardwert angegeben ist
USER_NAME	([ID])	Datenbankbenutzername für den mit ID angegebenen Benutzer

Tabelle 11.8: Systemfunktionen

Die Funktionen USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, CURRENT_TIMESTAMP und APP_NAME sind sogenannte NILADIC-Funktionen. Diese rufen keine Werte von SQL Server ab, sondern stellen einen Standardwert bereit, der automatisch bei einer INSERT- oder UPDATE-Anweisung in eine Zeile einer Tabelle eingesetzt wird, wenn der betreffende Wert nicht angegeben ist.

CAST und CONVERT

Mit den Funktionen CAST und CONVERT konvertiert man einen Ausdruck explizit von einem Datentyp in einen anderen.

Syntax:

```
CAST (Ausdruck AS Datentyp)
CONVERT (Datentyp[(Länge)], Ausdruck [, Stil])
```

Die Funktionalität von CAST und CONVERT ist ähnlich. CAST entspricht dem Standard SQL-92 und ist CONVERT vorzuziehen. Allerdings müssen Sie zum Beispiel CONVERT verwenden, wenn Sie ein Datum in ein bestimmtes Ausgabeformat umwandeln wollen, da sich in der Funktion CAST kein Stilparameter angeben läßt.

Argumente:

Ausdruck ist jeder gültige SQL-Server-Ausdruck.

Der *Datentyp* muß vom Zielsystem unterstützt werden. Benutzerdefinierte Typen sind nicht zulässig.

Der Parameter *Länge* kann optional bei den Zeichentypen nchar, nvarchar, char, varchar, binary und varbinary angegeben werden.

Stil bezeichnet entweder das Datumsformat bei der Umwandlung von Datumswerten (datetime, smalldatetime) in Zeichenfolgen (nchar, nvarchar, char, varchar) oder das Zeichenfolgenformat bei der Umwandlung von Gleitkomma- und Währungsdaten (float, real, money, smallmoney) in Zeichenfolgen.

Tabelle 11.9 gibt die Stilkonstanten für Datumswerte, Tabelle 11.10 die Stilkonstanten für Gleitkommawerte und Tabelle 11.11 für Währungswerte an.

Ohne Jahrhundert yy	Mit Jahrhundert yyyy	Standard	Eingabe/Ausgabe
-	0 oder 100	Standard	mon dd yyyy hh:mi AM (oder PM)
1	101	USA	mm/dd/yy
2	102	ANSI	yy.mm.dd
3	103	Britisch/Französisch	dd/mm/yy
4	104	Deutsch	dd.mm.yy
5	105	Italienisch	dd-mm-yy
6	106	-	dd mon yy
7	107	-	mon dd, yy
8	108	-	hh:mm:ss
-	9 oder 109	Standard + Millisekunden	mon dd yyyy hh:mi:ss:ms AM (oder PM)
10	110	USA	mm-dd-yy
11	111	Japan	yy/mm/dd
12	112	ISO	yymmdd
-	13 oder 113	Europ. Standard + Millisekunden	dd mon yyyy hh:mi:ss:ms (24 h)
14	114	-	hh:mi:ss:ms (24 h)
-	20 oder 120	ODBC-konform	yyyy-mm-dd hh:mi:ss (24 h)
-	21 oder 121	ODBC-konform (mit Millisekunden)	yyyy-mm-dd hh:mi:ss.ms (24 h)

Tabelle 11.9: Stilkonstanten für Datumswerte

In der letzten Spalte bedeutet »Eingabe«, daß eine Umwandlung von einer Zeichenfolge in den Datentyp datetime erfolgt, und »Ausgabe« steht für die Umwandlung von einem Datumswert in eine Zeichenfolge.

Wert	Ausgabe
0 (Standard)	Maximal sechsstellig. Gegebenenfalls in der wissenschaftlichen Notation einzusetzen.
1	Immer achtstellig. Immer in der wissenschaftlichen Notation verwendet.

2	Immer 16stellig. Immer in der wissenschaftlichen Notation verwendet.

Tabelle 11.10: Stilkonstanten für Gleitkommawerte

Wert	Ausgabe
0 (Standard)	Keine Tausendertrennzeichen, zwei Ziffern nach Dezimalzeichen
1	Tausendertrennzeichen vor dem Dezimalzeichen, zwei Ziffern danach
2	Keine Tausendertrennzeichen, vier Ziffern nach dem Dezimalzeichen.

Tabelle 11.11: Stilkonstanten für Währungswerte

Rückgabetyp:

Die Funktion CONVERT gibt den gleichen Wert wie Datentyp 0 zurück.

Beispiel:

Die folgende Anweisung ruft das aktuelle Systemdatum ab und gibt es mit vierstelliger Jahreszahl aus:

```
select convert(char(10),getdate(),104) as Heute
```

Das Ergebnis lautet:

```
Heute
----------
06.04.1999
```

11.4.9 Statistische Systemfunktionen

Die Funktionen dieser Kategorie liefern spezielle Werte zum Computersystem und zur Netzwerkübertragung:

- @@CPU_BUSY: Anzahl der Millisekunden, die SQL Server von der CPU beansprucht hat
- @@IDLE: Anzahl der Millisekunden, die sich SQL Server im Leerlauf befunden hat
- @@IO_BUSY: Anzahl der Millisekunden für Ein-/Ausgabeoperationen
- @@PACK_RECEIVED: Anzahl der vom Netzwerk gelesenen Eingabepakete
- @@PACK_SENT: Anzahl der an das Netzwerk gelieferten Ausgabepakete
- @@PACKET_ERRORS: Anzahl der Netzwerkpaketfehler
- @@TIMETICKS: Anzahl der Mikrosekunden pro Zeitsignal

- @@TOTAL_ERRORS: Anzahl der Lese-/Schreibfehler
- @@TOTAL_READ: Anzahl der Lesezugriffe
- @@TOTAL_WRITE: Anzahl der Schreibzugriffe

11.4.10 Text- und Image-Funktionen

Die Funktionen dieser Kategorie beziehen sich auf Operationen mit text- und image-Werten. Zu diesen Funktionen zählen PATINDEX, die bereits weiter vorn in diesem Kapitel in Verbindung mit den Zeichenfolgenfunktionen behandelt wurde, sowie TEXTPTR und TEXTVALID, die vor allem für SQL Server selbst von Bedeutung sind und auf die hier nicht weiter eingegangen wird.

---

kapitel 12 gespeicherte prozeduren

12.1 the same procedure ...

gespeicherte prozeduren sind eigentlich nichts weiter als eine gruppe von transact-sql-anweisungen, die man - ähnlich einer funktion - unter einem bestimmten namen aufruft und die als ganzes ausgeführt werden. nach dem erstellen einer gespeicherten prozedur kompiliert sql server die anweisungen und speichert die prozedur in einer datenbank. durch die kompilierung ergeben sich erhebliche geschwindigkeitsvorteile gegenüber der separaten ausführung der in der prozedur verpackten sql-anweisungen. außerdem erleichtern gespeicherte prozeduren die ausführung von routinearbeiten. zum lieferumfang von sql server gehören daher zahlreiche gespeicherte systemprozeduren, und sie können auch selbst eigene prozeduren erstellen.

allerdings sei auch auf die nachteile von gespeicherten prozeduren hingewiesen. für das erstellen und bearbeiten der prozeduren stellt sql server kein ausgefeiltes werkzeug bereit, wie man es von anderen programmiersprachen kennt. transact-sql bietet auch keine sprachelemente wie arrays oder benutzerdefinierte datentypen, in denen sich verschiedenartige einzelelemente zusammenfassen lassen. damit ist es schwierig, komplexe programmlogik zu realisieren. darüber hinaus gibt es keine möglichkeit, die gespeicherten prozeduren in übersichtlicher form zu verwalten. vielleicht erstellen sie sich selbst mit sql server eine datenbank, in der sie die gespeicherten prozeduren verwalten?

analog zu den funktionen in höheren programmiersprachen können gespeicherte prozeduren in sql server parameter übernehmen und rückgabewerte liefern.

die anweisungen und befehle sind bestandteil der sprache transact-sql. das gilt insbesondere auch für die steuerungsstrukturen. deshalb ist ihr einsatz nicht auf gespeicherte prozeduren beschränkt. zum beispiel kann man alle anweisungen, befehle, funktionen oder variablen auch außerhalb von gespeicherten prozeduren in dienstprogrammen wie osql oder sql-server-query analyzer verwenden und sogenannte ad-hoc-abfragen durchführen.

12.2 variablen

in variablen lassen sich werte speichern. das können zum beispiel ergebnisse von berechnungen, rückgabewerte aus gespeicherten prozeduren oder zählerstände in schleifen sein. jeder variablen ist ein speicherbereich zugeordnet, dessen größe vom datentyp der variablen abhängig ist. bevor man variablen einsetzen kann, muß man sie deklarieren. dann kann man der variablen werte zuweisen und werte aus der variablen abrufen.

12.2.1 variablen deklarieren (declare)

mit der anweisung declare lassen sich variablen mit den standarddatentypen von sql server erzeugen. die mit dieser anweisung deklarierten variablen müssen mit dem symbol @ beginnen. die syntax der declare-anweisung sieht wie folgt aus:

```
declare @variable1 datentyp [, @variable2 datentyp ...]
```

in ein und derselben declare-anweisung lassen sich mehrere variablen erzeugen. das folgende beispiel deklariert die variablen wert1, wert2 und wert3 mit den datentypen int, varchar(20) und datetime:

```
declare @wert1 int, @wert2 varchar(20), @wert3 datetime
```

12.2.2 werte zuweisen (set, select)

einer zuvor deklarierten variablen weist man mit der set-anweisung einen wert zu. die syntax lautet:

```
set @lokalevariable = ausdruck
```

der *ausdruck* kann jeder gültige sql-server-ausdruck sein.

die anweisungen im folgenden beispiel weisen den im letzten abschnitt deklarierten variablen werte zu:

```
set @wert1 = 67
set @wert2 = 'y2k-test'
set @wert3 = '1.7.00'
```

mit der select-anweisung läßt sich einer variablen ebenfalls ein wert zuweisen:

```
select @lokalevariable = ausdruck
```

allerdings empfiehlt microsoft, vorzugsweise die set-anweisung zu verwenden.

12.2.3 variablen verwenden

die folgende anweisung zeigt die im letzten beispiel zugewiesenen werte an:

```
select @wert1, @wert2, @wert3
```

die nächste anweisung addiert zum datumswert in der variablen @wert3 die anzahl tage, die in der variablen @wert1 enthalten ist. die darauffolgende select-anweisung zeigt das ergebnis an:

```
select @wert3 = @wert3 + @wert1
select @wert3
```

insgesamt ergibt sich folgende ergebnismenge:

```
----------- -------------------- ---------------------------
67          y2k-test             2000-07-01 00:00:00.000


---------------------------
2000-09-06 00:00:00.000
```

welche wirkungen die zuweisung mit hilfe einer select-anweisung haben kann, zeigt das nächste beispiel:

```
-- var2.sql

declare @preis money
set @preis = 1234.56
use pubs
select @preis=price from titles
  where price>100
select @preis as preis
```

die erste anweisung deklariert die variable @preis mit dem datentyp money. die set-anweisung weist der variablen @preis den anfangswert 1234.56 zu. jetzt kommt das eigentlich interessante an diesem beispiel: die select-anweisung führt eine abfrage bezüglich der spalte price in der tabelle titles aus und weist das ergebnis der variablen @preis zu. aufgrund der bedingung in der where-klausel ist die ergebnismenge leer. damit behält die variable @preis den mit der set-anweisung zugewiesenen wert bei:

```
preis
---------------------
1234.5600
```

kommentieren sie jetzt die where-klausel aus:

```
declare @preis money
set @preis = 1234.56
use pubs
select @preis=price from titles
  --where price>100
select @preis as preis
```

die ergebnismenge ist jetzt nicht mehr leer. wenn die select-anweisung - wie in diesem fall - mehrere werte zurückgibt, wird der variablen der zuletzt zurückgegebene wert zugewiesen. das ergebnis lautet jetzt:

```
preis
---------------------
14.9900
```

wenn sie im zuweisungsausdruck eine skalare unterabfrage verwenden und diese unterabfrage keine zeilen zurückgibt, erhält die variable den wert null.

12.2.4 gültigkeitsbereich

der gültigkeitsbereich einer variablen reicht von dem punkt ihrer deklaration bis zum ende des stapels oder der gespeicherten prozedur, wo sie deklariert wurde. man spricht deshalb auch von lokalen variablen.

das folgende beispiel liefert einen fehler, weil zwischen der variablendeklaration und der zuweisung des wertes das schlüsselwort go steht, das einen stapel abschließt.

```
-- var3.sql
set nocount on
declare @wert1 int
go
set @wert1 = 1234
go
```

12.3 transact-sql-steuerungsstrukturen

wenn sie bereits mit programmiersprachen wie c++ oder visual basic gearbeitet haben, werden ihnen die strukturen zur ablaufsteuerung von transact-sql vertraut vorkommen. dennoch gibt es verschiedene eigenheiten zu beachten, da diese strukturen speziell auf die arbeit mit datenbanken zugeschnitten sind und die syntax teilweise von den bekannten strukturen abweicht.

blöcke (begin...end)

die schlüsselwörter begin und end kennzeichnen einen block von transact-sql-anweisungen. die im block eingeschlossenen anweisungen werden als einheit ausgeführt. wenn ein block nur eine anweisung enthält, kann man auf die schlüsselwörter begin...end verzichten. beispiele für blöcke finden sie bei der nachfolgenden behandlung der steuerungsstrukturen.

die begriffe *block* und *stapel* sind genau auseinanderzuhalten. ein block ist eine logische zusammenfassung mehrerer transact-sql-anweisungen zu einer einheit, während ein stapel eine physikalische einheit von transact-sql-anweisungen ist, die sql server in einen sogenannten *ausführungsplan* kompiliert und als solches insgesamt ausführt. ein stapel kann mehrere blöcke enthalten. umgekehrt trifft dies nicht zu: ein block kann nicht über mehrere stapel reichen.

bedingte ausführung (if...else)

mit der steuerungsstruktur if...else lassen sich anweisungen abhängig von einer bedingung ausführen. die syntax lautet:

```
if ausdruck
  anweisung1
[else
  anweisung2]
```

mit *ausdruck* wird die bedingung formuliert. die auswertung des ausdrucks muß einen booleschen wert ergeben. ist dieser wert true, kommt *anweisung1* zur ausführung. bei false überspringt das programm die *anweisung1* und setzt mit der darauffolgenden anweisung fort.

wenn der optionale else-zweig vorhanden ist und *ausdruck* das ergebnis false liefert, wird die alternative *anweisung2* ausgeführt.

bei *anweisung1* und *anweisung2* kann es sich um einzelne anweisungen oder in begin und end eingeschlossene anweisungsblöcke handeln. wenn der *ausdruck* eine select-anweisung enthält, ist diese in klammern einzuschließen.

die folgende anweisung testet, ob es sich bei der aktuellen datenbank nicht um pubs handelt. wenn das der fall ist, liefert die bedingung das ergebnis true, und die anweisung use pubs macht die

verlegerdatenbank zur aktuellen datenbank.

```
if db_name()<>'pubs'
  use pubs
```

in einer if-anweisung verwendet man häufig das schlüsselwort exists mit einer nachfolgenden select-anweisung. das nächste beispiel zeigt einen ausschnitt aus dem skript instpubs.sql. die if-anweisung in diesem codefragment prüft, ob die datenbank pubs vorhanden ist:

```
if exists (select * from sysdatabases where name='pubs')
begin
  raiserror('dropping existing pubs database ....',0,1)
  drop database pubs
end
```

die systemtabelle sysdatabases verzeichnet alle in sql server vorhandenen datenbanken. die in klammern stehende select-anweisung bildet eine unterabfrage. mit dem schlüsselwort exists ermittelt man, ob die unterabfrage zeilen zurückgibt. wenn die datenbank pubs existiert, ergibt die auswertung des bedingungsausdrucks true, und der in begin und end eingeschlossene anweisungsblock wird ausgeführt (siehe dazu auch die erläuterungen im anhang b für die zeilen 17 bis 22 der datei instpubs.sql).

schleifen (while)

mit schleifenkonstruktionen lassen sich anweisungen oder anweisungsblöcke mehrfach ausführen, solange eine bestimmte bedingung zutrifft. gegenüber anderen programmiersprachen, die mehrere schleifentypen (zum beispiel for...next, do...loop until) kennen, gibt es in transact-sql nur die while-schleife.

sieht man einmal von compilerspezifischen optimierungen ab, ist die vielzahl der schleifentypen in anderen programmiersprachen eigentlich überflüssig. wenn man die schleifenbedingung geeignet formuliert und die möglichkeit hat, die bedingung entweder am beginn oder am ende der schleife zu testen, kommt man mit einem grundtyp für die konstruktion von schleifen aus.

die grundlegende syntax einer while-schleife hat in transact-sql folgendes aussehen:

syntax:

```
while ausdruck
  begin
    anweisung(en)
  end
```

der boolesche *ausdruck* stellt die zu testende schleifenbedingung dar. solange (englisch: while) die bedingung den wert true ergibt, wird die *anweisung* im rumpf der schleife ausgeführt. wenn der *ausdruck* eine select-anweisung enthält, ist diese in klammern einzuschließen. die *anweisung* kann eine einzelne transact-sql-anweisung oder ein in begin und end eingeschlossener anweisungsblock sein.

beispiele:

das folgende programm zeigt, wie sich ein zähler mit einer while-schleife realisieren läßt.

```
1:  declare @counter as int
2:  set @counter = 0
3:  while @counter < 10
4:    begin
5:      set @counter = @counter + 1
6:      print @counter
7:    end
8:  print 'schleife beendet'
```

als erstes deklariert das programm die ganzzahlige variable @counter und initialisiert sie mit dem wert 0. daran schließt sich die while-schleife an. der bedingungsausdruck liefert true, solange die variable @counter kleiner als 10 ist. da die variable den anfangswert 0 aufweist, tritt das programm in die schleife ein. die erste anweisung im rumpf der schleife addiert den wert 1 zur variablen @counter. anstelle der set-klausel kann auch select stehen. der ausdruck wird dann wie das ergebnis einer abfrage behandelt. eine einfache zuweisungsoperation der form @counter = @counter + 1 ist syntaktisch nicht korrekt.

zeile 6 gibt den inhalt der variablen @counter mit der print-anweisung aus. wenn das programm in zeile 7 das ende der schleife erreicht, springt es an den beginn der schleife in zeile 3. hier wird der wert von @counter erneut getestet. nach dem ersten schleifendurchlauf ist der wert von @counter gleich 1, der bedingungsausdruck liefert false, und das programm durchläuft die schleife ein zweites mal. dieser ablauf setzt sich fort, bis der wert in @counter den wert 10 erhält. dann liefert der test das ergebnis false. daraufhin setzt das programm mit der ersten anweisung nach der schleife fort und weist mit der print-anweisung in zeile 8 auf das verlassen der schleife hin. die ausgabe sieht wie folgt aus:

```
1
2
3
4
5
6
7
8
9
10
```

```
schleife beendet
```

schleifen vorzeitig verlassen (break)

die syntax der while-schleife läßt sich mit dem optionalen schlüsselwort break erweitern:

```
while ausdruck
begin
  anweisung(en)
  break
  anweisung(en)
end
```

bei einem break beendet das programm den aktuellen anweisungsblock und setzt die programmausführung mit der ersten anweisung nach diesem block fort.

das folgende beispiel ist eine modifikation des obigen codefragments. der rumpf der while-schleife enthält jetzt eine break-anweisung (zeile 7):

```
1:  declare @counter as int
2:  set @counter = 0
3:  while @counter < 10
4:    begin
5:      set @counter = @counter + 1
6:      print @counter
7:      break
8:    end
9:  print 'schleife beendet'
```

die ausgabe sieht folgendermaßen aus:

```
1
schleife beendet
```

nachdem das programm im ersten schleifendurchlauf die variable @counter inkrementiert und deren wert ausgegeben hat, bewirkt die break-anweisung, daß der zur schleife gehörende block sofort verlassen wird. die schleife gibt den zählerstand also nur einmal aus, und das programm springt sofort zu zeile 9, um das ende der schleife zu verkünden.

die break-klausel setzt man normalerweise in verbindung mit einer bedingungsanweisung ein, die einen zusätzlichen abbruchtest innerhalb des schleifenkörpers ausführt. vor allem aber ist break dafür vorgesehen, eine innere schleife in einer verschachtelten while-konstruktion zu verlassen. darauf geht der entsprechende abschnitt weiter unten in diesem kapitel ein.

schleife am beginn fortsetzen (continue)

eine weitere optionale ergänzung der while-syntax stellt das schlüsselwort continue dar, das die unmittelbare fortsetzung des programmablaufs am beginn der schleifenkonstruktion, d.h. mit dem test der schleifenbedingung, bewirkt:

```
while ausdruck
begin
  anweisung(en)
  [break]
  anweisung(en)
  [continue]
end
```

das folgende beispiel zeigt die wirkung von continue:

```
1:  declare @counter as int
2:  set @counter = 0
3:  while @counter < 5
4:  begin
5:    set @counter = @counter + 1
6:    if @counter=4
7:      continue
8:    else
9:      print @counter
10: end
11: print 'schleife beendet'
```

die ausgabe sieht folgendermaßen aus:

```
1
2
3
5
schleife beendet
```

erreicht die zählervariable im vierten schleifendurchlauf den wert 4, verzweigt das programm zur continue-klausel in zeile 7 und springt damit sofort zurück nach zeile 3. die auf continue bis zum ende des blocks folgenden anweisungen (hier die print-anweisung in zeile 9) werden nicht mehr ausgeführt. das programmbeispiel überspringt also die ausgabe des zählerwertes 4.

genau wie break verwendet man auch die continue-klausel normalerweise in verbindung mit einer bedingungsanweisung.

die darstellung der syntax in der online-dokumentation ist ungenau:

```
while boolean_expression
  {sql_statement | statement_block}
  [break]
  {sql_statement | statement_block}
  [continue]
```

aus dieser darstellung geht nicht hervor, daß die nach while und dem bedingungsausdruck folgenden anweisungen einschließlich break und continue innerhalb ein und desselben blocks stehen müssen. da es sich um mehrere anweisungen handelt, sind diese aber in begin und end einzuschließen. die zwischen while und break bzw. zwischen break und continue stehenden anweisungen können ihrerseits einzelanweisungen oder anweisungsblöcke sein.

da die break-klausel vor allem dafür vorgesehen ist, eine innere schleife bei einer verschachtelten schleifenkonstruktion zu verlassen, kann es zu schwer auffindbaren fehlern führen, wenn sie den rumpf der schleife nicht genau definieren.

bei einer nicht verschachtelten while-schleife wie

```
declare @counter as int
set @counter = 0
while @counter < 5
  set @counter = @counter + 1
  break
print 'schleife beendet'
```

weist sie sql server sofort auf den fehler hin:

```
server: nachr.-nr. 135, schweregrad 15, status 1, zeile 5
eine break-anweisung kann außerhalb des bereichs einer while-
anweisung nicht verwendet werden.
```

die angeführten beispiele zur while-anweisung in der online-dokumentation sind allerdings korrekt.

schleifen verschachteln

von verschachtelten schleifen spricht man, wenn innerhalb einer schleife eine weitere schleife enthalten ist. die tiefe der verschachtelung ist in transact-sql nicht begrenzt. allerdings wird man in der praxis kaum mehr als drei schleifen ineinander verschachteln.

im folgenden beispiel sind zwei schleifen ineinander verschachtelt:

```
1:  declare @aussen as int
2:  declare @innen as int
3:  declare @ausgabe as varchar(200)
4:  set @aussen = 0
5:  while @aussen < 5
6:  begin
7:    set @innen = 0
8:    while @innen<4
9:    begin
10:     set @innen=@innen+1
11:     set @ausgabe='innen: ' + cast(@innen as varchar(3))
12:     print @ausgabe
13:     /*break*/
14:   end
15:   set @aussen = @aussen + 1
16:   set @ausgabe = 'aussen: ' +  cast(@aussen as varchar(3))
17:   print @ausgabe
18: end
19: print 'schleife beendet'
```

die variable @aussen zählt die schleifendurchläufe der äußeren schleife, die variable @innen die durchläufe der inneren schleife. die innere schleife gibt den aktuellen zählerstand in zeile 12, die äußere in zeile 17 aus. jede schleife inkrementiert die ihr zugedachten variablen um 1.

die äußere schleife setzt als erstes den anfangswert der variablen @innen auf 0. dadurch wird die sich anschließende innere schleife viermal durchlaufen. erst wenn die innere schleife vollständig abgearbeitet ist, setzt sich der programmablauf in der äußeren schleife mit zeile 15 fort. die äußere schleife inkrementiert die variable @aussen, gibt den aktuellen wert aus und testet erneut in zeile 5, ob der endwert der variablen @aussen erreicht ist. solange der wert von @aussen kleiner als 5 ist, wiederholen sich die beschriebenen abläufe. als ergebnis erhält man:

```
innen: 1
innen: 2
innen: 3
```

```
innen: 4
aussen: 1
innen: 1
innen: 2
innen: 3
innen: 4
aussen: 2
innen: 1
innen: 2
innen: 3
innen: 4
aussen: 3
innen: 1
innen: 2
innen: 3
innen: 4
aussen: 4
innen: 1
innen: 2
innen: 3
innen: 4
aussen: 5
schleife beendet
```

wenn sie die kommentarzeichen um die break-klausel in zeile 13 entfernen und die anweisungen erneut ausführen, entsteht folgende ausgabe:

```
innen: 1
aussen: 1
innen: 1
aussen: 2
innen: 1
aussen: 3
innen: 1
aussen: 4
innen: 1
aussen: 5
schleife beendet
```

bei jedem durchlauf der äußeren schleife kann die innere schleife die variable @innen nur einmal inkrementieren, weil die break-anweisung in zeile 13 bewirkt, daß die innere schleife sofort verlassen wird.

kommentieren sie nun die begin- und end-klauseln der inneren while-schleife aus:

```
1:  declare @aussen as int
2:  declare @innen as int
3:  declare @ausgabe as varchar(200)
4:  set @aussen = 0
5:  while @aussen < 5
6:  begin
7:    set @innen = 0
8:    while @innen<4
9:    /*begin*/
10:     set @innen=@innen+1
11:     set @ausgabe='innen: ' + cast(@innen as varchar(3))
12:     print @ausgabe
13:     break
14:   /*end*/
15:   set @aussen = @aussen + 1
16:   set @ausgabe = 'aussen: ' +  cast(@aussen as varchar(3))
17:   print @ausgabe
18: end
19: print 'schleife beendet'
```

die anweisungen in den zeilen 11 bis 13 beziehen sich nun auf die äußere schleife. insbesondere gilt das break in zeile 13 nicht mehr für die innere, sondern die äußere schleife.

das ergebnis lautet:

```
innen: 4
schleife beendet
```

dieses experiment sollte noch einmal deutlich machen, wie wichtig die schlüsselwörter begin und end in einer schleifenkonstruktion sind.

weitere beispiele für schleifen

in den mit sql server im verzeichnis \mssql7\install installierten skripts finden sie viele beispiele für den praktischen einsatz von einfachen und verschachtelten while-schleifen mit und ohne break bzw. continue.

der folgende ausschnitt aus dem skript dropall.sql zeigt eine interessante konstruktion.

```
96: while (17=17)
97:    begin
```

```
98:
99:    fetch        next
100:      from      csr_17_todrop
101:      into      @obj_type
102:               ,@obj_name
103:
104:   if (@@fetch_status <> 0)
105:      begin
106:      deallocate csr_17_todrop
107:      break
108:      end
109:
... weitere anweisungen
153:
154:   end -- loop 17
```

der testausdruck in zeile 96 liefert immer true. die while-schleife dient hier nur als äußere hülle, um eine ganze folge von anweisungen - die auswertung eines cursors - in eleganter weise wiederholt auszuführen. mit der ominösen zahl 17 hat der programmierer lediglich eine numerierung der schleifen zur besseren orientierung vorgenommen, wie man anhand des kommentars in zeile 154 erkennen kann. das eigentliche geheimnis der schleife liegt in den zeilen 104 bis 108. wenn die funktion @@fetch_status einen wert ungleich 0 zurückgibt, führt das programm den if-zweig in den zeilen 105 bis 108 aus und trifft in zeile 107 auf die break-anweisung. damit wird die while-schleife beendet, und das programm setzt mit der nächsten anweisung nach der schleife (zeile 155) fort.

wenn der testausdruck unpassend formuliert ist oder die anweisungen in der schleife so gestaltet sind, daß die abbruchbedingung nie erfüllt und die schleife auch nicht durch eine break-anweisung verlassen werden kann, entsteht eine endlosschleife, die sich nur noch durch äußeren eingriff beenden läßt **ç**+**c** bei osql oder **á**+**à** in query analyzer).

im folgenden beispiel testet die bedingung in zeile 5, ob der zähler den wert 100 noch nicht erreicht hat. in zeile 7 hat der programmierer statt des beabsichtigten pluszeichens irrtümlich ein minuszeichen geschrieben. damit wird die zählvariable @counter dekrementiert, erreicht immer kleinere negative werte und kann den wert 100 überhaupt nicht erreichen - die schleife läuft endlos und läßt sich nur durch äußeren eingriff unterbrechen. wenn der zähler den wertebereich des datentyps int überschreitet, meldet sql server wiederholt überlauffehler, beendet aber die schleife nicht.

```
1:  declare @ausgabe as varchar(200)
2:  declare @counter as int
3:  set @counter=0
4:  print 'schleife begonnen'
5:  while (@counter<100)
6:  begin
7:    set @counter=@counter-1
```

```
8:    print @counter
9:  end
10: print 'schleife beendet'
```

programmsprung (goto)

die anweisung goto bewirkt einen unbedingten programmsprung zu der anweisung, die im sprungziel angegeben ist.

syntax:

sprungziel definieren:

```
marke:
```

unbedingten sprung zur marke ausführen:

```
goto marke
```

die syntax besteht aus zwei teilen: der sprungmarke und der eigentlichen goto-anweisung. mit der *marke* definieren sie den punkt, an dem das programm bei ausführung einer goto-anweisung fortsetzen soll. die marke muß den regeln für bezeichner entsprechen und ist mit einem doppelpunkt abzuschließen.

der sprungbereich ist auf den aktuellen stapel beschränkt. die marke und die goto-anweisung müssen sich also im selben stapel befinden. sprünge zu marken außerhalb des stapels sind nicht zulässig. allerdings ist es möglich, in eine prozedur, einen anweisungsblock oder eine steuerungsstruktur - also auch in eine while-schleife - zu springen. die marke kann vor oder nach der goto-anweisung stehen.

an keiner anweisung erhitzen sich die gemüter der programmierer so sehr wie an goto. die einen lehnen diese anweisung rundweg ab, weil sie der strukturierten programmierung widerspricht und zu undurchschaubarem code führt, andere befürworten sie als hintertürchen, wenn man damit umständliche programmkonstruktionen umgehen kann. wenn sie die beispielskripts von sql server studieren, finden sie massenhaft goto-anweisungen. urteilen sie selbst.

das folgende beispiel zeigt eine goto-anweisung, die den sprung in eine while-schleife bewirkt. eine derartige konstruktion dürfte wohl wasser auf die mühlen der goto-gegner sein.

```
1:  declare @counter as int
2:  set @counter = 0
3:  while @counter < 5 or @counter>=10
4:  begin
```

```
5:    set @counter=@counter+1
6:    print @counter
7:    if @counter>15 break
8:    continue
9:   schleife:
10:   set @counter=10
11:   print 'wieder in der schleife'
12: end
13: if @counter=5 goto schleife
14: print 'schleife beendet'
```

der anfangswert der zählvariablen ist 0. gemäß der schleifenbedingung in zeile 3 wird die schleife zunächst fünfmal durchlaufen, bis @counter (durch die anweisung in zeile 5) den wert 5 erreicht hat. die continue-anweisung in zeile 8 bewirkt, daß der schleifendurchlauf sofort wieder bei zeile 3 beginnt und die anweisung in zeile 10, die einen neuen anfangswert für @counter festlegt, nicht zur ausführung kommt.

nach diesem ersten zyklus ist @counter gleich 5. die if-anweisung nach dem ende der schleife in zeile 13 ergibt damit true, und die anweisung goto schleife bewirkt einen sprung zurück in die while-schleife zur marke in zeile 9. daraufhin setzt die anweisung in zeile 10 den wert der variablen @counter auf 10, gibt in zeile 11 die meldung wieder in der schleife aus, und die ausführung der schleife setzt sich fort. wenn die while-anweisung in zeile 3 die bedingung testet, liefert der zweite teil des ausdrucks den wert true. der zweite zyklus läuft nun genau wie der erste ab, außer daß @counter nicht von 0 bis 5, sondern von 10 bis 16 inkrementiert wird. sobald @counter den wert 16 erreicht hat, ergibt der test in der if-anweisung von zeile 7 das ergebnis true. die daraufhin ausgeführte break-anweisung bewirkt das unmittelbare verlassen der schleife. da @counter jetzt den wert 16 aufweist, liefert der test in zeile 13 den wert false. das programm gibt abschließend die meldung in zeile 14 aus.

die ausgabe sieht wie folgt aus:

```
1
2
3
4
5
wieder in der schleife
11
12
13
14
15
16
schleife beendet
```

rücksprung aus prozedur (return)

die anweisung return beendet eine abfrage oder prozedur sofort und unbedingt.

syntax:

```
return [ausdruck]
```

mit dem optionalen *ausdruck* kann man einen wert an die aufrufende prozedur oder anwendung übermitteln. der ausdruck muß eine ganze zahl ergeben. die mit sql server installierten gespeicherten prozeduren geben in der regel bei fehlerfreier ausführung den wert 0 zurück. negative werte kennzeichnen einen fehler. in sql server sind momentan die rückgabewerte von 0 bis -14 definiert (siehe tabelle 12.1). die werte -15 bis -99 sind für zukünftige erweiterungen reserviert.

rückgabewert	beschreibung
0	erfolgreiche ausführung
-1	objekt fehlt
-2	datentypfehler
-3	prozeß als deadlockopfer gewählt
-4	berechtigungsfehler
-5	syntaxfehler
-6	verschiedene benutzerfehler
-7	ressourcenfehler, beispielsweise speichermangel
-8	nicht fatales internes problem
-9	systemgrenze erreicht
-10	fataler interner software-fehler
-11	fataler interner software-fehler
-12	tabelle oder index beschädigt
-13	datenbank beschädigt
-14	hardware-fehler

tabelle 12.1: reservierte rückgabewerte für gespeicherte prozeduren

bei gespeicherten prozeduren dürfen sie das schlüsselwort return nicht anstelle eines break verwenden. eine while-schleife würde damit zwar verlassen werden, die auf die schleife folgenden anweisungen

kämen aber ebenfalls nicht mehr zur ausführung, weil das return sofort die gesamte prozedur beendet.

zeitverzögerung (waitfor)

wenn sie anweisungen erst nach einer bestimmten zeitspanne oder zu einem bestimmten zeitpunkt ausführen wollen, können sie mit waitfor eine verzögerungszeit oder einen genauen zeitpunkt angeben. die zeitsperre bezieht sich nur auf die verbindung, in der die waitfor-anweisung steht. es ist durchaus möglich, daß sie unter demselben anmeldenamen eine zweite verbindung zu sql server aufbauen und parallel zur - suspendierten - ersten verbindung anweisungen ohne zeitverzögerung ausführen.

die syntax der waitfor-anweisung sieht folgendermaßen aus:

syntax:

```
waitfor { delay 'zeit' | time 'zeit' }
```

argumente:

bei delay setzt sql server die ausführung für die angegebene zeitspanne aus, bei time wartet sql server bis zum angegebenen zeitpunkt. im parameter *zeit* legen sie die zeitspanne (delay) oder den zeitpunkt (time) bis jeweils maximal 24 stunden fest. eine datumsangabe ist nicht möglich.

mit der anweisung

```
waitfor delay '00:00:10'
```

setzen sie die ausführung aller nachfolgenden anweisungen in der aktuellen verbindung um 10 sekunden aus.

die von einer waitfor-anweisung betroffene verbindung bleibt bestehen, nur die ausführung von anweisungen wird ausgesetzt. deshalb sollten sie waitfor nicht für längere zeiträume verwenden.

wenn während einer laufenden verbindung der sql-server-dienst beendet wird, erhalten sie die folgende fehlermeldung:

```
[microsoft][odbc sql server driver][named
pipes]connectioncheckfordata (peeknamedpipe()).
[microsoft][odbc sql server driver][named pipes]verbindung ist
unterbrochen.

verbindung unterbrochen
```

die zeitversetzt oder zu einem bestimmten zeitpunkt auszuführende anweisung kommt damit nicht mehr zum zuge. um aufgaben zu einem bestimmten termin auszuführen, empfiehlt sich die terminplanung mit dem sql-server-agenten (siehe kapitel 23).

prozeduren ausführen (execute)

mit der anweisung execute (kurzform exec) lassen sich prozeduren - systemprozeduren, benutzerdefinierte gespeicherte prozeduren oder erweiterte gespeicherte prozeduren - ausführen. es ist auch möglich, eine übergebene zeichenfolge in einem transact-sql-stapel auszuführen. die syntax der execute-anweisung lautet:

syntax:

```
exec[ute] [@return_status =]
  {prozedurname [; nummer] | @prozedurnamevar }
  [[@parameter =] {wert | @variable [output] | [default] } ]
  [,...n]
[with recompile]
```

argumente:

die optionale variable *@return_status* nimmt den rückgabewert einer gespeicherten prozedur auf. diese ganzzahlige variable ist zuvor im aktuellen stapel oder in der aufrufenden prozedur zu deklarieren.

der *prozedurname* bezeichnet den namen der auszuführenden prozedur, entweder als literal angegebenen name oder als lokal definierte variable *@prozedurnamevar*, die den namen der prozedur darstellt. wenn sie prozeduren gruppiert haben, geben sie die jeweilige *nummer* der prozedur mit einem nachgestellten semikolon an.

wenn parameter an die prozedur zu übergeben sind, können sie diese in der reihenfolge der definition angeben oder mit dem benannten parameter @parameter = wert ohne einhaltung der reihenfolge übergeben. die gemischte übergabe mit und ohne benannte parameter ist nicht zulässig.

@variable bezeichnet eine variable, die einen wert an die prozedur übergibt und/oder von dieser zurückliefert. wenn sie einen wert zurückgeben wollen, müssen sie das immer mit einer variablen abwickeln.

das schlüsselwort output weist darauf hin, daß die prozedur einen wert zurückgibt. beim erstellen der prozedur müssen sie den betreffenden parameter ebenfalls mit dem schlüsselwort output deklarieren. konstanten lassen sich nicht mit output an eine prozedur übergeben.

das schlüsselwort default bewirkt, daß die prozedur für den betreffenden parameter einen standardwert annimmt.

in ein und derselben execute-anweisung können sie mehrere parameter übergeben.

wenn sie die option with recompile angeben, erstellt sql server den ausführungsplan neu.

beispiele:

die anweisung

```
exec sp_helpdb
```

führt die gespeicherte prozedur sp_helpdb ohne parameter aus und liefert damit angaben zu allen in sql server gespeicherten datenbanken. wenn sie informationen zu einer bestimmten datenbank abrufen wollen, übergeben sie die datenbank als parameter wie im folgenden beispiel:

```
exec sp_helpdb pubs
```

damit liefert sp_helpdb lediglich informationen zur datenbank pubs.

zeichenfolgen ausführen (execute)

wenn sie transact-sql-anweisungen dynamisch erstellen und ausführen wollen, schreiben sie die anweisungen in eine zeichenfolge und führen diese mit execute aus. die syntax der execute-anweisung lautet für diesen fall:

```
exec[ute] ({@zeichenfolgenvar | [n]'transact-sql-zeichenfolge'}
  [+...n])
```

in der lokalen variablen *@zeichenfolgenvar* geben sie die auszuführenden anweisungen an. alternativ können sie eine konstante *transact-sql-zeichenfolge* formulieren. mit dem optionalen *n* gilt für die transact-sql-zeichenfolge der datentyp nvarchar, andernfalls varchar.

das folgende beispiel zeigt, wie man transact-sql-zeichenfolgen in verbindung mit lokalen variablen in einer execute-anweisung ausführt:

```
use master
declare @tabelle as varchar(20)
set @tabelle = 'authors'
exec ('use pubs ' + 'select * from ' + @tabelle)
select * from titles
```

die anweisung soll daten aus der datenbank pubs abrufen. warum beginnt dann das beispiel mit use

master? den grund erfahren sie gleich. auf jeden fall ist master erst einmal die aktuelle datenbank.

die lokale variable @tabelle nimmt den namen der tabelle auf, aus der die select-anweisung daten auswählen soll. die erste transact-sql-zeichenfolge in der execute-anweisung ändert den datenbankkontext, so daß jetzt die datenbank pubs zur aktuellen datenbank avanciert. die nächste transact-sql-zeichenfolge stellt den ersten teil der select-abfrage - und zwar im kontext der datenbank pubs - dar. den erforderlichen tabellennamen stellt die lokale variable @tabelle bereit. die in der execute-anweisung angegebene zeichenfolge ist demnach als

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

zu interpretieren. wie erwartet, liefert die ergebnismenge die daten der tabelle authors.

die nächste - nach execute stehende - anweisung lautet:

```
select * from titles
```

und das ergebnis lautet:

```
server: nachr.-nr. 208, schweregrad 16, status 1, zeile 1
ungültiger objektname 'titles'.
```

was ist hier passiert? der mit use pubs innerhalb der execute-anweisung vollzogene kontextwechsel ist nur bis zur vollständigen ausführung der execute-anweisung wirksam. danach ist master wieder die aktuelle datenbank. und in master gibt es keine tabelle titles. jetzt wissen sie auch, warum das beispiel mit use master begonnen hat: um die gültigkeitsdauer eines kontextwechsels zu demonstrieren.

für die ausführung von zeichenfolgen empfiehlt microsoft statt der execute-anweisung die neu mit sql server 7.0 eingeführte gespeicherte prozedur sp_executesql, die wesentlich flexibler ist.

wenn sie allerdings skripts erstellen, die sowohl unter sql server 6.x als auch 7.0 laufen sollen, müssen sie zeichenfolgen mit execute ausführen, da sp_executesql in früheren versionen nicht vorhanden war. das in anhang b erläuterte skript instpubs.sql enthält in den zeilen 912 bis 934 beispiele für die ausführung von zeichenfolgen mittels execute.

benutzerdefinierte meldungen (print)

mit der transact-sql-anweisung print geben sie eine benutzerdefinierte meldung an den client zurück. die syntax der anweisung print lautet:

```
print 'zeichenfolge' | @variable | @@funktion | ausdruck
```

die anweisung

```
print 'das ist ein test'
```

gibt schlicht und einfach die meldung

```
das ist ein test
```

aus. die auszugebende zeichenfolge können sie auch einer lokalen variablen zuweisen, um dann diese variable mit print auszugeben:

```
declare @test varchar(30)
set @test='das ist ein anderer test'
print @test
```

bei verwendung einer funktion muß diese einen zeichenfolgenwert zurückgeben:

```
print convert(varchar(30), getdate())
```

diese anweisung holt mit der funktion getdate das aktuelle systemdatum und wandelt es mit hilfe von convert in eine zeichenfolge um.

schließlich können sie eine zeichenfolge auch mit operatoren in form eines ausdrucks zusammenbauen:

```
print 'am '+ convert(varchar(30), getdate()) + ' ausgeführt.'
```

der kern dieser anweisung entspricht dem letzten beispiel. neu dazugekommen sind die beiden strings am beginn und am ende. die verbindung der teilstrings erfolgt mit dem plusoperator. das ergebnis sieht folgendermaßen aus:

```
am jun 15 1999  5:26pm ausgeführt.
```

verwenden sie den befehl raiserror, um eine benutzerdefinierte fehlermeldung auszugeben, wenn die meldung eine von @@error zurückgebbare fehlernummer enthält. in den beispielskripts zu sql server wird der befehl raiserror oftmals auch anstelle von print eingesetzt, um eine einfache meldung auszugeben.

fehlermeldungen (raiserror)

die transact-sql-anweisung raiserror gibt eine benutzerdefinierte fehlermeldung aus. außerdem setzt sie ein systemflag, um auf einen fehler hinzuweisen. die anweisung raiserror hat folgende syntax:

```
raiserror ({msg_id | msg_str}{, schweregrad, status}
[, argument [,...n]] )
[with option[,...n]]
```

der parameter *msg_id* steht für die nummer einer benutzerdefinierten fehlermeldung, die in der systemtabelle sysmessages gespeichert ist. alternativ können sie in *msg_str* eine zeichenfolge in der art der c-funktion printf angeben. die zeichenfolge kann ersetzbare parameter und formatsteuerzeichen enthalten. mit den parametern *schweregrad* und *status* weisen sie den benutzer auf die art des fehlers hin. die sich anschließenden argumente müssen in anzahl und typ den ersetzbaren parametern entsprechen.

in der klausel with *option* lassen sich folgende optionen angeben:

- log: schreibt den fehler in das fehler- und anwendungsprotokoll des servers
- nowait: sendet die meldungen sofort an den client
- seterror: legt unabhängig vom schwergrad den von @@error zurückgegebenen wert auf *msg_id* oder 50000 fest

das folgende beispiel deklariert die variable @var1 als zeichenfolge und weist ihr den wert das ist ein string. zu. die anweisung raiserror gibt eine benutzerdefinierte meldung aus, wobei der parameter %s durch den text in @var1 ersetzt wird:

```
declare @var1 varchar(100)
set @var1 = 'das ist ein string.'
raiserror('der text lautet: %s ',1,1,@var1)
```

als ergebnis erscheint:

```
meldung 50000, ebene 1, status 50000:
der text lautet: das ist ein string.
```

bedingungsliste (case)

wenn sie mit basic oder c programmieren, sind ihnen schalteranweisungen wie select case oder switch geläufig. derartige steuerungsstrukturen dienen dazu, in abhängigkeit von einer bedingung einen block von anweisungen auszuführen. die case-anweisung von transact-sql wertet zwar ebenfalls bedingungen aus, verzweigt dann aber nicht wie die eingangs genannten strukturen zum zugehörigen programmblock, sondern gibt den wert eines ausdrucks zurück.

die case-anweisung weist zwei syntaxformen auf. die sogenannte einfache case-funktion sieht wie folgt aus:

```
case eingabeausdruck
  when whenausdruck then ergebnisausdruck
  [...n]
  [
    else alternativergebnisausdruck
  ]
end
```

die syntax der komplexen case-funktion lautet:

```
case
  when boolescherausdruck then ergebnisausdruck
    [...n]
  [
    else alternativergebnisausdruck
  ]
end
```

der *eingabeausdruck* in der ersten syntaxform kann jeder gültige sql-server-ausdruck sein. das ergebnis dieses ausdrucks wird dann in der angegebenen reihenfolge der when-klauseln mit dem jeweiligen *whenausdruck* auf gleichheit getestet. sobald ein test das ergebnis true liefert, gibt die case-funktion den zugehörigen *ergebnisausdruck* zurück. wenn kein *whenausdruck* mit *eingabeausdruck* übereinstimmt und eine else-klausel angegeben ist, liefert die case-funktion den *alternativergebnisausdruck* zurück. ist auch keine else-klausel vorhanden, lautet der rückgabewert null.

während die einfache case-funktion nacheinander zwei ausdrücke vergleicht, wertet die komplexe case-funktion boolesche ausdrücke aus. ergibt ein *boolescherausdruck* in der when-klausel das ergebnis true, liefert die case-funktion den zugehörigen ergebnisausdruck zurück. die weitere auswertung läuft analog zur einfachen case-funktion ab.

das erste beispiel zeigt den einsatz der einfachen case-funktion. in der tabelle authors der datenbank pubs ist in der spalte state die abkürzung des bundesstaates für die adresse des autors verzeichnet. die folgende anweisung zeigt den vollständigen namen des jeweiligen bundesstaates an:

```
-- casee.sql
use pubs
select distinct staat =
  case state
    when 'al' then 'alabama'
    when 'ak' then 'alaska'
    when 'az' then 'arizona'
    when 'ca' then 'california'
    when 'co' then 'colorado'
    when 'ct' then 'connecticut'
...
    when 'cz' then 'canal zone'
    when 'dc' then 'district of columbia'
    when 'gu' then 'guam'
    when 'pr' then 'puerto rico'
    when 'vi' then 'virgin islands'
    else 'unbekannt'
  end
, state as abkürzung from authors
```

mit dieser anweisung listen sie alle staaten auf, in der autoren tätig sind. das ergebnis lautet:

```
staat                abkürzung
-------------------- ---------
california           ca
indiana              in
kansas               ks
maryland             md
michigan             mi
oregon               or
tennessee            tn
utah                 ut
```

anstelle der konstruktion staat = case ... end können sie auch die syntaxform case ... end as staat verwenden, wie es das zweite beispiel zeigt.

im zweiten beispiel wertet die komplexe form der case-funktion die spalte price der tabelle titles aus und gibt das preisniveau des jeweiligen bereichs zurück:

```
-- casek.sql
use pubs
```

```
select price as preis,
  case
    when price is null then 'preis nicht bekannt'
    when price<10 then 'billigware'
    when price>=10 and price<20 then 'mittleres niveau'
    else 'gehobener preis'
  end as preisniveau
from titles
```

das ergebnis lautet:

```
preis                 preisniveau
--------------------- -------------------
19.9900               mittleres niveau
11.9500               mittleres niveau
2.9900                billigware
...
7.9900                billigware
20.9500               gehobener preis
11.9500               mittleres niveau
14.9900               mittleres niveau
```

12.4 gespeicherte prozeduren erstellen

nachdem sie nun mit variablen und steuerungsstrukturen vertraut sind, können sie gespeicherte prozeduren erstellen, die mit diesen elementen der sprache transact-sql arbeiten. sql server bietet zwar einen assistenten zum erstellen gespeicherter prozeduren, der ihnen aber nur dann einen nutzen bringt, wenn sie mit der syntax der transact-sql-anweisung create procedure vertraut sind. deshalb erläutert der nächste abschnitt zunächst, wie sie gespeicherte prozeduren per transact-sql erstellen.

12.4.1 gespeicherte prozeduren per transact-sql erstellen

auf der ebene von transact-sql erstellen sie gespeicherte prozeduren mit der anweisung create procedure, die folgende syntax hat:

```
create procedure prozedurname [; nummer]
[
  {@parametername datentyp} [varying][= standardwert][output]
]
[,...n]
[with
```

```
  {recompile | encryption | recompile, encryption}
]
[for replication]
as
  sql_anweisungen
```

der *prozedurname* muß den regeln für bezeichner entsprechen und innerhalb der datenbank und für jeden besitzer eindeutig sein. in sql server 7.0 können sie eine lokale temporäre gespeicherte prozedur erstellen, indem sie dem namen ein nummernzeichen (#) voranstellen. die prozedur ist dann nur für die verbindung gültig, die die gespeicherte prozedur erstellt hat. eine globale temporäre gespeicherte prozedur kennzeichnen sie mit zwei nummernzeichen (##). auf diese prozedur können dann alle benutzerverbindungen zugreifen.

die optionale *nummer* bietet die möglichkeit, prozeduren gleichen namens in einer gruppe zusammenzufassen und durch die angehängte nummer zu unterscheiden. sinn und zweck dieser übung ist es, alle prozeduren der gruppe in einem zug löschen zu können.

an eine gespeicherte prozedur können sie parameter übergeben und über die parameter auch werte zurückerhalten. der *parametername* muß mit einem at-zeichen beginnen und den regeln für bezeichner entsprechen. die parameter sind lokal zur prozedur und stehen in keinem zusammenhang mit gleichen parameternamen in anderen prozeduren.

als *datentyp* ist jeder in sql server definierte typ möglich.

das schlüsselwort varying bezeichnet die als ausgabeparameter unterstützte ergebnismenge und ist nur für cursorparameter gültig.

wenn sie einen *standardwert* bereitstellen, können sie die prozedur aufrufen, ohne einen wert für den jeweiligen parameter angeben zu müssen. allerdings sind nur konstanten und der wert null zulässig. beispielsweise können sie keine funktion wie getdate() als standardwert verwenden, um etwa das aktuelle datum an die prozedur zu übergeben.

mit dem schlüsselwort output kennzeichnen sie einen parameter, der einen wert an die aufrufende prozedur zurückgibt. das schlüsselwort muß auch im aufruf der gespeicherten prozedur erscheinen.

der angegebene platzhalter [,...n] bedeutet, daß sie bis zu 1024 parameter spezifizieren können.

die option recompile bewirkt, daß sql server die prozedur bei jeder ausführung neu kompiliert. damit verzichten sie zwar auf die vorteile eines bereits kompilierten ausführungsplans, dennoch läuft die prozedur schneller als die einzelnen transact-sql-anweisungen. verwenden sie diese option, wenn die ausführung der prozedur in hohem maße von übergebenen parametern abhängig ist.

der text einer gespeicherten prozedur wird in der systemtabelle syscomments abgelegt. mit der option encryption verhindern sie, daß unbefugte den text der prozedur einsehen können.

die option for replication markiert eine gespeicherte prozedur für die ausschließliche verwendung in einem replikationsvorgang. die optionen with recompile und for replication schließen sich gegenseitig aus.

die eigentliche funktionalität einer gespeicherten prozedur definieren sie mit den *sql_anweisungen* im anschluß an das schlüsselwort as.

das folgende beispiel erstellt eine gespeicherte prozedur, die das im parameter @indatum übergebene datum in die form tag.monat.jahr konvertiert und im ausgabeparameter @outdatum zurückgibt:

```
-- kdatum.sql
create proc kdatum @indatum datetime, @outdatum char(10) output
as
  select @outdatum=convert(char(10),@indatum,104)
go
```

diese gespeicherte prozedur rufen sie beispielsweise wie folgt auf:

```
declare @heute varchar(10)
declare @sdatum datetime
set @sdatum=getdate()
exec kdatum @sdatum, @heute output
select @heute as heute
```

zunächst werden zwei variablen deklariert, die für die übergabe der parameter vorgesehen sind. die set-anweisung weist der variablen @sdatum das aktuelle systemdatum zu. die gespeicherte prozedur kdatum wandelt das in @sdatum übergebene datum um und gibt es im ausgabeparameter @heute zurück. schließlich zeigt die select-anweisung das konvertierte datum an:

```
heute
----------
20.04.1999
```

die im as-abschnitt angegebenen sql-anweisungen müssen sie nicht in begin und end einschließen, weil die anweisung create procedure allein in einem stapel erscheinen muß und das ende des stapels gleichzeitig das ende des anweisungsblocks darstellt.

12.4.2 assistent zur erstellung gespeicherter prozeduren

wie sie gesehen haben, läßt sich eine gespeicherte prozedur per transact-sql problemlos erstellen. der assistent zur erstellung gespeicherter prozeduren ist vor allem dann hilfreich, wenn sie prozeduren für das einfügen, aktualisieren oder löschen von daten in bestimmten tabellen erstellen wollen.

dieser abschnitt zeigt, wie sie mit dem assistenten eine gespeicherte prozedur für die datenbank lotto erstellen. führen sie dazu die folgenden schritte aus:

1. erweitern sie im enterprise manager die konsolenstruktur mindestens bis zum gewünschten server. klicken sie dann auf die schaltfläche **assistenten auswählen**, oder wählen sie aus dem menü **extras** den befehl **assistenten**. im dialogfeld **assistent auswählen** erweitern sie den zweig **datenbank** und wählen dann den eintrag *assistent für erstellung gespeicherter prozeduren* aus.
2. klicken sie im startdialogfeld auf **weiter**. im nächsten dialogfeld wählen sie die datenbank aus, in der die gespeicherte prozedur zu erstellen ist (siehe abbildung 12.1). klicken sie auf **weiter**.

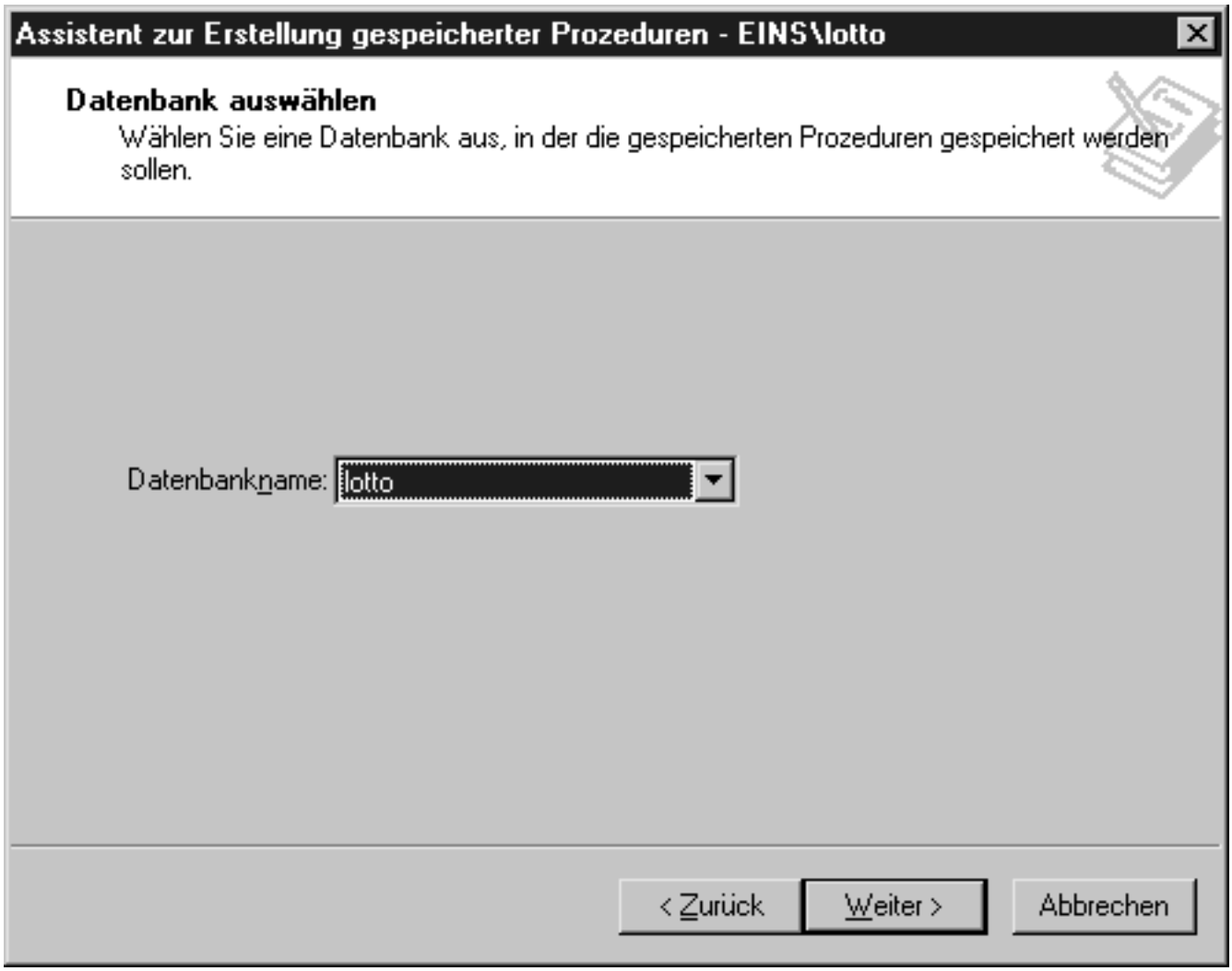


abbildung 12.1: das dialogfeld datenbank auswählen

3. im dialogfeld **gespeicherte prozeduren auswählen** (siehe abbildung 12.2) *müssen* sie eine aktion - d.h. einfügen, löschen oder aktualisieren - markieren, da sonst die schaltfläche **weiter** deaktiviert bleibt und sie den assistenten nicht beenden können. schalten sie für das beispiel das kontrollkästchen in der spalte **einfügen** ein.

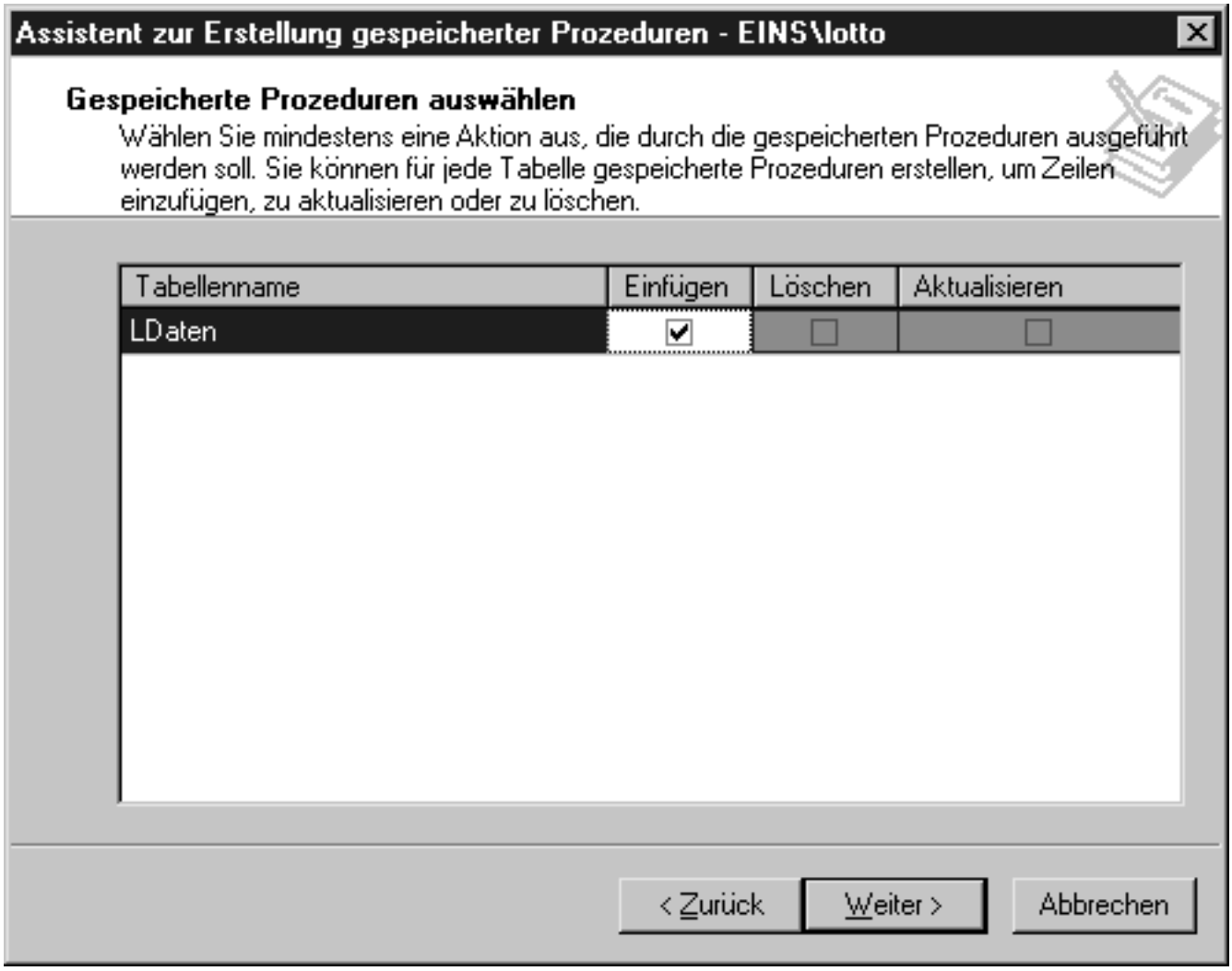


abbildung 12.2: wählen sie hier eine aktion aus, die die gespeicherte prozedur ausführen soll

4. nachdem sie auf **weiter** geklickt haben, erscheint bereits das letzte dialogfeld des assistenten (siehe abbildung 12.3). allerdings dürfte ihre arbeit damit in den wenigsten fällen abgeschlossen sein. klicken sie auf die schaltfläche **bearbeiten**, um die gespeicherte prozedur entsprechend ihren wünschen zu verändern.

Bild

abbildung 12.3: klicken sie im letzten dialogfeld auf bearbeiten, um die prozedur ihren vorstellungen anzupassen

5. im dialogfeld **eigenschaften gespeicherter prozeduren** (siehe abbildung 12.4) können sie zunächst einmal den namen der prozedur ändern und festlegen, welche spalten als parameter an die prozedur zu übergeben sind. geben sie als namen zum beispiel sp_neueziehung ein.

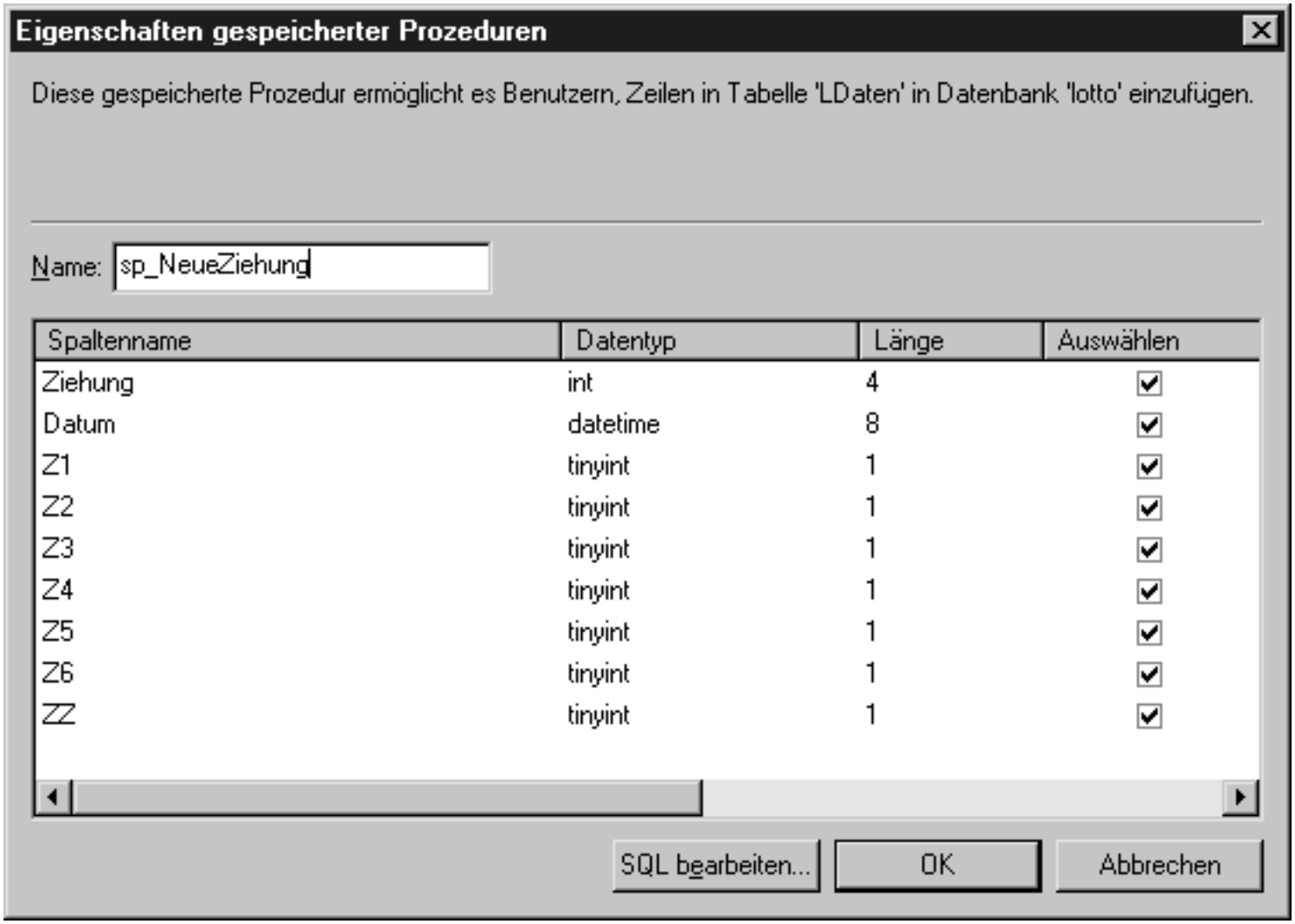


abbildung 12.4: das dialogfeld eigenschaften gespeicherter prozeduren

6. im beispiel sollen alle spalten als übergabeparameter verwendet werden. allerdings soll es möglich sein, das datum explizit anzugeben oder das aktuelle systemdatum als standardwert zu übernehmen. da die übergabe einer funktion als standardwert nicht erlaubt ist, legen sie null als standardwert fest und testen innerhalb der prozedur auf diesen wert. wenn der parameter für das datum den wert null enthält, setzen sie den datumswert auf das systemdatum. bearbeiten sie die transact-sql-anweisung wie in abbildung 12.5 zu sehen. klicken sie auf die schaltfläche **analysieren**, um sich von der korrekten syntax der anweisung zu überzeugen.

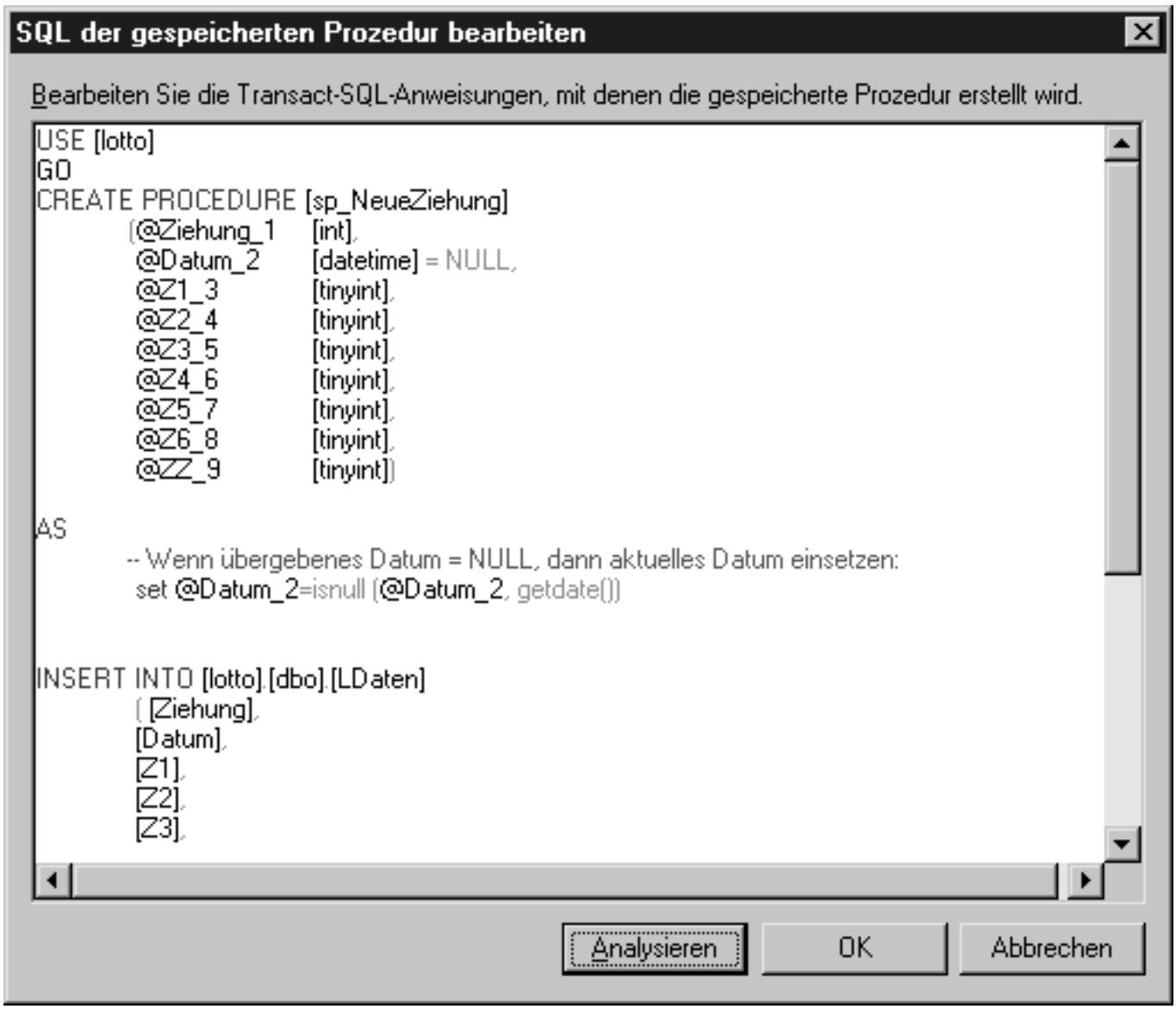


abbildung 12.5: hier können sie die transact-sql-anweisung für die gespeicherte prozedur verändern

7. klicken sie auf **ok** und dann im letzten dialogfeld des assistenten auf **fertigstellen**, um die gespeicherte prozedur erstellen zu lassen.

obwohl sie den namen geändert haben, zeigt das letzte dialogfeld des assistenten immer noch den automatisch generierten namen an. das ist aber lediglich ein schönheitsfehler. die prozedur wird unter dem von ihnen gewählten namen erstellt, wie sie sich durch einen blick in den ordner **gespeicherte prozeduren** der datenbank lotto überzeugen können.

die neu erstellte prozedur können sie nun wie folgt aufrufen:

```
exec sp_neueziehung 2282,'3.7.99',6,15,23,24,25,32,31
```

wie sieht es nun aus, wenn sie das datum im aufruf der gespeicherten prozedur nicht angeben, um den standardwert für das datum einsetzen zu lassen? probieren sie folgenden aufruf aus:

```
exec sp_neueziehung 2282,,6,15,23,24,25,32,31
```

das ergebnis ist ein syntaxfehler. für dieses problem gibt es mehrere lösungen. wenn sie explizit im aufruf den wert null für das datum eintragen, ist die syntax wieder korrekt, und der test auf null in der gespeicherten prozedur bewirkt, daß das aktuelle systemdatum verwendet wird. die nächste möglichkeit besteht darin, mit benannten parametern zu arbeiten, d.h. die namen der parameter zusammen mit den werten anzugeben. das ist aber noch umständlicher, als gleich ein richtiges datum einzugeben. am besten ist es, die prozedur so zu formulieren, daß der parameter für das datum an letzter stelle steht. dann können sie das datum im aufruf der prozedur wahlweise angeben oder einfach weglassen.

genaugenommen ist es überflüssig, den standardwert null in der gespeicherten prozedur zu definieren, da ein nicht übergebener parameter automatisch als null behandelt wird. das beispiel soll lediglich zeigen, wie und wo sie den standardwert unterbringen. echte standardwerte geben sie zum beispiel an, wenn eine prozedur zeilen einfügt und für bestimmte spalten keine null-werte zugelassen sind. dann stellen sie einen standardwert bereit oder fangen einen unzulässig übergebenen null-wert mit programmlogik ab, damit die gespeicherte prozedur keine fehler erzeugt.

12.4.3 gespeicherte prozeduren ändern

um die gespeicherte prozedur sp_neueziehung des obigen beispiels zu ändern, erweitern sie die konsolenstruktur in der datenbank lotto, klicken auf den ordner **gespeicherte prozeduren** und klicken dann im detailbereich mit der rechten maustaste auf die prozedur sp_neueziehung. aus dem kontextmenü wählen sie den befehl **eigenschaften**. ändern sie im dialogfeld **eigenschaften** die definition der gespeicherten prozedur, wie es abbildung 12.6 zeigt.

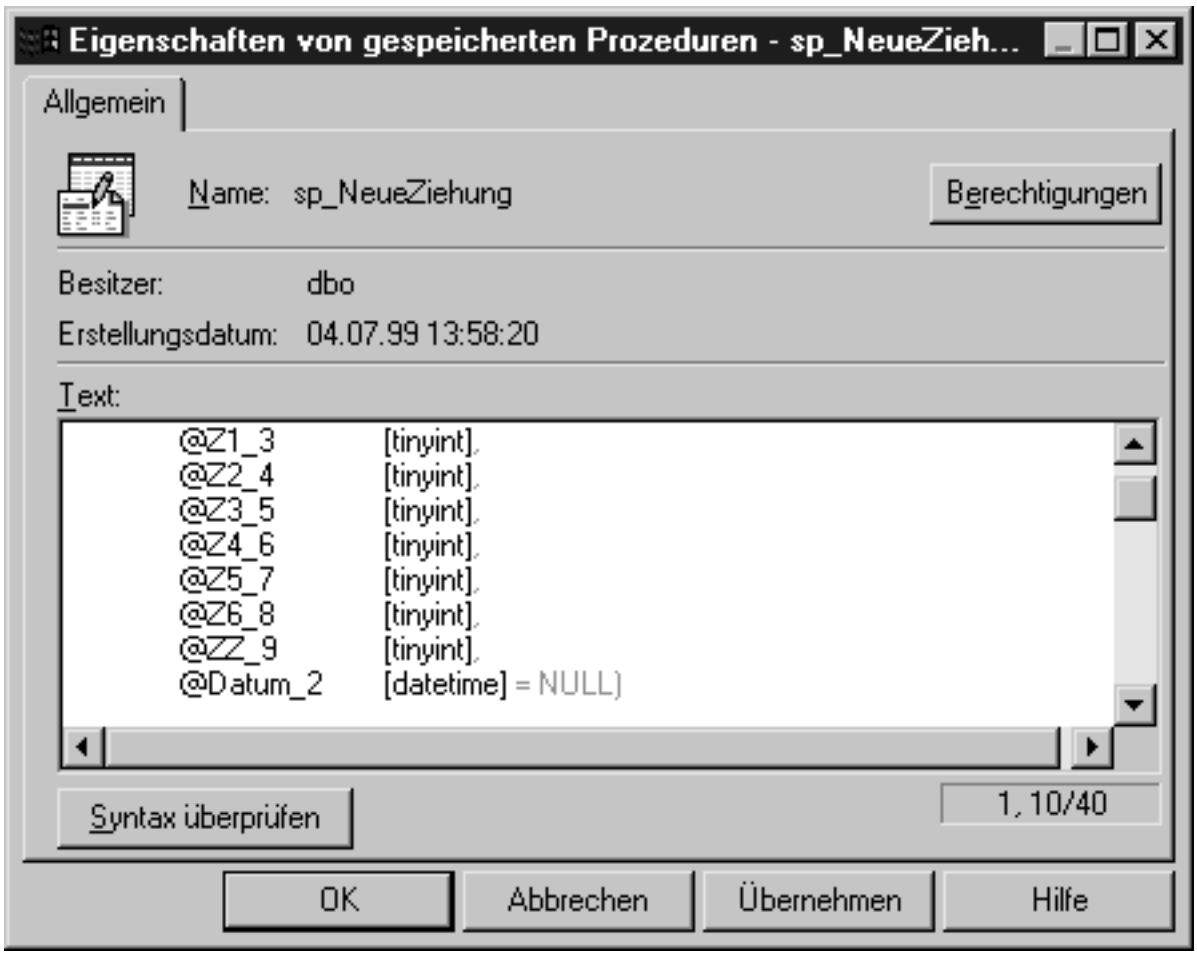


abbildung 12.6: im eigenschaftsdialogfeld können sie die definition einer gespeicherten prozedur ändern

klicken sie zunächst auf **syntax prüfen**, um sich von der korrekten änderung zu überzeugen. anschließend wählen sie erst **übernehmen** und danach **ok**.

der aufruf der gespeicherten prozedur sp_neueziehung läßt sich jetzt wie folgt formulieren:

```
exec sp_neueziehung 2282,6,15,23,24,25,32,31
```

mit angabe eines datums sieht der aufruf so aus:

```
exec sp_neueziehung 2282,6,15,23,24,25,32,31,'3.7.99'
```

eine gespeicherte prozedur können sie auch per transact-sql ändern. die syntax der anweisung alter procedure ist bis auf das schlüsselwort alter anstelle von create die gleiche wie für create procedure.

12.5 gespeicherte prozeduren entfernen

um eine gespeicherte prozedur zu löschen, klicken sie im detailbereich des enterprise managers mit der rechten maustaste auf die prozedur und wählen **löschen** aus dem kontextmenü. es erscheint das dialogfeld **objekte löschen**, in dem sie sich zunächst über die abhängigkeiten informieren können. klicken sie dann auf **alle löschen**, um die prozedur zu entfernen.

per transact-sql löschen sie eine prozedur mit der anweisung drop procedure:

```
drop procedure prozedurliste
```

in der *prozedurliste* können sie mehrere durch komma getrennte prozedurnamen angeben und somit mehrere gespeicherte prozeduren auf einmal löschen.

die prozedur sp_neueziehung werden sie dann wie folgt wieder los:

```
drop procedure sp_neueziehung
```

12.6 prozeduren verschachteln

wenn man eine prozedur aus einer anderen prozedur heraus aufruft, spricht man von verschachtelten prozeduren. die aufgerufene prozedur ist in der aufrufenden prozedur verschachtelt. die ausführung der prozeduren erfolgt damit auf verschiedenen ebenen.

das folgende beispiel zeigt mit der funktion @@nestlevel die aktuelle ausführungsebene von gespeicherten prozeduren an.

```
1:  create procedure innerproc as
2:  select @@nestlevel as 'innere ebene'
3:  go
4:  create procedure outerproc as
5:  select @@nestlevel as 'äußere ebene'
6:  exec innerproc
7:  go
8:
9:  select @@nestlevel as 'ausgangsebene'
10: go
11:
12: execute outerproc
13: go
14: exec innerproc
```

```
15: exec outerproc
```

zunächst werden in den zeilen 1 bis 3 und 4 bis 7 zwei gespeicherte prozeduren erstellt: die erste prozedur innerproc zeigt lediglich die ausführungsebene an. die zweite prozedur outerproc zeigt die ausführungsebene an und ruft dann die innere prozedur innerproc auf, die ihrerseits die ausführungsebene anzeigt.

zeile 9 zeigt zunächst die ausführungsebene an, ohne daß bereits eine gespeicherte prozedur aufgerufen wurde:

```
ausgangsebene
-------------
0
```

die oberste aufrufebene wird mit der nummer 0 gekennzeichnet. als nächstes ruft zeile 12 die gespeicherte prozedur outerproc auf, die folgendes ergebnis liefert:

```
äußere ebene
------------
1

innere ebene
------------
2
```

die erste ergebnismenge kommt vom aufruf der funktion @@nestlevel, der ersten anweisung in der gespeicherten prozedur outerproc.

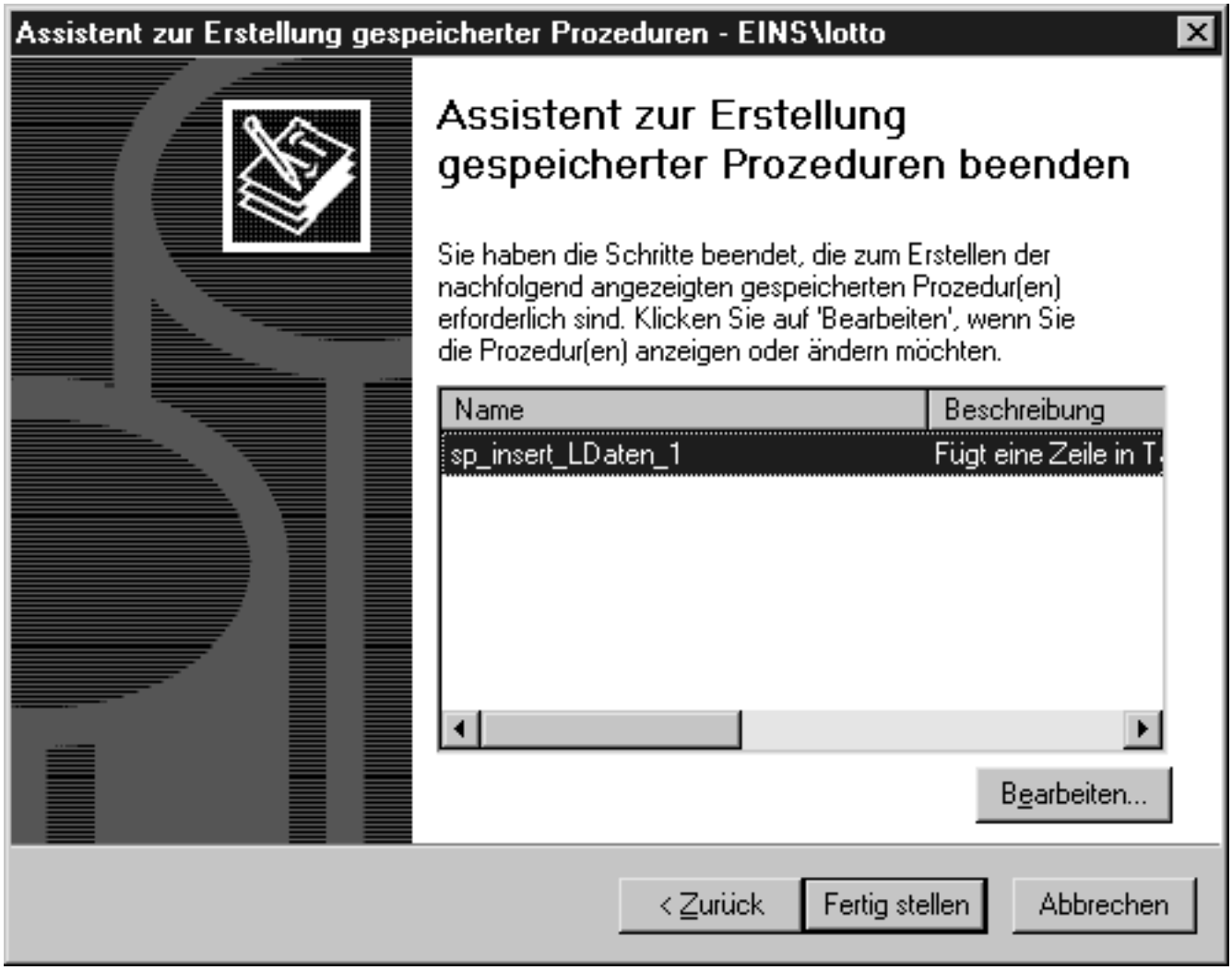
```
innere ebene
------------
1

äußere ebene
------------
1

innere ebene
------------
2
```

12.7 gespeicherte systemprozeduren

in sql server existieren zahlreiche vordefinierte systemprozeduren, mit deren hilfe administrative und informative aufgaben leichter abzuwickeln sind. außerdem werden befehle der datendefinitionssprache (ddl), die nicht im sprachumfang von sql-92 enthalten sind, in form von gespeicherten prozeduren ausgeführt. anhang c bringt eine übersicht über die gespeicherten systemprozeduren von sql server. dort finden sie auch hinweise auf die kapitel, in denen eine bestimmte prozedur näher behandelt wird.

12.8 prozeduren gruppieren

mehrere prozeduren dürfen den gleichen namen tragen, wenn sie sich durch die zugeordnete id unterscheiden. damit kann man gespeicherte prozeduren in einer gruppe zusammenfassen und bei bedarf mit einer einzigen anweisung löschen. die systemtabelle syscolumns verzeichnet die nummer jeder prozedur einer gruppe in der spalte number.


Assistent zur Erstellung gespeicherter Prozeduren - EINS\lotto
Assistent zur Erstellung gespeicherter Prozeduren beenden
Sie haben die Schritte beendet, die zum Erstellen der nachfolgend angezeigten gespeicherten Prozedur(en) erforderlich sind. Klicken Sie auf 'Bearbeiten', wenn Sie die Prozedur(en) anzeigen oder ändern möchten.
Name
Beschreibung
sp_insert_LDaten_1
Fügt eine Zeile in T
Bearbeiten...
< Zurück
Fertig stellen
Abbrechen

kapitel 13 sichten

13.1 auf einen blick

eine sicht ist eine virtuelle tabelle, deren inhalt durch eine abfrage erzeugt wird. während tabellen permanent gespeichert werden, existiert eine sicht nicht in form eines gespeicherten satzes von datenwerten in der datenbank. in der abfrage, die die sicht definiert, sind die zeilen und spalten angegeben, aus denen die sicht bestehen soll. die sicht wird dynamisch erzeugt, sobald man sich darauf bezieht.

bild

abbildung 13.1: beispiel für eine sicht aus der datenbank pubs

abbildung 13.1 zeigt das beispiel der sicht titleview aus der datenbank pubs.

in dieser sicht sind spalten aus verschiedenen tabellen der datenbank pubs enthalten:

- title aus der tabelle titles,
- au_ord aus der tabelle titleauthor,
- au_lname aus der tabelle authors,
- price aus der tabelle titles,
- ytd_sales aus der tabelle titles,
- pub_id aus der tabelle titles.

abbildung 13.2 zeigt das datenbankdiagramm mit den tabellen, die der sicht zugrundeliegen.

bild

abbildung 13.2: tabellen, aus denen die sicht titleview erstellt wird

das gleiche ergebnis wie in abbildung 13.1 hätte man auch mit der anweisung

```
select title, au_ord, au_lname, price, ytd_sales, pub_id
from authors, titles, titleauthor
where authors.au_id = titleauthor.au_id
   and titles.title_id = titleauthor.title_id
```

erhalten können. wenn man abfragen häufig in der gleichen form benötigt, ist es sehr umständlich, zeitaufwendig und fehlerträchtig, diese anweisungen immer wieder einzutippen. wie abbildung 13.1 gezeigt hat, war lediglich die anweisung

```
select * from titleview
```

erforderlich, um die gewünschten spalten abzurufen.

das ist aber nicht der einzige vorteil, den sichten bieten. man verwendet sichten unter anderem, um

- ein neues virtuelles tabellenformat zu erzeugen
- komplexe abfragen zu vereinfachen
- sicherheitsfunktionen zu realisieren, indem man berechtigungen für eine sicht, nicht jedoch für die zugrundeliegende(n) tabelle(n) gewährt und nur ausgewählte - und keine vertraulichen - daten in die sicht übernimmt
- daten zu filtern
- ausgewählte daten zu importieren und zu exportieren
- zwischen einheiten umzurechnen

13.2 sichten erstellen

die beispiele in diesem abschnitt arbeiten mit der datenbank pubs. mit den werkzeugen von sql server erstellen sie folgende sichten:

- eine einfache sicht, die lediglich auf einer tabelle basiert und ein neues tabellenformat erzeugt
- eine sicht, die aus der spalte price der tabelle titles eine sicht mit berechneten spalten erzeugt, um den anteil der umsatzsteuer auszuweisen und die preise in euro anzugeben.

13.2.1 sichterstellungs-assistent

der sichterstellungs-assistent bietet sowohl eine bequeme möglichkeit, eine sicht interaktiv zu erstellen, als auch eine hervorragende einführung in die transact-sql-befehle, mit denen man sichten erzeugen kann. eigentlich stellt der sichterstellungs-assistent eine symbiose aus einem herkömmlichen assistenten mit einem editor für transact-sql-anweisungen dar. die folgenden schritte zeigen an einem beispiel, wie man eine sicht auf ausgewählte tabellen und spalten der datenbank pubs erzeugt.

1. erweitern sie die konsolenstruktur im enterprise manager bis zur datenbank pubs. (es genügt auch, die struktur bis zum server oder dem ordner **datenbanken** zu erweitern. der assistent fragt ohnehin nach der zu verwendenden datenbank.)
2. wählen sie im menü **extras** den befehl **assistenten**, oder klicken sie in der symbolleiste des enterprise managers auf die schaltfläche **assistenten auswählen**. im dialogfeld **assistent auswählen** erweitern sie den zweig **datenbanken**, markieren den eintrag *sichterstellungs-assistent* und klicken auf **ok**.
3. klicken sie im startdialogfeld des assistenten auf **weiter**. im zweiten dialogfeld wählen sie aus dem drop-down-listenfeld **datenbankname** die datenbank pubs aus. klicken sie auf **weiter**.
4. im dritten schritt wählen sie die tabellen aus, die der sicht zugrundeliegen. schalten sie das kontrollkästchen für die tabelle titles ein, da die sicht den titel und den preis anzeigen soll (siehe

abbildung 13.3). klicken sie auf **weiter**.

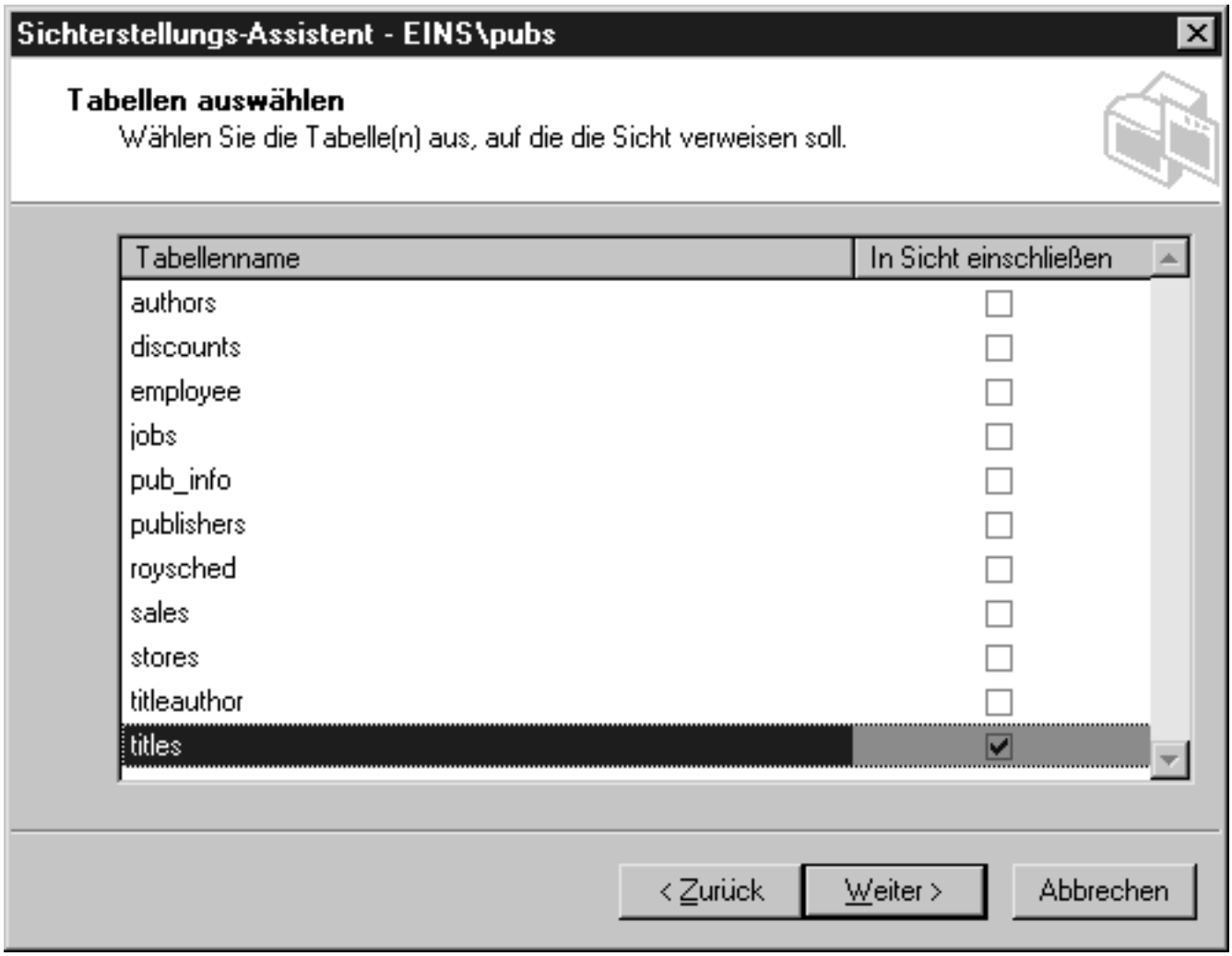


abbildung 13.3: im dritten dialogfeld des sichterstellungs-assistenten wählen sie die tabellen aus

5. das vierte dialogfeld zeigt alle spalten für alle tabellen an, die sie im vorherigen schritt markiert haben (siehe abbildung 13.4). die spalten sind unter **spaltenname** in der form *tabellenname.spaltenname* aufgeführt. das dialogfeld zeigt außerdem den datentyp der einzelnen spalten zur orientierung an. schalten sie unter **spalte auswählen** die kontrollkästchen für die spalten title und price der tabelle titles ein (d.h. titles.title bzw. titles.price).

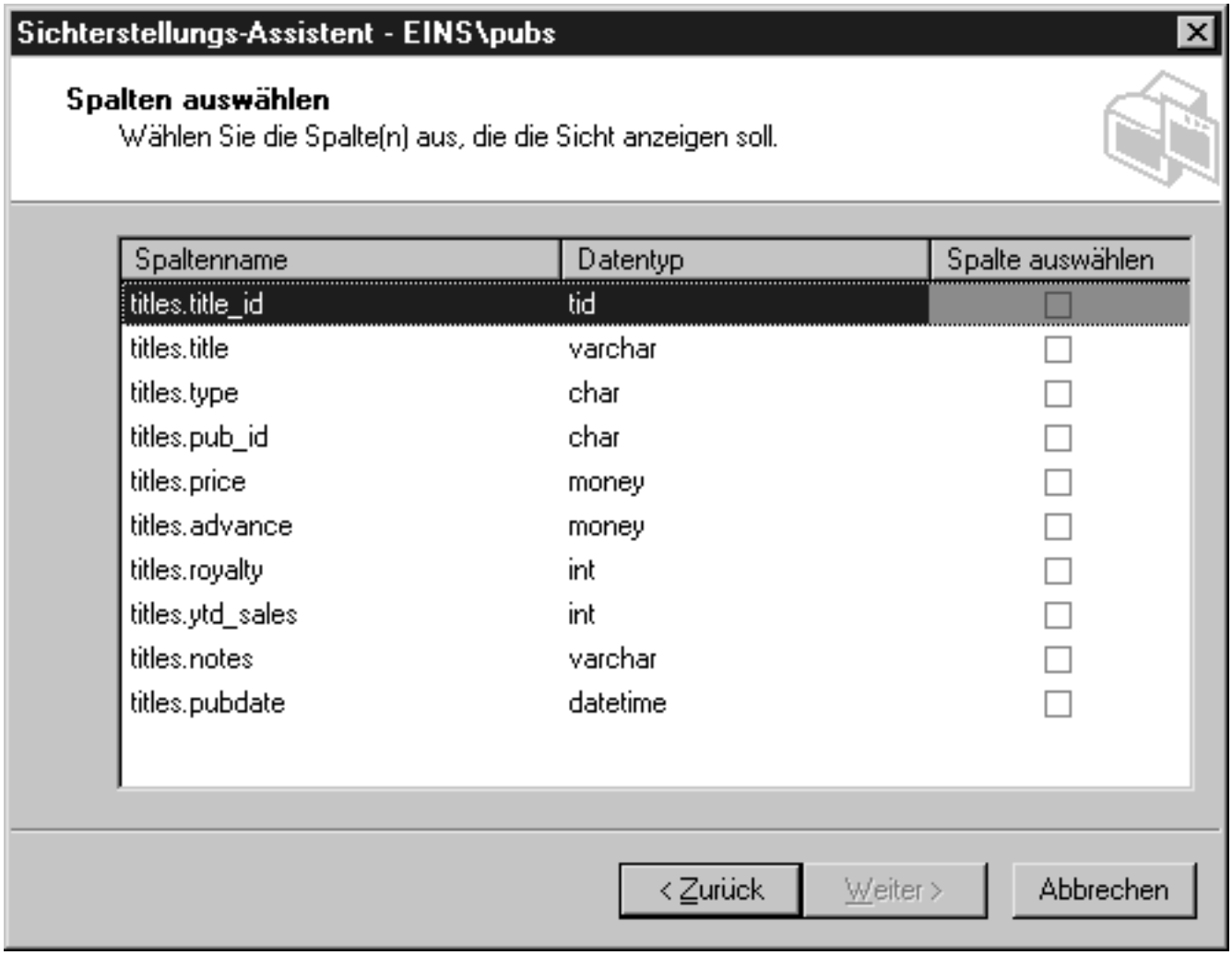


abbildung 13.4: das dialogfeld zeigt alle spalten für alle im vorherigen schritt markierten tabellen an

6. in kapitel 5 wurde bereits darauf hingewiesen, daß kenntnisse zu transact-sql manchmal auch beim einsatz der grafischen werkzeuge erforderlich sind. der sichterstellungs-assistent ist ein beispiel dafür. im fünften dialogfeld lassen sich beschränkungen definieren. diese sind in form von where-klauseln zu formulieren. wenn die sicht zum beispiel nur titel anzeigen soll, für die tatsächlich ein preis ausgewiesen ist (die preisspalte also keine null-werte enthält), geben sie die in abbildung 13.5 dargestellte where-klausel ein. klicken sie auf **weiter**.

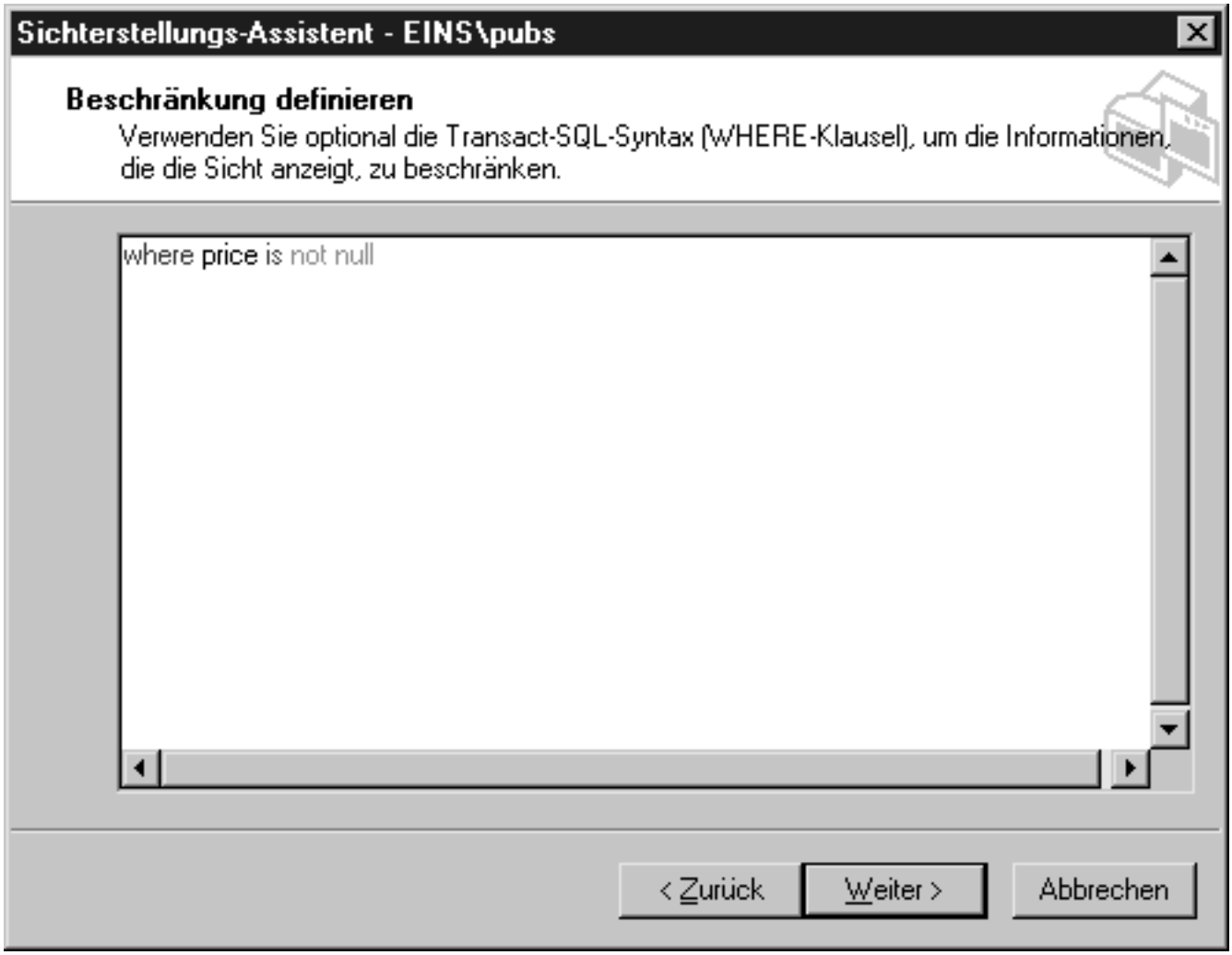


abbildung 13.5: mit where-klauseln lassen sich beschränkungen definieren

7. im nächsten schritt fordert sie der assistent auf, die sicht zu benennen. legen sie den namen für das beispiel im feld **sichtname** mit titelpreisview fest, und klicken sie auf **weiter**.
8. das letzte dialogfeld zeigt die generierte transact-sql-anweisung, mit der die sicht erstellt wird (siehe abbildung 13.6). bei bedarf können sie die anweisung in diesem dialogfeld bearbeiten oder auch zurückgehen, um andere tabellen, spalten oder bedingungen festzulegen. wenn sie die anweisung übernehmen möchten, klicken sie auf **fertigstellen**. der assistent erstellt daraufhin die sicht und zeigt eine bestätigung an.

bild

abbildung 13.6: das letzte dialogfeld zeigt die generierte transact-sql-anweisung an

bezüglich einer sicht können sie genauso eine abfrage ausführen wie zu einer tabelle. wenn sie beispielsweise im sql server query analyzer die anweisung

```
select * from titelpreisview
```

ausführen, erhalten sie das gewünschte ergebnis, wie es abbildung 13.7 zeigt.

bild

abbildung 13.7: ausführen einer abfrage bezüglich der sicht titelpreisview

13.2.2 sichten im enterprise manager erstellen

wenn sie nicht gerade darauf aus sind, die transact-sql-anweisung im sichterstellungs-assistenten zu bearbeiten, bietet sich der assistent nur für einfachste sichten an. mit dem enterprise manager lassen sich komplexere sichten erstellen. die sicht im folgenden beispiel übernimmt die spalten title und price (wie im letzten beispiel) aus der zugrundeliegenden tabelle titles der datenbank pubs und fügt zwei berechnete spalten hinzu: den anteil der umsatzsteuer bezüglich der spalte price und den in euro umgerechneten preis.

diese sicht erstellen sie mit dem enterprise manager in folgenden schritten:

1. erweitern sie die konsolenstruktur bis zur datenbank pubs.
2. klicken sie im ordner **datenbanken** mit der rechten maustaste auf pubs, und wählen sie aus dem kontextmenü den befehl **neu / sicht**. sie können auch die datenbank pubs erweitern, mit der rechten maustaste auf **sichten** klicken und aus dem kontextmenü **neue sicht** wählen.
3. klicken sie mit der rechten maustaste auf den diagrammbereich (siehe abbildung 13.8), und wählen sie aus dem kontextmenü den befehl **tabelle hinzufügen**.

das fenster **neue sicht** gliedert sich von oben nach unten in den *diagrammbereich*, den *kriterienbereich*, den *sql-bereich* und den *ergebnisbereich*. die einzelnen bereiche lassen sich über die schaltflächen der symbolleiste (dritte bis sechste von links) ein- und ausblenden.

bild

abbildung 13.8: der diagrammbereich im fenster neue sicht mit geöffnetem kontextmenü

4. markieren sie auf der registerkarte **tabellen** im dialogfeld **tabelle hinzufügen** die tabelle titles (siehe abbildung 13.9). klicken sie auf die schaltfläche **hinzufügen**. daraufhin wird die tabelle in den diagrammbereich des fensters **neue sicht** übernommen. das dialogfeld **tabelle hinzufügen** bleibt geöffnet, um weitere tabellen oder sichten (registerkarte **sichten**) in die neue sicht aufnehmen zu können. das beispiel verwendet nur eine tabelle. klikken sie auf **schliessen**.

sichten

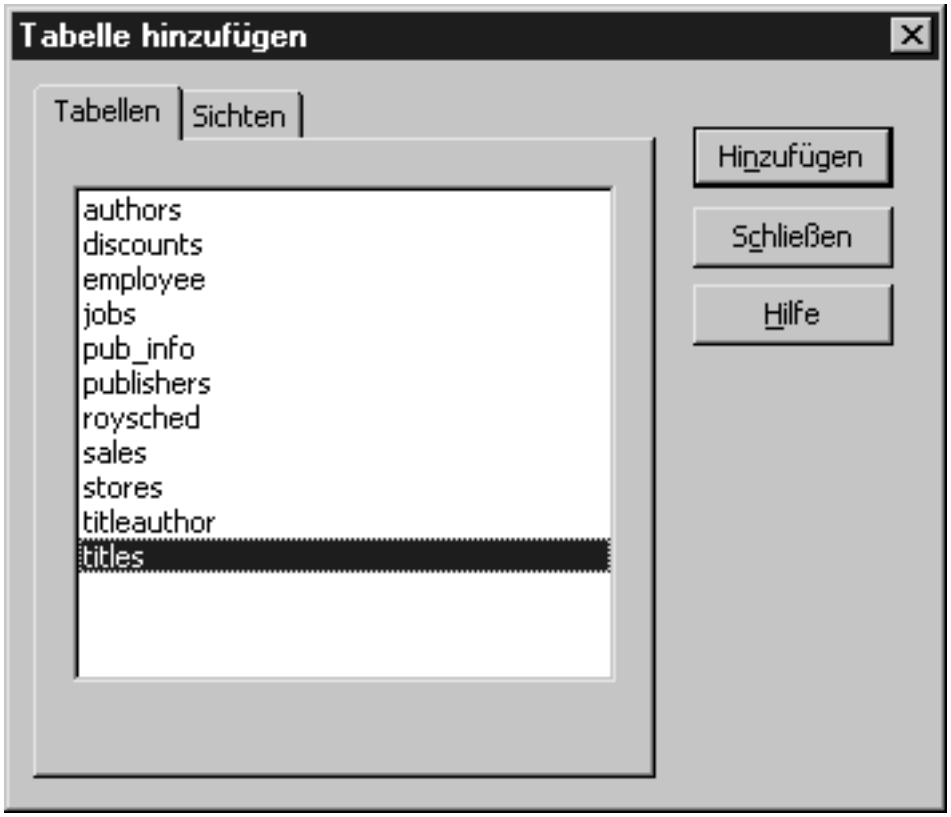


abbildung 13.9: das dialogfeld tabelle hinzufügen mit aktiver registerkarte tabellen

5. um spalten der ausgewählten tabellen in das abfrageergebnis aufzunehmen, markieren sie die spalte entweder in der tabelle oder klicken im kriterienbereich auf eine zelle unter *spalte* und wählen die spalte aus der drop-down-liste aus. wenn sie den mauszeiger in der tabelle auf den spaltennamen setzen, erscheint eine quickinfo mit dem namen und dem datentyp der spalte (siehe abbildung 13.10). wählen sie für das beispiel die spalten title und price aus.

bild

abbildung 13.10: spalten in das abfrageergebnis aufnehmen

6. als nächstes fügen sie der sicht drei berechnete spalten hinzu. die umsatzsteuer für bücher beträgt derzeit 7 prozent. um den nettopreis zu berechnen, ist der wert in der spalte price durch 1,07 zu dividieren. der anteil der umsatzsteuer ergibt sich aus price minus nettopreis.

geben sie in die nächsten freien felder unter *spalte* die folgenden drei ausdrücke ein:

```
convert(money; price / 1,07)
convert(money; price - price / 1,07)
convert(money; price / 1,95583)
```

wenn sie im kriterienbereich einen ausdruck unter *spalte* eingeben und dann das feld verlassen, erscheint automatisch unter *alias* ein virtueller spaltenname (expr1, expr2 usw. in der voreinstellung). überschreiben sie den vorgegebenen spaltennamen durch eine aussagekräftige bezeichnung (im beispiel netto, ust und euro).

wenn sie die ausdrücke in die felder des kriterienbereichs eingeben, müssen sie die gültigen

ländereinstellungen beachten. in einer transact-sql-anweisung trennen sie zum beispiel die parameter der funktion convert durch kommas und verwenden als dezimaltrennzeichen für gleitkommazahlen den punkt. im kriterienbereich müssen sie statt dessen die parameter jeweils durch semikolon trennen und als dezimaltrennzeichen das komma verwenden. im sql-bereich bleibt es bei der englischen schreibweise (parameter durch komma getrennt, dezimal*punkt*).

7. ersetzen sie die vorgegebenen aliasnamen durch netto, ust und euro. bei berechneten spalten sind aliasnamen erforderlich. um die verwendung von optionalen aliasnamen zu zeigen, geben sie bitte für die spalte price den namen preis ein. die spalte title erhält im beispiel keinen alias. wir kommen weiter unten darauf zurück.
8. in der spalte *kriterien* des kriterienbereichs lassen sich bedingungen formulieren. die hier eingegebenen ausdrücke erscheinen in der where-klausel der select-anweisung. tragen sie für die spalte price die bedingung is not null ein.

wenn sie nach einer spalte gruppieren möchten, klicken sie mit der rechten maustaste auf die spalte und wählen den befehl **gruppieren nach**. daraufhin wird im kriterienbereich eine zusätzliche spalte *gruppieren nach* eingefügt. klicken sie in das gewünschte feld, um eine gruppenfunktion auszuwählen. bei aktiviertem **gruppieren nach** erscheint in der transact-sql-anweisung eine group by-klausel, und die where-klausel wird zur having-klausel (siehe dazu kapitel 10).

9. klicken sie im sql-server-bereich mit der rechten maustaste, und wählen sie den befehl **ausführen**. sql server startet daraufhin die abfrage und gibt die ergebnismenge im ergebnisbereich aus (siehe abbildung 13.11).

Bild

abbildung 13.11: für die neu erstellte sicht wurde eine abfrage ausgeführt

10. für die sicht (d.h. die abfrage) können sie weitere optionen festlegen. klicken sie dazu an einer beliebigen stelle im fenster **neue sicht** mit der rechten maustaste, und wählen sie den befehl **eigenschaften**. daraufhin wird das dialogfeld **eigenschaften** mit der registerkarte **abfrage** angezeigt (siehe abbildung 13.12). eine ganze zahl oder ein prozentwert im feld **top** bewirkt, daß nur die ersten zeilen in der ergebnismenge erscheinen. wenn das kontrollkästchen **distinct-werte** eingeschaltet ist, werden alle duplikate aus der ergebnismenge gefiltert.

abbildung 13.12: im dialogfeld eigenschaften können sie weitere optionen für die abfrage festlegen

11. wenn sie mit dem test der sicht zufrieden sind, klicken sie auf die erste schaltfläche in der symbolleiste, um die sicht zu speichern. geben sie im dialogfeld **speichern unter** den namen der sicht ein, zum beispiel preiseuroview. die sicht gehört nun zur datenbank pubs.

wenn sie im diagrammbereich, kriterienbereich oder sql-bereich änderungen vornehmen, wirken diese änderungen unmittelbar auf die anderen bereiche. gegebenenfalls müssen sie erst den cursor in einen anderen bereich setzen, damit die änderungen sichtbar werden. zum beispiel können sie im sql-bereich eine weitere tabelle in der from-klausel angeben. die tabelle erscheint dann automatisch im diagrammbereich (nachdem sie den cursor in einen anderen bereich gesetzt haben).

13.2.3 sichten mit transact-sql erstellen (create view)

im letzten schritt des sichterstellungs-assistenten haben sie bereits einen einblick in die transact-sql-anweisung create view erhalten. beim erstellen der sicht mit dem enterprise manager konnten sie sogar verfolgen, wie sich bestimmte elemente in dieser anweisung niederschlagen. die syntax der anweisung create view sieht folgendermaßen aus:

syntax:

```
create view sichtname [(spaltenliste)]
[with encryption]
as
  selectanweisung
[with check option]
```

aus der syntax geht hervor, daß eine sicht im wesentlichen aus einer select-anweisung besteht, die von einem zusätzlichen gerüst zur sicht gemacht wird.

tabelle 13.1 erläutert die argumente der anweisung create view.

argument	bedeutung
sichtname	name der sicht. die sichtnamen müssen den regeln für bezeichner entsprechen. optional kann man den besitzer der sicht angeben.
spaltenliste	durch komma getrennte liste der namen für die spalten in der sicht. wenn keine spaltennamen angegeben sind, übernehmen die spalten der sicht die namen der zugrundeliegenden spalten. den spaltennamen kann man angeben, wenn die spalte in der sicht einen anderen namen als die zugrundeliegende spalte erhalten soll. der spaltenname ist erforderlich, wenn die spalte in der sicht von einem arithmetischen ausdruck, einer funktion oder einer konstanten abgeleitet wird oder wenn der spaltenname doppelt vorkommen würde.
with encryption	verschlüsselt die einträge in der systemtabelle syscomments, die den text der create view-anweisung speichert.
as	schlüsselwort, das die nachfolgenden aktionen für die sicht einleitet.
select-anweisung	eine select-anweisung, die spalten aus mehreren tabellen und anderen sichten enthalten kann.
with check option	bei dieser option müssen alle datenänderungen in der sicht den in der select-anweisung festgelegten kriterien entsprechen.

tabelle 13.1: argumente der anweisung create view

für die select-anweisung gelten in einer sicht folgende einschränkungen:

- die klauseln order by, compute und compute by sind nicht zulässig.
- das schlüsselwort into darf nicht verwendet werden.
- die select-anweisung darf nicht auf eine temporäre tabelle verweisen.

es empfiehlt sich nicht, sichten zu verschachteln. wenn die select-anweisung auf andere sichten verweist und diese vielleicht wieder auf andere sichten, wird dieses gebilde schnell undurchschaubar. löscht man eine sicht, bleiben die davon abhängigen sichten erhalten, sie sind aber nicht mehr funktionsfähig, weil die basisdaten fehlen.

das folgende beispiel zeigt einen ausschnitt aus dem skript instpubs.sql. die anweisung erzeugt die sicht titleview aus den tabellen authors, titles und titleauthor. die spalten title, price, ytd_sales und pub_id entnimmt die sicht aus der tabelle titles, die spalte au_ord aus der tabelle titleauthor und die spalte au_lname aus der tabelle authors. als bedingung gilt, daß die spalten au_id der tabellen authors und titleauthor sowie die spalten title_id der tabellen titles und titleauthor jeweils gleiche werte aufweisen müssen.

```
833:create view titleview
834:as
835:select title, au_ord, au_lname, price, ytd_sales, pub_id
836:from authors, titles, titleauthor
837:where authors.au_id = titleauthor.au_id
838:   and titles.title_id = titleauthor.title_id
839:
840:go
```

wenn sie diese sicht mit

```
use pubs
select * from titleview
```

aufrufen, erhalten sie die eingangs in abbildung 13.1 dargestellte ergebnismenge.

13.3 sichten ändern

wenn man eine sicht entwickelt, kann es durchaus vorkommen, daß man zum beispiel zusätzliche spalten aufnehmen muß oder die bedingung in der where-klausel ändern will. es ist also die definition der sicht zu ändern. das läßt sich sowohl mit dem enterprise manager als auch per transact-sql erledigen.

13.3.1 sichten im enterprise manager ändern

ähnlich zu den weiter vorn beschriebenen schritten, nach denen sie die sicht preiseuroview erstellt haben, können sie die sicht auch ändern. führen sie dazu folgende schritte aus:

1. erweitern sie die konsolenstruktur bis zur datenbank, in der die zu ändernde sicht enthalten ist, und erweitern sie den ordner der datenbank (im beispiel pubs).
2. klicken sie auf **sichten**, so daß im rechten fensterbereich eine liste der sichten erscheint.
3. klicken sie mit der rechten maustaste auf die zu ändernde sicht (im beispiel preiseuroview), und wählen sie aus dem kontextmenü den befehl **sicht bearbeiten**.
4. im fenster mit diagrammbereich, kriterienbereich, sql-bereich und ausgabebereich können sie jetzt die sicht bearbeiten, wie es im abschnitt »sichten im enterprise manager erstellen« beschrieben wurde.
5. nachdem sie die sicht gespeichert haben, steht die neue definition der sicht zur abfrage bereit.

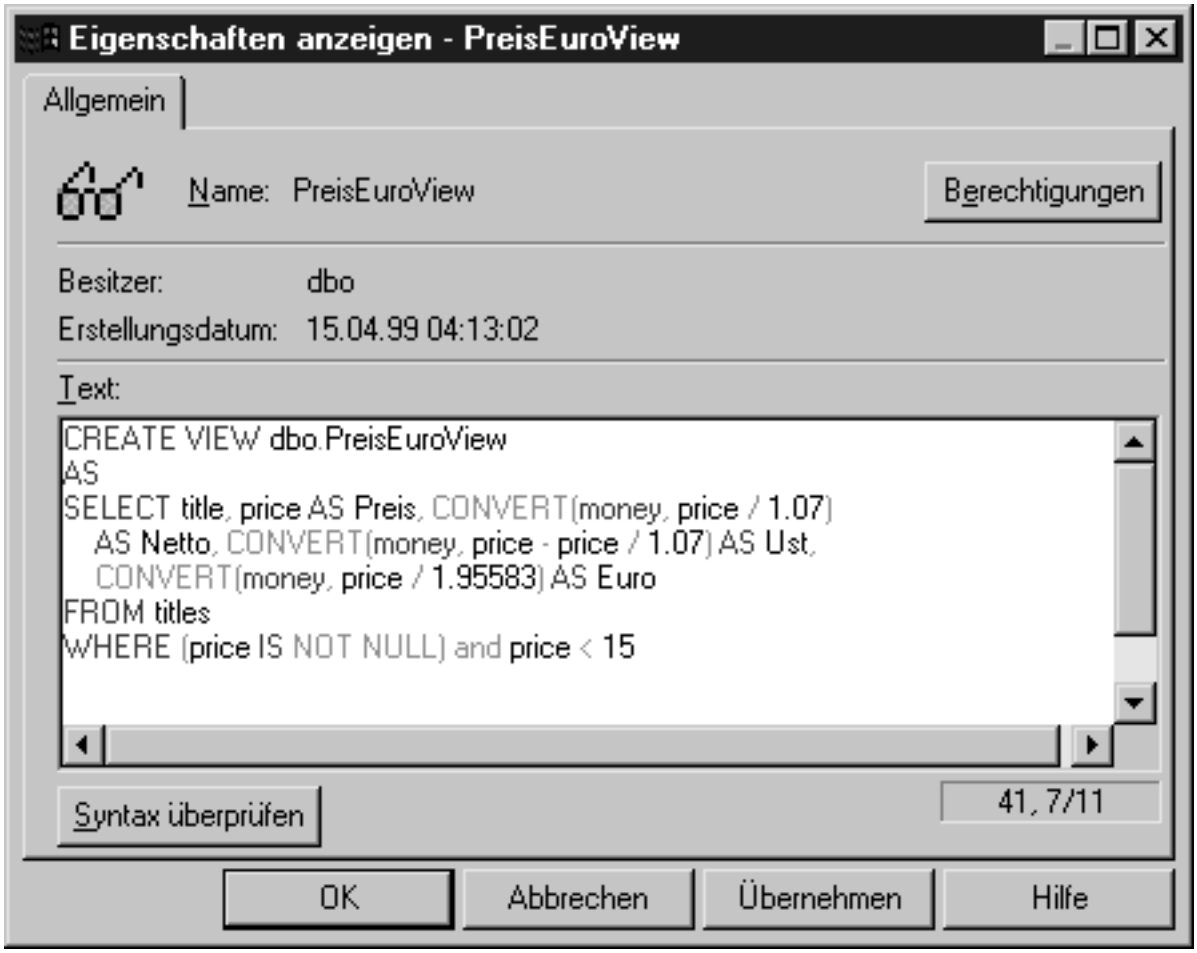


abbildung 13.13: im dialogfeld eigenschaften anzeigen können sie die transact-sql-anweisung der sicht bearbeiten

wenn sie die transact-sql-anweisung, die die sicht definiert, direkt bearbeiten möchten, können sie dazu das dialogfeld **eigenschaften** für die sicht aufrufen. dazu doppelklicken sie auf den namen der sicht in der liste des rechten fensterbereichs oder klicken mit der rechten maustaste auf den namen und wählen **eigenschaften** aus dem kontextmenü. es erscheint das dialogfeld **eigenschaften anzeigen** - *sichtname* (siehe abbildung 13.13).

im beispiel wurde für die sicht preiseuroview die where-klausel in der letzten zeile um die bedingung

```
and price < 15
```

erweitert. nachdem sie eine änderung vorgenommen haben, können sie sofort die korrekte syntax kontrollieren. klicken sie dazu auf die schaltfläche **syntax überprüfen**. wenn alles ok ist, klicken sie auf **übernehmen**. von diesem zeitpunkt an steht die neue definition der sicht in der datenbank bereit.

13.3.2 sichten mit transact-sql ändern (alter view)

sql server 7.0 erlaubt es jetzt, die definition einer sicht per transact-sql mit der anweisung alter view zu ändern. die syntax dieser anweisung sieht folgendermaßen aus:

```
alter view sichtname [(spaltenliste)]
[with encryption]
as
  selectanweisung
[with check option]
```

es zeigt sich, daß die syntax bis auf das schlüsselwort alter mit der syntax der anweisung create view identisch ist. vor allem aber bleiben die berechtigungen erhalten, die sie in der alten definition festgelegt haben. das gilt auch für spalten, die sie in der neuen definition umbenennen. kommen allerdings neue objekte hinzu, müssen sie auch die berechtigungen dafür festlegen. wenn sie dagegen eine sicht löschen und dann neu erstellen, gehen alle berechtigungen verloren.

13.4 mit sichten arbeiten

in den abschnitten zum erstellen von sichten haben sie bereits einige beispiele kennengelernt, wie man abfragen bezüglich einer sicht ausführt. die datenübertragung zwischen den basistabellen und der sicht ist keine einbahnstraße. man kann sowohl daten abrufen als auch ändern, einfügen und löschen. allerdings ist die modifikation von daten in einer sicht nicht so uneingeschränkt wie bei einer normalen tabelle möglich.

13.4.1 daten abrufen

aus einer sicht lassen sich die daten genau wie aus einer tabelle abrufen. man kann also auch bestimmte spalten auswählen und bedingungen angeben:

```
use pubs
select cast(title as char(30))as titel, preis from preiseuroview
where euro>10
```

diese anweisung ruft alle zeilen der sicht preiseuroview ab, in denen der preis höher als 10 euro ist. ausgewählt werden nur die spalten title und euro.

```
titel                          euro
------------------------------ ---------------------
the busy executive's database  10.2207
straight talk about computers  10.2207
silicon valley gastronomic tre 10.2207
but is it user friendly?       11.7341
secrets of silicon valley      10.2258
computer phobic and non-phobic 11.0388
```

```
prolonged data deprivation: fo 10.2207
onions, leeks, and garlic: coo 10.7116

(8 row(s) affected)
```

13.4.2 spaltennamen

bei einer abfrage bezüglich einer sicht geben sie die spaltennamen so an, wie sie in der sicht - und nicht in der basistabelle - lauten. wenn sie die sicht preiseuroview nach den weiter oben beschriebenen schritten mit dem enterprise manager erstellt haben, weist die spalte title keinen aliasnamen und die spalte price den aliasnamen preis auf. die abfrage im letzten beispiel hat deshalb die spalten title und preis referenziert - und nicht title und price.

13.4.3 daten ändern

wie bereits erwähnt, ist eine sicht eine gespeicherte abfrage. das heißt aber nicht, daß man die daten nur abrufen kann. mit einer sicht lassen sich die daten auch modifizieren - d.h. mit den anweisungen update, insert und delete aktualisieren, einfügen und löschen. die änderungen an den daten einer sicht »greifen auf die basistabellen durch«.

die daten einer sicht aktualisiert man mit einer update-anweisung gemäß der folgenden syntax:

```
update sichtname
  set spaltenname = ausdruck
```

das erste beispiel soll die spalte preis der sicht mit dem faktor 2 multiplizieren. damit sich die ergebnisse nachprüfen lassen, rufen wir zunächst mit

```
select cast(title as varchar(30)) as titel, preis from
preiseuroview
```

die ursprünglichen preise ab:

```
titel                          preis
------------------------------ ---------------------
the busy executive's database  19.9900
cooking with computers: surrep 11.9500
you can combat computer stress 2.9900
straight talk about computers  19.9900
silicon valley gastronomic tre 19.9900
```

```
the gourmet microwave          2.9900
but is it user friendly?       22.9500
secrets of silicon valley      20.0000
computer phobic and non-phobic 21.5900
is anger the enemy?            10.9500
life without fear              7.0000
prolonged data deprivation: fo 19.9900
emotional security: a new algo 7.9900
onions, leeks, and garlic: coo 20.9500
fifty years in buckingham pala 11.9500
sushi, anyone?                 14.9900

(16 row(s) affected)
```

als nächstes wird die spalte preis der sicht mit der anweisung

```
update preiseuroview
set preis = preis * 2
```

aktualisiert. fragen sie die preisspalte zur kontrolle wieder mit der obigen select-anweisung ab. die preise haben sich verdoppelt. schließlich ist zu klären, ob sich die geänderten preise auch in der zugrundeliegenden basistabelle titles niederschlagen. führen sie dazu die folgende anweisung aus:

```
select cast(title as varchar(30)) as titel, price from titles
```

die ergebnismenge bestätigt die durchgeführte aktualisierung:

```
titel                          price
------------------------------ ---------------------
the busy executive's database  39.9800
cooking with computers: surrep 23.9000
you can combat computer stress 5.9800
straight talk about computers  39.9800
silicon valley gastronomic tre 39.9800
the gourmet microwave          5.9800
the psychology of computer coo null
but is it user friendly?       45.9000
secrets of silicon valley      40.0000
net etiquette                  null
computer phobic and non-phobic 43.1800
```

```
is anger the enemy?            21.9000
life without fear              14.0000
prolonged data deprivation: fo 39.9800
emotional security: a new algo 15.9800
onions, leeks, and garlic: coo 41.9000
fifty years in buckingham pala 23.9000
sushi, anyone?                 29.9800

(18 row(s) affected)
```

im ergebnis sind zwar die null-werte enthalten, die in der sicht mit der where-klausel unterdrückt wurden, die preise haben sich aber tatsächlich in der basistabelle verdoppelt.

eine verdopplung von null-werten hätte wieder null ergeben. wie sieht es aber aus, wenn die sicht zum beispiel nur die zeilen der basistabelle abruft, in denen der preis kleiner als 15 ist? wirkt dann die aktualisierung der preisspalte auf *alle* zeilen der basistabelle oder nur auf die zeilen, die für die sicht ausgewählt wurden? um das zu klären, ändern wir die where-klausel in der definition der sicht wie folgt ab:

```
where (price is not null) and price < 15
```

wenn sie jetzt die ergebnisse der sicht mit der anweisung

```
select cast(title as varchar(30)) as titel, preis from
preiseuroview
```

abrufen, erhalten sie nur noch 8 zeilen in der ergebnismenge (unter der voraussetzung, daß sie diese abfrage auf die noch nicht geänderten bzw. wieder in den ursprungszustand versetzten daten der tabelle titles ausführen):

```
titel                          preis
------------------------------ ---------------------
cooking with computers: surrep 11.9500
you can combat computer stress 2.9900
the gourmet microwave          2.9900
is anger the enemy?            10.9500
life without fear              7.0000
emotional security: a new algo 7.9900
fifty years in buckingham pala 11.9500
sushi, anyone?                 14.9900
```

```
(8 row(s) affected)
```

jetzt führen sie wieder die oben beschriebene update-anweisung aus, um die preise mit 2 zu multiplizieren:

```
update preiseuroview
set preis = preis * 2
```

sql server zeigt zur bestätigung die meldung:

```
(8 row(s) affected)
```

in der sicht wurden offensichtlich die acht ausgewählten zeilen aktualisiert. zur kontrolle fragen sie die sicht ab:

```
titel                          preis
------------------------------ ---------------------
you can combat computer stress 5.9800
the gourmet microwave          5.9800
life without fear              14.0000

(3 row(s) affected)
```

in der ergebnismenge der sichtabfrage erscheinen nur noch drei zeilen, weil die zusätzliche bedingung price < 15 in der where-klausel wirkt und von den acht verdoppelten preisen nur noch drei in den bereich kleiner als 15 fallen. was aber ist in der zugrundeliegenden tabelle titles passiert? probieren sie es aus:

```
select cast(title as varchar(30)) as titel, price from titles
```

das ergebnis sieht folgendermaßen aus (die kennzeichnung »verdoppelt« für aktualisierte zeilen gehört nicht zur ergebnismenge):

```
titel                          price
------------------------------ ---------------------
```

```
the busy executive's database  19.9900
cooking with computers: surrep 23.9000     verdoppelt
you can combat computer stress 5.9800      verdoppelt
straight talk about computers  19.9900
silicon valley gastronomic tre 19.9900
the gourmet microwave          5.9800      verdoppelt
the psychology of computer coo null
but is it user friendly?       22.9500
secrets of silicon valley      20.0000
net etiquette                  null
computer phobic and non-phobic 21.5900
is anger the enemy?            21.9000     verdoppelt
life without fear              14.0000     verdoppelt
prolonged data deprivation: fo 19.9900
emotional security: a new algo 15.9800     verdoppelt
onions, leeks, and garlic: coo 20.9500
fifty years in buckingham pala 23.9000     verdoppelt
sushi, anyone?                 29.9800     verdoppelt

(18 row(s) affected)
```

fazit: wenn man die daten einer sicht aktualisiert, greifen die änderungen nur für die zeilen, die tatsächlich in der sicht enthalten sind, auf die zugrundeliegenden tabellen durch.

13.4.4 daten einfügen

eine sicht bietet auch die möglichkeit, daten einzufügen. die neuen zeilen werden auch in die basistabelle übernommen.

wie sie sicherlich schon erraten haben, fügt man daten in eine sicht mit einer insert-anweisung gemäß der folgenden syntax ein:

```
insert [into] sichtname
[(spaltenliste)] values (ausdrucksliste)
```

in der *spaltenliste* kann man - durch kommas getrennt - diejenigen spalten festlegen, für die werte einzufügen sind. in der values-liste müssen genauso viele werte erscheinen wie spalten in der spaltenliste genannt sind. außerdem muß die reihenfolge der werte in der values-liste mit der reihenfolge der spalten in der sicht übereinstimmen. wenn eine spalte nicht in der spaltenliste aufgeführt ist, muß sql server automatisch einen wert bereitstellen können. das ist dann möglich, wenn

- es sich um eine identity-spalte handelt. in diesem fall nimmt sql server den nächsten identitätswert.
- für die spalte ein standardwert oder ein timestamp-wert definiert ist.

- die spalte null-werte zuläßt. sql server fügt dann einen null-wert ein.

aufgrund der einschränkungen, die für das modifizieren von daten in einer sicht gelten (siehe den abschnitt weiter unten in diesem kapitel), läßt sich die bisher als beispiel verwendete sicht preiseuroview nicht in der momentanen form einsetzen, um daten einzufügen. ändern sie deshalb für das nachfolgende beispiel die definition der sicht wie folgt ab:

```
use pubs
go
alter view preiseuroview
as
select title_id, title, price as preis
from titles
where (price is not null)
```

aus der sicht wurden die berechneten spalten entfernt, und die spalte title_id ist neu hinzugekommen.

jetzt lassen sich daten mit der anweisung insert in die sicht einfügen:

```
use pubs
insert preiseuroview
(title_id, title, preis) values('fl1234','sql server 7.0
kompendium',99.95)
```

aus der sicht können sie mit der folgenden abfrage feststellen, ob der neue titel eingefügt wurde:

```
select title_id, cast(title as varchar(30)) as titel,
       preis from preiseuroview
```

interessant ist aber auch, ob die daten in der basistabelle der sicht erscheinen. das läßt sich mit der folgenden abfrage überprüfen:

```
select title_id, cast(title as varchar(30)) as titel,
       price from titles
```

in der ergebnismenge ist der über die sicht eingefügte titel gekennzeichnet:

```
title_id titel                          price
-------- ------------------------------ ---------------------
```

```
bu1032   the busy executive's database  19.9900
bu1111   cooking with computers: surrep 11.9500
bu2075   you can combat computer stress 2.9900
bu7832   straight talk about computers  19.9900
fl1234   sql server 7.0 kompendium      99.9500     eingefügt
mc2222   silicon valley gastronomic tre 19.9900
mc3021   the gourmet microwave          2.9900
mc3026   the psychology of computer coo null
pc1035   but is it user friendly?       22.9500
pc8888   secrets of silicon valley      20.0000
pc9999   net etiquette                  null
ps1372   computer phobic and non-phobic 21.5900
ps2091   is anger the enemy?            10.9500
ps2106   life without fear              7.0000
ps3333   prolonged data deprivation: fo 19.9900
ps7777   emotional security: a new algo 7.9900
tc3218   onions, leeks, and garlic: coo 20.9500
tc4203   fifty years in buckingham pala 11.9500
tc7777   sushi, anyone?                 14.9900

(19 row(s) affected)
```

13.4.5 daten löschen

aus einer sicht kann man daten löschen, wenn die sicht auf genau eine basistabelle verweist. die syntax zum löschen von daten sieht wie folgt aus:

```
delete [from] sichtname
where suchbedingung
```

den im letzten abschnitt mit insert eingefügten datensatz löschen sie mit der folgenden anweisung:

```
delete from preiseuroview
where title_id = 'fl1234'
```

fragen sie die sicht oder die basistabelle ab, um das löschen des datensatzes nachzuprüfen.

13.4.6 einschränkungen

in einer sicht lassen sich daten modifizieren, und diese änderungen spiegeln sich auch in den basistabellen der sicht wider. allerdings sind bestimmte regeln und einschränkungen zu beachten, die sich aus dem speziellen charakter einer sicht ergeben.

um daten in einer sicht - und damit in den basistabellen - modifizieren zu können, muß die sicht *aktualisierbar* sein. dazu muß die sichtdefinition folgende voraussetzungen erfüllen:

- in der from-klausel der sichtdefinition ist mindestens eine tabelle angegeben.
- die auswahlliste enthält weder aggregatfunktionen noch die klauseln group by, union, distinct und top. eine unterabfrage in der from-klausel darf allerdings aggregatfunktionen enthalten, wenn die damit gebildeten werte nicht geändert werden.
- die auswahlliste der sicht darf keine abgeleiteten spalten enthalten.

darüber hinaus gelten folgende einschränkungen für die modifizierung von daten einer sicht:

- wenn die sichtdefinition die klausel with check option enthält, sind nur solche änderungen zulässig, nach denen die daten entsprechend den bedingungen in der where-klausel weiterhin in der sicht erscheinen. sql server bricht die verarbeitung einer zeile ab, wenn die daten nach der änderung nicht mehr in der sicht enthalten sein würden.
- die einer sicht zugrundeliegenden daten können aus mehreren tabellen stammen. eine änderung von daten in der sicht ist jedoch nur für jeweils eine basistabelle möglich.
- die delete-anweisung läßt sich nur bei sichten verwenden, die auf genau eine basistabelle verweisen.
- wenn spalten in der basistabelle keine null-werte enthalten dürfen und diese spalten nicht bestandteil der sicht sind, muß gewährleistet sein, daß beim einfügen von daten für die null-spalten standardwerte erzeugt werden.

die sicht preiseuroview (in der ursprünglichen fassung, die sie mit dem enterprise manager erstellt haben) enthält berechnete spalten und ist damit nicht aktualisierbar. wenn sie zum beispiel mit der anweisung

```
insert preiseuroview
(title, preis) values('kompendium', 99.95)
```

daten in die sicht einfügen wollen, erhalten sie die fehlermeldung

```
server: nachr.-nr. 4406, schweregrad 16, status 1, zeile 1
sicht 'preiseuroview' ist nicht aktualisierbar, da eine spalte
der sicht abgeleitet oder konstant ist.
```

es ist ebenfalls nicht möglich, eine berechnete (abgeleitete) oder konstante spalte einer sicht zu aktualisieren. so führt die anweisung

```
update preiseuroview
set euro=preis/1.5
```

zur gleichen fehlermeldung wie im letzten beispiel.

wenn sie die berechneten spalten aus der sichtdefinition entfernen und daten einfügen wollen, erhalten sie die fehlermeldung

```
server: nachr.-nr. 515, schweregrad 16, status 2, zeile 1
der wert null kann in spalte 'title_id', tabelle
'pubs.dbo.titles' nicht eingefügt werden. die spalte lässt null-
werte nicht zu. insert schlägt fehl.
die anweisung wurde beendet.
```

erst wenn sie die null-spalte title_id in die sicht aufnehmen und für diese spalte einen wert angeben, lassen sich daten in der sicht modifizieren.

die in den obigen beispielen mehrfach veränderte sichtdefinition verfehlt natürlich das ziel, das wir mit der sicht ursprünglich verfolgt haben. es kam hier aber darauf an, die auswirkungen der verschiedenen einschränkungen beim modifizieren von daten zu zeigen.

13.4.7 daten exportieren

kapitel 9 hat gezeigt, wie man daten mit bulk insert in eine tabelle einfügt. das massenkopieren funktioniert aber nicht nur für tabellen, sondern auch für sichten. das beispiel in diesem abschnitt soll die ziehungen ab 2200 aus der datenbank lotto exportieren. das ließe sich zwar direkt aus der tabelle realisieren, dieses kapitel soll aber zeigen, daß es auch mit sichten funktioniert. die transact-sql-anweisung bulk insert eignet sich nur zum importieren. deshalb greifen wir hier auf das dienstprogramm bcp zurück, um die daten aus der sicht in die ausgabedatei last100.txt zu schreiben.

erstellen sie zunächst folgende sicht:

```
use lotto
go
create view last100
as
select ziehung, convert(varchar(15),datum,104)as datum,
       z1,z2,z3,z4,z5,z6,zz from ldaten
where ziehung >=2200
```

fragen sie die sicht mit

```
select * from last100
```

ab, um sich von der korrekten arbeitsweise zu überzeugen.

wechseln sie als nächstes zur ms-dos-eingabeaufforderung, und geben sie die folgende anweisung ein:

```
bcp lotto..last100 out last100.txt -c -t"," -usa -seins -p
```

diese anweisung können sie auf jedem client-pc ausführen. wenn sie den befehl bcp aus dem verzeichnis aufrufen, in das sie die textdatei schreiben wollen, sparen sie sich die angabe des pfades für die ausgabedatei. die anmeldung erfolgt auf dem server eins mit dem benutzernamen sa ohne kennwort. die datensätze werden im textformat (option -c) in die datei last100.txt exportiert (parameter out). als feldtrennzeichen ist das komma festgelegt. weitere hinweise zum dienstprogramm bcp finden sie in kapitel 9.

13.5 verschachtelte sichten

sichten müssen nicht nur auf tabellen basieren, sondern können auch daten aus anderen sichten beziehen. erstellen sie als beispiel die folgende sicht:

```
create view preiseuro
as
select title, preis, euro from preiseuroview
```

diese sicht ruft die spalten title, preis und euro aus der weiter oben erstellten sicht preiseuroview ab. beachten sie, daß die spalte euro in der sicht preiseuroview eine berechnete spalte darstellt. wenn sie die sicht mit der anweisung

```
select preis, euro from preiseuro
```

abrufen, erhalten sie die ergebnismenge

```
preis                euro
-------------------- --------------------
19.9900              10.2207
11.9500              6.1099
```

```
2.9900                  1.5288
19.9900                 10.2207
19.9900                 10.2207
2.9900                  1.5288
22.9500                 11.7341
20.0000                 10.2258
21.5900                 11.0388
10.9500                 5.5986
7.0000                  3.5790
19.9900                 10.2207
7.9900                  4.0852
20.9500                 10.7116
11.9500                 6.1099
14.9900                 7.6643
```

13.6 abhängigkeit von sichten

die in der aktuellen datenbank definierten sichten können sie mit der gespeicherten prozedur sp_help abrufen. die prozedur gibt namen, besitzer und objekttyp der sichten aus.

eine abhängigkeit der sichten läßt sich mit der gespeicherten prozedur sp_depends gemäß folgender syntax ermitteln:

```
sp_depends [@objname =] 'objekt'
```

zum beispiel zeigen sie mit

```
exec sp_depends preiseuroview
```

die abhängigkeiten der sicht preiseuroview an. wenn sie die beiden sichten preiseuroview und die darauf basierende sicht preiseuro wie in diesem kapitel beschrieben erstellt haben, liefert die obige anweisung folgende ausgabe:

```
in der aktuellen datenbank verweist das angegebene objekt auf:
name                    type              updated selected column
----------------------- ----------------- ------- -------- ------
dbo.titles              user table        no      no       title
dbo.titles              user table        no      no       price
```

```
in der aktuellen datenbank bestehen folgende verweise auf das
angegebene objekt:
name                    type
--------------------------------------- ---------------
dbo.preiseuro           view
```

die prozedur zeigt zwei ergebnismengen an: eine für die objekte, auf denen die sicht basiert, die zweite für objekte, die von der sicht abhängen.

13.7 sichten löschen

in diesem kapitel haben sie in der datenbank pubs zwei sichten erstellt: preiseuroview und preiseuro. dieser abschnitt zeigt, wie sie die sichten mit dem enterprise manager und mit transact-sql wieder löschen können.

13.7.1 enterprise manager

mit dem enterprise manager löschen sie eine sicht in folgenden schritten:

1. erweitern sie die konsolenstruktur bis zur datenbank, die die sicht enthält. erweitern sie den ordner der datenbank, und klicken sie auf **sichten**, so daß im rechten fensterbereich eine liste der sichten für die datenbank erscheint.
2. klicken sie mit der rechten maustaste auf die zu löschende sicht, und wählen sie **löschen** aus dem kontextmenü, oder drücken sie **ç**. es erscheint das dialogfeld **objekte löschen** (siehe abbildung 13.14).

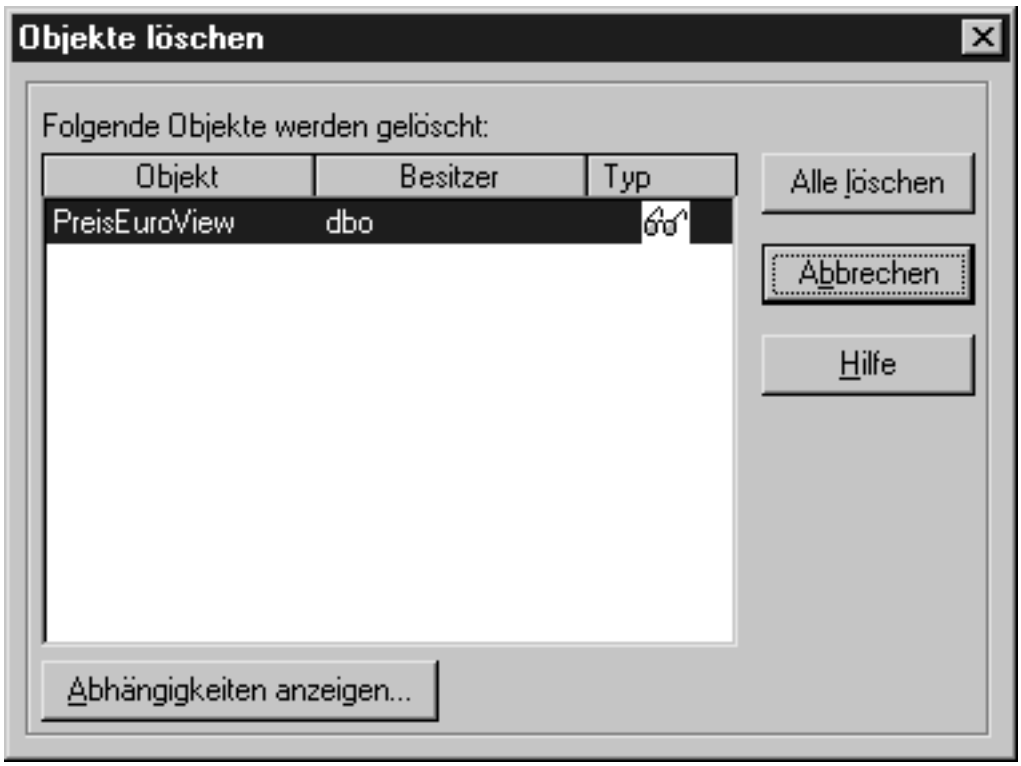


abbildung 13.14: das dialogfeld objekte löschen wird zur bestätigung eingeblendet, wenn sie eine sicht löschen wollen

3. eine sicht kann sowohl von tabellen bzw. anderen sichten abhängig sein als auch selbst als basis für andere sichten dienen. deshalb sollten sie sich zunächst über die abhängigkeiten informieren, bevor sie die sicht löschen. klicken sie dazu im dialogfeld **objekte löschen** auf die schaltfläche **abhängigkeiten anzeigen**.
4. im dialogfeld **abhängigkeiten für** *sichtname* (siehe abbildung 13.15) sind auf der registerkarte **allgemein** die abhängigkeiten aufgeführt. klicken sie auf **schliessen**, wenn sie sich über die abhängigkeiten informiert haben.

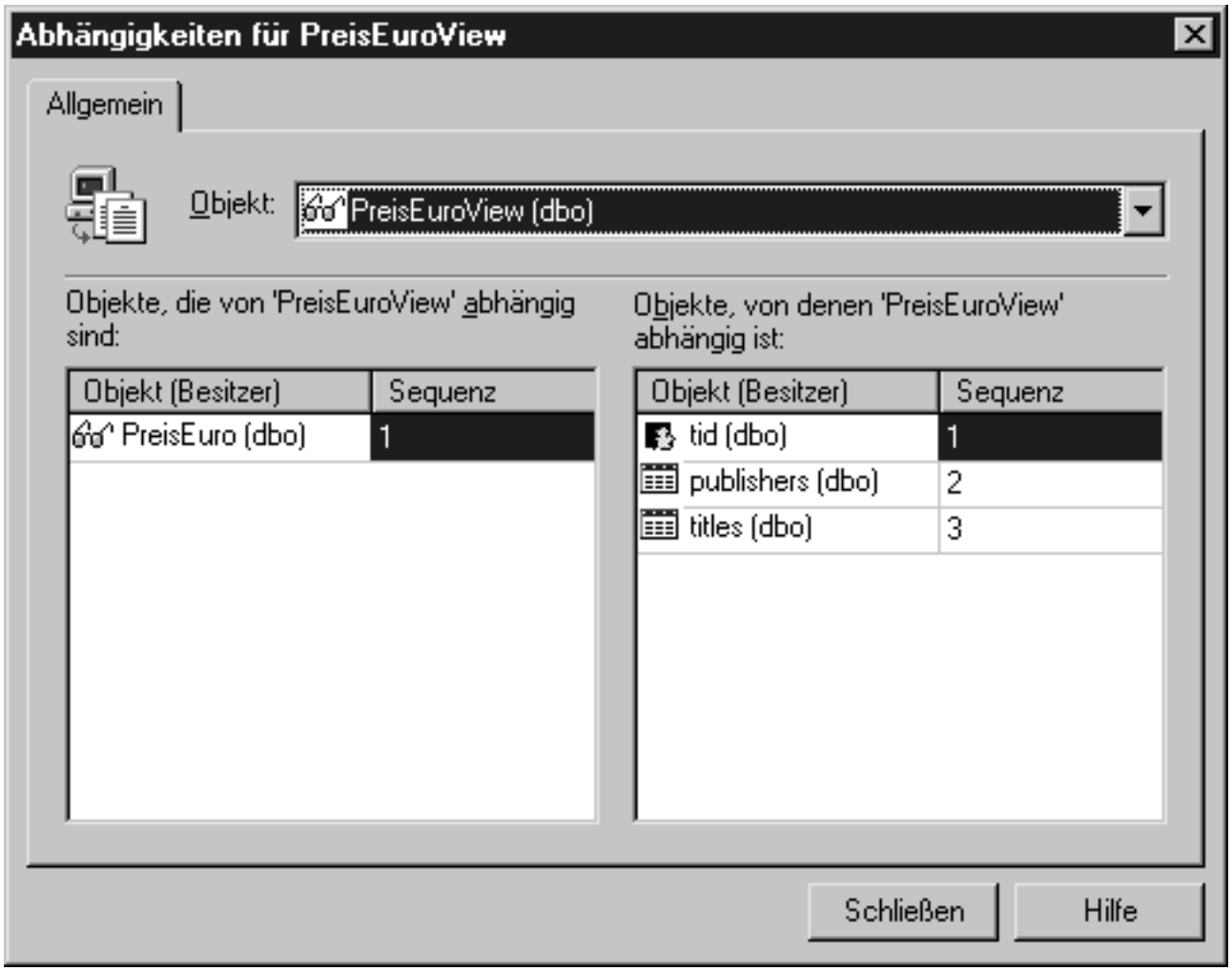


abbildung 13.15: das dialogfeld zeigt alle abhängigkeiten der sicht

5. um die sicht zu löschen, klicken sie im dialogfeld **objekte löschen** auf die schaltfläche **alle löschen**. eine weitere bestätigung ist nicht erforderlich.

13.7.2 sichten per transact-sql löschen

mit der anweisung drop view löschen sie eine oder mehrere sichten aus der aktuellen datenbank. die syntax hat folgendes aussehen:

```
drop view sichtliste
```

die anweisung löscht die in der *sichtliste* - durch komma getrennt - aufgeführten sichten. um zum beispiel die sicht preiseuroview zu löschen, führen sie folgende anweisung aus:

```
drop view preiseuroview
```

sql server löscht die sicht ohne weitere rückfrage. verwenden sie die anweisung drop view also mit vorsicht, und informieren sie sich vorher mit der gespeicherten prozedur sp_depends über die abhängigkeiten der sicht.

13.7.3 verweis auf gelöschte sichten

wenn sie eine sicht, die auf eine andere sicht verweist, löschen, beschwert sich sql server erst dann, wenn sie die gelöschte sicht referenzieren. haben sie im letzten beispiel die sicht preiseuroview gelöscht und führen die folgende abfrage aus

```
select * from preiseuro
```

erhalten sie folgende fehlermeldung:

```
server: nachr.-nr. 208, schweregrad 16, status 1,  prozedur
preiseuro, zeile 3
ungültiger objektname 'preiseuroview'.
server: nachr.-nr. 4413, schweregrad 16, status 1, zeile 1
 sicht 'preiseuro' konnte wegen vorheriger bindungsfehler nicht
verwendet werden.
```

SQL Server Query Analyzer - [Abfrage - eins.pubs.EINS\FRANK LANGENAU - (unbenannt) - use pubs selec...*]

Datei Bearbeiten Ansicht Abfrage Fenster ?

DB: pubs

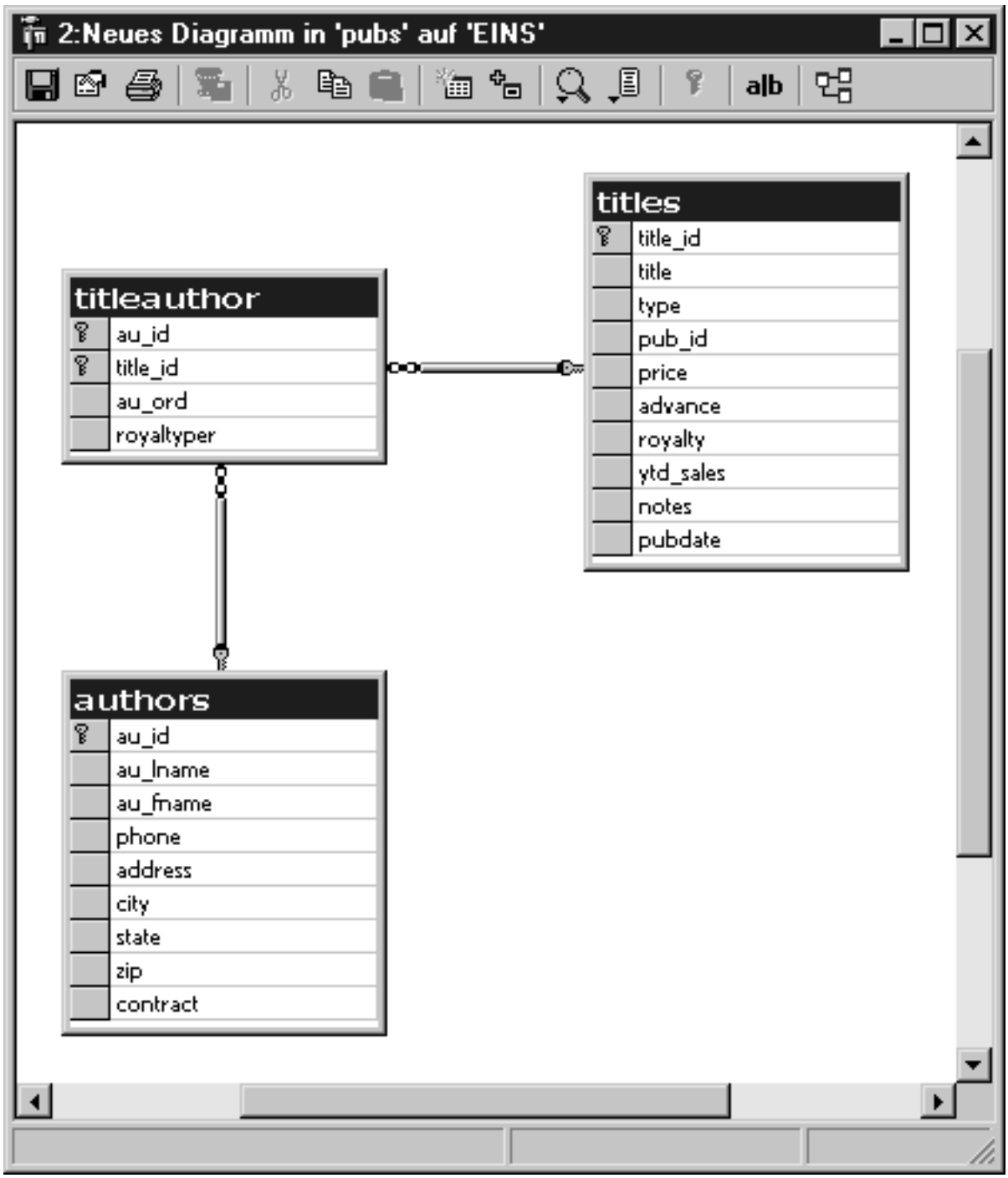
```
use pubs
select * from titleview
```

title	au_ord	au_lname	price	ytd_sales	pub_id
The Busy Executive's...	1	Bennet	19.9900	4095	1389
Fifty Years in Bucki...	1	Blotchet-Halls	11.9500	15096	0877
But Is It User Friendly?	1	Carson	22.9500	8780	1389
The Gourmet Microwave	1	DeFrance	2.9900	22246	0877
Silicon Valley Gastr...	1	del Castillo	19.9900	2032	0877
Secrets of Silicon V...	1	Dull	20.0000	4095	1389
The Busy Executive's...	2	Green	19.9900	4095	1389
You Can Combat Compu...	1	Green	2.9900	18722	0736
Sushi, Anyone?	3	Gringlesby	14.9900	4095	0877
Secrets of Silicon V...	2	Hunter	20.0000	4095	1389
Computer Phobic AND ...	1	Karsen	21.5900	375	0877
Net Etiquette	1	Locksley	NULL	NULL	1389
Emotional Security: ...	1	Locksley	7.9900	3336	0736
Cooking with Compute...	1	MacFeather	11.9500	3876	1389
Computer Phobic AND ...	2	MacFeather	21.5900	375	0877
Cooking with Compute...	2	O'Leary	11.9500	3876	1389

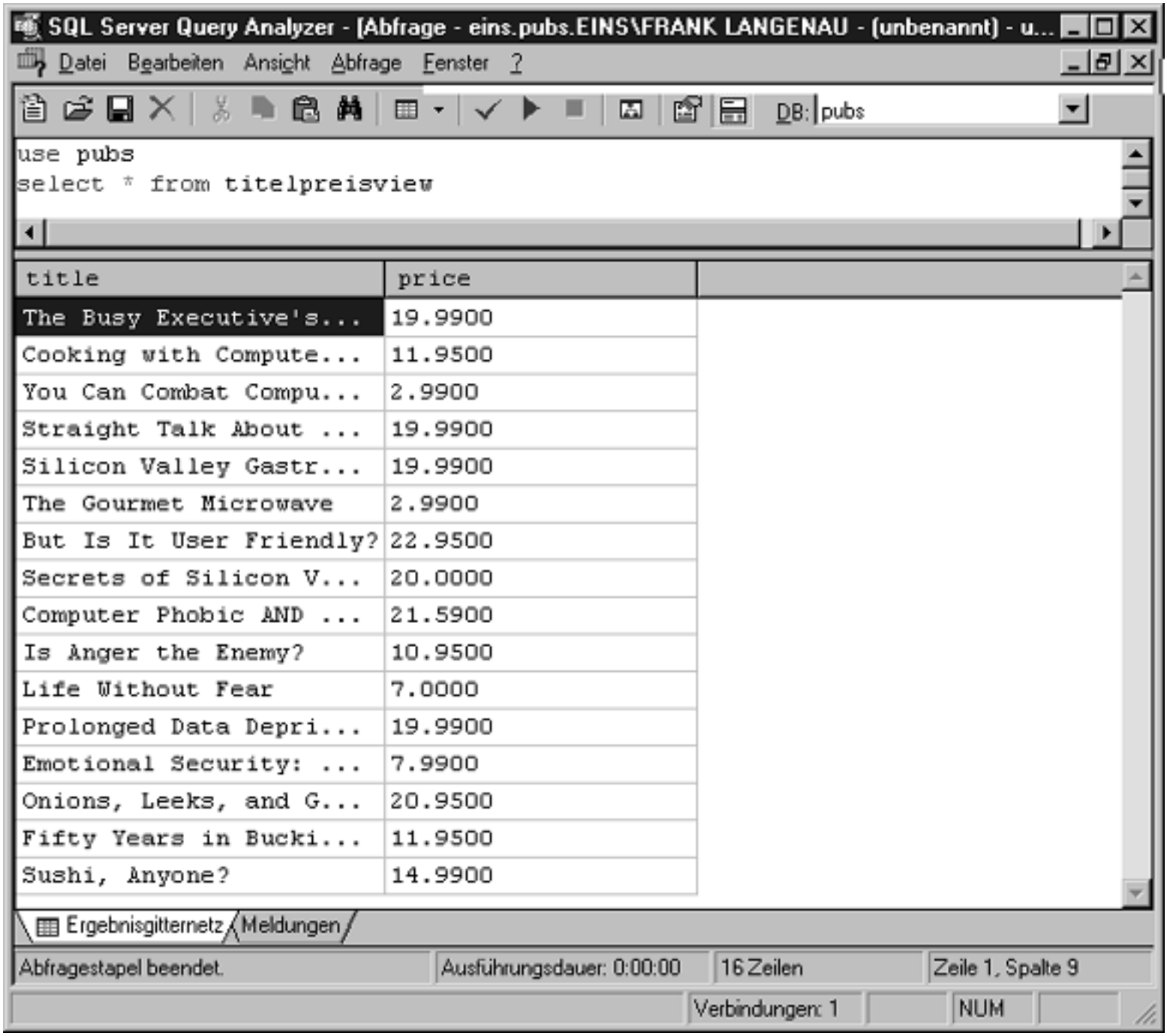
Ergebnisgitternetz | Meldungen

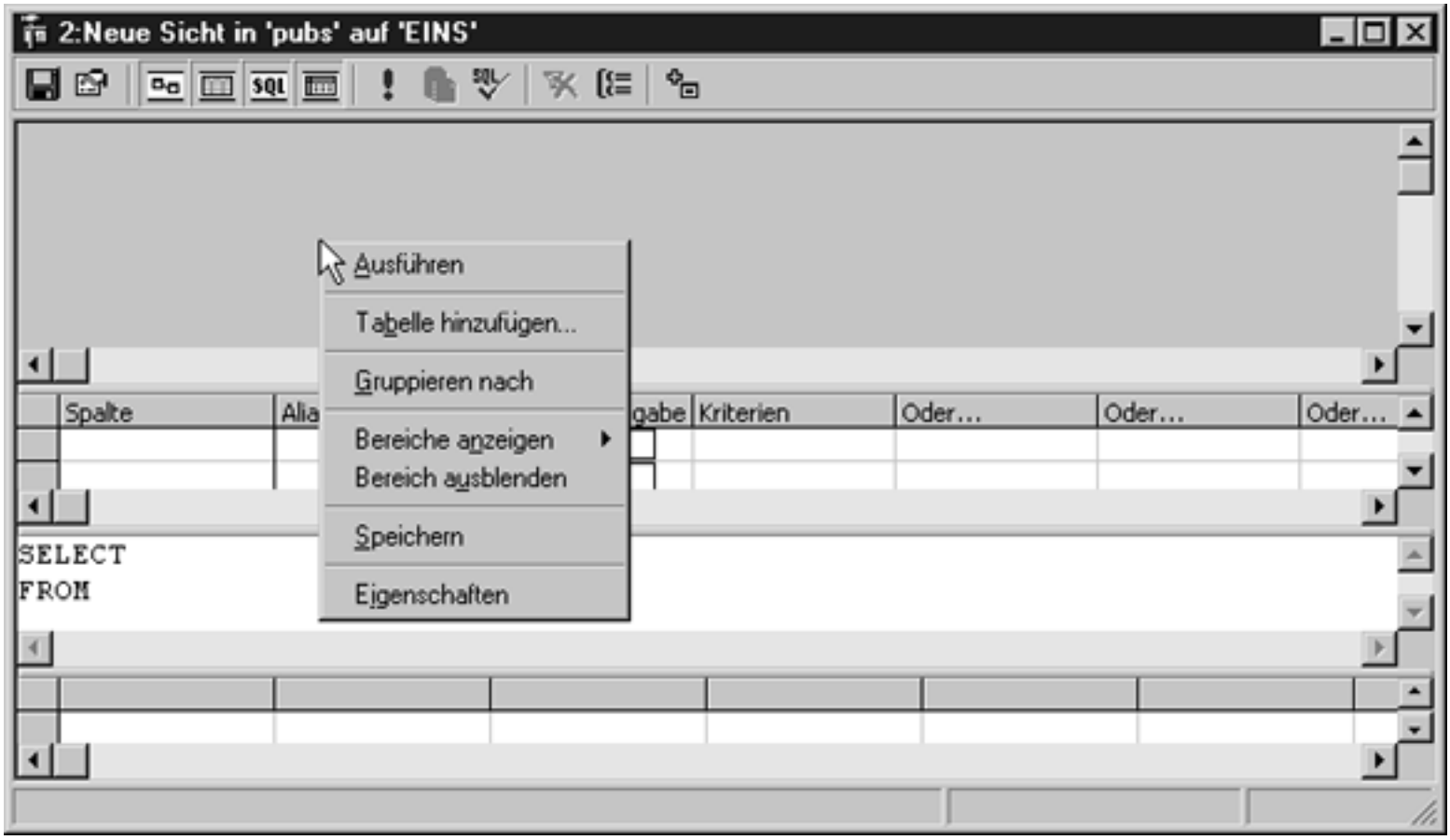
Abfragestapel beendet. | Ausführungsdauer: 0:00:00 | 25 Zeilen | Zeile 2, Spalte 24

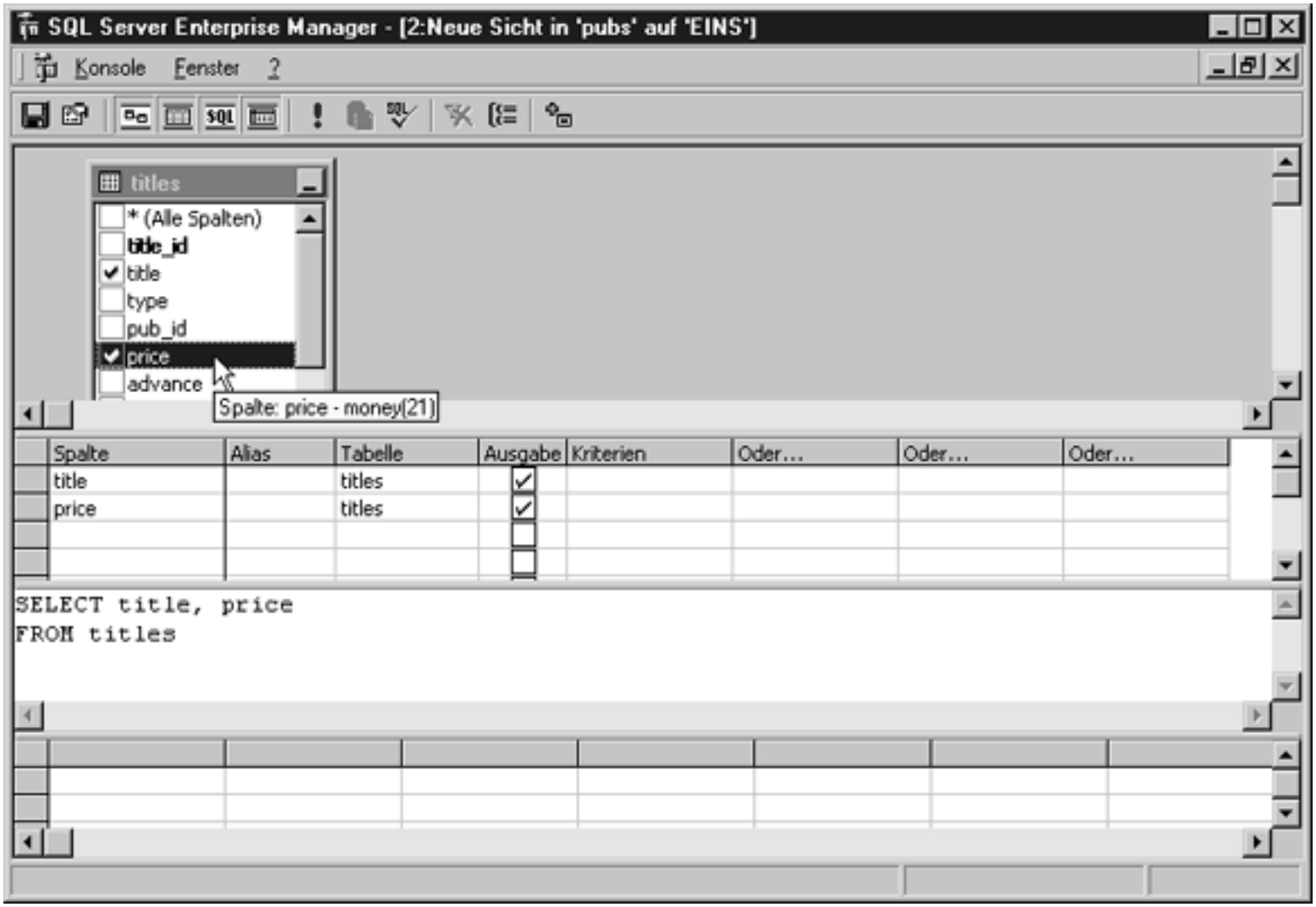
Verbindungen: 1 | NUM

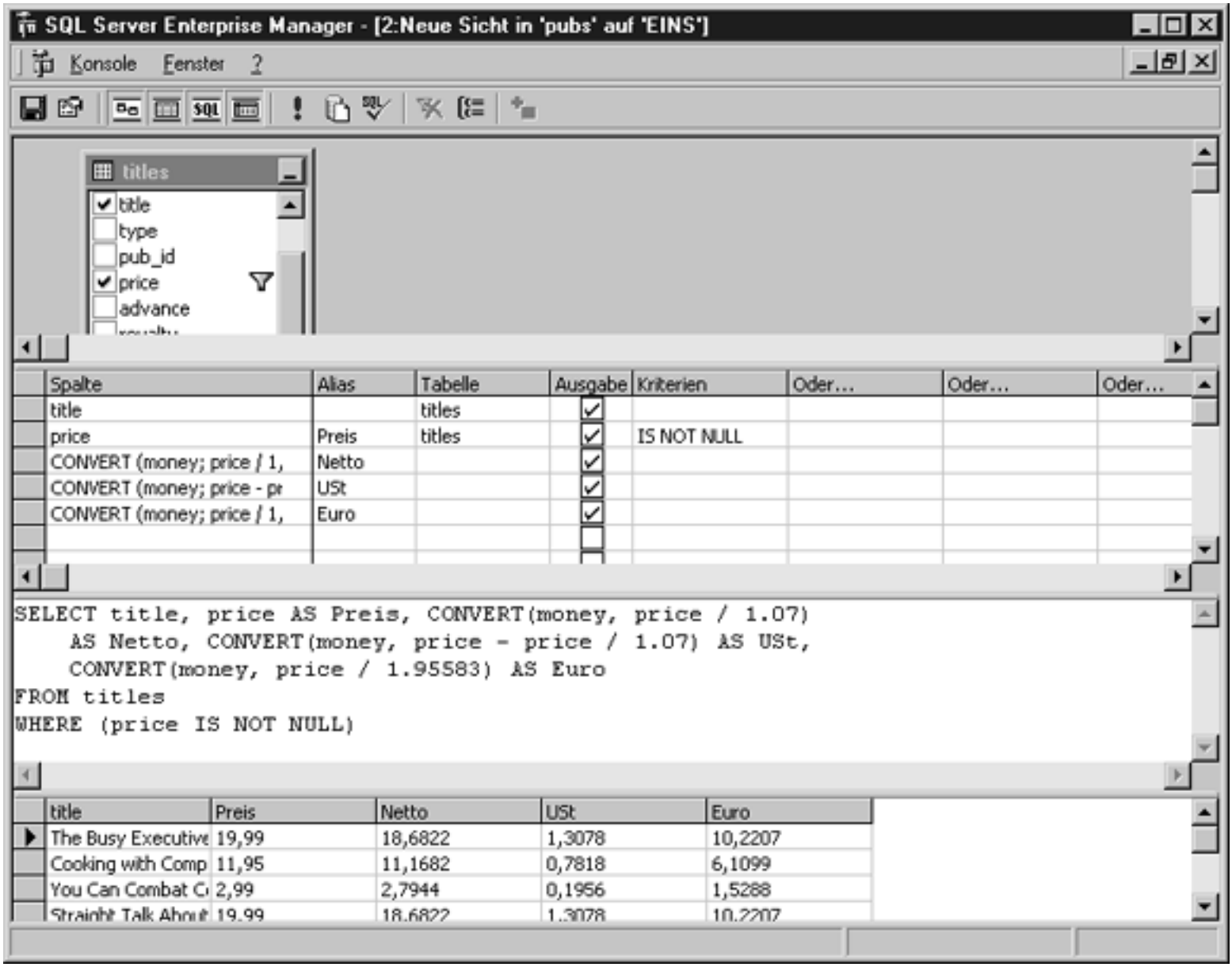

2:Neues Diagramm in 'pubs' auf 'EINS'
titles
title_id
title
type
pub_id
price
advance
royalty
ytd_sales
notes
pubdate
titleauthor
au_id
title_id
au_ord
royaltyper
authors
au_id
au_lname
au_fname
phone
address
city
state
zip
contract


Sichterstellungs-Assistent - EINS
Sichterstellungs-Assistenten beenden
Sie haben die Schritte beendet, die zum Erstellen der nachfolgend angezeigten Sicht erforderlich sind. Sie können, falls gewünscht, die Anweisungen in diesem Fenster bearbeiten, um sie anzupassen.
USE [pubs]
GO
CREATE VIEW [TitelPreisView]
AS SELECT [titles].[title], [titles].[price]
FROM [titles]
where price is not null
< Zurück
Fertig stellen
Abbrechen


SQL Server Query Analyzer - [Abfrage - eins.pubs.EINS\FRANK LANGENAU - (unbenannt) - u...
Datei Bearbeiten Ansicht Abfrage Fenster ?
DB: pubs
use pubs
select * from titelpreisview
title price
The Busy Executive's... 19.9900
Cooking with Compute... 11.9500
You Can Combat Compu... 2.9900
Straight Talk About ... 19.9900
Silicon Valley Gastr... 19.9900
The Gourmet Microwave 2.9900
But Is It User Friendly? 22.9500
Secrets of Silicon V... 20.0000
Computer Phobic AND ... 21.5900
Is Anger the Enemy? 10.9500
Life Without Fear 7.0000
Prolonged Data Depri... 19.9900
Emotional Security: ... 7.9900
Onions, Leeks, and G... 20.9500
Fifty Years in Bucki... 11.9500
Sushi, Anyone? 14.9900
Ergebnisgitternetz
Meldungen
Abfragestapel beendet.
Ausführungsdauer: 0:00:00
16 Zeilen
Zeile 1, Spalte 9
Verbindungen: 1
NUM


2:Neue Sicht in 'pubs' auf 'EINS'
Ausführen
Tabelle hinzufügen...
Gruppieren nach
Bereiche anzeigen
Bereich ausblenden
Speichern
Eigenschaften
Spalte
Alia
gabe
Kriterien
Oder...
Oder...
Oder...
SELECT
FROM


SQL Server Enterprise Manager - [2:Neue Sicht in 'pubs' auf 'EINS']
Konsole
Fenster
?
titles
* (Alle Spalten)
title_id
title
type
pub_id
price
advance
Spalte: price - money(21)
Spalte
Alias
Tabelle
Ausgabe
Kriterien
Oder...
Oder...
Oder...
title
titles
price
titles
SELECT title, price
FROM titles


SQL Server Enterprise Manager - [2:Neue Sicht in 'pubs' auf 'EINS']
Konsole
Fenster
?
titles
title
type
pub_id
price
advance
Spalte
Alias
Tabelle
Ausgabe
Kriterien
Oder...
Oder...
Oder...
title
titles
price
Preis
titles
IS NOT NULL
CONVERT (money; price / 1,
Netto
CONVERT (money; price - pr
USt
CONVERT (money; price / 1,
Euro
SELECT title, price AS Preis, CONVERT(money, price / 1.07)
AS Netto, CONVERT(money, price - price / 1.07) AS USt,
CONVERT(money, price / 1.95583) AS Euro
FROM titles
WHERE (price IS NOT NULL)
title
Preis
Netto
USt
Euro
The Busy Executive
19,99
18,6822
1,3078
10,2207
Cooking with Comp
11,95
11,1682
0,7818
6,1099
You Can Combat C
2,99
2,7944
0,1956
1,5288
Straight Talk About
19,99
18,6822
1,3078
10,2207

kapitel 14 indizes

14.1 die stecknadel im heuhaufen ...

... müssen sie nicht gerade suchen, um bestimmte daten in einer datenbank zu finden. allerdings wäre es recht mühsam und langwierig, eine große tabelle zeile für zeile nach einem eintrag zu durchsuchen.

wie gehen sie zum beispiel vor, wenn sie aus dem katalog eines elektronikversands einen artikel bestellen möchten? vielleicht blättern sie zum zeitvertreib den katalog durch und notieren sich dabei die bestellnummern auf ihrem wunschzettel. es kann aber auch sein, daß sie schon etwas konkretes ins auge gefaßt haben und nun schnell die bestellnummer brauchen. das durchblättern von anfang an dauert ihnen dann natürlich zu lange. also sehen sie im alphabetisch geordneten inhaltsverzeichnis der artikel nach, um die betreffende seite zu ermitteln und diese dann direkt aufzuschlagen. der elektronikanbieter hat seinen katalog mit einem sogenannten *index* ausgestattet, der verweise (d.h. die seitennummern) auf die artikelbeschreibungen enthält.

das schnelle und gezielte aufsuchen von inhalten ist auch das ziel, das indizes in einer datenbank verfolgen. wenn der benutzer in einer umfangreichen datenbank nach einem bestimmten eintrag sucht und das datenbank-managementsystem die datensätze zeilenweise lesen und mit den gestellten bedingungen vergleichen muß, kann es ziemlich lange dauern, bis der benutzer die gewünschten informationen erhält. ordnet man aber den datensätzen der tabelle einen index zu, läßt sich der eintrag über den verweis im index in kürzester zeit lokalisieren und abrufen.

indizes kann man auch für mehrere spalten oder kombinationen von spalten erstellen. allerdings sollte man immer das rechte maß im auge behalten. zu viele indizes bremsen das system. einerseits läßt sich das abrufen von daten beschleunigen, andererseits verlangsamen sich operationen wie einfügen, aktualisieren und löschen. anhand der nutzung einer datenbank muß man abwägen, ob indizes vorteilhaft sind. wenn es sich überwiegend um abfragen handelt, bringen indizes geschwindigkeitsvorteile. dagegen können indizes hemmend wirken, wenn vorwiegend datenaktualisierungen auszuführen sind.

14.2 struktur von sql-server-indizes

sql server speichert tabellen in form von seiten, die jeweils 8 kbyte groß sind. die organisation der tabellen erfolgt entweder als gruppierte tabellen - das sind tabellen, die mit einem gruppierten index verknüpft sind - oder als heap. was darunter zu verstehen ist, erfahren sie in den folgenden abschnitten.

indizes verwaltet sql server als b-bäume, eine erweiterung der binären bäume zu vielweg-bäumen. dabei unterteilt man den baum in mehrere teilbäume (mit jeweils mehreren *knoten*) und faßt diese als gleichzeitig zugreifbare einheit auf. ein derartiger teilbaum heißt *seite* (in sql server eine *indexseite*). ein b-baum beginnt auf der obersten ebene - der *wurzel* - und kann sich über mehrere zwischenebenen erstrecken, bis die unterste ebene mit den sogenannten *blattknoten* erreicht ist. blattknoten besitzen keine nachfolger mehr und zeigen entweder auf einen datensatz in einer tabelle oder stellen selbst die daten

dar.

weiterführende informationen zu b-bäumen (und bäumen im allgemeinen) finden sie in »*algorithmen und datenstrukturen«* von niklaus wirth, erschienen im b.g. teubner verlag.

je nachdem, wie der index physikalisch gespeichert wird, unterscheidet sql server zwischen gruppierten und nicht gruppierten indizes. die art der speicherung wirkt sich auf die leistungsbilanz aus. je mehr knoten (sprich schlüssel) auf einer indexseite enthalten sind, desto weniger indexebenen sind erforderlich, und sql server muß weniger indexseiten lesen, um die daten für einen bestimmten schlüssel bereitzustellen.

14.2.1 gruppierte indizes

bei einem *gruppierten index* enthält die unterste ebene eines b-baums, d.h. die blattknoten, die eigentlichen datenseiten. die daten sind physikalisch in der reihenfolge des index gespeichert. aus diesem grund ist auch nur ein gruppierter index pro tabelle möglich. die datenseiten sind in einer doppelt verketteten liste miteinander verknüpft.

14.2.2 nicht gruppierte indizes

tabellen, die nicht über einen gruppierten index verfügen, bezeichnet sql server als *heaps* (zu deutsch etwa halde). datenseiten und indexseiten sind vollständig voneinander getrennt. die reihenfolge der datenzeilen innerhalb der datenseiten und die reihenfolge der datenseiten selbst ist willkürlich. weiterhin sind die datenseiten nicht durch eine verkettete liste verbunden.

nicht gruppierte indizes sind ebenfalls als b-bäume realisiert, wobei aber die blattknoten der untersten ebene nicht die eigentlichen daten, sondern nur einen zeiger auf die datenzeilen enthalten. daraus ergibt sich auch, daß nicht gruppierte indizes gegenüber vergleichbaren gruppierten indizes eine indexebene mehr aufweisen.

die anzahl der nicht gruppierten indizes beschränkt sql server auf 249. ein indexeintrag (schlüssel) kann aus maximal 16 spalten bestehen und darf nicht mehr als 900 byte umfassen. damit möglichst viele indexeinträge auf eine indexseite bzw. datenseite passen, sollten sie die breite eines indexeintrags möglichst gering halten. auf diese weise verringern sie die erforderlichen zugriffe auf den datenträger, wenn sie daten suchen oder neue daten einfügen. auf die leistungsbilanz hat auch die anzahl der indizes einen einfluß, da jeweils alle indizes bei update-, delete- und insert-operationen anzupassen sind. wenn sie für viele spalten oder spaltenkombinationen indizes anlegen, beschleunigen sie zwar die abfrageleistung, bremsen aber gleichzeitig die leistung bei datenänderungen. hier ist ein ausgewogenes mittelmaß gefragt. empfohlen werden ein gruppierter index und etwa zwei bis sechs nicht gruppierte indizes für eine tabelle.

14.2.3 eindeutige und nicht eindeutige indizes

bei einem *eindeutigen index* sind doppelte index- oder schlüsselwerte nicht zulässig. jeder indexwert kommt nur ein einziges mal in der tabelle vor. in der voreinstellung läßt sql server mehrdeutige indizes zu. da bei mehrdeutigen indizes zusätzliche suchschritte erforderlich sind, nachdem die erste übereinstimmung gefunden wurde, sollte man nach möglichkeit eindeutige indizes verwenden.

bei gruppierten indizes bildet sql server intern einen zusätzlichen schlüsselwert, damit der index eindeutig wird. wenn aber ihre tabellenstruktur ohnehin so ausgelegt ist, daß sie einen eindeutigen index erstellen könnten, sollten sie diese aufgabe nicht an sql server delegieren.

14.2.4 ein- und mehrspaltige indizes

wie bereits erwähnt, kann ein index aus bis zu 16 spalten einer tabelle bestehen. oftmals genügt aber bereits eine einzige spalte, um einen eindeutigen index zu erstellen. das ist beispielsweise bei artikelnummern der fall. dagegen ist es bei einer tabelle mit den namen der mitarbeiter sicherlich notwendig, mehrere spalten - etwa vorname und nachname, eventuell noch geburtsdatum - zu verknüpfen, damit ein eindeutiger index entsteht.

14.3 indizes erstellen

die folgenden abschnitte zeigen, wie sie indizes erstellen. sql server bietet auch hier wieder einen assistenten, der sie durch die einzelnen schritte führt. aber auch die im anschluß behandelte transact-sql-anweisung create index dürfte ihnen keine schwierigkeiten bereiten.

14.3.1 indexerstellungs-assistent

im enterprise manager erstellen sie einen index komfortabel mit dem indexerstellungs-assistenten in folgenden schritten:

1. wählen sie einen server aus, und klicken sie in der symbolleiste des enterprise managers auf die schaltfläche **assistenten auswählen**. (sie können auch die konsolenstruktur bis zur gewünschten tabelle erweitern und den indexerstellungs-assistenten von dieser ebene aus starten.)
2. im dialogfeld **assistent auswählen** erweitern sie den zweig **datenbank**, markieren den eintrag *indexerstellungs-assistent* und klicken auf **ok**. es erscheint das startdialogfeld des assistenten (siehe abbildung 14.1).

abbildung 14.1: startdialogfeld des indexerstellungs-assistenten

3. klicken sie auf **weiter**. im zweiten dialogfeld wählen sie die datenbank und die tabelle aus, für die sie den index erstellen wollen (siehe abbildung 14.2).

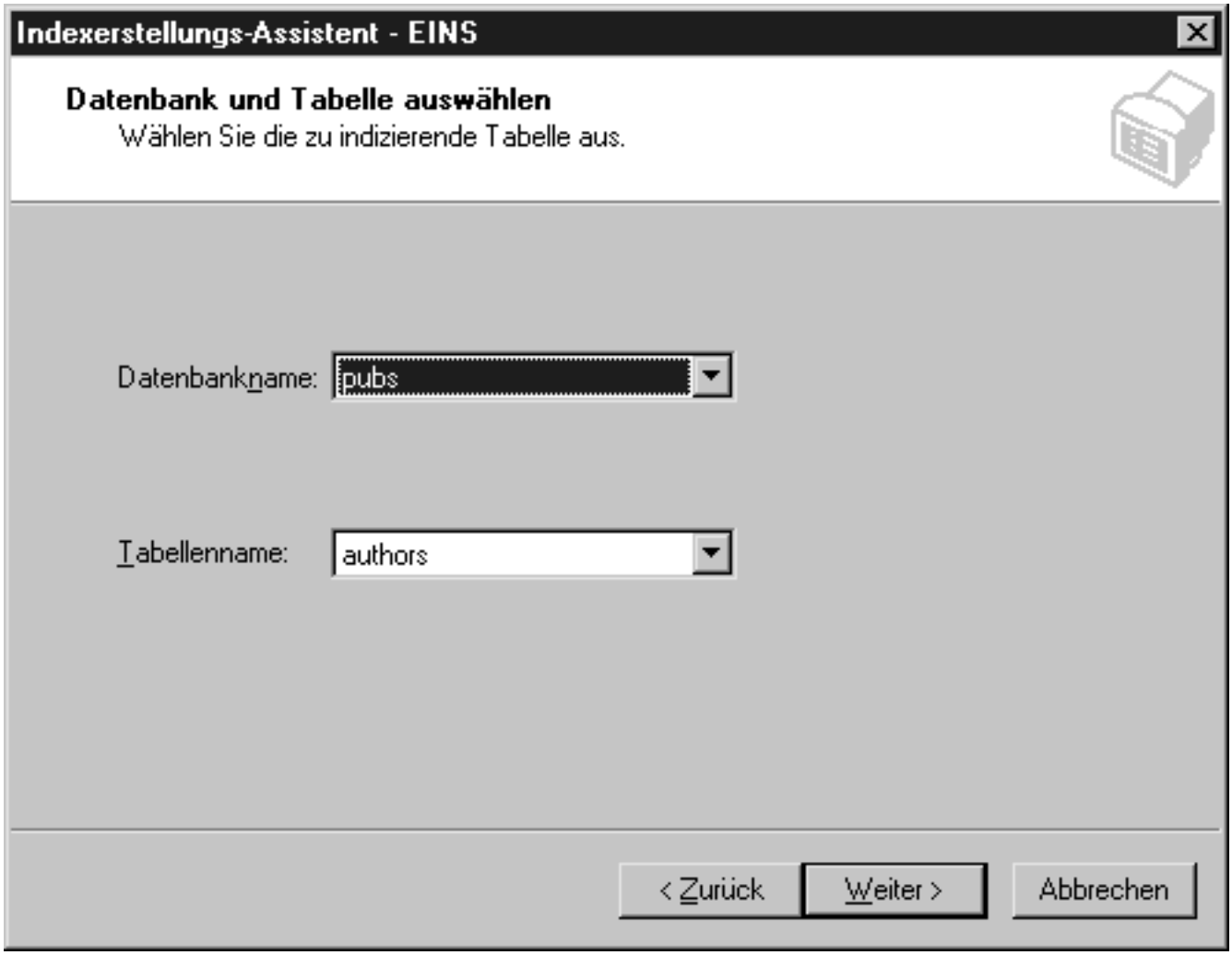


abbildung 14.2: auswahl von datenbank und tabelle

4. falls für die tabelle bereits indizes vorhanden sind (wie es für die tabelle authors der datenbank pubs der fall ist), erscheint ein zusätzliches dialogfeld mit informationen zu den bereits vorhandenen indizes (siehe abbildung 14.3).

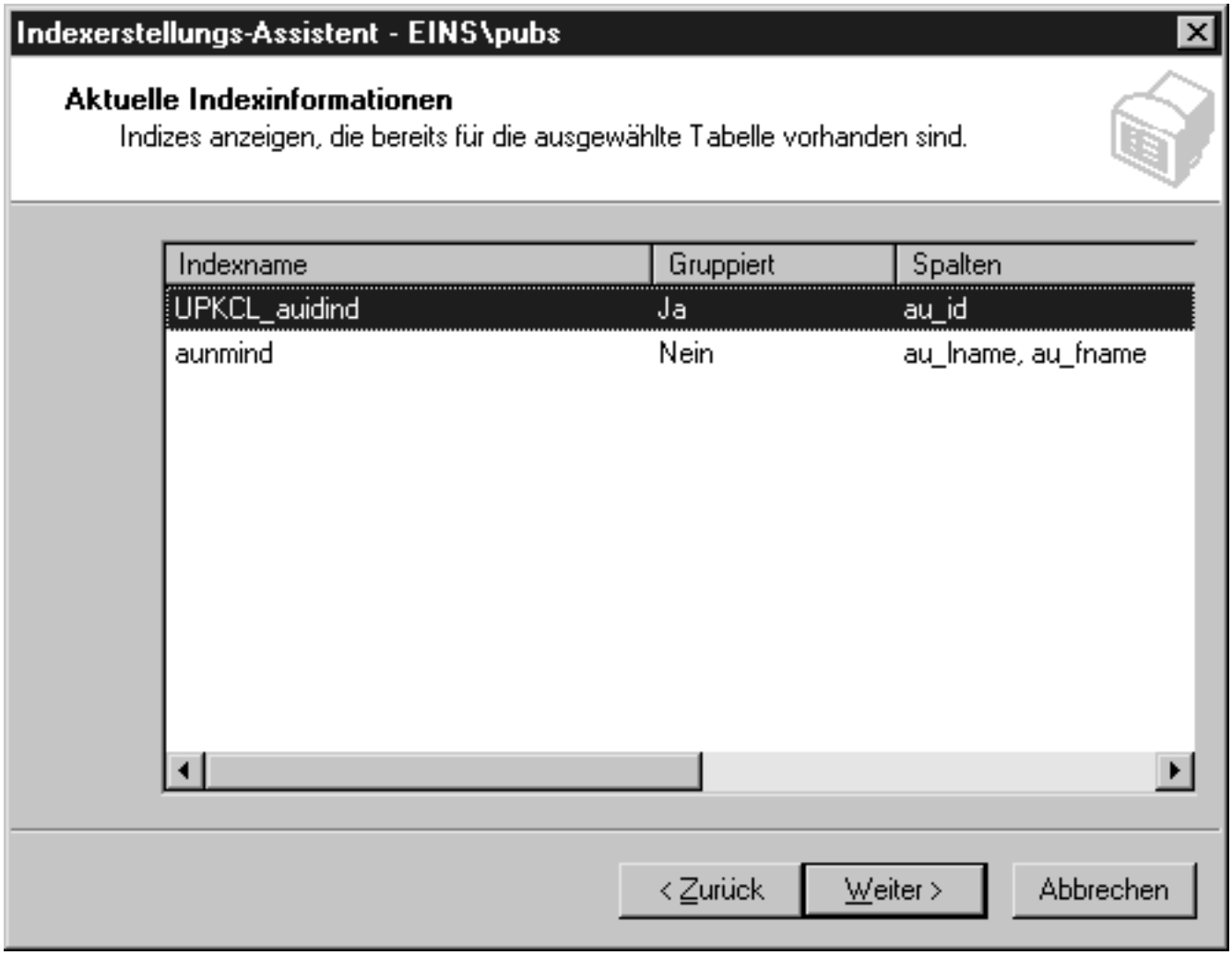


abbildung 14.3: informationen über bereits vorhandene indizes

5. im nächsten dialogfeld legen sie die spalten fest, die sie in den index einbeziehen wollen (siehe abbildung 14.4).

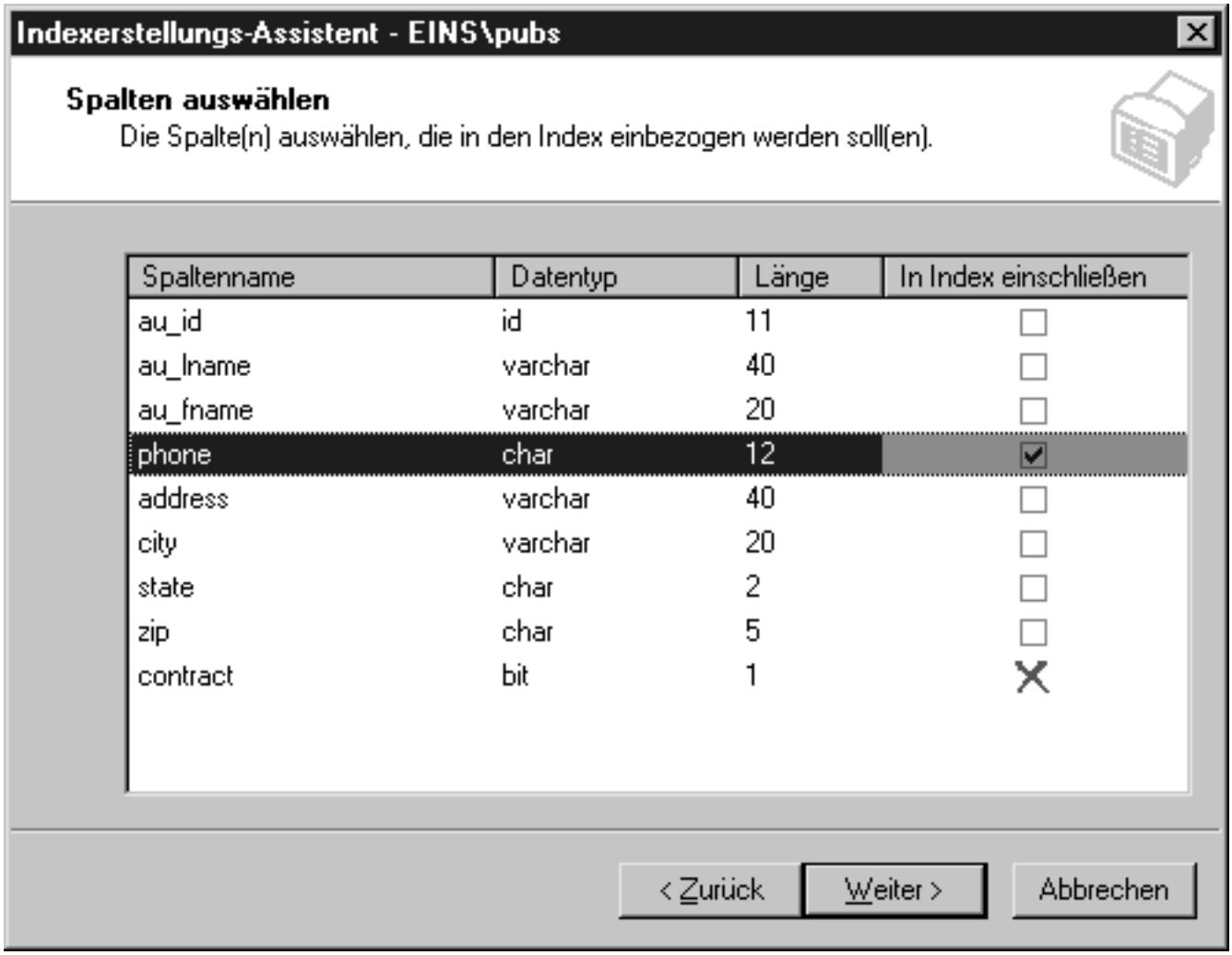


abbildung 14.4: spalten für den neuen index auswählen

für spalten der datentypen bit, text und image können sie keinen index erstellen. im dialogfeld **spalten auswählen** (siehe abbildung 14.4) ist deshalb die spalte contract mit dem datentyp bit durch ein rotes kreuz gekennzeichnet.

6. im dialogfeld **indexoptionen festlegen** geben sie die art des neuen index an (siehe abbildung 14.5).

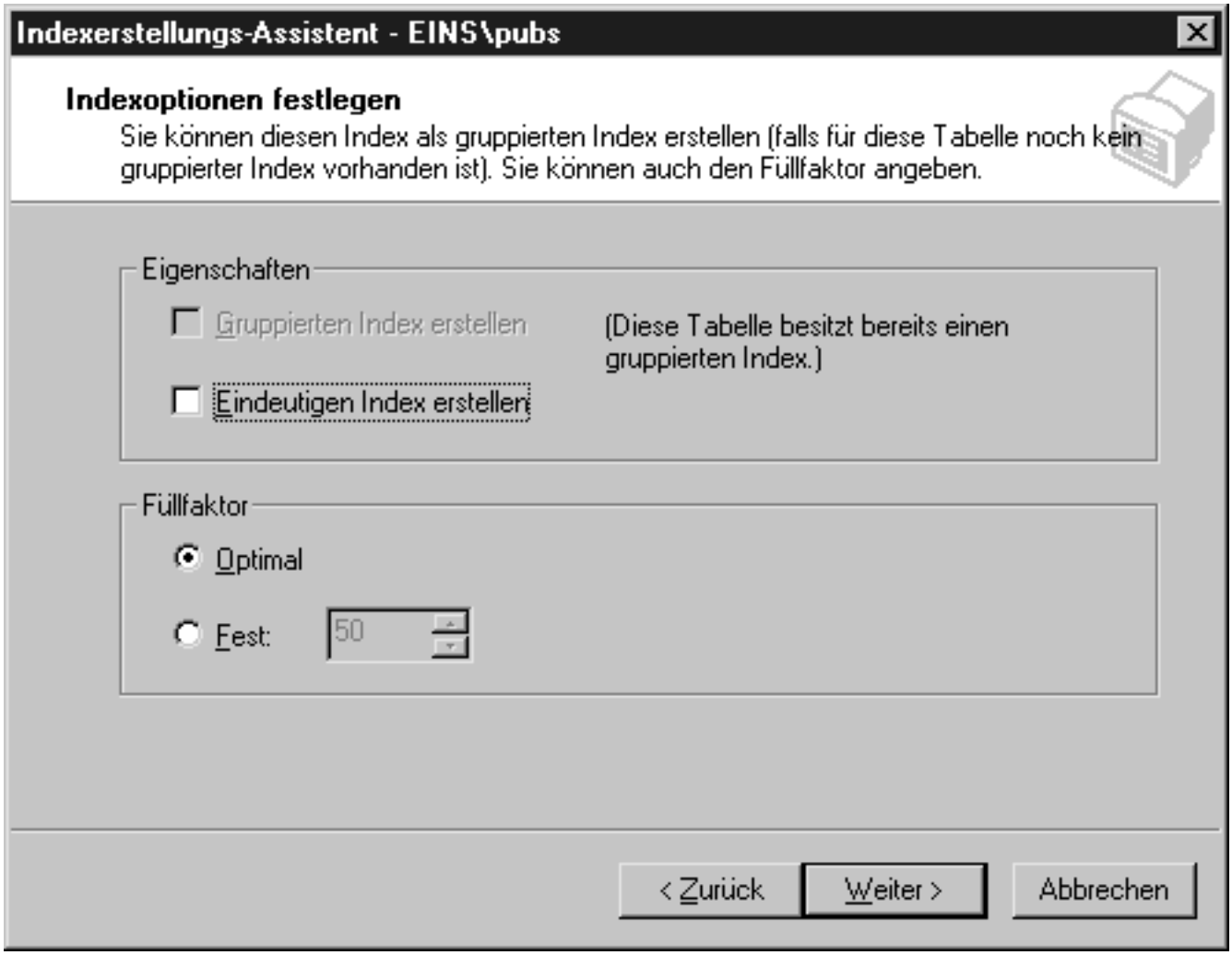


abbildung 14.5: indexoptionen festlegen

folgende indexoptionen lassen sich festlegen:

- **gruppierten index erstellen**: wenn sie dieses kontrollkästchen einschalten, erstellt der assistent auf der basis der ausgewählten spalte(n) einen gruppierten index für die tabelle. da sql server nur einen gruppierten index pro tabelle erlaubt, ist das kontrollkästchen deaktiviert, wenn bereits ein gruppierter index definiert ist.
- **eindeutigen index erstellen**: da ein eindeutiger index die höchste selektivität aufweist, sollten sie dieses kontrollkästchen einschalten, wenn die werte in der ausgewählten spalte oder der kombination von spalten eindeutige schlüssel repräsentieren können. wenn die ausgewählten spalten allerdings zu doppelten schlüsseln führen, kann der assistent den index nicht erstellen. abbildung 14.6 zeigt als beispiel die fehlermeldung, wenn sie einen eindeutigen index für die spalte fname der tabelle employee in der datenbank pubs erstellen wollen.

bild

abbildung 14.6: fehlermeldung beim versuch, einen eindeutigen index zu erstellen

- **füllfaktor**: mit den optionen dieses abschnitts legen sie fest, wieviel platz sql server auf jeder indexseite für neue einträge reserviert, d.h. wie dicht die indexeinträge gepackt werden. kommt ein neuer eintrag auf eine volle indexseite hinzu, verschiebt sql server etwa die hälfte der seite auf eine neue seite. diese sogenannte *seitenteilung* nimmt natürlich eine gewisse zeit in anspruch. hier ist

also ein kompromiß zwischen schnellen aktualisierungen (niedrigem füllfaktor) und hoher abfrageleistung (großer füllfaktor in richtung 100 prozent) zu schließen. bei einem hohen füllfaktor erhöht sich die abfrageleistung, weil der index dichter gepackt ist (d.h. weniger speicher benötigt), folglich weniger indexebenen erforderlich sind und sql server bei einer suche weniger seiten lesen muß.

- im letzten dialogfeld des indexerstellungs-assistenten (siehe abbildung 14.7) können sie den vorgegebenen namen für den index überschreiben.

aus dem indexnamen sollte zumindest im ansatz hervorgehen, auf welchen spalten der index basiert. darüber hinaus können sie sich weitere regeln für die vergabe von indexnamen zu eigen machen. vielleicht sind ihnen die ominösen namen im dialogfeld **aktuelle indexinformationen** des indexerstellungs-assistenten (siehe abbildung 14.3) aufgefallen. hierbei handelt es sich nicht um namen, die der assistent generiert hat, sondern um die namen, die im skript instpubs.sql beim erstellen der datenbank pubs angegeben sind. zum beispiel hat upkcl_auidind folgende bedeutung: *u*nique *p*rimary *k*ey *cl*ustered für die spalte *au_id* - d.h. eindeutiger primärschlüssel, gruppiert. das suffix ind steht für index. beachten sie, daß ein indexname aus maximal 30 zeichen bestehen darf.

abbildung 14.7: index fertigstellen

8. wenn sie mehrere spalten für den index festgelegt haben, können sie die reihenfolge der spalten im index mit den schaltflächen **nach oben** bzw. **nach unten** ändern. markieren sie die entsprechende spalte im listenfeld **eingeschlossene spalten**, und ändern sie dann die reihenfolge mit hilfe der

schaltflächen. nachdem sie alle schritte abgeschlossen haben, klicken sie auf die schaltfläche **fertigstellen**, um den index erstellen zu lassen.

14.3.2 index per transact-sql erstellen

fast so einfach wie mit dem indexerstellungs-assistenten können sie einen index auch mit der transact-sql-anweisung create index anlegen. die syntax lautet:

```
create index [unique] [clustered | nonclustered]
       index indexname
       on tabellenname (spaltenliste)
[with
    [pad_index]
    [[,] fillfactor = füllfaktor]
    [[,] ignore_dup_key]
    [[,] drop_existing]
    [[,] statistics_norecompute]
[on dateigruppe]
```

argumente:

unique spezifiziert, daß für diesen index keine duplikate erlaubt sind.

die option clustered bewirkt einen gruppierten index. das hat zur folge, daß die daten der tabelle physisch sortiert werden und die blattknoten des index darstellen. gruppierte indizes müssen eindeutig sein. ideal ist es, wenn bereits die für den index vorgesehenen spalten einen eindeutigen index ermöglichen und sie das schlüsselwort unique angeben können. bei einem gruppierten, aber nicht eindeutigen index (unique nicht spezifiziert) ergänzt sql server eine spalte mit einem 4 byte großen wert, der die eindeutigkeit sicherstellt.

mit der option nonclustered spezifizieren sie einen index, der als normaler b-baum realisiert ist und ein vollkommen selbständiges objekt darstellt.

wenn sie die beschriebenen optionen nicht angeben, gilt als voreinstellung ein nicht eindeutiger und nicht gruppierter index.

für *indexname* geben sie einen namen an, der innerhalb der tabelle eindeutig sein muß, in einer datenbank aber mehrfach vorkommen kann.

die mit *tabellenname* bezeichnete tabelle enthält die spalten, die zu indizieren sind.

in der *spaltenliste* geben sie durch komma getrennt und in der gewünschten reihenfolge alle spalten an, aus denen ein zusammengesetzter index bestehen soll. nicht zulässig sind spalten der datentypen ntext, text, image und bit sowie berechnete spalten oder funktionen.

wie bereits weiter oben in diesem kapitel erwähnt, legt der füllfaktor fest, wieviel platz auf jeder

datenseite freizuhalten ist. der füllfaktor, den sie in der create index-anweisung mit fillfactor festlegen, bezieht sich nur auf die blattseiten. wenn sie die option pat_index angeben, hält sql server auch auf den zwischenebenen des index den mit fillfactor spezifizierten prozentsatz frei. die option pat_index hat also nur dann sinn, wenn sie auch fillfactor explizit spezifizieren.

die option ignore_dup_key erlaubt es, daß sie in eine tabelle mit eindeutig gruppiertem index auch zeilen einfügen können, deren spaltenwerte zu doppelten schlüsseln führen. normalerweise löst das eine fehlermeldung aus, und sql server führt einen rollback für die gesamte transaktion aus. mit ignore_dup_key verwirft sql server die betreffenden zeilen und gibt nur eine warnung aus.

ein beispiel soll das verdeutlichen. die folgenden anweisungen erstellen die datenbank test, machen sie zur aktuellen datenbank und legen die tabelle ttab mit den spalten sp1 und sp2 in der datenbank test an:

```
create database test
go
use test
go
create table ttab (sp1 int, sp2 varchar(29))
go
```

als nächstes wird ein eindeutiger gruppierter index namens testind mit der spalte sp1 der tabelle ttab erstellt, wobei die option ignore_dup_key spezifiziert ist:

```
create unique clustered index testind on ttab (sp1) with
ignore_dup_key
go
```

fügen sie nun die folgenden vier zeilen ein:

```
insert ttab values (1, 'eins')
insert ttab values (2, 'zwei')
insert ttab values (2, 'drei')
insert ttab values (4, 'vier')
go
```

nach der dritten zeile bringt sql server die meldung:

```
server: nachr.-nr. 3604, schweregrad 16, status 1, zeile 7
doppelter schlüssel wurde ignoriert.
```

rufen sie nun alle zeilen aus der tabelle ab:

```
select * from ttab
```

das ergebnis lautet:

```
sp1         sp2
----------- ------------------------------
1           eins
2           zwei
4           vier

(3 row(s) affected)
```

da die dritte einzufügende zeile den gleichen schlüssel liefert wie die bereits eingefügte zweite zeile, verwirft sie sql server und zeigt die warnung an. im endeffekt finden sie nur drei zeilen in der tabelle vor.

die option drop_existing bewirkt, daß sql server den bereits vorhandenen index löscht und neu erstellt. wenn sie einen gruppierten index mit drop_existing neu erstellen und sich die schlüssel nicht ändern, werden vorhandene nicht gruppierte indizes nicht beeinflußt. die option drop_existing bietet sich also an, wenn sie einen index reorganisieren wollen.

mit der option statistics_norecompute unterdrücken sie die automatische neuberechnung von indexstatistiken. diese option ist nicht zu empfehlen, da der optimierer von sql server möglicherweise nicht die optimalen ausführungspläne für abfragen dieser tabelle wählt. wenn sie dennoch diese option angegeben haben, können sie die anweisung update statistics ohne die klausel norecompute ausführen, um die automatische neuberechnung zu aktivieren (siehe den entsprechenden abschnitt weiter unten in diesem kapitel).

das skript instpubs.sql erstellt in den zeilen 808 bis 828 mehrere indizes mit create index. anhang b geht ausführlich auf dieses skript ein.

14.4 indizes verwalten

indizes brauchen hin und wieder etwas pflege, damit sie wirkungsvoll bleiben. das kann manuell geschehen oder über wartungsroutinen. auf jeden fall brauchen sie erst einmal informationen zu den betreffenden indizes.

14.4.1 informationen zu tabellen und indizes

im enterprise manager können sie sich einen überblick über die tabellen und indizes einer datenbank verschaffen. erweitern sie dazu die konsolenstruktur bis zur gewünschten datenbank, und klicken sie auf den datenbankordner. daraufhin lädt sql server die datenbankinformationen und zeigt sie im rechten fensterausschnitt des enterprise managers an. unter der rubrik *tabellen & indizes* finden sie eine übersicht über die derzeit in einer datenbank vorhandenen tabellen und indizes sowie deren aktuelle größen (siehe abbildung 14.8).

bild

abbildung 14.8: der enterprise manager mit der taskpad-ansicht (rubrik tabellen & indizes) für die datenbank pubs

informationen zu den indizes einer tabelle können sie mit der gespeicherten prozedur sp_helpindex abrufen:

```
sp_helpindex [@objname =] 'tabellenname'
```

die ergebnismenge liefert drei spalten mit folgenden angaben:

- index_name: name des index
- index_description: beschreibung des index
- index_keys: spalten, aus denen der index aufgebaut ist

die anweisung

```
use pubs
exec sp_helpindex authors
```

liefert angaben zu den indizes der tabelle authors. (aus satztechnischen gründen stehen die drei spalten der ergebnismenge jeweils untereinander.)

```
index_name
index_description
index_keys
-------------------------------------------------------------
upkcl_auidind
clustered, unique, primary key located on primary
au_id

aunmind
nonclustered located on primary
```

```
au_lname, au_fname

authors_index_1
nonclustered, unique located on primary
au_fname
```

14.4.2 indizes hinzufügen, bearbeiten oder löschen

indizes können sie sowohl mit dem enterprise manager als auch per transact-sql hinzufügen, bearbeiten oder löschen. im enterprise manager führen sie zunächst die folgenden schritte aus:

1. erweitern sie die konsolenstruktur bis zur tabelle, für die sie die indizes verwalten wollen. klicken sie auf den ordner **tabellen**, um die tabellen im rechten fensterausschnitt auflisten zu lassen.
2. klicken sie im rechten fensterausschnitt mit der rechten maustaste auf die gewünschte tabelle. im kontextmenü wählen sie **alle aufgaben** und im daraufhin eingeblendeten menü den befehl **indizes verwalten**. im dialogfeld **indizes verwalten** ist die im schritt 1 ausgewählte tabelle bereits vorgegeben. allerdings können sie hier auch andere datenbanken und tabellen auswählen (siehe abbildung 14.9).

abbildung 14.9: das dialogfeld indizes verwalten

in der mitte des dialogfelds **indizes verwalten** sind die vorhandenen indizes aufgelistet. wenn sie einen index bearbeiten oder löschen wollen, markieren sie die entsprechende indexzeile und klicken dann auf **bearbeiten** bzw. **löschen**. wollen sie zum beispiel den weiter oben erstellten index authors_index_1 wieder loswerden, markieren sie die dritte zeile und klicken auf **löschen**. jetzt brauchen sie nur das daraufhin erscheinende meldungsfeld mit **ja** zu bestätigen - schon ist der index im nirwana verschwunden.

über die schaltfläche **bearbeiten** gelangen sie zum dialogfeld **vorhandenen index bearbeiten** (siehe abbildung 14.10).

bild

abbildung 14.10: das dialogfeld vorhandenen index bearbeiten

über dieses dialogfeld können sie zusätzliche spalten in den index aufnehmen, spalten entfernen und andere optionen festlegen. die schaltfläche **sql bearbeiten** öffnet das dialogfeld transact-sql-skript

bearbeiten, das die transact-sql-anweisung für das erstellen des index zeigt (siehe abbildung 14.11). als beispiel wurde hier die option fillfactor=50 ergänzt. nachdem sie die transact-sql-anweisung geändert haben, klicken sie auf die schaltfläche **analysieren**. sql server überprüft daraufhin die anweisung und sagt ihnen, ob alles in ordnung ist und die gewählten optionen für den betreffenden index zulässig sind. wenn sie die änderungen übernehmen wollen, klicken sie auf **ausführen**, andernfalls auf **abbrechen**.

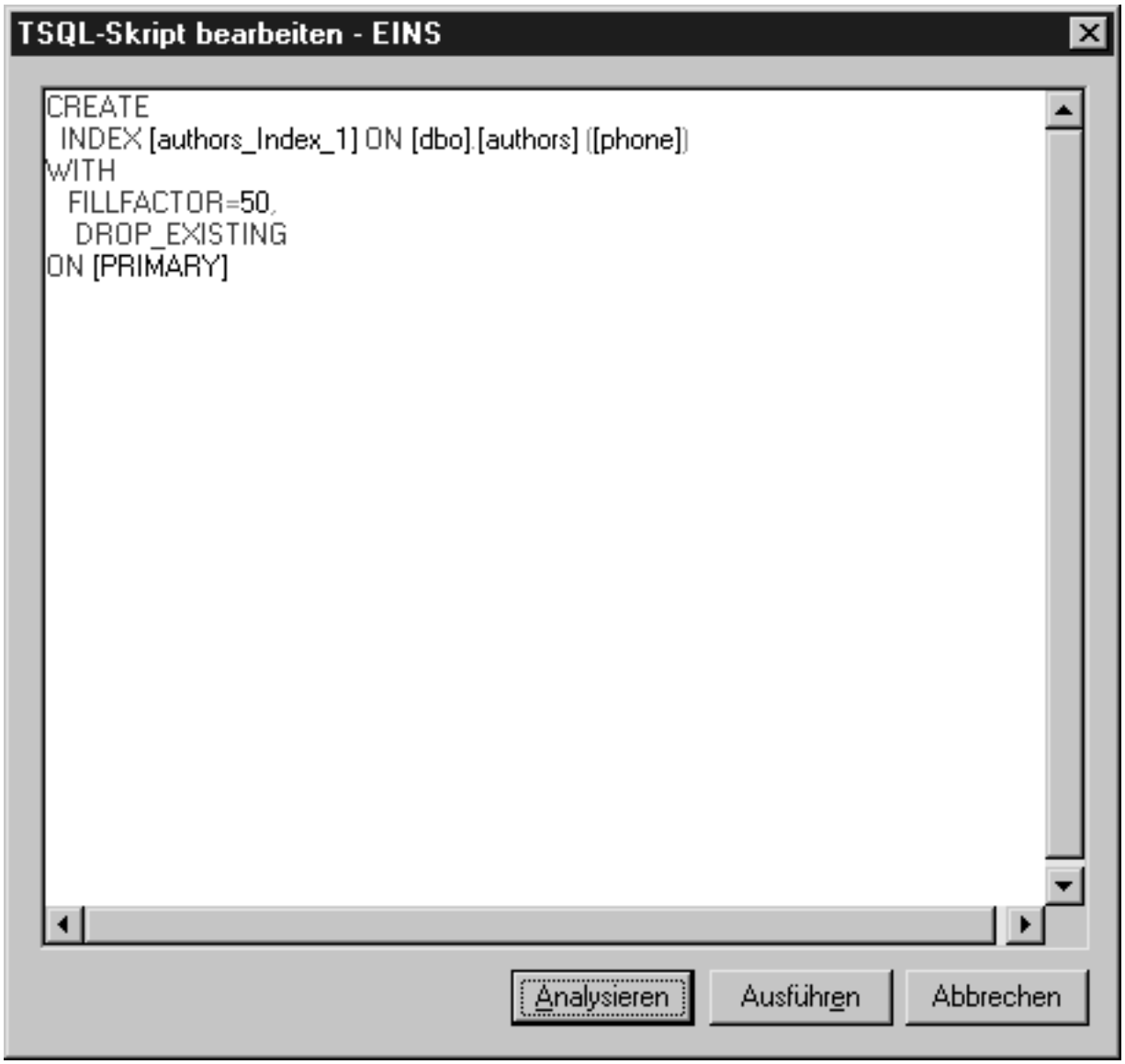


abbildung 14.11: in diesem dialogfeld können sie die transact-sql-anweisung bearbeiten

haben sie **ausführen** gewählt, übernimmt sql server die neue indexdefinition und kehrt dann zum dialogfeld **indizes verwalten** zurück. markieren sie zur kontrolle den geänderten index (im beispiel authors_index_1), und klicken sie auf **bearbeiten**. die ergänzte option fillfactor=50 ist jetzt im dialogfeld **vorhandenen index bearbeiten** zu sehen: das kontrollkästchen **füllfaktor** ist eingeschaltet, die prozentzahl steht daneben.

über das dialogfeld **indizes verwalten** können sie auch einen neuen index anlegen. klicken sie dazu auf die schaltfläche **neu**. das dialogfeld **neuen index erstellen** entspricht im wesentlichen dem in abbildung 14.10 dargestellten dialogfeld **vorhandenen index bearbeiten**. allerdings ist jetzt das feld **indexname** aktiviert. tragen sie hier einen passenden namen für den neuen index ein. die übrigen schritte entsprechen denen für die bearbeitung eines vorhandenen index.

einen index können sie auch über das dialogfeld **eigenschaften** (siehe nächster abschnitt) bearbeiten.

14.4.3 indizes umbenennen oder bearbeiten

man könnte eigentlich erwarten, daß sich im dialogfeld **indizes verwalten** ein index auch umbenennen läßt. dem ist aber leider nicht so. wenn sie per enterprise manager einen index umbenennen möchten, brauchen sie zunächst ein datenbankdiagramm. sollte für die betreffende datenbank noch kein datenbankdiagramm vorhanden sein, erstellen sie es nach dem in kapitel 6 beschriebenen verfahren. achten sie darauf, daß mindestens die tabelle eingebunden ist, für die sie einen index umbenennen möchten. zum datenbankdiagramm beziehungen nach dem beispiel aus kapitel 6 gehören alle tabellen der datenbank pubs.

die folgenden schritte zeigen am beispiel der datenbank pubs, wie sie einen index mit hilfe des datenbankdiagramms beziehungen umbenennen:

1. klicken sie im ordner der datenbank pubs auf **diagramme**. doppelklicken sie dann im rechten fensterbereich des enterprise managers auf das datenbankdiagramm beziehungen, um es zu öffnen.
2. klicken sie mit der rechten maustaste auf die tabelle authors (die tabelle, die den umzubenennenden index enthält), und wählen sie **eigenschaften** aus dem kontextmenü.

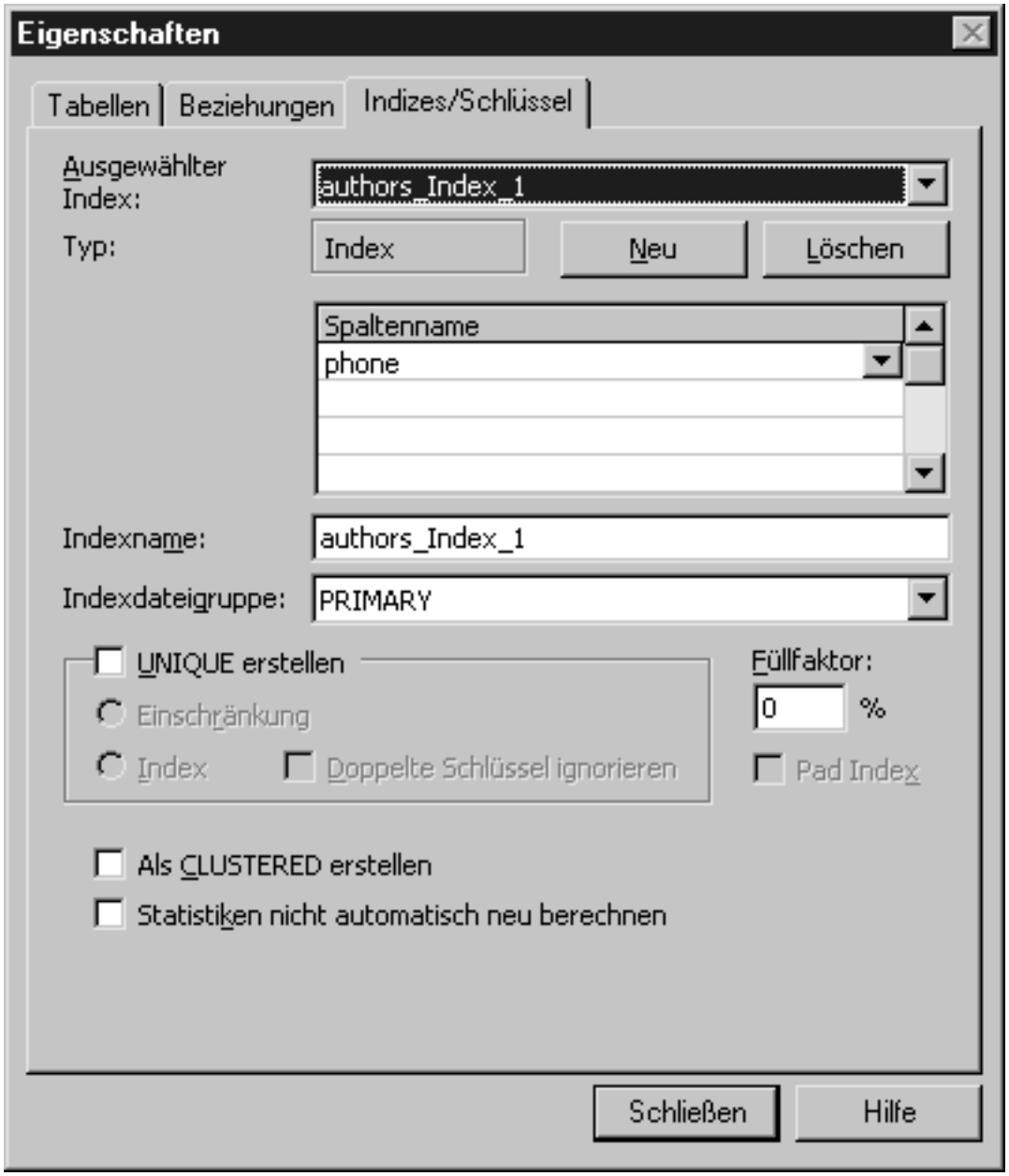


abbildung 14.12: das dialogfeld eigenschaften mit der registerkarte indizes/schlüssel

3. im dialogfeld **eigenschaften** (siehe abbildung 14.12) wählen sie zunächst im kombinationsfeld **ausgewählter index** den index aus, den sie umbenennen wollen - im beispiel den weiter oben erstellten index authors_index_1, falls sie ihn noch nicht gelöscht haben.
4. im bearbeitungsfeld **indexname** überschreiben sie einfach den bisherigen namen für den index durch den neuen.
5. nachdem sie alle änderungen (gegebenenfalls auch an anderen indizes) vorgenommen haben, klicken sie auf **schliessen**, um die änderungen zu übernehmen. eine schaltfläche **abbrechen** gibt es nicht - geändert ist geändert.

im dialogfeld **eigenschaften** lassen sich weitere modifikationen an einem index vornehmen. wenn sie im feld **spalten** in eine leere zeile klicken, erscheint ein drop-down-pfeil, über den sie eine liste der verfügbaren spalten einblenden und neue spalten in den index aufnehmen können. das kontrollkästchen **unique erstellen** schalten sie ein, wenn der index eindeutig sein soll. um einen gruppierten index zu erstellen, aktivieren sie das kontrollkästchen **als clustered erstellen**. wenn allerdings bereits ein

gruppierter index für die tabelle vorhanden ist, weist sie eine meldung wie in abbildung 14.13 darauf hin.

bild

abbildung 14.13: hinweis auf vorhandenen gruppierten index

die beiden zuletzt genannten optionen entsprechen denjenigen, die sie beim erstellen eines index im dialogfeld **indexoptionen festlegen** (siehe abbildung 14.5) eingestellt haben.

14.5 indexauswahl

wie eingangs erwähnt, bringen indizes in der regel leistungsverbesserungen, können aber auch hinderlich sein, wenn vorwiegend einfüge- und aktualisierungsoperationen auszuführen sind. daraus ergibt sich, daß man nicht aufs geratewohl indizieren sollte. es stellt sich also die frage, was zu indizieren ist, was man besser nicht indizieren sollte und ob gruppierte oder nicht gruppierte indizes vorteilhafter sind.

für einen index bieten sich spalten an, die in

- primär- oder fremdschlüsselspalten verwendet werden,
- abfragen mit order by und group by vorkommen,
- aggregatfunktionen verwendet werden,
- einer where-klausel genau wie angegeben als bedingung formuliert sind.

dagegen eignen sich indizes nicht bzw. nur schlecht

- für tabellen, die nur wenige zeilen enthalten,
- für spalten mit einer schlechten selektivität (beispielsweise eine spalte, die das geschlecht eines mitarbeiters angibt und damit nur zwei mögliche werte enthält),
- wenn spalten sehr breit sind (ein index ist auf 900 byte begrenzt, bei unicode also nur 450 zeichen),
- für tabellen, die häufig an umfangreichen transaktionen (einfüge- und löschoperationen) beteiligt sind,
- wenn spalten kaum oder gar nicht in einer abfrage verwendet werden,
- für spalten der datentypen text, image oder bit.

der vorrat an gruppierten indizes ist sehr begrenzt - nämlich genau auf einen. demgegenüber können sie bis zu 249 nicht gruppierte indizes erstellen. gruppierte indizes bieten sich an

- wenn die suchbedingung in einer abfrage mit konkreten werten formuliert ist, beispielsweise where spalte1 = 17,
- wenn die suchbedingung in einer abfrage als bereich angegeben ist, beispielsweise where spalte1 between 2200 and 2260,
- wenn häufig abfragen auf spalten mit order by und group by vorkommen,
- bei spalten für fremdschlüssel.

außerdem sollten sie darauf achten, daß die schlüssel in einem gruppierten index möglichst kurz sind.

nachdem sie einen gruppierten index für eine tabelle festgelegt haben, bleiben ihnen nur noch nicht gruppierte indizes für die restlichen indexaufgaben.

nicht gruppierte indizes verwenden sie bei

- spalten, auf die sie aggregatfunktionen (zum beispiel min, max) anwenden.
- bei *abgedeckten abfragen*. das sind abfragen, bei denen alle angegebenen spalten innerhalb desselben index enthalten sind. wenn sie zum beispiel in einer tabelle mitarbeiter einen index für die spalten vorname und nachname definieren, läßt sich eine abfrage wie

```
select vorname from mitarbeiter where nachname = "white"
```

- vollständig über den index realisieren, ohne daß ein zugriff auf die eigentlichen datenseiten erforderlich ist.
- abfragen mit kleinen ergebnismengen.
- einsatz der dbcc-anweisung dbreindex, um indizes dynamisch neu zu erstellen.

allgemein gilt, daß sie spalten indizieren sollten, die in der where-klausel von abfragen auftauchen und dem optimalen ausführungsplan von sql server entsprechen.

eine hilfe für den optimalen einsatz von indizes stellt der indexoptimierungs-assistent dar, auf den kapitel 24 näher eingeht.

ebenfalls in kapitel 24 erfahren sie, wie man mit hilfe des sql server query analyzer vorschläge für indizes erhalten kann.

14.6 indexeigenschaften

ein schlüsselwert kann auf genau eine zeile oder auch mehrere zeilen zeigen. diese eigenschaft drückt die *selektivität* eines index aus. die höchste selektivität hat ein eindeutiger schlüsselwert, während ein schlüssel, der sich auf sehr viele zeilen bezieht, eine niedrige selektivität hat. die qualität von indizes läßt sich aus den zugehörigen verteilungsstatistiken ablesen. die dbcc-anweisung show_statistics liefert einen bericht zur verteilungsstatistik eines index und damit auch der selektivität.

```
dbcc show_statistics (tabelle, ziel)
```

tabelle gibt den namen der tabelle an, für die sie statistische informationen abrufen wollen. *ziel* bezeichnet ein objekt (indexname oder auflistung), für das die informationen zu liefern sind.

informationen zum index employee_ind rufen sie mit der folgenden anweisung ab:

```
use pubs
dbcc show_statistics (employee, employee_ind)
```

das ergebnis lautet:

```
statistik für index 'employee_ind'.
updated              rows        rows sampled
-------------------- ----------- ------------
apr 15 1999  4:08am  43          43

steps       density                  average key length
----------- ------------------------ -------------------------
43          2.3255814e-2             13.372094

(1 row(s) affected)

all density              columns
------------------------ ---------------------------------------
2.3255814e-2             lname
2.3255814e-2             lname, fname
2.3255814e-2             lname, fname, minit

(3 row(s) affected)

steps
------------------------------
accorti
afonso
...
thomas
tonini

(43 row(s) affected)

dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.
```

die spalten der ergebnismenge haben folgende bedeutung:

- *updated*: datum und uhrzeit der letzten statistikaktualisierung
- *rows*: anzahl der zeilen in der tabelle
- *rows sampled*: anzahl der zeilen, die in der statistik berücksichtigt wurden
- *steps*: anzahl der verteilungsschritte
- *density*: selektivität des index

- *average key length*: durchschnittliche schlüssellänge
- *all density*: selektivität des angegebenen spaltenpräfix im index
- *columns*: name des spaltenpräfix, auf den sich der wert von *all density* bezieht
- *steps*: nummer der histogrammwerte

14.7 volltextindizes

sql server bietet mit der version 7.0 erstmalig die möglichkeit, textspalten und andere zeichenspalten zu indizieren. bisher ließen sich zeichenwerte nur mit hilfe der vergleichsoperatoren oder der funktionen zum mustervergleich suchen. die suche war außerdem auf spalten der datentypen char und varchar beschränkt.

was ist nun unter einer volltextindizierung zu verstehen? nehmen sie als beispiel die datenbank pubs, die in der tabelle titles alle vorhandenen titel verzeichnet. wenn sie nach einem bestimmten wort innerhalb eines beliebigen titels suchen, mußten sie bisher die tabelle zeilenweise abarbeiten und zum beispiel mit der funktion patindex (siehe kapitel 11) feststellen, ob das betreffende wort in der spalte title enthalten ist. wesentlich schneller und eleganter läßt sich das mit einer volltextsuche durchführen.

14.7.1 microsoft search

die unterstützung der volltextsuche in sql server 7.0 wird von einem unabhängigen modul - dem microsoft-search-dienst - realisiert. dieses modul unterstützt sowohl die indizierung als auch die abfragen. der microsoft-search-dienst steht nur unter win-dows nt server zur verfügung und gehört nicht zur standardinstallation. während einer benutzerdefinierten installation müssen sie die server-komponente *volltextsuche* auswählen (siehe dazu schritt 2 der benutzerdefinierten installation in kapitel 2). wenn die volltextsuche installiert ist, erscheint im sql-server-dienst-manager der dienst *microsoft search*.

14.7.2 sql server und volltextindizes

von herkömmlichen sql-server-indizes unterscheiden sich volltextindizes in mehreren grundsätzlichen punkten, von denen tabelle 14.1 die wichtigsten nennt.

sql-server-indizes	volltextindizes
gehören zur datenbank, in der sie definiert sind, und werden von dieser datenbank gesteuert	werden im dateisystem gespeichert, aber von der datenbank verwaltet
mehrere indizes pro tabelle möglich	nur ein volltextindex pro tabelle zulässig
beim einfügen, aktualisieren oder löschen von daten paßt sql server die indizes automatisch an die geänderten datenbestände an	aktualisierung muß angefordert werden, kann per terminplan oder spezieller anforderung geschehen

tabelle 14.1: normale indizes und volltextindizes

tabelle 14.1 macht auch deutlich, daß volltextindizes ein eigenleben führen. zur pflege eines volltextindex gehört vor allem, daß man ihn regelmäßig aktualisiert. dieser vorgang heißt *auffüllung*. kapitel 23 zeigt, wie sich wartungsaufgaben nach einem terminplan ausführen lassen.

sql server unterstützt die volltextindizierung mit einem assistenten, auf den der nächste abschnitt eingeht.

14.7.3 volltextindizierungs-assistent

die folgenden schritte zeigen, wie sie mit hilfe des volltextindizierungs-assistenten einen volltextindex für die tabelle titles der datenbank pubs erstellen. ziel ist es, alle datensätze aus der tabelle heraussuchen zu können, die ein bestimmtes stichwort enthalten. hierfür bieten sich die spalten title (titel) und notes (bemerkungen) an. die spalte type (gebiet) könnte man zwar auch in den volltextindex aufnehmen, allerdings läßt sich eine abfrage bezüglich dieser spalte mit einem normalen index einfacher erledigen.

1. erweitern sie die konsolenstruktur bis zum ordner **datenbanken** oder bis zur datenbank, in der sie den volltextkatalog erstellen wollen. wählen sie aus dem menü **extras** den befehl **volltextindizierung**. klicken sie im startdialogfeld des volltextindizierungs-assistenten auf **weiter**.
2. wenn sie den ordner datenbanken und nicht die datenbank selbst in der konsolenstruktur markiert haben, müssen sie zunächst im zweiten dialogfeld des assistenten die datenbank festlegen. wählen sie im beispiel die datenbank pubs.
3. im dritten dialogfeld des assistenten - das als zweites erscheint, wenn sie in der konsolenstruktur bereits die datenbank markiert haben - legen sie die zu indizierende tabelle fest (siehe abbildung 14.14). wählen sie für das beispiel die tabelle [dbo].[titles] aus dem drop-down-listenfeld **tabelle auswählen**. klicken sie auf **weiter**.

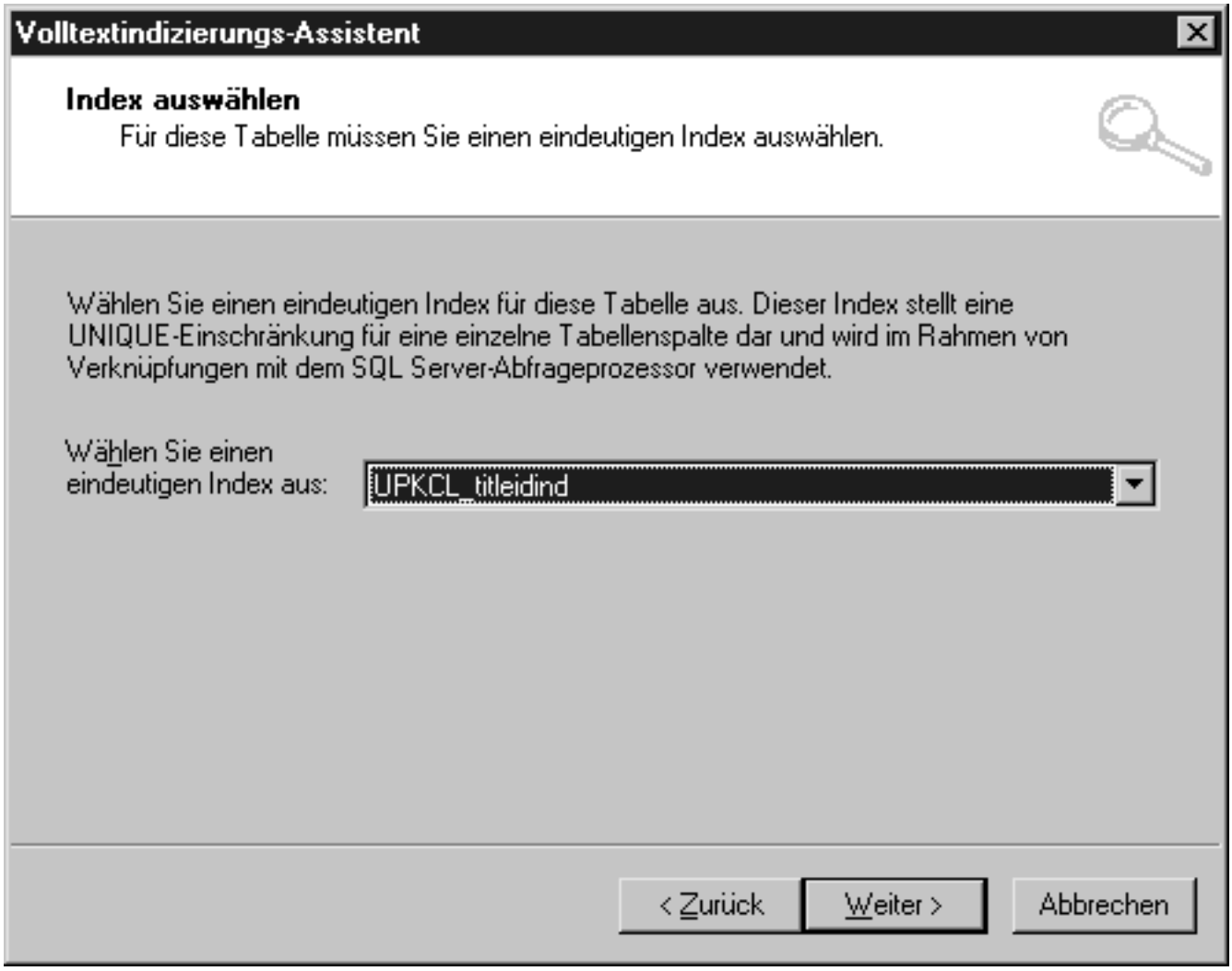


abbildung 14.14: in diesem dialogfeld müssen sie einen eindeutigen index auswählen

abbildung 14.15: im dritten dialogfeld des volltextindizierungs-assistenten legen sie die tabelle fest

4. das nächste dialogfeld verlangt die auswahl eines eindeutigen index (siehe abbildung 14.15). für die tabelle titles ist nur ein derartiger index definiert: upkcl_titleidind. klicken sie auf **weiter**.
5. im dialogfeld **tabellenspalten auswählen** legen sie die zeichenspalten fest, die in den volltextindex einfließen sollen. entsprechend dem eingangs formulierten ziel markieren sie die spalten title und notes (siehe abbildung 14.16). klicken sie auf die schaltfläche **hinzufügen**, um die beiden spalten in das listenfeld **hinzugefügte spalten** zu übernehmen. klicken sie auf **weiter**.

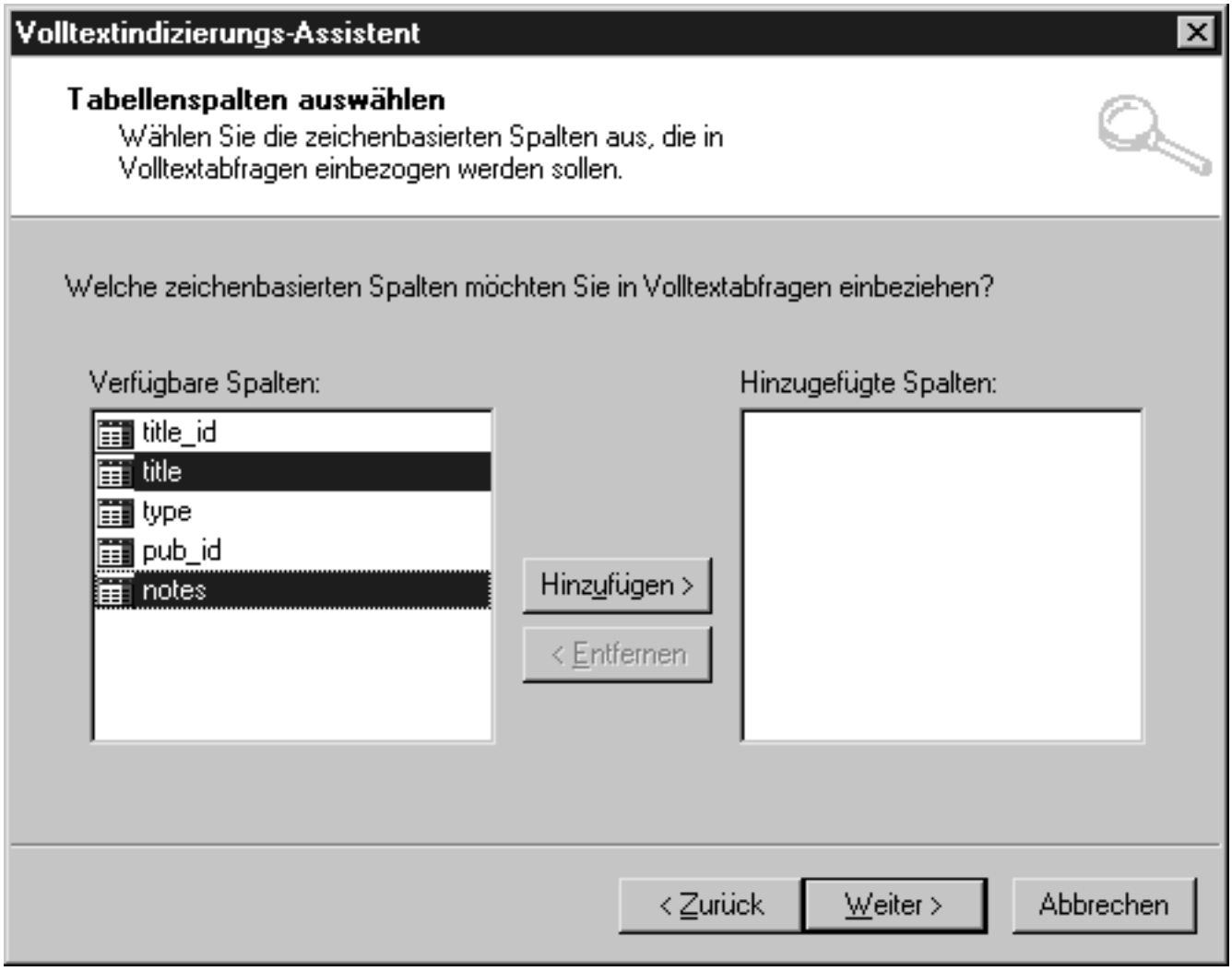


abbildung 14.16: auswahl der zu indizierenden zeichenspalten

6. die volltextindizes einer datenbank werden in einem oder mehreren katalog(en) zusammengefaßt. im dialogfeld **katalog auswählen** (siehe abbildung 14.17) legen sie fest, in welchen katalog der volltextindex aufzunehmen ist. da momentan noch kein katalog vorhanden ist, müssen sie einen neuen katalog erstellen. tragen sie im feld **name** eine passende bezeichnung ein - im beispiel deskriptoren. klicken sie auf **weiter**.

abbildung 14.17: geben sie den namen für einen neuen katalog ein

7. im nächsten dialogfeld können sie optional einen terminplan für die auffüllung festlegen. für das beispiel übergehen sie dieses dialogfeld einfach. den volltextindex können sie auch manuell auffüllen. klicken sie auf **weiter**. das letzte dialogfeld faßt die gewählten einstellungen zusammen (siehe abbildung 14.18). klicken sie auf **fertigstellen**, um den volltextindex anlegen zu lassen.

bild

abbildung 14.18: letztes dialogfeld des volltextindizierungs-assistenten

8. es erscheint nun ein dialogfeld, das den fortschritt anzeigt. wenn das erstellen des volltextindex abgeschlossen ist, zeigt dieses dialogfeld den hinweis an, daß sie den index noch auffüllen müssen (siehe abbildung 14.19). klicken sie auf **ok**.

abbildung 14.19: fertigmeldung nach erstellen des volltextindex

14.7.4 volltextindex auffüllen

der erstellte volltextindex allein bringt ihnen noch gar nichts. zunächst müssen sie den katalog auffüllen. da noch kein terminplan für das automatische auffüllen existiert, erledigen sie das manuell.

erweitern sie dazu im enterprise manager die konsolenstruktur bis zu den einträgen der datenbank pubs. klicken sie auf den ordner **volltextkataloge**, um die vorhandenen kataloge im detailbereich anzuzeigen.

momentan ist nur der gerade erstellte volltextkatalog deskriptoren vorhanden. klikken sie mit der rechten maustaste darauf. aus dem kontextmenü wählen sie den befehl **auffüllen starten** und aus dem daraufhin angezeigten untermenü den befehl **vollständiges auffüllen** (siehe abbildung 14.20). das meldungsfeld **volltextsuche** weist darauf hin, daß das auffüllen des volltextkatalogs erfolgreich gestartet wurde. bestätigen sie mit **ok**.

falls nach dem auffüllen noch keine angaben in der spalte **zuletzt aufgefüllt am** erscheinen, markieren sie in der konsolenstruktur den eintrag **volltextkataloge** und wählen aus dem menü **vorgang** den befehl **aktualisieren**.

Bild

abbildung 14.20: den volltextkatalog auffüllen

wenn sie per transact-sql überprüfen wollen, ob das auffüllen des volltextkatalogs abgeschlossen ist, führen sie folgende anweisung aus:

```
use pubs
select fulltextcatalogproperty('deskriptoren','populatestatus')
```

der rückgabewert 0 zeigt an, daß die auffüllung beendet ist und sich der dienst für den volltextkatalog im leerlauf befindet.

14.7.5 abfragen mit volltextindizes

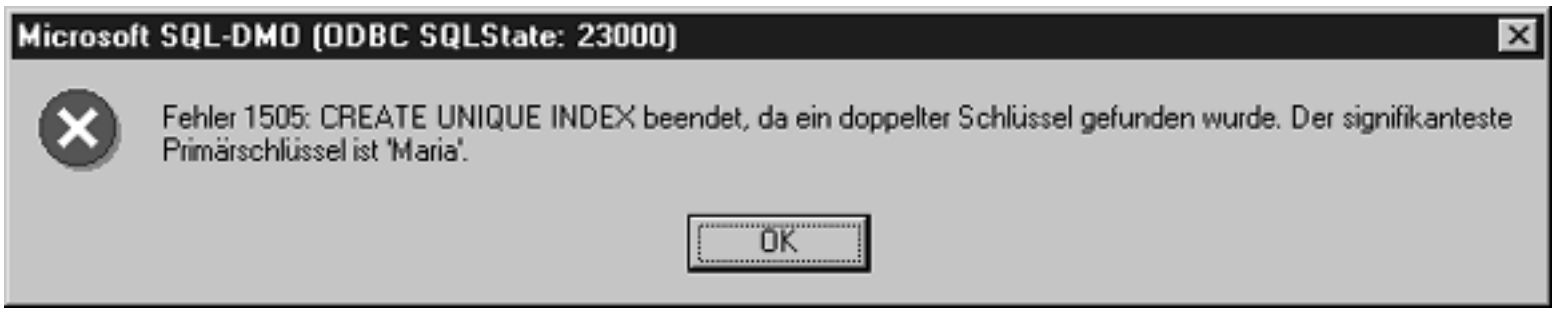
nachdem sie den volltextkatalog aufgefüllt haben, können sie abfragen bilden und dabei in der where-klausel das prädikat contains angeben:

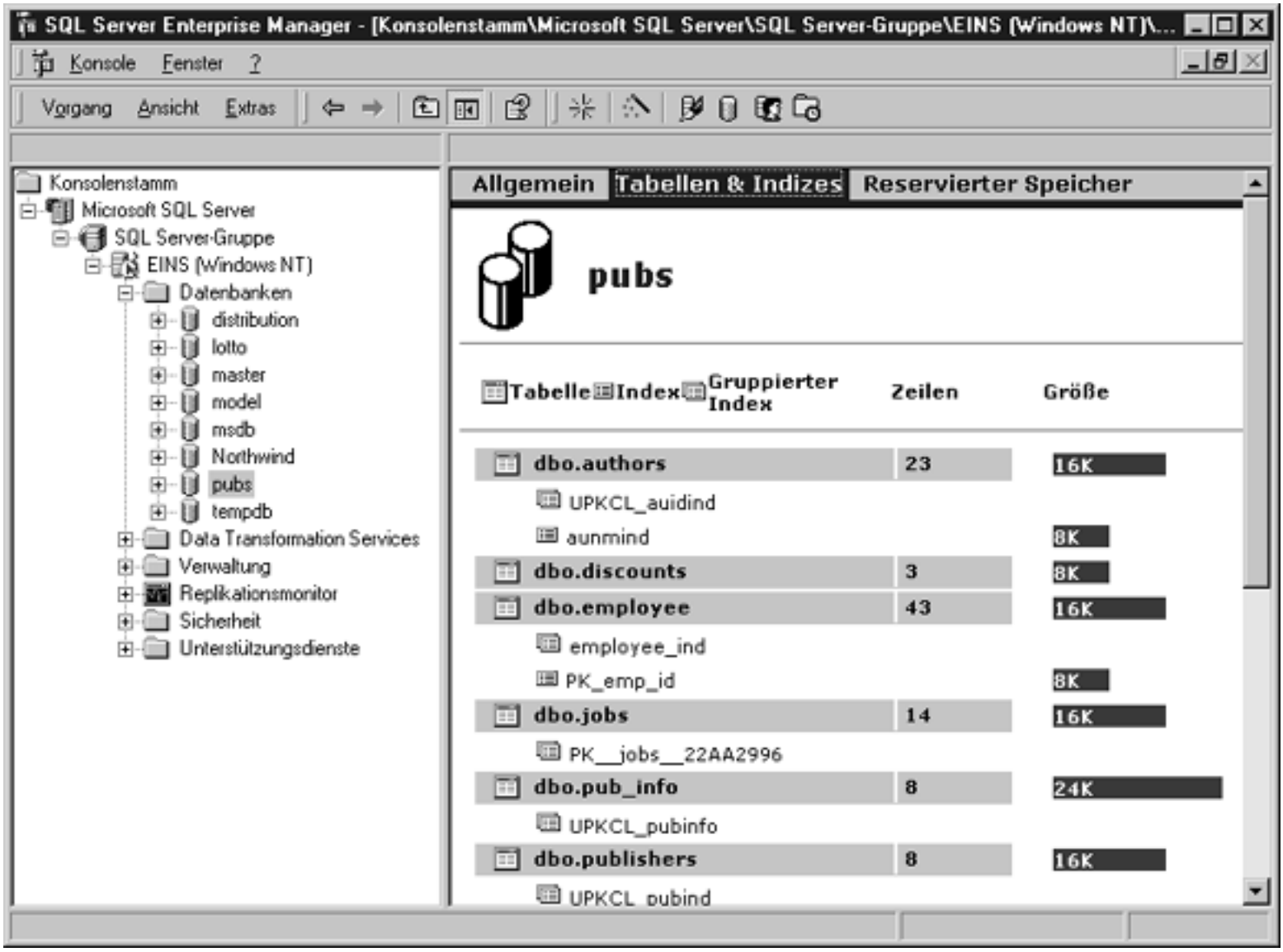
```
select title_id, title from titles
where contains (title, 'cooking')
```

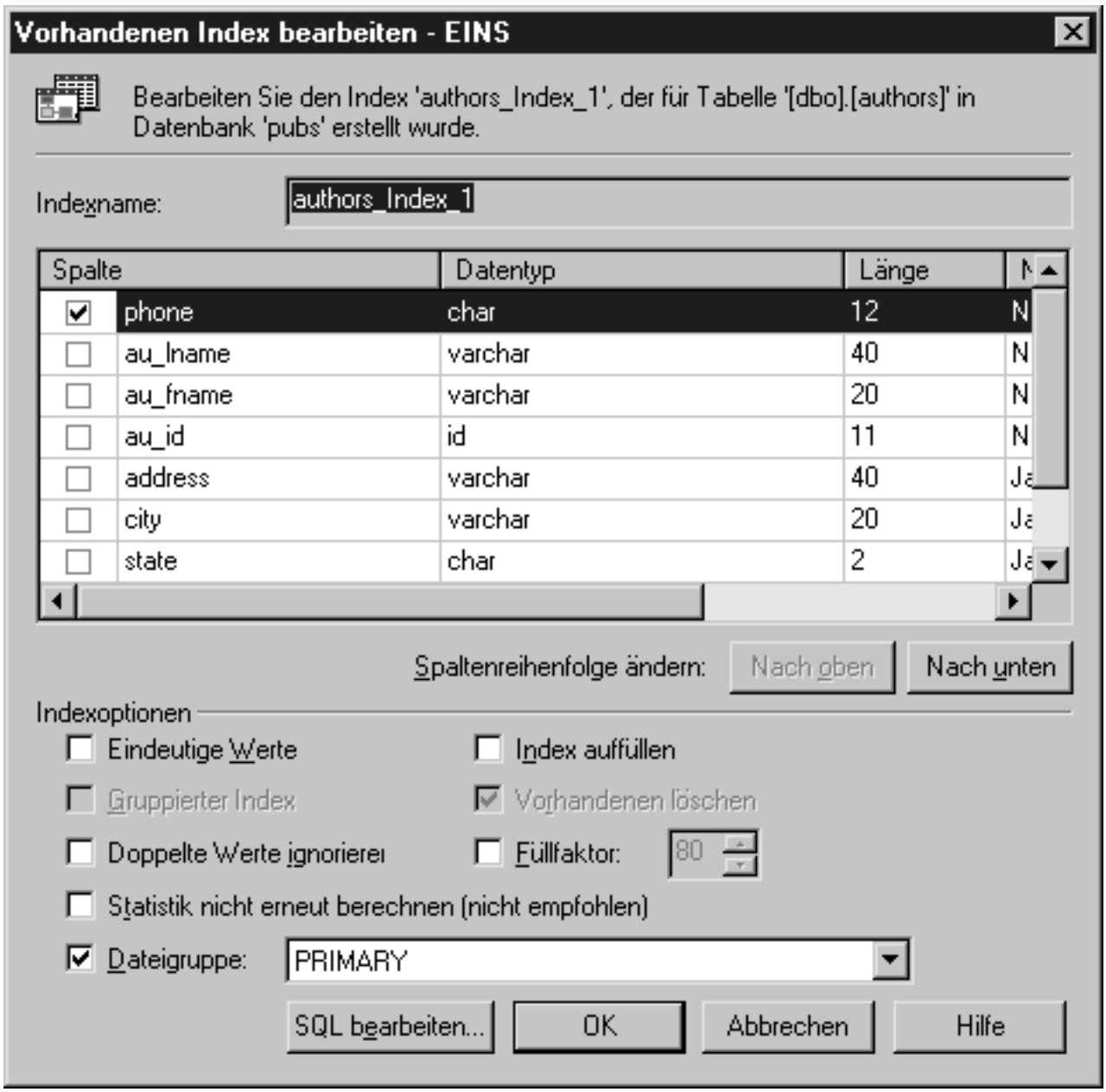
als ergebnis erhalten sie:

```
title_id title
-------- ------------------------------------------------------
bu1111   cooking with computers: surreptitious balance sheets
mc3026   the psychology of computer cooking
tc3218   onions, leeks, and garlic: cooking secrets of the medi

(3 row(s) affected)
```

das thema volltextindizes ist damit keineswegs abgeschlossen, läßt sich an dieser stelle aber nicht vollständig behandeln. es sei auf die online-dokumentation verwiesen, in der sie zahlreiche beispiele und weiterführende erläuterungen finden.

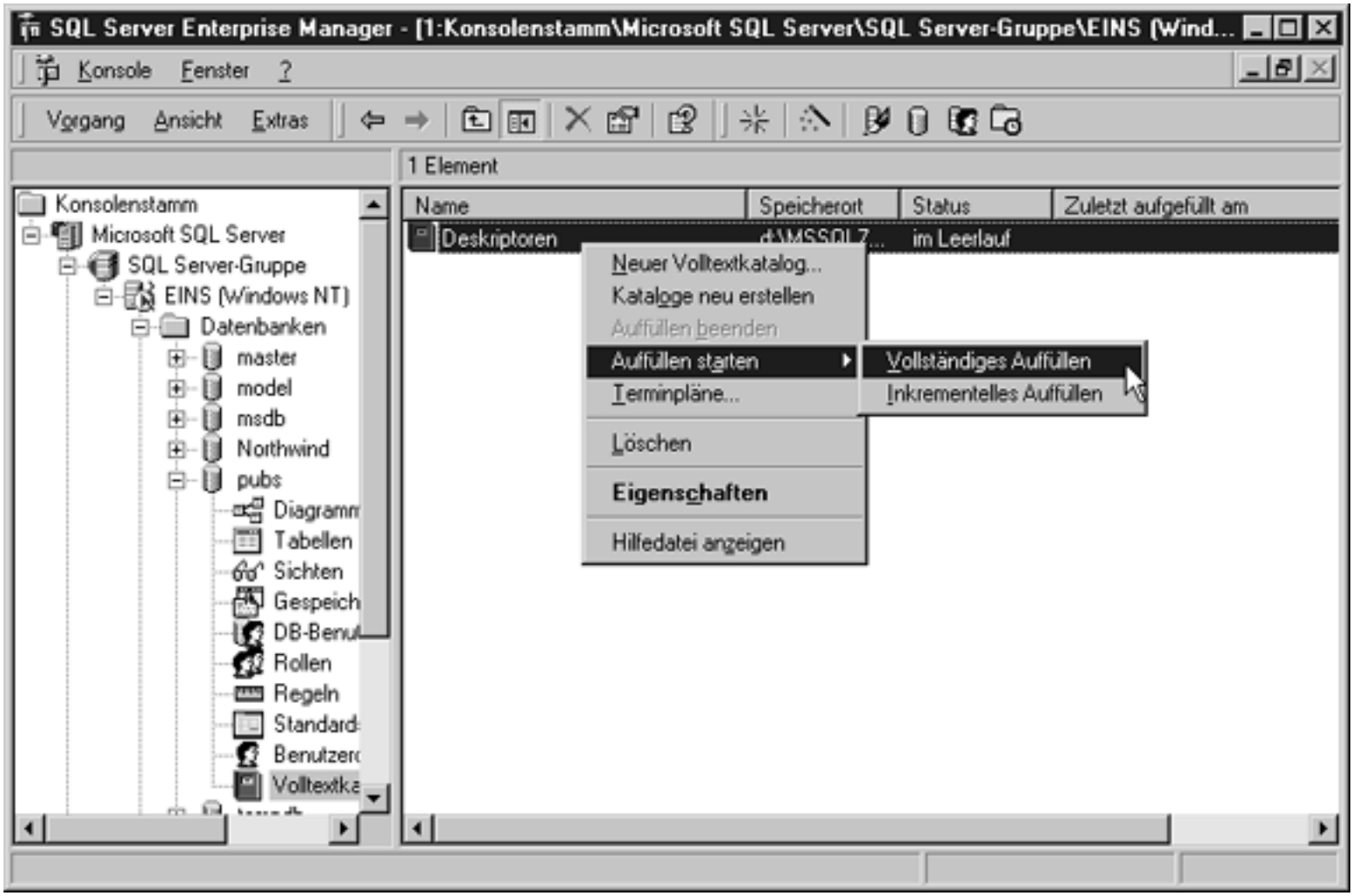

Microsoft SQL-DMO (ODBC SQLState: 23000)
Fehler 1505: CREATE UNIQUE INDEX beendet, da ein doppelter Schlüssel gefunden wurde. Der signifikanteste Primärschlüssel ist 'Maria'.
OK


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
distribution
lotto
master
model
msdb
Northwind
pubs
tempdb
Data Transformation Services
Verwaltung
Replikationsmonitor
Sicherheit
Unterstützungsdienste
Allgemein
Tabellen & Indizes
Reservierter Speicher
pubs
Tabelle
Index
Gruppierter Index
Zeilen
Größe
dbo.authors 23 16K
UPKCL_auidind
aunmind 8K
dbo.discounts 3 8K
dbo.employee 43 16K
employee_ind
PK_emp_id 8K
dbo.jobs 14 16K
PK__jobs__22AA2996
dbo.pub_info 8 24K
UPKCL_pubinfo
dbo.publishers 8 16K
UPKCL_pubind


Vorhandenen Index bearbeiten - EINS
Bearbeiten Sie den Index 'authors_Index_1', der für Tabelle '[dbo].[authors]' in Datenbank 'pubs' erstellt wurde.
Indexname:
authors_Index_1
Spalte
Datentyp
Länge
phone
char
12
au_lname
varchar
40
au_fname
varchar
20
au_id
id
11
address
varchar
40
city
varchar
20
state
char
2
Spaltenreihenfolge ändern:
Nach oben
Nach unten
Indexoptionen
Eindeutige Werte
Index auffüllen
Gruppierter Index
Vorhandenen löschen
Doppelte Werte ignorieren
Füllfaktor:
80
Statistik nicht erneut berechnen (nicht empfohlen)
Dateigruppe:
PRIMARY
SQL bearbeiten...
OK
Abbrechen
Hilfe


SQL Server Enterprise Manager
Es kann nur ein gruppierter Index für Tabelle 'authors' erstellt werden.
Ändern Sie die Gruppierungs-Eigenschaftseinstellung für Index 'UPKCL_auidind', bevor Sie eine zweiten Index erstellen.
OK
Hilfe


Volltextindizierungs-Assistent
SQL Server-Volltextindizierungs-Assistenten beenden
Sie haben die erforderlichen Schritte zum Konfigurieren einer Tabelle für Volltextindizierung beendet. Eine Kurzbeschreibung Ihrer Auswahl wird nachstehend angezeigt.
Datenbank: pubs
Tabelle: [dbo].[titles]
Eindeutiger Schlüssel: UPKCL_titleidind
Spalten:
notes
title
< Zurück
Fertig stellen
Abbrechen


SQL Server Enterprise Manager - [1:Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Wind...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
1 Element
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
master
model
msdb
Northwind
pubs
Diagramm
Tabellen
Sichten
Gespeich
DB-Benut
Rollen
Regeln
Standard
Benutzer
Volltextka
Name
Speicherort
Status
Zuletzt aufgefüllt am
Deskriptoren
im Leerlauf
Neuer Volltextkatalog...
Kataloge neu erstellen
Auffüllen beenden
Auffüllen starten
Terminpläne...
Löschen
Eigenschaften
Hilfedatei anzeigen
Vollständiges Auffüllen
Inkrementelles Auffüllen

Kapitel 15 Cursor

15.1 Freie Auswahl

Wenn Sie mit einer Textverarbeitung einen Brief schreiben, haben Sie den gesamten Text oder zumindest einen größeren Ausschnitt vor sich und können mit dem Cursor von Zeile zu Zeile gehen, um bestimmte Textstellen gezielt anzusteuern und zu bearbeiten. In Datenbanken bietet der Datenbankcursor ein vergleichbares Instrument.

Die Ergebnismenge einer Abfrage kennt normalerweise kein Konzept einer »nächsten Zeile«, und einzelne Zeilen der Ergebnismenge lassen sich auch nicht individuell verarbeiten. Bei Abfragen werden die Zeilen der Ergebnismenge an den Client in der Reihenfolge übermittelt, in der sie in der Ergebnismenge vorkommen. Die Anwendung muß die Zeilen in dieser Reihenfolge und die Ergebnismenge als Einheit verarbeiten. Nachdem eine SQL-Anweisung über eine Verbindung zur Datenbank ausgelöst wurde, bleibt der Anwendung nichts weiter übrig, als alle Zeilen abzurufen, bis das Ende der Abfragemenge erreicht ist. Alternativ kann die Anwendung die Abfrage abbrechen und damit auf die restlichen Datensätze verzichten.

Unter Umständen ist eine Anwendung aber darauf angewiesen, bestimmte Zeilen der Ergebnismenge in ausgewählten Zeilen auszuwerten. Dazu müssen sich Zeilen (oder einige aufeinanderfolgende Zeilen) der Ergebnismenge in Variablen der Anwendung übertragen lassen. Insbesondere trifft das auf interaktive Anwendungen zu, die Datensätze online verarbeiten.

Cursor unterstützen eine derartige Arbeitsweise, indem sie die gesamte Ergebnismenge offenlegen. Im einzelnen bieten Cursor die Möglichkeit,

- auf bestimmte Zeilen der Ergebnismenge zu positionieren,
- eine Zeile oder einen Block von Zeilen relativ zur aktuellen Position in der Ergebnismenge abzurufen,
- die Daten im Zeilenbereich der aktuellen Cursorposition zu ändern,
- verschiedene Ebenen der Sichtbarkeit gegenüber Änderungen, die von anderen Benutzern an den Datenbankdaten vorgenommen werden und die sich in der Ergebnismenge widerspiegeln, einzuführen.

Wenn man mit einem Cursor arbeiten will, sind bestimmte Schritte zu befolgen:

1. Als erstes müssen Sie den Cursor erzeugen.
2. Anschließend öffnen Sie den Cursor, um in einer gespeicherten Prozedur oder Anwendung darauf zugreifen zu können.
3. Schließlich können Sie die Daten zeilenweise lesen und Änderungen an der aktuellen Cursorposition vornehmen.
4. Wenn Sie die Arbeiten am Cursor beendet haben, schließen Sie den Cursor.

5. Als letzter Schritt ist der Cursor freizugeben, um ihn vollständig zu verwerfen.

Cursor sind keine beständigen Datenbankobjekte, da es sich im Grunde nur um eine spezielle Form einer Ergebnismenge handelt.

15.2 Cursorarten

SQL Server unterstützt die folgenden drei Cursorarten:

- Transact-SQL-Cursor
- API-Servercursor
- Clientcursor

Die als API-Servercursor und Clientcursor realisierten Cursorarten sind für Anwendungsentwickler von Bedeutung, die über entsprechende Datenbankschnittstellen (OLE DB, ODBC, ADO) auf SQL Server zugreifen. Die Bezeichnungen verraten bereits, wo diese Cursorarten implementiert werden: API-Servercursor auf dem Server, Clientcursor auf dem jeweiligen Client, der die Anwendung ausführt.

Im Rahmen dieses Buches geht es ausschließlich um Transact-SQL-Cursor, im folgenden einfach als Cursor bezeichnet. Die Implementierung auf dem Server bietet mehrere Vorteile. Unter anderem ist die Netzwerkbelastung in der Regel geringer, und es steht die volle Palette der Cursoroperationen für das Abrufen und die Aktualisierung von Daten zur Verfügung.

Cursor setzt man vor allem in Transact-SQL-Skripts, gespeicherten Prozeduren und Triggern ein.

15.3 Cursor erzeugen (DECLARE)

Wie eine SELECT-Anweisung gibt auch ein Cursor eine Ergebnismenge zurück, nur in einer speziellen Form. Das drückt sich auch in der Syntax der Anweisung DECLARE CURSOR aus, mit der man einen Cursor erzeugt. Die einfachste Syntaxform zeigt deutlich, daß es sich um eine erweiterte SELECT-Anweisung handelt:

```
DECLARE Cursorname CURSOR
FOR Selectanweisung
```

SQL Server unterstützt sowohl die Syntax nach dem Standard SQL-92 als auch eine erweiterte Transact-SQL-Syntax.

Syntax gemäß SQL-92:

```
DECLARE Cursorname [INSENSITIVE] [SCROLL] CURSOR
FOR Auswahlanweisung
[FOR {READ ONLY | UPDATE [OF Spaltenliste]}]
```

Erweiterte Transact-SQL-Syntax:

```
DECLARE Cursorname CURSOR
[LOCAL | GLOBAL]
[FORWARD ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC | FAST_FORWARD]
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
[TYPE_WARNING]
FOR Auswahlanweisung
[FOR UPDATE [OF Spaltenliste]]
```

Die für beide Syntaxformen zutreffenden Argumente lauten:

Cursorname gibt entsprechend den Regeln für Bezeichner den Namen an, den der Cursor erhalten soll.

Die *Auswahlanweisung* definiert in Form einer SELECT-Anweisung die Ergebnismenge für den Cursor.

Option	Beschreibung
Cursorname	Der Name des Cursors (muß den Regeln für Bezeichner entsprechen).
`INSENSITIVE`	Änderungen an den zugrundeliegenden Tabellen spiegeln sich nicht im Cursor wider. Bei dieser Option sind keine Aktualisierungen des Cursors erlaubt.
`SCROLL`	Definiert einen bildlauffähigen Cursor, d.h., es sind die FETCH-Operationen PRIOR, FIRST, LAST, ABSOLUTE n und RELATIVE n möglich.
Auswahlanweisung	Eine SELECT-Anweisung. Die Klauseln COMPUTE, COMPUTE BY, FOR BROWSE und INTO sind innerhalb der Auswahlanweisung nicht zulässig.
`READ ONLY`	Unterbindet Aktualisierungen über diesen Cursor.
UPDATE [OF *Spaltenliste*]	Spezifiziert aktualisierbare Spalten innerhalb des Cursors. Fehlt die optionale Klausel OF *Spaltenliste*, lassen sich alle Spalten aktualisieren.

Tabelle 15.1: Optionen der SQL-92-Syntax

Option	Beschreibung
`LOCAL`	Der Gültigkeitsbereich des Cursors ist lokal bezüglich des Stapels, der gespeicherten Prozedur oder des Triggers, wo er erstellt wurde.

GLOBAL	Der Cursor ist für die gesamte Verbindung gültig.
FORWARD_ONLY	Definiert einen Vorwärtscursor. Die Cursordaten lassen sich nur mit dem Befehl FETCH NEXT abrufen.
STATIC	Der Cursor erlaubt keine Änderungen, und Änderungen an den zugrundeliegenden Tabellen spiegeln sich nicht im Cursor wider.
KEYSET	Mitgliedschaft und Reihenfolge der Zeilen im Cursor liegen fest, wenn man den Cursor öffnet. Der Cursor verwaltet nur die Schlüssel auf Basistabellen, die der Cursor verwendet.
DYNAMIC	Änderungen an den Basistabellen spiegeln sich im Cursor wider.
FAST_FORWARD	Definiert einen FORWARD_ONLY-, READ_ONLY-Cursor. Bei dieser Cursoroption ist die Leistung optimiert.
SCROLL_LOCKS	Beim Einlesen der Daten in den Cursor sperrt SQL Server die Zeilen. Diese Option garantiert, daß Aktualisierungen oder Löschungen an einem Cursor immer erfolgreich verlaufen.
OPTIMISTIC	Aktualisierungen oder Löschungen an einem Cursor können scheitern, wenn sich die zugrundeliegenden Cursordaten seit dem Einlesen in den Cursor geändert haben.
TYPE_WARNING	Der Client erhält eine Warnmeldung, falls der Cursor implizit aus dem angeforderten Typ in einen anderen umgewandelt wird.

Tabelle 15.2: Spezielle Optionen der erweiterten Transact-SQL-Syntax

Die beiden Syntaxformen lassen sich nicht miteinander mischen. Die erweiterte Transact-SQL-Syntax ist nicht abwärtskompatibel zu vorherigen Versionen von SQL Server.

Die Anweisung

```
1:  declare lfuenf cursor
2:  scroll
3:  for select Ziehung, z1, z2, z3, z4, z5, z6, zz from ldaten
4:  for update
```

deklariert den Cursor lfuenf für die Datenbank Lotto. Die SELECT-Anweisung bestimmt, daß die Spalten Ziehung und Z1 bis ZZ aus der Tabelle LDaten abgerufen werden. Die Klausel FOR UPDATE legt fest, daß man den Cursor aktualisieren kann. Die Option SCROLL in Zeile 2 ist erforderlich, wenn man bei der Arbeit mit dem Cursor den Datensatzzeiger auf eine bestimmte Position einstellen will.

15.4 Cursor öffnen (OPEN)

Nachdem der Cursor deklariert ist, öffnet man den Cursor im zweiten Schritt mit der Anweisung:

```
OPEN [GLOBAL] Cursorname
```

Der Parameter *Cursorname* bezeichnet den Namen des in der DECLARE-Anweisung festgelegten Cursors. Die OPEN-Anweisung sollte unmittelbar auf die Deklaration folgen.

Die Option GLOBAL gibt an, daß sich der Cursorname auf einen globalen Cursor bezieht.

Mit der Funktion @@CURSOR_ROWS läßt sich die Anzahl der Zeilen des zuletzt geöffneten Cursors ermitteln.

Den im letzten Beispiel erstellten Cursor lfuenf öffnen Sie mit

```
open lfuenf
```

15.5 Daten vom Cursor abrufen (FETCH)

Nach diesen Vorbereitungen ist es nun möglich, die Daten des Cursors mit der Transact-SQL-Anweisung FETCH zeilenweise abzurufen. Wenn es sich um einen *Vorwärtscursor* handelt, kann man nur von einer Zeile zur nächsten gehen, bis das Ende des Cursors erreicht ist. Besitzt der Cursor dagegen Bildlauffähigkeit, kann man mit den verschiedenen Formen der FETCH-Anweisung an beliebige Positionen im Cursor springen. Die Syntax der FETCH-Anweisung lautet:

```
FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n]
FROM]
[GLOBAL] Cursorname
[INTO Variablenliste]
```

Für einen *bildlauffähigen* Cursor bietet Transact-SQL die in Tabelle 15.3 dargestellten Anweisungen. Ob der Cursor bildlauffähig ist, entscheiden Sie beim Erzeugen des Cursors. Bei einem Vorwärtscursor ist nur die Operation FETCH NEXT zulässig.

Abrufoperation	Beschreibung
`FETCH FIRST`	Ruft die erste Zeile im Cursor ab.
`FETCH NEXT`	Ruft die nächste Zeile im Cursor ab.

FETCH PRIOR	Ruft die Zeile ab, die vor der zuletzt abgerufenen liegt.
FETCH LAST	Ruft die letzte Zeile im Cursor ab.
FETCH ABSOLUTE n	Ruft die Zeile an der Position *n* ab. Positive Werte für *n* beziehen sich auf die Nummer der Zeile von Beginn des Cursors an gerechnet, negative Werte vom Ende an gerechnet. Zum Beispiel ruft FETCH ABSOLUTE -1 die letzte Zeile im Cursor ab.
FETCH RELATIVE n	Gibt die *n*-te Zeile nach der aktuellen Zeile zurück.

Tabelle 15.3: Abrufoperationen für einen Cursor

Nach dem Öffnen des Cursors gilt die aktuelle Position 0.

Cursorname bezeichnet den Namen des Cursors, wie Sie ihn in der DECLARE-Anweisung festgelegt haben.

Die Option GLOBAL bedeutet, daß sich *Cursorname* auf einen globalen Cursor bezieht.

Das folgende Beispiel setzt voraus, daß die Variable @lfd als int deklariert wurde. Die Variable nimmt die absolute Position der laufenden Ziehung auf.

```
1:  set @lfd = -5
2:  fetch absolute @lfd from lfuenf
3:  while @lfd<-1
4:    begin
5:    set @lfd=@lfd+1
6:    fetch next from lfuenf
7:    end
```

Der Cursor lfuenf ruft alle Ziehungen (sprich Zeilen) der Tabelle LDaten ab. Um die letzten fünf Ziehungen anzuzeigen, muß man den Datensatzzeiger auf die fünfte Position vor dem Ende des Cursors setzen. Zeile 1 legt dazu den Wert der Variablen @lfd auf -5 fest.

Zeile 2 ruft den ersten der gewünschten Datensätze mit der Anweisung FETCH ABSOLUTE n aus dem Cursor ab. Die WHILE-Schleife in den Zeilen 3 bis 7 ruft die nächsten Datensätze mit der Anweisung FETCH NEXT aus dem Cursor ab und inkrementiert die Zählvariable @lfd um 1. Das Ende des Cursors und damit die letzte gespeicherte Ziehung ist erreicht, wenn die Variable @lfd auf den Wert -1 hochgezählt wurde.

Die Ergebnismenge sieht folgendermaßen aus:

```
Ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
2256        4    12   19   27   41   45   48
```

```
Ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
2257        4    6    11   17   20   48   23

Ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
2258        18   26   30   32   40   46   33

Ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
2259        10   15   25   27   32   46   49

Ziehung     z1   z2   z3   z4   z5   z6   zz
----------- ---- ---- ---- ---- ---- ---- ----
2260        4    5    18   20   22   41   40
```

Die Anweisung FETCH NEXT in Zeile 6 hätte man auch als

```
6:    fetch absolute @@lfd from lfuenf
```

schreiben können.

Auch die Abbruchbedingung in der WHILE-Schleife läßt sich eleganter formulieren. Darauf geht der Abschnitt »Cursorfunktionen« weiter unten in diesem Kapitel ein.

15.6 Cursor schließen (CLOSE)

Nachdem Sie die Arbeiten mit dem Cursor beendet haben, müssen Sie den Cursor schließen. Das hebt die Zuordnung der Ergebnismenge zum Cursor auf und gibt alle Sperren für die Zeilen frei, auf die der Cursor positioniert ist. Die eigentlichen Daten des Cursors sind noch vorhanden, auch wenn der Zugriff erst wieder möglich ist, wenn Sie den Cursor erneut öffnen. Das bedeutet aber auch, daß die vom Cursor beanspruchten Ressourcen weiterhin belegt sind. Verwenden Sie die CLOSE-Anweisung, wenn Sie den Cursor bald wiederverwenden möchten.

Die Syntax für das Schließen eines Cursors lautet:

```
CLOSE [GLOBAL] Cursorname
```

Die Anweisung CLOSE darf nicht auf einen bereits geschlossenen oder noch nicht geöffneten Cursor angewendet werden. SQL Server liefert in diesem Fall eine Fehlermeldung.

Die Anweisung

```
close lfuenf
```

schließt den Cursor lfuenf.

15.7 Cursor freigeben (DEALLOCATE)

Um die von einem Cursor belegten Ressourcen wieder freizugeben, entfernen Sie mit der Anweisung

```
DEALLOCATE [GLOBAL] Cursorname
```

alle Cursorverweise. Nachdem der letzte Verweis entfernt wurde, gibt SQL Server alle Datenstrukturen, die den Cursor bilden, frei. Mit dieser Anweisung schließt man gleichzeitig den Cursor. Eine Anweisung CLOSE *Cursorname* nach DEALLOCATE führt zu einem Fehler. Verwenden Sie die DEALLOCATE-Anweisung, um die Ressourcen eines Cursors möglichst schnell freizugeben, wenn Sie den Cursor nicht erneut öffnen möchten.

15.8 Daten ändern

Die Daten in der aktuellen Zeile des Cursors kann man ändern, wobei die Änderungen auf die zugrundeliegenden Tabellen durchgreifen. Man spricht hier von positionierten Operationen, da sich das Aktualisieren oder Löschen von Daten auf die momentane Position im Cursor bezieht.

Die für das Aktualisieren und Löschen von Cursorzeilen relevante Syntax der UPDATE- bzw. DELETE-Anweisung sieht folgendermaßen aus:

Syntax:

```
UPDATE Tabellenname
SET Spaltenname1 = {Ausdruck1 | NULL | (Auswahlanweisung)}
[,  Spaltenname2 = {Ausdruck2 | NULL | (Auswahlanweisung)} ...]
WHERE CURRENT OF Cursorname
```

Syntax:

```
DELETE FROM Tabellenname
WHERE CURRENT OF Cursorname
```

Die Klausel WHERE CURRENT OF als das bestimmende Element dieser UPDATE-/DELETE-Anweisungen legt fest, daß sich die Operation auf die aktuelle Position des mit *Cursorname* spezifizierten Cursors bezieht.

Der *Tabellenname* gibt die zu aktualisierende Tabelle an.

In der SET-Klausel legt man die neuen Werte für die angegebene Spalte fest.

Wenn Sie eine Zeile aus dem Cursor löschen, wird diese bei einem aktualisierbaren Cursor auch in der zugrundeliegenden Tabelle gelöscht. Wenn Sie den Cursor erneut öffnen und über FETCH ABSOLUTE n auf die Zeile zugreifen wollen, die Sie vorher gelöscht haben, erhalten Sie statt dessen die nächste Zeile zurück.

15.9 Cursorfunktionen

Mit den Cursorfunktionen kann man den aktuellen Status eines Cursors überprüfen.

@@CURSOR_ROWS

Die Funktion @@CURSOR_ROWS gibt die Anzahl der Zeilen zurück, die sich im momentan geöffneten Cursor befinden.

Syntax:

```
@@CURSOR_ROWS
```

Rückgabetyp:

Die Funktion gibt einen Wert vom Typ integer zurück. Die möglichen Werte faßt Tabelle 15.4 zusammen.

Rückgabewert	Beschreibung
-m	Der Wert m gibt die Anzahl der momentan im Keyset vorhandenen Zeilen an, wenn der Cursor asynchron aufgefüllt wird. Wenn SQL Server bereits 5678 Zeilen in den Cursor abgerufen hat, liefert die Funktion @@CURSOR_ROWS das Ergebnis -5678 zurück.
-1	Dieser Wert kennzeichnet einen dynamischen Cursor. Die Anzahl der Zeilen ändert sich ständig, weil der Cursor alle Änderungen widerspiegelt. Deshalb kennzeichnet der Wert -1, daß die Anzahl der Zeilen unbekannt ist.

0	Der Cursor enthält keine Zeilen, die den angegebenen Bedingungen entsprechen. Die Funktion gibt diesen Wert auch zurück, wenn der zuletzt geöffnete Cursor geschlossen oder freigegeben wurde.
n	Liefert die Gesamtzahl der Zeilen eines vollständig gefüllten Cursors.

Tabelle 15.4: Rückgabewerte der Funktion @@CURSOR_ ROWS

Beispiel:

Wenn Sie den in den obigen Beispielen verwendeten Cursor lfuenf geöffnet haben, gibt die Anweisung

```
select @@cursor_rows as Gesamtzahl
```

die Gesamtzahlen der Zeilen im Cursor zurück:

```
Gesamtzahl
-----------
2260
```

@@FETCH_STATUS

Die Funktion @@FETCH_STATUS gibt den Status der zuletzt ausgeführten FETCH-Anweisung zurück. Es ist nicht ersichtlich, auf welchen Cursor sich diese Funktion bezieht, da sie global für alle Cursor einer Verbindung gilt.

Syntax:

```
@@FETCH_STATUS
```

Rückgabetyp:

Die Funktion @@FETCH_STATUS gibt einen Wert vom Typ integer zurück. Tabelle 15.5 listet die möglichen Rückgabewerte auf.

Rückgabewert	Beschreibung
0	Die FETCH-Anweisung war erfolgreich.
-1	Die FETCH-Anweisung ist fehlgeschlagen, oder die abzurufende Zeile lag außerhalb der Ergebnismenge.

-2	Die abzurufende Zeile ist nicht in der Ergebnismenge enthalten.

Tabelle 15.5: Rückgabewerte der Funktion @@FETCH_ STATUS

Beispiel:

Das im Abschnitt zur FETCH-Anweisung weiter oben in diesem Kapitel angeführte Beispiel läßt sich mit der Funktion @@FETCH_STATUS wie folgt formulieren:

```
set @lfd = -5
fetch absolute @lfd from lfuenf
while @@fetch_status=0
  fetch next from lfuenf
```

Allerdings zeigt die Ausgabe einen Schönheitsfehler. Nachdem die FETCH NEXT-Anweisung in der Schleife die letzte im Cursor vorhandene Zeile (Ziehung) abgerufen hat, liefert @@FETCH_STATUS den Wert 0 (erfolgreiche Ausführung) zurück. Deshalb erfolgt ein weiterer Schleifendurchlauf. Die FETCH NEXT-Anweisung gibt zuerst die Spaltenüberschriften aus, schlägt dann aber fehl, weil eine Zeile außerhalb der Ergebnismenge abgerufen wird. Jetzt erst liefert @@FETCH_STATUS das Ergebnis -1.

15.10 Asynchrones Füllen

Bei großen Keyset-gesteuerten oder statischen Cursor läßt sich eine Leistungsverbesserung erreichen, wenn der Cursor asynchron aufgefüllt wird. Bei kleinen Cursor ist durch den zusätzlichen Aufwand keine Verbesserung zu erwarten. Mit dem Parameter cursor threshold, der sich mit der gespeicherten Prozedur sp_configure einstellen läßt, legt man die Schwelle fest, ab der SQL Server den Cursor asynchron auffüllt.

---

kapitel 16 datenintegrität

16.1 der 30. februar 2001 ...

... ist sicherlich kein datum, für das man einen wichtigen geschäftstermin planen sollte. ein monatliches gehalt von 357000 dm könnte den pförtner eines unternehmens zwar glücklich machen, auf dauer aber zu einigen problemen in der firma führen. diese beispiele sollen lediglich zeigen, daß die eingabe ungültiger, nicht zulässiger oder widersinniger daten durchaus vorkommen kann. datenbankanwendungen oder datenbank-managementsysteme sollten aber in der lage sein, derartige daten zu erkennen und zurückzuweisen. damit sind wir beim thema datenintegrität.

unter *datenintegrität* versteht man die fehlerfreiheit, genauigkeit und zuverlässigkeit - kurz die qualität - von daten. insbesondere ist die datenintegrität bei der veränderung von daten sicherzustellen. und eine veränderung der daten findet bereits statt, wenn der benutzer daten in die datenbank eingibt.

zur datenintegrität gehört auch die einhaltung sogenannter *geschäftsregeln*. eine verletzung dieser regeln stellt keinen fehler im eigentlichen sinne dar, sondern ist als abweichung von der norm im kontext der unternehmensdaten anzusehen.

für die durchsetzung der integrität spielen die schlüsselbeziehungen, die zwischen den einzelnen tabellen einer datenbank bestehen, eine wichtige rolle. die folgenden abschnitte erläutern, was schlüssel sind und welche arten von schlüsseln existieren.

16.2 beziehungen

beziehungen sind das salz in der suppe des relationalen datenbankmodells. es handelt sich dabei um verknüpfungen zwischen tabellen, wobei der fremdschlüssel der einen tabelle auf den primärschlüssel der anderen tabelle verweist. abbildung 16.1 zeigt einen ausschnitt aus dem datenbankdiagramm für die datenbank pubs. die spalten, die den primärschlüssel bilden, sind durch ein schlüsselsymbol gekennzeichnet. zwischen der tabelle publishers und der tabelle titles besteht eine sogenannte 1:n-beziehung, d.h., ein verleger (publisher) läßt sich zu mehreren titeln in beziehung setzen, während ein titel nicht zu mehreren verlegern gehören kann.

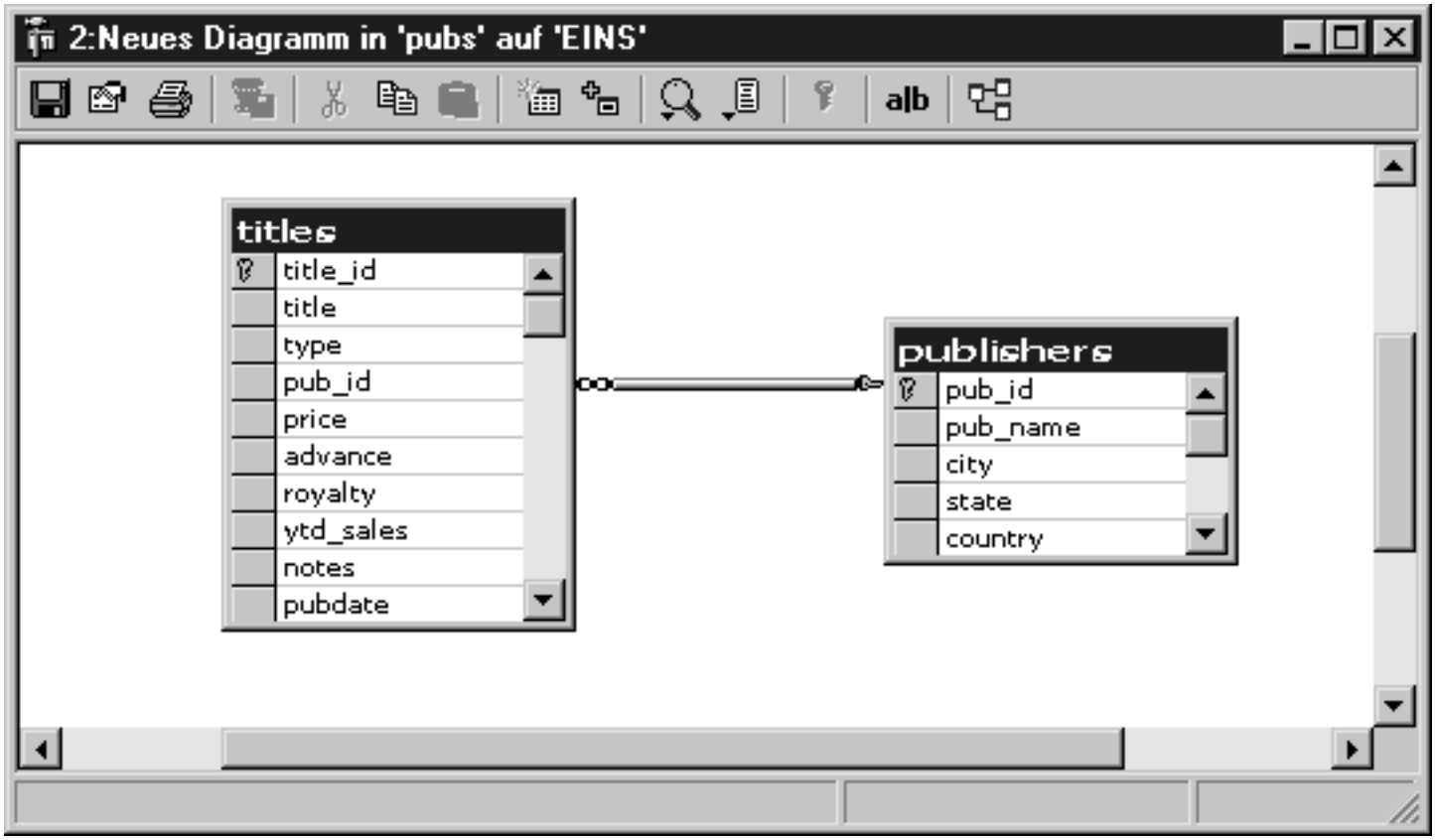


abbildung 16.1: beziehungen zwischen zwei tabellen der datenbank pubs

mit dem einrichten derartiger schlüsselbeziehungen läßt sich erreichen, daß die daten von verknüpften tabellen zu jedem zeitpunkt übereinstimmen. das ist vor allem von bedeutung, wenn sie abfragen mit den join-operatoren (siehe kapitel 10) durchführen. wie sie schlüsselbeziehungen mit hilfe eines datenbankdiagramms erstellen, erläutert der entsprechende abschnitt weiter unten in diesem kapitel nach der behandlung der einschränkungen.

16.2.1 schlüssel

entsprechend der ersten normalform (siehe kapitel 4) erhält jede tabelle einen schlüssel, der die zeilen eindeutig identifiziert. prinzipiell ist ein schlüssel zwar keine pflicht, bei mehreren zueinander in beziehung stehenden tabellen sind schlüssel aber unabdingbar.

kandidatenschlüssel (ersatzschlüssel)

jede spalte oder kombination von spalten, deren werte die zeilen in einer tabelle eindeutig kennzeichnen, kommt für einen schlüssel in frage. es handelt sich sozusagen um kandidaten für einen schlüssel - deshalb auch die bezeichnung *kandidatenschlüssel*. eine tabelle kann mehrere kandidatenschlüssel enthalten. der datenbankadministrator wählt dann einen davon als primärschlüssel aus. ist nur ein kandidatenschlüssel vorhanden, wird er automatisch zum primärschlüssel.

primärschlüssel

ein primärschlüssel (englisch: primary key) ist eine spalte oder kombination von spalten, deren werte eine zeile innerhalb einer tabelle als eindeutig kennzeichnen. spalten, die für einen primärschlüssel

vorgesehen sind, dürfen keine null-werte enthalten.

fremdschlüssel

ein fremdschlüssel (englisch: foreign key) ist eine spalte oder eine kombination aus mehreren spalten, deren werte mit dem primärschlüssel oder eindeutigen schlüssel in derselben oder einer anderen tabelle übereinstimmen. im gegensatz zu primärschlüsseln dürfen in fremdschlüsseln auch null-werte vorkommen.

die als primärschlüssel in der einen tabelle und als fremdschlüssel in der anderen tabelle definierten spalten müssen nicht die gleichen namen tragen. allerdings lassen sich bei gleichen namen die bestehenden beziehungen besser erkennen.

16.3 arten der integrität

wenn man die datenintegrität gewährleisten will, muß man zunächst wissen, wo verletzungen auftreten können und wie man diese unterbinden kann. dabei sind folgende fragen zu beantworten:

- welche arten der integrität gibt es?
- wie legt man integritätsregeln fest?
- wo legt man integritätsregeln fest?
- wie reagiert man auf verletzungen der integritätsregeln?

die datenintegrität läßt sich sowohl in der datenbank selbst als auch in der anwendung, die auf die datenbank zugreift, definieren und durchsetzen. dabei bietet die durchsetzung der integrität auf der seite der datenbank eine reihe von vorteilen. wenn zum beispiel mehrere unterschiedliche anwendungen auf ein und dieselbe datenbank zugreifen und sich änderungen an den geschäftsregeln ergeben, müßten diese änderungen in allen anwendungen programmiert werden. definiert man dagegen die geschäftsregeln in der datenbank selbst, sind die änderungen nur an einer stelle zentral vorzunehmen und von der konkreten anwendung unabhängig. wir beschäftigen uns hier ausschließlich mit der durchsetzung der datenintegrität in der datenbank, d.h. auf der server-seite.

in sql server kann man die datenintegrität

- deklarativ und
- prozedural

definieren bzw. durchsetzen.

nach der ersten methode legen sie die integritätsregeln bereits bei der deklaration der datenbankobjekte (d.h. der tabellen) fest. die regeln gehören dann als fester bestandteil zum objekt. die prozedurale methode stützt sich auf die programmgesteuerte durchsetzung der datenintegrität. für diesen zweck lassen sich spezielle gespeicherte prozeduren - sogenannte trigger - erstellen, die verletzungen der integrität überwachen und geeignete gegenmaßnahmen einleiten können.

wo kommt es überhaupt zu verletzungen? auf die eingabe der daten wurde bereits hingewiesen. damit ist der fall aber noch längst nicht abgeschlossen. in einer relationalen datenbank sind in der regel mehrere tabellen miteinander verknüpft. ein wert, der in einer einzelnen tabelle durchaus in ordnung ist, kann bei

einer verknüpfung zu einer unzulässigen kombination führen. wenn der benutzer daten löscht, kann durch das fehlen bestimmter daten eine verknüpfung zu anderen tabellen ungültig werden.

sql server unterscheidet vier kategorien der integrität:

- entitätsintegrität
- domänenintegrität
- referentielle integrität
- benutzerdefinierte integrität

16.3.1 entitätsintegrität

eine zeile stellt in einer bestimmten tabelle eine entität dar, wenn sich jede zeile in dieser tabelle eindeutig identifizieren läßt. die entitätsintegrität soll diese eindeutigkeit sicherstellen. das kann man mit indizes oder einschränkungen für die bezeichnerspalte(n) bzw. primärschlüssel erreichen (primary key-einschränkung, unique-einschränkung, identity-eigenschaft). es empfiehlt sich, die entitätsintegrität immer auf der untersten ebene durch indizes zu erzwingen. diese indizes können teil von primary key- und unique-einschränkungen sein oder auch unabhängig von einschränkungen erstellt worden sein.

16.3.2 domänenintegrität

die menge aller zulässigen werte in einer spalte bezeichnet man als *domäne*. demgemäß bezieht sich die domänenintegrität auf die einhaltung der zulässigen werte in einer bestimmten spalte einer tabelle. in den bereich der domänenintegrität fällt auch die null-zulässigkeit. mittel zur durchsetzung der domänenintegrität sind default-definitionen (standardwerte), foreign key-einschränkungen, check-einschränkungen und die definition von spalten als not null. für die domänenintegrität empfiehlt sich die verwendung von check-einschränkungen.

16.3.3 referentielle integrität

die integrität der beziehungen zwischen datenzeilen über mehrere tabellen hinweg läßt sich mit hilfe der referentiellen integrität durchsetzen. in sql server basiert die durchsetzung der referentiellen integrität auf beziehungen zwischen primär- und fremdschlüsseln bzw. zwischen fremdschlüsseln und eindeutigen schlüsseln. als instrument zur durchsetzung der referentiellen integrität kommen foreign key-einschränkungen und check-einschränkungen in frage, wobei foreign key-einschränkungen zu bevorzugen sind.

16.3.4 benutzerdefinierte integrität

zur benutzerdefinierten integrität zählt sql server sogenannte geschäftsregeln, die sich keiner der anderen kategorien zuordnen lassen. prinzipiell können sie geschäftsregeln auch mit den bereits genannten arten der datenintegrität realisieren.

16.4 einschränkungen

microsoft empfiehlt, die datenintegrität in sql server vorzugsweise mit einschränkungen durchzusetzen, d.h. erst in zweiter linie auf trigger (siehe nächstes kapitel), regeln und standardwerte zurückzugreifen. sql server unterscheidet fünf klassen der einschränkungen:

- not null
- check
- primary key
- foreign key
- unique

diese einschränkungen können sie beim erstellen einer tabelle mit create table definieren. die um einschränkungen erweiterte syntax der anweisung create table hat folgendes aussehen:

```
create table [datenbankname.[besitzer].| besitzer.]
tabellenname
(
  { spaltendefinition
  | spaltenname as berechneterspaltenausdruck
  | tabelleneinschränkung
} [, ...n]
)
```

in kapitel 8 wurde die *spaltendefinition* in der form

```
spaltenname datentyp
```

angegeben. die vollständige syntax der *spaltendefinition* unter einbeziehung der einschränkungen lautet:

```
{spaltenname datentyp}
[[default konstanterausdruck]
|[identity [(startwert, inkrement) [not for replication]]]
]
[rowguidcol]
[spalteneinschränkung] [...n]
```

die syntax der *spalteneinschränkung* ist folgendermaßen aufgebaut:

```
[constraint einschränkungsname]
{
  [ null | not null ]
  | [ { primary key | unique }
      [clustered | nonclustered]
      [with fillfactor = füllfaktor]
      [on {dateigruppe | default} ]]
    ]
  | [ [foreign key]
  references reftabelle [ (refspalte) ]
  [not for replication]
  ]
  | check [not for replication]
    (logischerausdruck)
}
```

für die *tabelleneinschränkung* gilt folgende syntax:

```
[constraint einschränkungsname]
{
  [ {primary key | unique}
    [ clusterd | nonclustered]
    { (spaltenliste) }
    [ with fillfactor = füllfaktor ]
    [on {dateigruppe | default} ]
  ]
  | foreign key
    [ (spaltenliste) ]
    references reftabelle [ (refspaltenliste) ]
    [not for replication]
  | check [not for replication]
    (suchbedingungen)
}
```

wie die syntax erkennen läßt, unterscheidet sql server zwischen einschränkungen auf spaltenebene und einschränkungen auf tabellenebene. das ganze geheimnis dieses unterschieds besteht in folgendem:

- *spalteneinschränkungen* sind teil der spaltendefinition und nur für die betreffende spalte gültig.
- *tabelleneinschränkungen* werden unabhängig von den spaltendefinitionen deklariert und können sich auf mehrere spalten beziehen.

die nächsten abschnitte gehen auf die einzelnen einschränkungen und die relevanten syntaxelemente der transact-sql-anweisung create table ein. die beispiele zeigen ausschnitte aus den definitionen der tabellen

für die datenbank pubs. im installationsskript für diese datenbank beginnen die definitionen der tabellen in zeile 68. das kommentierte listing finden sie im anhang b. die in den nachstehenden beispielen angegebenen zeilennummern beziehen sich ebenfalls auf dieses listing.

16.4.1 not null

für die werte einer spalte können sie null-werte explizit zulassen oder verbieten. zum beispiel darf eine als primärschlüssel vorgesehene spalte keine null-werte enthalten. der folgende ausschnitt aus der definition der tabelle publishers zeigt die festlegung der spalte pub_id als not null:

```
96: create table publishers
97: (
98:    pub_id         char(4)           not null
99:
```

16.4.2 check

in sql server 7.0 sind check-einschränkungen gegenüber den regeln zu bevorzugen, um die werte in spalten zu beschränken und damit die domänenintegrität durchzusetzen. außerdem kann eine spalte mehrere check-einschränkungen, jedoch nur eine regel erhalten. ebenfalls aus der tabelle publishers stammt der folgende auszug:

```
102:         check (pub_id in ('1389', '0736', '0877', '1622',
             '1756')
103:            or pub_id like '99[0-9][0-9]'),
```

diese einschränkung bezieht sich noch auf die spalte pub_id und bewirkt, daß der benutzer in die tabelle publishers nur zeilen eintragen kann, die in der spalte pub_id einen wert enthalten, der

- in der liste 1389, 0736, 0877, 1622 und 1756 aufgeführt ist

oder

- dem muster 99*nn* entspricht, wobei für *nn* die ziffernfolgen 00 bis 99 möglich sind.

16.4.3 primary key

die entitätsintegrität läßt sich mit primary key-einschränkungen durchsetzen. außerdem sind primärschlüssel für die referentielle integrität erforderlich, damit der fremdschlüssel der anderen tabelle auf den primärschlüssel verweisen kann. bei primary key-einschränkungen ist automatisch eindeutigkeit gewährleistet. in eine so gekennzeichnete spalte kann der benutzer keine werte eingeben, die zu einem doppelten schlüsselwert führen würden.

wie zu jedem topf ein deckel gehört, hat auch jeder primärschlüssel einen zugehörigen index. da sql

server in einem index maximal 16 spalten erlaubt, kann ein primärschlüssel aus maximal 16 spalten bestehen.

der folgende ausschnitt zeigt die vollständige definition der spalte pub_id in der tabelle publishers:

```
96: create table publishers
97: (
98:    pub_id         char(4)          not null
99:
100:         constraint upkcl_pubind primary key clustered
101:
102:         check (pub_id in ('1389', '0736', '0877', '1622',
             '1756')
103:            or pub_id like '99[0-9][0-9]'),
104:
```

die klausel constraint upkcl_pubind in zeile 100 ist optional. damit legen sie einen namen - hier upkcl_pubind - für die einschränkung fest. wenn sie diese klausel nicht angeben, generiert sql server automatisch einen eindeutigen namen.

die eigentliche definition der spalte pub_id als primärschlüssel erfolgt mit den schlüsselwörtern primary key. das schlüsselwort clustered gibt an, daß ein gruppierter index für die primary key-einschränkung erstellt wird.

16.4.4 foreign key

mit foreign key-einschränkungen legen sie eine spalte oder eine kombination von spalten fest, um sie in beziehung zum primärschlüssel einer anderen tabelle setzen und somit die referentielle integrität durchsetzen zu können.

ein primärschlüssel in der einen tabelle allein genügt nicht, um die referentielle integrität zu erzwingen. sie müssen in der anderen tabelle auch einen fremdschlüssel definieren, um eine beziehung zwischen beiden tabellen herzustellen und damit die durchsetzung der referentiellen integrität zu ermöglichen.

in der tabelle publishers ist die spalte pub_id als primärschlüssel deklariert. wie sie abbildung 16.1 entnehmen können, verweist die tabelle titles auf die tabelle publishers. demzufolge müßte in der tabelle titles eine foreign key-einschränkung definiert sein. der folgende ausschnitt zeigt den relevanten teil dieser einschränkungsdefinition:

```
116:create table titles
117:(
...
128:   pub_id         char(4)              null
129:
130:         references publishers(pub_id),
```

```
131:
```

wo aber stehen die schlüsselwörter foreign key? die angabe von foreign key ist nur erforderlich, wenn sie die einschränkung auf tabellenebene definieren (um mehrere spalten angeben zu können). im beispiel handelt es sich um eine spalteneinschränkung, bei der die schlüsselwörter foreign key entsprechend der oben angegebenen syntax optional sind. zeile 130 deklariert die fremdschlüsselbeziehung zur spalte pub_id der tabelle publishers.

mit foreign key-einschränkungen können sie die referentielle integrität nur für tabellen innerhalb derselben datenbank und auf demselben server realisieren. falls sie die referentielle integrität zu anderen datenbanken gewährleisten wollen, müssen sie auf trigger (siehe kapitel 17) ausweichen.

16.4.5 unique

die eindeutigkeit von spalten, die nicht zum primärschlüssel gehören, läßt sich mit unique-einschränkungen erzwingen. im gegensatz zu primary key-einschränkungen erlauben unique-einschränkungen auch null-werte. außerdem ist es möglich, mehrere unique-einschränkungen für eine tabelle zu definieren. unique-einschränkungen verwendet man, um eindeutigkeit auch bei nicht-primärschlüsselspalten zu erzwingen. der fremdschlüssel einer tabelle kann sowohl auf primärschlüsselspalten als auch spalten mit unique-einschränkung verweisen.

eine als unique deklarierte spalte läßt sich nicht als primärschlüssel oder teil des primärschlüssels verwenden. sql server legt einen eindeutigen index an, um die eindeutigkeit einer unique-spalte sicherzustellen. in der definition können sie festlegen, ob das ein gruppierter (clustered) oder nicht gruppierter (nonclustered) index sein soll. da man einen gruppierten index vorzugsweise für eine primärschlüsselspalte verwendet, definieren sie eine unique-spalte in der regel als nonclustered. wenn sie die art des index nicht explizit angeben, gilt nonclustered als standardeinstellung.

da die datenbank pubs keine unique-spalten enthält, verwendet das folgende beispiel eine fiktive tabelle tabellea, deren spalte sp12 als unique deklariert wird:

```
create table tabellea (
sp11 int not null primary key,
sp12 int unique nonclustered,
sp13 varchar(40))
go
```

16.4.6 einschränkungen im enterprise manager definieren

die genannten einschränkungen können sie auch mit dem enterprise manager erstellen und ändern. die im folgenden beschriebenen schritte beziehen sich zwar auf das bearbeiten einer vorhandenen tabelle, grundsätzlich führen sie aber die gleichen schritte aus, wenn sie eine neue tabelle erstellen und dafür einschränkungen festlegen wollen.

1. erweitern sie die konsolenstruktur bis zur datenbank, zu der die tabelle gehört. klicken sie auf **tabellen**, um alle tabellen im detailausschnitt anzuzeigen.
2. klicken sie mit der rechten maustaste auf die tabelle, für die sie eine einschränkung festlegen oder ändern möchten. wählen sie aus dem kontextmenü den befehl **tabelle bearbeiten**. daraufhin wird die tabelle zur bearbeitung geöffnet (siehe abbildung 16.2).

bild

abbildung 16.2: tabelle publishers zur bearbeitung geöffnet

3. wenn sie check- oder unique-einschränkungen festlegen wollen, gehen sie direkt zu schritt 6 weiter.
4. in der zur bearbeitung geöffneten tabelle können sie die null-zulässigkeit (d.h. die not null-einschränkung) und den primärschlüssel (die primary key-einschränkung) festlegen: schalten sie das kontrollkästchen unter *null zulassen* für den jeweiligen spaltennamen aus, um die not null-einschränkung zu definieren. den primärschlüssel legen sie fest, indem sie die betreffende spalte oder - bei einem zusammengesetzten primärschlüssel - mit gedrückter **ç**-taste alle zu diesem schlüssel gehörenden spalten markieren, mit der rechten maustaste klikken und aus dem kontextmenü den befehl **primärschlüssel festlegen** wählen. sql server prüft, ob sich für die angegebenen spalten ein primärschlüssel erstellen läßt, und kennzeichnet im erfolgsfall die ausgewählten spalten im zeilenmarkierer mit einem schlüsselsymbol. andernfalls erscheint ein meldungsfeld mit hinweisen, warum sich der primärschlüssel nicht erstellen läßt und wie sie abhilfe schaffen können (siehe abbildung 16.3).

bild

abbildung 16.3: beispiel einer fehlermeldung, wenn sich der primärschlüssel nicht erstellen läßt

5. wenn sie lediglich die not null- oder primary key-einschränkung festlegen wollten, können sie jetzt die tabelle schließen und die änderungen speichern. zur festlegung von check- oder unique-einschränkungen geht es mit schritt 6 weiter.
6. klicken sie in der zur bearbeitung geöffneten tabelle mit der rechten maustaste. wählen sie aus dem kontextmenü den befehl **eigenschaften**. daraufhin erscheint das dialogfeld **eigenschaften** (siehe abbildung 16.4), in dem sie bei bedarf auch eine andere tabelle auswählen können. um eine check-einschränkung festzulegen, gehen sie zu schritt 7, für eine unique-einschränkung zu schritt 8.

bild

abbildung 16.4: das dialogfeld eigenschaften für die tabelle publishers

7. eine check-einschränkung definieren sie auf der registerkarte **tabellen**. abbildung 16.4 zeigt eine bereits bestehende einschränkung für die spalte pub_id der tabelle publishers. klicken sie auf **neu**, um eine neue einschränkung festzulegen, und geben sie die entsprechende klausel in das feld **einschränkungsausdruck** ein. im bearbeitungsfeld **einschränkungsname** können sie den vorgegebenen namen bei bedarf überschreiben. das kontrollkästchen **vorhandene daten bei erstellung überprüfen** ist bei einer neuen einschränkung standardmäßig eingeschaltet und sollte eingeschaltet bleiben, da dann die neue einschränkung auch auf bereits in der tabelle vorhandene daten wirkt. andernfalls umgehen sie den mechanismus der integritätsprüfung, was ja sicherlich

nicht ihr ziel ist. das gleiche gilt für das kontrollkästchen **einschränkung für insert und update aktivieren**. wenn sie es ausschalten, deaktivieren sie die integritätsprüfung für neu hinzugefügte daten oder für vorhandene daten, die aktualisiert werden. dagegen können sie das dritte kontrollkästchen für die replikation ausschalten, wenn die integritätsprüfung beim kopieren in eine andere datenbank nicht erforderlich ist. klicken sie auf **schliessen**, um die neue einschränkung in die tabelle zu übernehmen. wenn sie noch eine unique-einschränkung festlegen wollen, geht es bei schritt 8 weiter.

8. um eine unique-einschränkung festzulegen, aktivieren sie die registerkarte **indizes/schlüssel** (siehe abbildung 16.5).

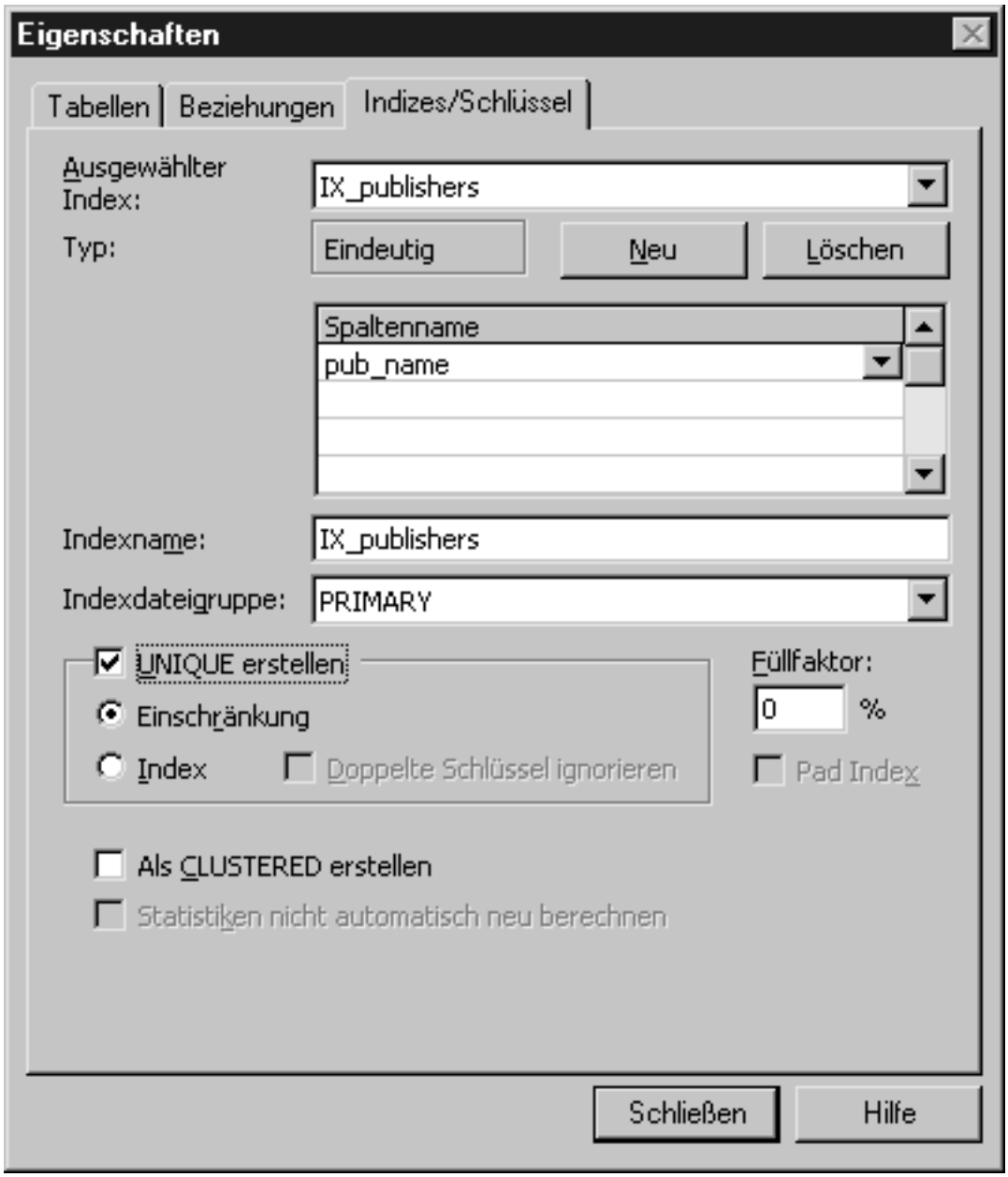


abbildung 16.5: unique-einschränkungen legen sie auf der registerkarte indizes/schlüssel fest

klicken sie auf **neu**. markieren sie in der liste **spaltenname** die spalte(n), die sie in die unique-einschränkung aufnehmen wollen. schalten sie das kontrollkästchen **unique erstellen** ein. auf die optionen clustered und nonclustered , die sie über das kontrollkästchen **als clustered erstellen** steuern, hat bereits der abschnitt zur unique-einschränkung weiter vorn in diesem kapitel hingewiesen. klikken sie

auf **schliessen**, um die einschränkung zu übernehmen.

die auf diese weise definierten einschränkungen werden erst in die tabelle übernommen, wenn sie die tabelle speichern. wollen sie die änderungen verwerfen, schließen sie die tabelle einfach über die schaltfläche **schliessen** in der titelleiste und wählen im bestätigungsdialogfeld die schaltfläche **nein**.

16.4.7 beziehungen im enterprise manager definieren

dieser abschnitt zeigt am beispiel von zwei generischen tabellen tabellea und tabelleb, wie sie beziehungen mit hilfe eines datenbankdiagramms einrichten. die beiden tabellen gehören zur datenbank test. die struktur der tabellen ist aus abbildung 16.6 ersichtlich. um die beispiele nachzuvollziehen, können sie die tabellen im enterprise manager oder mit der folgenden transact-sql-anweisung erstellen:

```
create database test
go
use test
go
create table tabellea (
sp11 int not null,
sp12 int,
sp13 varchar(40))
go

create table tabelleb (
sp21 int,
sp22 int,
sp23 varchar(40)
)
go
```

bild

abbildung 16.6: struktur der beispieltabellen tabellea und tabelleb

die spalte sp11 der tabellea ist als primärschlüssel vorgesehen und deshalb bereits als not null definiert. in tabelleb soll die spalte sp21 als fremdschlüssel fungieren und auf sp11 in tabellea verweisen. um diese beziehung einzurichten, erstellen sie zunächst ein datenbankdiagramm für beide tabellen der datenbank test. wenn sie die anzeigeoption *spalteneigenschaften* wählen, entspricht das diagramm der darstellung in abbildung 16.6. führen sie dann folgende schritte aus:

1. um die geforderte beziehung einrichten zu können, müssen sie als erstes einen primärschlüssel für tabellea festlegen. klicken sie dazu mit der rechten maustaste auf die spalte sp11 von tabellea, und wählen sie aus dem kontextmenü den befehl **primärschlüssel festlegen**. spalte sp11 erhält daraufhin ein schlüsselsymbol als kennzeichnung.
2. jetzt markieren sie die spalte sp21 in tabelleb. drücken sie dann die linke maustaste, halten sie sie

gedrückt, und ziehen sie eine beziehungslinie zur tabellea. wenn sie die maustaste lösen, erscheint das dialogfeld **beziehung erstellen** (siehe abbildung 16.7).

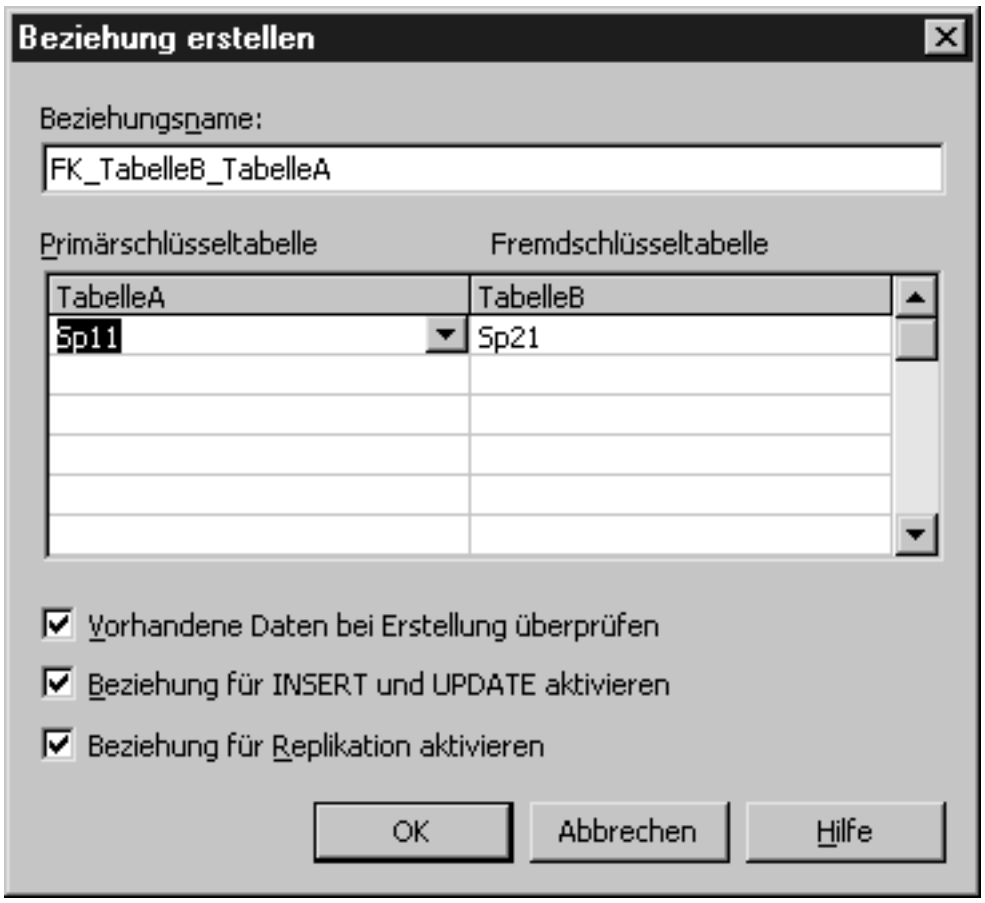


abbildung 16.7: das dialogfeld beziehung erstellen

3. in der liste **primärschlüsseltabelle** ist die spalte sp11 verzeichnet. diese spalte hat sql server automatisch ausgewählt, da sie sie als primärschlüssel festgelegt haben. die liste **fremdschlüsseltabelle** gibt die von ihnen markierte spalte sp21 an. von hier aus haben sie die beziehungslinie zur tabellea gezogen. nachdem die beziehung eingerichtet ist, wird spalte sp21 in tabelleb zum fremdschlüssel. klicken sie auf **ok**, um die beziehung einrichten zu lassen. speichern sie dann das diagramm.
4. das datenbankdiagramm zeigt jetzt eine beziehungslinie zwischen den beiden tabellen an (siehe abbildung 16.8). die fremdschlüsselseite ist mit einem unendlichkeitssymbol gekennzeichnet, die primärschlüsselseite mit einem schlüsselsymbol. wenn sie den mauszeiger auf die beziehungslinie setzen, wird die bezeichnung der beziehung eingeblendet.

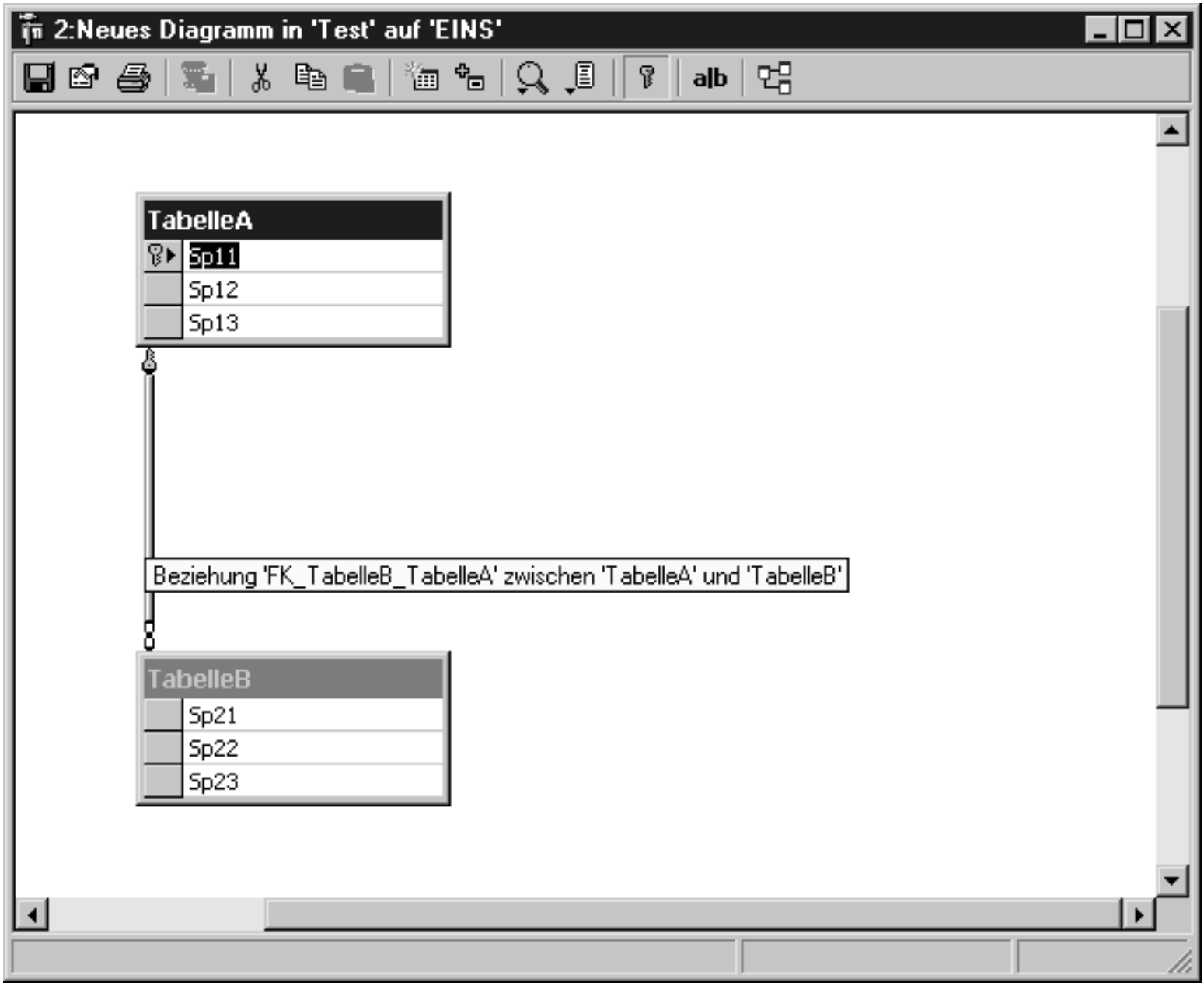


abbildung 16.8: eingerichtete beziehung zwischen tabellea und tabelleb

5. klicken sie mit der rechten maustaste auf die beziehungslinie, und wählen sie aus dem kontextmenü den befehl **eigenschaften**. daraufhin erscheint das bereits aus dem letzten abschnitt bekannte dialogfeld **eigenschaften**. gehen sie auf die registerkarte **beziehungen** (siehe abbildung 16.9). hier sind die in beziehung stehenden spalten aufgeführt. bei bedarf können sie auch den namen der beziehung ändern. klikken sie auf **schliessen**, um das dialogfeld zu schließen. das datenbankdiagramm können sie ebenfalls schließen.

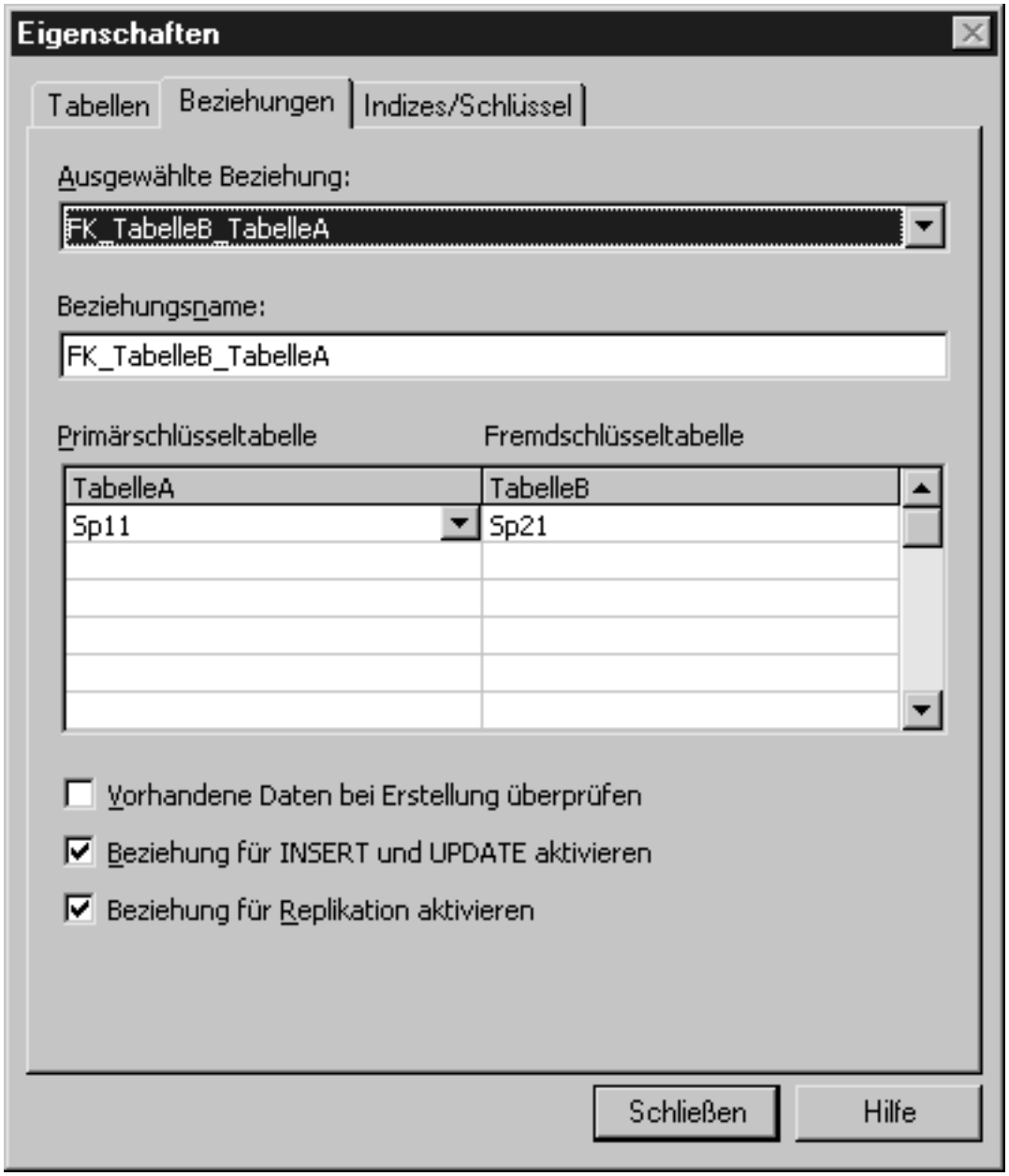


abbildung 16.9: das dialogfeld eigenschaften zeigt auf der registerkarte beziehungen die neu eingerichtete beziehung an

im datenbankdiagramm können sie beziehungen auch wieder löschen. klicken sie dazu mit der rechten maustaste auf die beziehungslinie, und wählen sie aus dem kontextmenü den befehl **beziehung aus datenbank löschen**.

abschließend können sie noch folgendes ausprobieren: löschen sie die datenbank test im enterprise manager oder mit der transact-sql-anweisung

```
drop database test
```

erstellen sie dann die datenbank mit den nachstehenden anweisungen neu. diese anweisungen enthalten bereits die definitionen für den primärschlüssel und die beziehung zwischen tabelleb und tabellea:

```
create database test
go
use test
go
create table tabellea (
sp11 int not null primary key,
sp12 int,
sp13 varchar(40))
go

create table tabelleb (
sp21 int foreign key references tabellea(sp11),
sp22 int,
sp23 varchar(40)
)
go
```

erstellen sie jetzt das datenbankdiagramm im enterprise manager erneut. das ergebnis ist das gleiche. allerdings weist die beziehung jetzt einen anderen namen auf, weil sql server automatisch einen namen mit einem global eindeutigen bezeichner generiert. das ist aber der einzige unterschied. öffnen sie zur kontrolle das dialogfeld **eigenschaften**, und gehen sie auf die registerkarte **beziehungen**. für die primärschlüsseltabelle ist spalte sp11 und für die fremdschlüsseltabelle sp21 angegeben - genau wie beim einrichten der beziehung über das datenbankdiagramm.

16.5 regeln

regeln sind (im gegensatz zu check-einschränkungen) selbständige datenbankobjekte. deshalb vollzieht sich die anwendung von regeln in zwei schritten:

- regel erstellen
- regel an einen benutzerdefinierten datentyp oder eine spalte binden

ein und dieselbe regel können sie zwar an mehrere spalten (bzw. benutzerdefinierte datentypen) binden, eine spalte läßt sich aber nur mit jeweils einer regel versehen. wenn sie mehrere verschiedene einschränkungen für eine spalte definieren wollen, müssen sie auf die weiter unten beschriebenen mechanismen (check-einschränkungen) ausweichen.

sql server stellt regeln aus gründen der abwärtskompatibilität bereit. vorzugsweise sollten sie auf check-einschränkungen zurückgreifen, um die werte in einer spalte zu beschränken.

regeln können sie sowohl mit dem enterprise manager als auch per transact-sql erstellen, ändern, löschen, an spalten binden oder die bindung aufheben. wenn sie mit dem enterprise manager arbeiten, erweitern sie die konsolenstruktur bis zur gewünschten datenbank. im zweig der betreffenden datenbank finden sie den eintrag **regeln** (siehe abbildung 16.10).

bild

abbildung 16.10: der eintrag regeln im enterprise manager für die datenbank lotto

16.5.1 regeln erstellen

mit dem enterprise manager erstellen sie eine neue regel in folgenden schritten:

1. klicken sie in der konsolenstruktur mit der rechten maustaste auf den eintrag **regeln** der datenbank, für die sie die regel hinzufügen möchten. wählen sie aus dem kontextmenü den befehl **neue regel**. es erscheint das dialogfeld **regeleigenschaften** (siehe abbildung 16.11).

abbildung 16.11: das dialogfeld regeleigenschaften

2. im eingabefeld **name** legen sie einen namen für die neue regel fest, beispielsweise zahlenbereich. die regel selbst formulieren sie im mehrzeiligen eingabefeld **text**:

```
@range >= 1 and @range <= 49
```

3. die zulässigen werte für spalten, an die die regel zahlenbereich gebunden ist, dürfen also zwischen 1 und 49 (jeweils inklusive) liegen.
4. bestätigen sie die eingegebene regel mit **ok**, damit sie sql server auf syntaxfehler untersuchen und »verinnerlichen« kann.

die regel können sie auch mit der transact-sql-anweisung create rule erstellen. diese anweisung hat folgende syntax:

```
create rule regelname as bedingungsausdruck
```

als *regelname* geben sie den namen der neu zu erstellenden regel an. im *bedingungsausdruck* können sie alle ausdrücke verwenden, die in einer where-klausel zulässig sind. dazu gehören operatoren und prädikate (beispielsweise in, between). verweise auf spalten oder andere datenbankobjekte sind nicht erlaubt. im bedingungsausdruck ist eine lokale variable enthalten. der name kann beliebig sein, muß allerdings mit einem at-zeichen beginnen.

das obige beispiel für eine regel sieht als transact-sql-anweisung folgendermaßen aus:

```
create rule zahlenbereich as @range >=1 and @range <=49
```

im nächsten schritt binden sie die regel an einen benutzerdefinierten datentyp oder eine spalte.

16.5.2 regeln binden

damit eine regel wirksam wird, müssen sie sie mit einem datenbankobjekt - d.h. mit einem benutzerdefinierten datentyp oder einer spalte - verbinden.

mit dem enterprise manager binden sie eine regel in folgenden schritten an eine oder mehrere spalten:

1. klicken sie in der konsolenstruktur auf den eintrag **regeln**. im rechten fensterbereich des enterprise managers erscheint daraufhin eine liste der vorhandenen regeln. doppelklicken sie auf die regel zahlenbereich. es erscheint wieder das dialogfeld **regeleigenschaften**, in dem jetzt die schaltflächen **udts binden** und **spalten binden** aktiviert sind. im beispiel wollen wir die regel an die spalten z1 bis zz der tabelle ldaten der datenbank lotto binden. klicken sie also auf die schaltfläche **spalten binden**. daraufhin erscheint das dialogfeld **regel an spalten binden** (siehe abbildung 16.12).

abbildung 16.12: das dialogfeld regel an spalte binden

2. im kombinationsfeld **tabelle** wählen sie die gewünschte tabelle aus, falls diese nicht bereits vorgegeben ist. die beiden listenfelder im unteren teil des dialogfelds zeigen die nicht gebundenen sowie die gebunden spalten der ausgewählten tabelle. markieren sie einen eintrag (d.h. eine spalte) im linken listenfeld, und klicken sie auf die schaltfläche **hinzufügen**, um die spalte in das rechte listenfeld zu übernehmen. die regel wird an die im rechten listenfeld aufgeführten spalten gebunden. wenn sie die bindung der regel für eine bestimmte spalte aufheben möchten, markieren sie die spalte im rechten listenfeld und klicken auf **entfernen**. mit den üblichen verfahren zur mehrfachauswahl können sie auch mehrere spalten auf einmal markieren und hinzufügen bzw. entfernen (wie in abbildung 16.12 für die spalten z1 bis zz gezeigt).
3. wenn sie spalten zwischen den listenfeldern verschoben haben, ist die schaltfläche **übernehmen** am unteren rand des dialogfelds aktiviert. klicken sie darauf, um die bindung zu übernehmen. durch klicken auf **ok** gelangen sie wieder zum dialogfeld **regeleigenschaften**. klicken sie auch hier erst auf **übernehmen** und dann auf **ok**, um die regeln an die gewählten spalten zu binden.

wenn sie im schritt 1 der obigen anweisungsfolge auf die schaltfläche **udts binden** klicken, gelangen sie zum dialogfeld **regel an benutzerdefinierte datentypen binden** (siehe abbildung 16.13).

abbildung 16.13: das dialogfeld regel an benutzerdefinierte datentypen binden

haben sie zum beispiel zahltyp (datentyp tinyint, keine null-werte) als benutzerdefinierten datentyp festgelegt und wollen die regel des letzten beispiels an diesen datentyp binden, schalten sie das kontrollkästchen **binden** in der zeile für diesen datentyp ein. die spalte **nur zukünftig** hat die bedeutung des parameters *futureonly_flag* (siehe nachstehende beschreibung der transact-sql-anweisung).

per transact-sql lassen sich regeln nicht so ohne weiteres (d.h. mit einer einfachen anweisung in der art von select) binden. sql server stellt deshalb die gespeicherte prozedur sp_bindrule bereit, um eine regel an eine spalte oder an einen benutzerdefinierten datentyp zu binden. die syntax dieser anweisung sieht folgendermaßen aus:

```
sp_bindrule [@rulename=] 'regelname',
[@objectname=] 'objektname'
[, [@futureonly=] 'futureonly_flag']
```

im parameter *regelname* geben sie den namen der regel an, die sie mit der anweisung create rule erstellt haben.

objektname bezeichnet den namen des objekts (tabelle und spalte bzw. benutzerdefinierter datentyp), an das die regel zu binden ist.

futureonly_flag kommt nur in verbindung mit benutzerdefinierten datentypen zur anwendung. dieses attribut verhindert, daß vorhandene spalten eines benutzerdefinierten datentyps die regel erben. wenn *futureonly_flag* gleich null ist (standardwert), bezieht sich die bindung der neuen regel nur auf spalten des benutzerdefinierten datentyps, für die noch keine regel definiert ist oder für die bereits die vorhandene regel zur anwendung kommt.

die regel zahlenbereich binden sie dann wie folgt an die spalte z1:

```
exec sp_bindrule 'zahlenbereich', 'ldaten.z1'
```

16.5.3 bindung von regeln aufheben

wenn sie im enterprise manager das dialogfeld **regel an spalten binden** (wie im letzten abschnitt beschrieben, siehe abbildung 16.12) öffnen, können sie die bindung von regeln an spalten aufheben, indem sie im rechten listenfeld die betreffende(n) spalte(n) markieren und auf **entfernen** klicken.

per transact-sql läßt sich die bindung einer regel an einen benutzerdefinierten datentyp oder eine spalte mit der gespeicherten prozedur sp_unbindrule aufheben:

```
sp_unbindrule [@objname=] 'objektname'
[, [@futureonly=] 'futureonly_flag']
```

der parameter *futureonly_flag* gilt auch hier nur für benutzerdefinierte datentypen. wenn sie dafür futureonly angeben, behalten vorhandene spalten dieses datentyps die regel bei.

16.5.4 regeln bearbeiten

um vorhandene regeln im enterprise manager zu ändern, erweitern sie die konsolenstruktur bis zum eintrag **regeln** der betreffenden datenbank, klicken im rechten fensterausschnitt des enterprise managers auf die zu ändernde regel und wählen **eigenschaften** aus dem kontextmenü. damit gelangen sie zum dialogfeld **regeleigenschaften** (siehe abbildung 16.14).

abbildung 16.14: im dialogfeld regeleigenschaften können sie eine vorhandene regel bearbeiten

bevor sie die definition einer regel ändern können, müssen sie die bindung der regel an spalten oder benutzerdefinierte datentypen aufheben. wenn sie das nicht beachten und versuchen, die regel per enterprise manager im dialogfeld **regeleigenschaften** zu ändern, bestraft sie sql server mit einer fehlermeldung wie in abbildung 16.15.

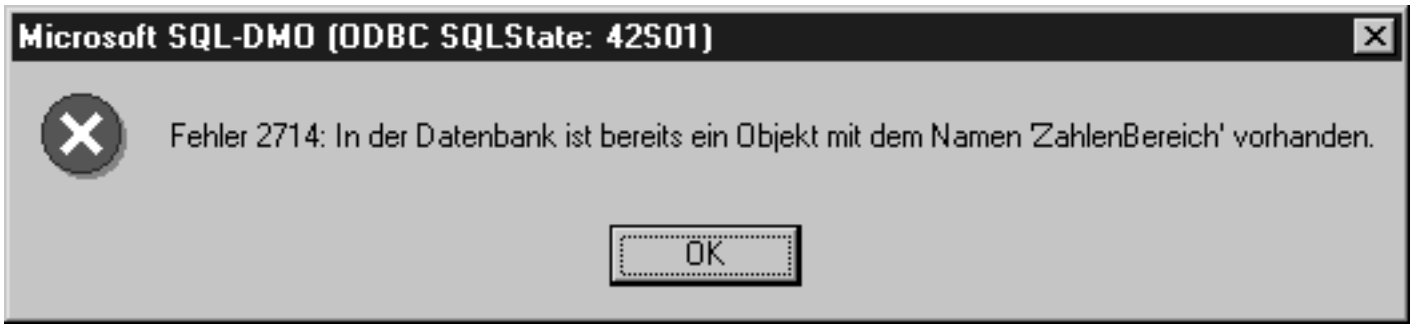


abbildung 16.15: hinweis auf bereits vorhandenes objekt

Bild

abbildung 16.16: fehler beim bearbeiten gebundener regeln

nachdem sie das meldungsfeld bestätigt haben, erscheint eine weitere fehlermeldung (siehe abbildung 16.16).

das dialogfeld **regeleigenschaften** können sie nur noch über die schaltfläche **abbrechen** verlassen. nun

aber werden sie stutzen: die regel ist nicht mehr im rechten fensterbereich des enterprise managers zu sehen. keine angst, die regel ist nicht verschwunden. markieren sie **regeln** in der konsolenstruktur, wählen sie aus dem menü **vorgang** den befehl **aktualisieren** - und schon ist die regel wieder zu sehen.

16.5.5 regeln umbenennen

um einer regel einen anderen namen zu geben, erweitern sie die konsolenstruktur im enterprise manager, so daß die regeln im detailbereich erscheinen. klicken sie mit der rechten maustaste auf die gewünschte regel, und wählen sie **umbenennen** aus dem kontextmenü. daraufhin können sie den namen der regel im rechten fensterbereich bearbeiten (umbenennen), wie sie es von objekten des windows-desktop her kennen. bestätigen sie den neuen namen mit **è** und das dialogfeld mit der frage »möchten sie dieses objekt wirklich umbenennen?« durch klicken auf **ja**.

wenn die regel an spalten oder benutzerdefinierte datentypen gebunden ist, zeigt das dialogfeld **regeleigenschaften** (bzw. die darüber erreichbaren dialogfelder **regel an ... binden**) noch nicht die gebundenen spalten bzw. datentypen an. diese informationen erscheinen erst bei der nächsten anmeldung an sql server.

16.6 regelverletzungen

wenn sie regeln oder check-einschränkungen definiert haben, kommt natürlich irgendwann der zeitpunkt, zu dem der benutzer einen wert eingibt, der den festgelegten bedingungen widerspricht. wenn sie zum beispiel die regel zahlenbereich an die spalte z1 der tabelle ldaten in der datenbank lotto gebunden haben und die anweisung

```
insert into ldaten (ziehung,z1,z2,z3,z4,z5,z6,zz)
                   values (2261,50,2,3,4,5,6,7)
```

ausführen, erhalten sie die fehlermeldung:

```
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 datenbank 'lotto', tabelle 'ldaten', spalte
'z1' auf.
die anweisung wurde beendet.
```

die zahl 50 für die spalte z1 ist laut regel zahlenbereich zu groß.

sql server beendet automatisch eine anweisung, die eine bedingung verletzt. wenn die anweisung allerdings in einer stapeltransaktion ausgeführt wird, setzt sql server die transaktion trotz des

aufgetretenen fehlers fort. damit die datenintegrität auf diese weise nicht verletzt wird, sollten sie mittels der funktion @@error eine fehlerprüfung durchführen. nach jeder erfolgreichen ausführung einer anweisung setzt sql server die fehlervariable auf 0. wenn bei einer fehlerprüfung die funktion @@error einen wert ungleich 0 zurückgibt, können sie die jeweilige transaktion rückgängig machen. auf transaktionen geht kapitel 18 näher ein.

16.7 standardwerte

mit standardwerten kann man eine spalte mit einem vorgegebenen wert füllen. das kann zum beispiel erforderlich sein, wenn man eine zeile in eine tabelle einfügt, aber der wert für eine bestimmte spalte noch nicht bekannt ist oder nicht bereitgestellt wird. da jede spalte einen wert enthalten muß, bietet sich für noch bekannte spalteninhalte ein standardwert an. das kann der wert null sein, wenn die spalte null-werte zuläßt. in der regel verwendet man aber 0 für numerische spalten und n/v für zeichenfolgenspalten.

16.7.1 standardwerte festlegen

wie bei regeln bzw. einschränkungen gibt es in sql server zwei prinzipielle möglichkeiten, um standardwerte festzulegen. beim ersten - zu bevorzugenden verfahren - legt man standardwerte mit dem schlüsselwort default beim erstellen der tabelle (per create table) fest. damit gehört der standardwert zur definition der tabelle.

beim zweiten verfahren erzeugt man mit der anweisung create default ein standardwertobjekt und bindet es mit der gespeicherten prozedur sp_bindefault an die betreffende(n) spalte(n).

zunächst soll das folgende beispiel zeigen, wie sql server mit einem nicht angegebenen wert verfährt. die anweisungsfolge fügt in die datenbank lotto eine ziehung ein, ohne das datum bereitzustellen:

```
insert into ldaten (ziehung,z1,z2,z3,z4,z5,z6,zz)
values (2262,1,2,3,4,5,6,7)
select ziehung, datum from ldaten
where ziehung = 2262
```

das ergebnis der select-anweisung lautet:

```
ziehung     datum
----------- --------------------------
2262        null
```

da in der definition der tabelle ldaten für die spalte datum lediglich der datentyp datetime angegeben ist, sind null-werte zulässig, und diesen wert setzt sql server als standardwert ein, wenn sie keinen anderen wert für die spalte datum angeben.

wenn sie als eifriger lottospieler jeden samstag die tabelle ldaten auf den neuesten stand bringen, bietet es sich an, die spalte datum mit einem standardwert zu definieren, der das aktuelle datum liefert. die definition der tabelle ldaten sieht dann folgendermaßen aus:

```
use lotto
/* tabelle für die ziehungsergebnisse erstellen */
create table ldaten (
ziehung int not null,
datum   datetime default getdate(), -- standardwert
z1      tinyint,
z2      tinyint,
z3      tinyint,
z4      tinyint,
z5      tinyint,
z6      tinyint,
zz      tinyint )
```

jetzt können sie daten einfügen, ohne das aktuelle datum angeben zu müssen. sql server ersetzt das fehlende datum durch den standardwert - das aktuelle systemdatum. dieses bevorzugte verfahren setzt allerdings voraus, daß sie den standardwert bereits beim erstellen der tabelle definieren.

wenn sie einen standardwert erst im nachhinein festlegen wollen, ohne die tabellendefinition selbst zu verändern, können sie mit der transact-sql-anweisung create default arbeiten:

```
create default standardwert as ausdruck
```

standardwert gibt den namen des standardwerts an, den sie noch an eine spalte binden müssen. der *ausdruck* darf nur konstante werte liefern. spaltennamen oder andere datenbankobjekte dürfen hier nicht auftauchen. allerdings sind neben konstanten auch mathematische ausdrücke und systemfunktionen (wie getdate im obigen beispiel) möglich.

nachdem sie das standardwertobjekt mit create default erstellt haben, binden sie es mit der gespeicherten prozedur sp_bindefault an eine spalte:

```
sp_bindefault [@defname =] 'standardwert',
[@objname =] 'objektname'
[, [@futureonly =] 'futureonly_flag']
```

standardwert bezeichnet den namen des standardwerts, wie sie ihn mit der anweisung create default erstellt haben. als *objektname* geben sie tabelle und spalte im format *tabelle.spalte* oder den

benutzerdefinierten datentyp an.

futureonly_flag kommt nur in verbindung mit benutzerdefinierten datentypen zur anwendung. dieses attribut verhindert, daß vorhandene spalten eines benutzerdefinierten datentyps den standardwert erben. wenn *futureonly_flag* gleich null ist (standardwert), bezieht sich die bindung des neuen standardwerts nur auf spalten des benutzerdefinierten datentyps, für die noch kein standardwert definiert ist oder für die bereits der vorhandene standardwert zur anwendung kommt.

mit create default und sp_bindefault erstellen sie den standardwert für die spalte datum der tabelle ldaten dann wie folgt:

```
use lotto
go
create default datumheute as getdate()
go
exec sp_bindefault datumheute, 'ldaten.datum'
```

die gespeicherte prozedur heißt sp_bindefault und nicht sp_binddefault.

die mit create default und sp_bindefault angelegten standardwerte lassen sich auch im enterprise manager einsehen bzw. ändern. erweitern sie dazu die konsolenstruktur bis zum eintrag **standards** der betreffenden datenbank, klicken sie im rechten fensterausschnitt des enterprise managers mit der rechten maustaste auf den gewünschten standardwert, und wählen sie **eigenschaften** aus dem kontextmenü. es erscheint das dialogfeld **standardeigenschaften** (siehe abbildung 16.17). über die schaltflächen **udts binden** bzw. **spalten binden** gelangen sie zu den entsprechenden dialogfeldern, in denen sie die bindung des standardwerts an benutzerdefinierte datentypen bzw. spalten vornehmen können (siehe abbildung 16.18).

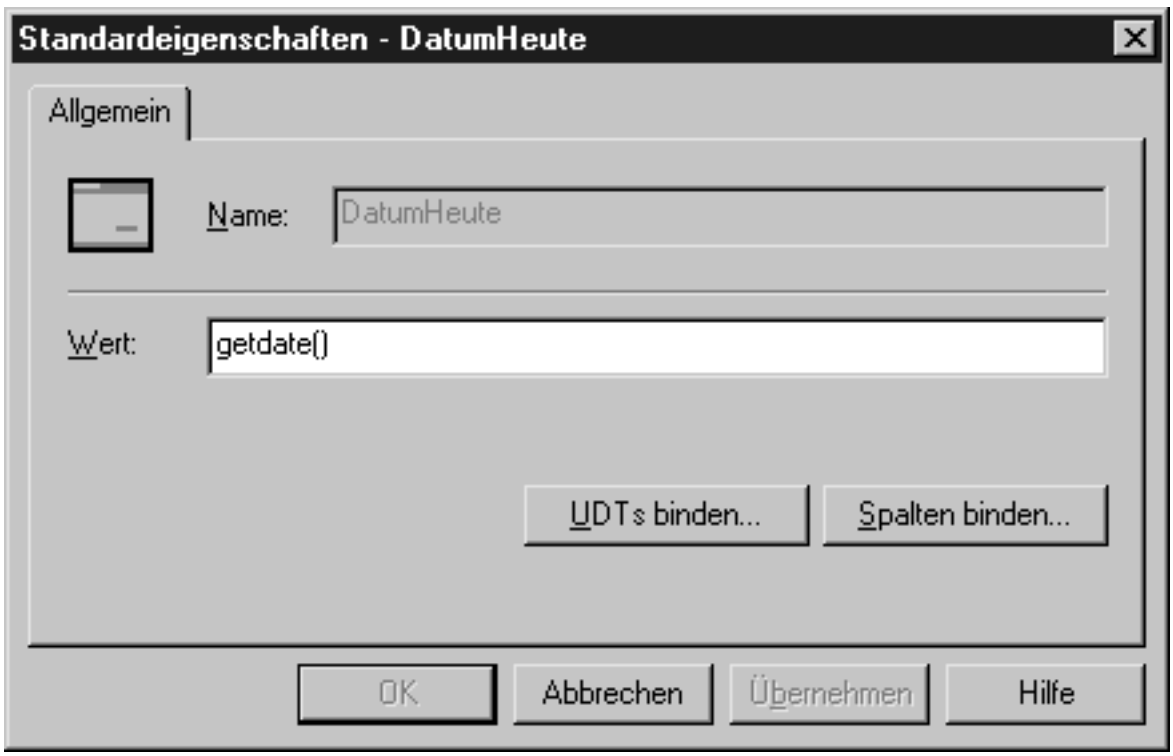


abbildung 16.17: das dialogfeld standardeigenschaften

im enterprise manager können sie auch einen neuen standardwert erstellen. klicken sie dazu in der konsolenstruktur mit der rechten maustaste auf den eintrag **standards**, und wählen sie aus dem kontextmenü den befehl **neuer standard**.

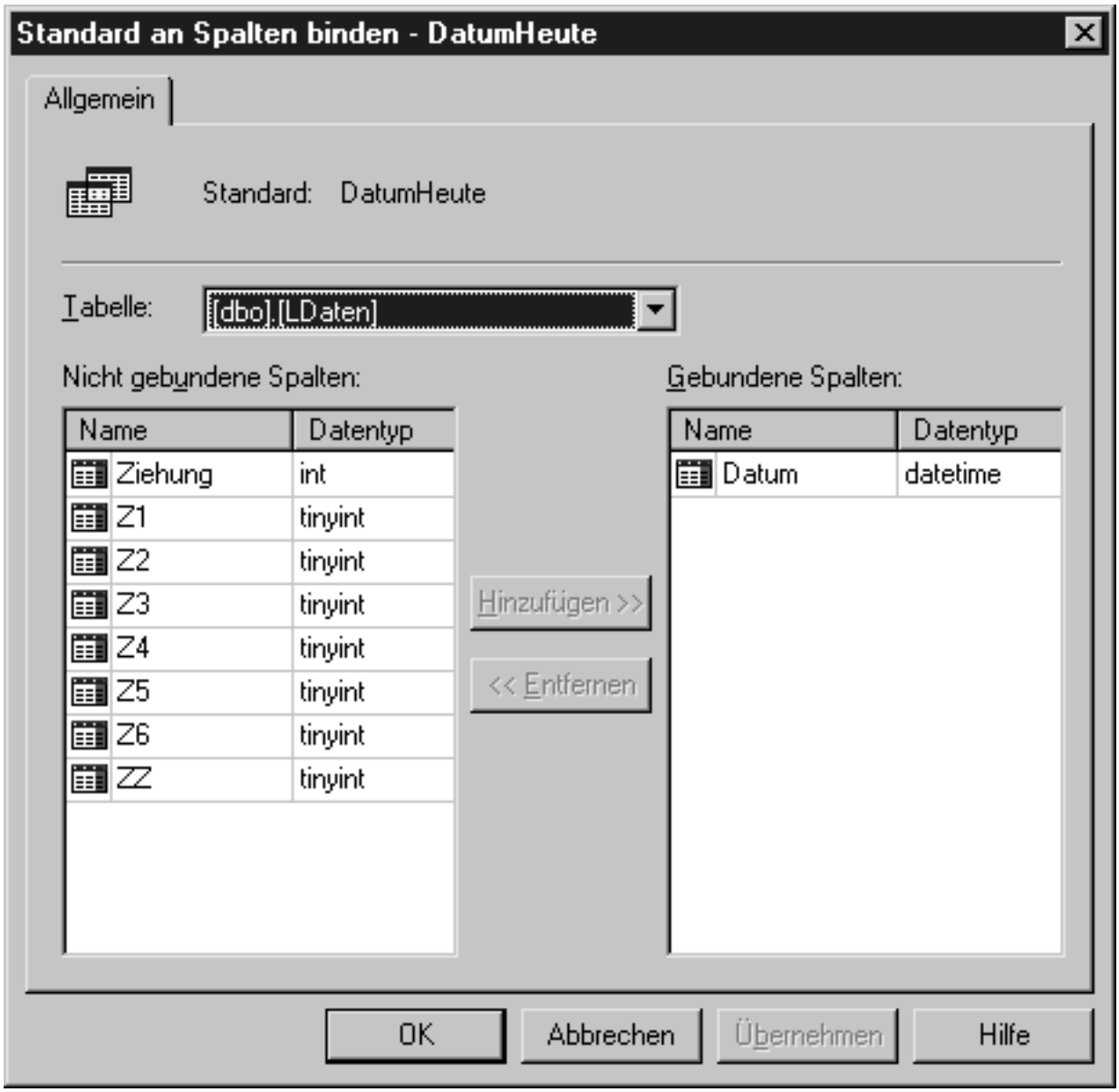
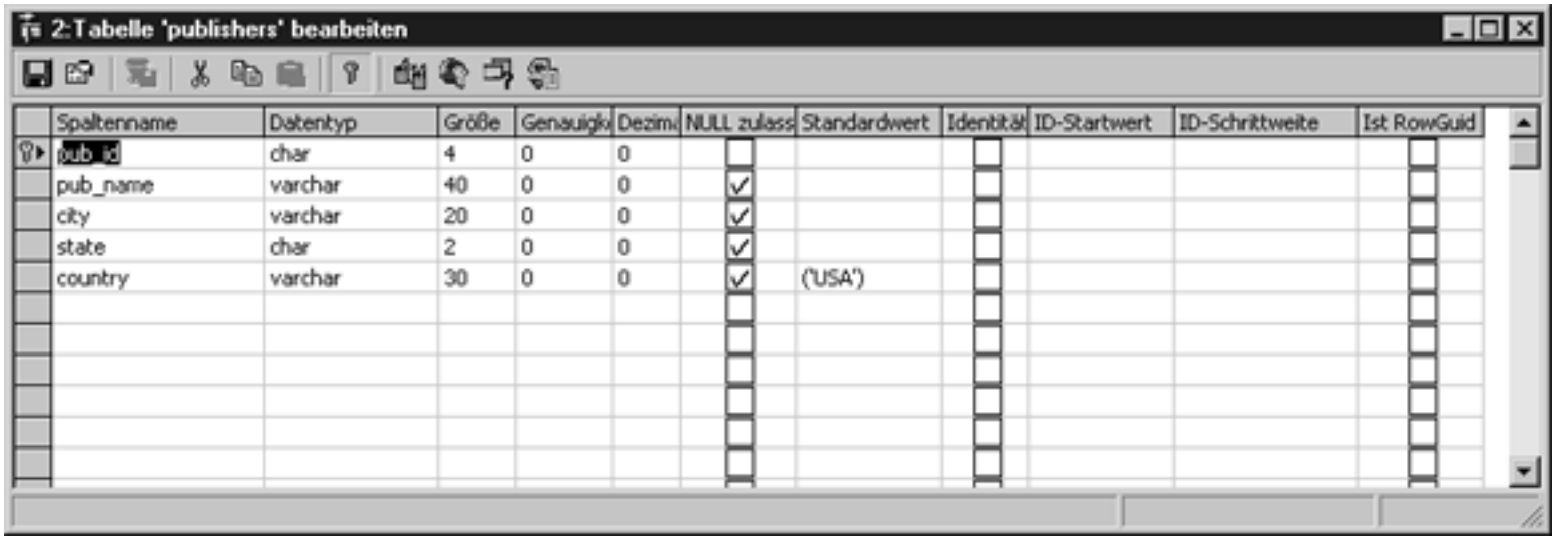
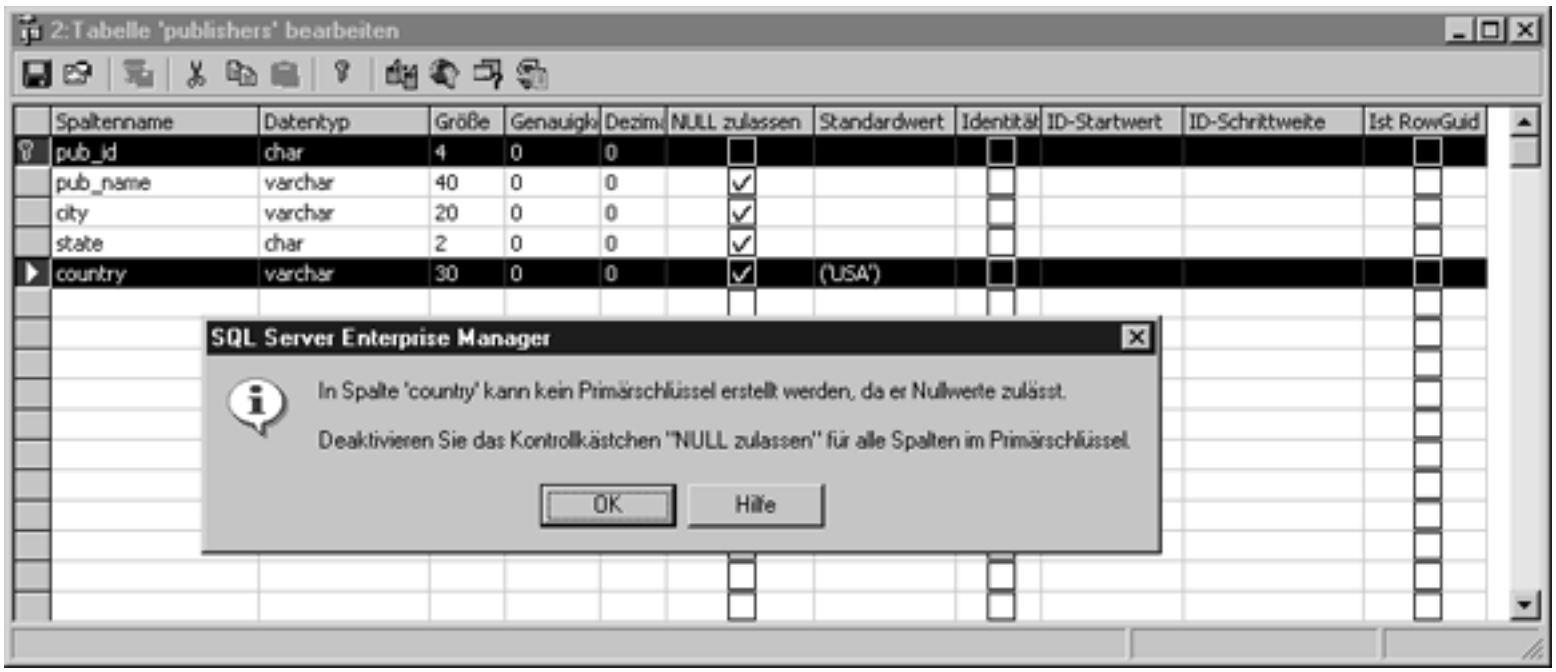
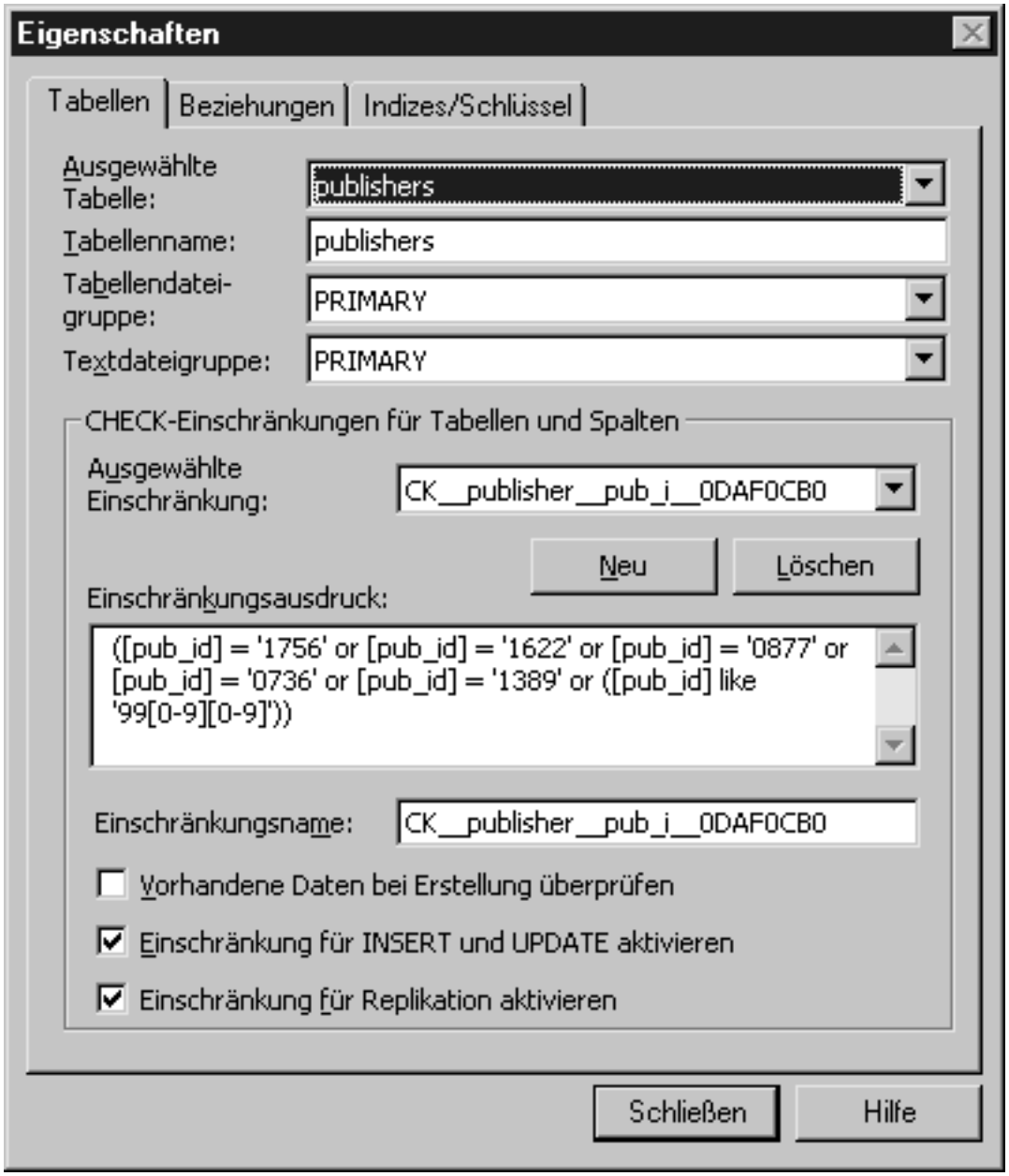


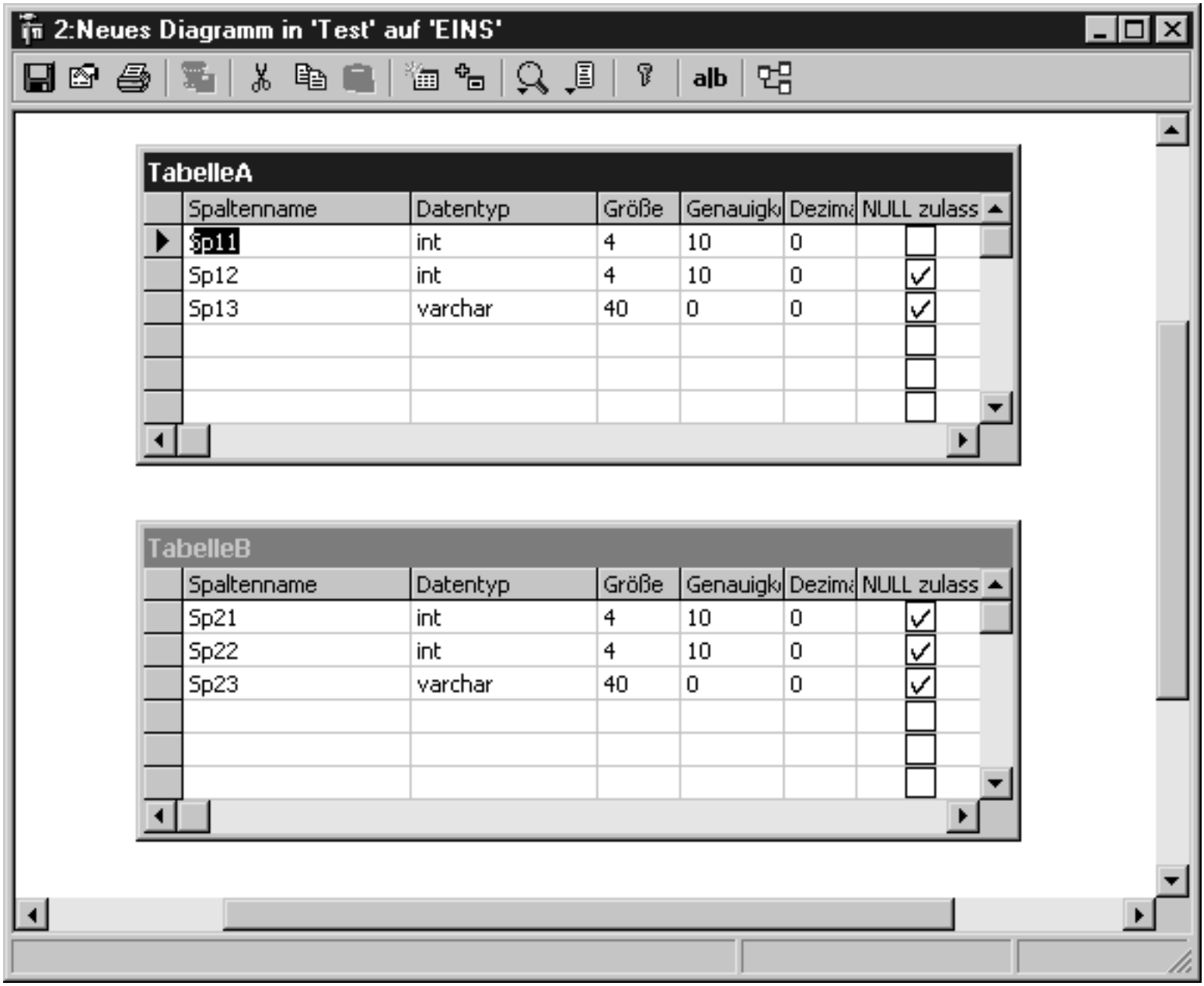
abbildung 16.18: das dialogfeld standard an spalten binden

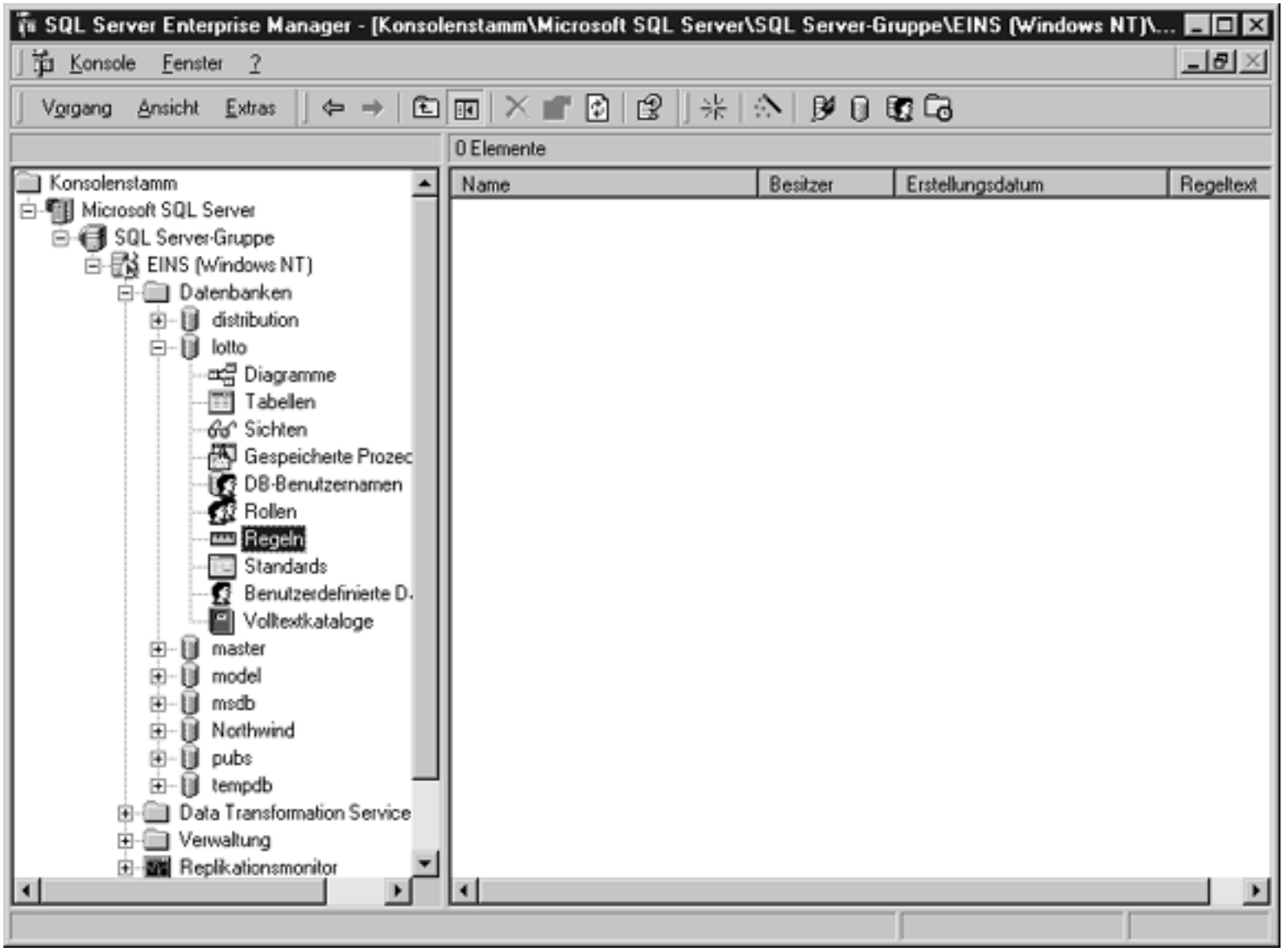


Spaltenname	Datentyp	Größe	Genauigk	Dezim	NULL zulass	Standardwert	Identität	ID-Startwert	ID-Schrittweite	Ist RowGuid
pub_id	char	4	0	0	☐		☐			☐
pub_name	varchar	40	0	0	☑		☐			☐
city	varchar	20	0	0	☑		☐			☐
state	char	2	0	0	☑		☐			☐
country	varchar	30	0	0	☑	('USA')	☐			☐


2: Tabelle 'publishers' bearbeiten
Spaltenname
Datentyp
Größe
Genauigk
Dezim
NULL zulassen
Standardwert
Identität
ID-Startwert
ID-Schrittweite
Ist RowGuid
pub_id
char
4
0
0
pub_name
varchar
40
0
0
city
varchar
20
0
0
state
char
2
0
0
country
varchar
30
0
0
('USA')
SQL Server Enterprise Manager
In Spalte 'country' kann kein Primärschlüssel erstellt werden, da er Nullwerte zulässt.
Deaktivieren Sie das Kontrollkästchen "NULL zulassen" für alle Spalten im Primärschlüssel.
OK
Hilfe


Eigenschaften
Tabellen
Beziehungen
Indizes/Schlüssel
Ausgewählte Tabelle:
publishers
Tabellenname:
publishers
Tabellendatei-gruppe:
PRIMARY
Textdateigruppe:
PRIMARY
CHECK-Einschränkungen für Tabellen und Spalten
Ausgewählte Einschränkung:
CK__publisher__pub_i__0DAF0CB0
Neu
Löschen
Einschränkungsausdruck:
([pub_id] = '1756' or [pub_id] = '1622' or [pub_id] = '0877' or [pub_id] = '0736' or [pub_id] = '1389' or ([pub_id] like '99[0-9][0-9]'))
Einschränkungsname:
CK__publisher__pub_i__0DAF0CB0
Vorhandene Daten bei Erstellung überprüfen
Einschränkung für INSERT und UPDATE aktivieren
Einschränkung für Replikation aktivieren
Schließen
Hilfe


2:Neues Diagramm in 'Test' auf 'EINS'
TabelleA
Spaltenname
Datentyp
Größe
Genauigk
Dezim
NULL zulass
Sp11
int
4
10
0
Sp12
int
4
10
0
Sp13
varchar
40
0
0
TabelleB
Spaltenname
Datentyp
Größe
Genauigk
Dezim
NULL zulass
Sp21
int
4
10
0
Sp22
int
4
10
0
Sp23
varchar
40
0
0


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
0 Elemente
Name
Besitzer
Erstellungsdatum
Regeltext
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
distribution
lotto
Diagramme
Tabellen
Sichten
Gespeicherte Prozec
DB-Benutzernamen
Rollen
Regeln
Standards
Benutzerdefinierte D.
Volltextkataloge
master
model
msdb
Northwind
pubs
tempdb
Data Transformation Service
Verwaltung
Replikationsmonitor


Microsoft SQL-DMO (ODBC SQLState: 42000)
Fehler 3716: Regel 'dbo.ZahlenBereich' kann nicht gelöscht werden, da sie an mindestens eine Spalte gebunden ist.
OK

Kapitel 17 Trigger

17.1 Heimlich, still und leise

Wenn Sie Daten in eine Tabelle einfügen bzw. dort ändern oder löschen, können im Hintergrund Operationen ablaufen, die Ihre Daten überprüfen, eventuell einer Änderung unterziehen oder abhängige Daten in anderen Tabellen modifizieren. Die Rede ist hier von Triggern, die auf eine bestimmte Bedingung hin in Aktion treten. Hauptsächlich setzt man Trigger ein, um Geschäftsregeln innerhalb der Datenbank - d.h. auf der Server-Seite - durchzusetzen. Das entlastet die Anwendungsseite und hat den Vorteil, daß sich erforderliche Änderungen der Geschäftsregeln an einer zentralen Stelle vornehmen lassen.

Die in Kapitel 16 behandelten Mechanismen der Einschränkungen, Bedingungen und Standardwerte können Sie sich als Filter vorstellen, die Ihre Daten genau unter die Lupe nehmen, *bevor* sie in die Datenbank gelangen. Dagegen kann ein Trigger erst dann die Daten manipulieren, nachdem alle derartigen Prüfungen erfolgreich abgeschlossen sind.

17.2 Das Wesen von Triggern

Trigger bedeutet Auslöser. SQL Server bezeichnet damit einen speziellen Typ einer gespeicherten Prozedur, die als Reaktion auf eine der folgenden Operationen ausgelöst - sprich ausgeführt - wird:

- INSERT
- UPDATE
- DELETE

SQL Server führt die in einem Trigger definierten Transact-SQL-Befehle im Sinne einer Transaktion aus, d.h. angefangen von der auslösenden Operation bis zur vollständigen und fehlerfreien Realisierung aller Anweisungen - oder überhaupt nicht.

Um die referentielle Integrität durchzusetzen, sollte man vorzugsweise mit Einschränkungen und Standardwerten arbeiten. Es gibt aber Situationen, in denen die erforderliche Funktionalität dieser Instrumente nicht ausreicht und man auf Trigger zurückgreifen muß. Außerdem haftet der in Kapitel 16 besprochenen deklarativen referentiellen Integrität (DRI) momentan noch der Mangel an, daß sie sich nicht über mehrere Datenbanken hinweg einsetzen läßt. Das kann ein Problem darstellen, wenn Sie in verteilten Anwendungen die Einschränkungen und Datenwerte in anderen Datenbanken und auf anderen Servern prüfen müssen.

17.3 Trigger erstellen

Trigger greifen in die Abläufe innerhalb einer Datenbank ein und ändern damit das Datenbankschema - zum Beispiel die Art und Weise des Tabellenzugriffs oder die Beziehungen anderer Objekte zu der Tabelle, für die ein Trigger definiert ist. Aufgrund der weitreichenden Konsequenzen müssen Sie der Datenbankbesitzer sein oder über die Rechte des Systemadministrators verfügen, wenn Sie einen Trigger erstellen wollen.

Die Tabelle, für die der Trigger ausgeführt wird, heißt auch *Triggertabelle*. Als Besonderheit ist anzumerken, daß man Trigger für Tabellen definieren kann, die noch gar nicht vorhanden sind. Dabei handelt es sich um die sogenannte *verzögerte Namensauflösung*.

Die vereinfachte Syntax der Anweisung CREATE TRIGGER hat folgendes Aussehen:

```
CREATE TRIGGER Triggername
ON Triggertabelle
FOR {[DELETE] [,] [INSERT] [,] [UPDATE]}
[WITH ENCRYPTION]
AS
  SQL_Anweisungen
```

Der *Triggername* muß den Regeln für Bezeichner entsprechen und innerhalb der Datenbank eindeutig sein. Optional können Sie den Besitzer des Triggers angeben.

Triggertabelle bezeichnet die Tabelle, für die der Trigger ausgeführt wird. Auch hier können Sie optional den Besitzer der Tabelle spezifizieren.

Die Schlüsselwörter WITH ENCRYPTION bewirken, daß SQL Server den Text der in der Systemtabelle syscomments gespeicherten Triggerdefinition verschlüsselt. Achten Sie darauf, daß Sie den Originaltext der Definition zur Verfügung haben, damit Sie den Trigger notfalls wiederherstellen können. Aus der Tabelle syscomments können Sie ihn nicht wieder auslesen. Außerdem ist die unverschlüsselte Version erforderlich, wenn Sie eine Versionsumstellung von SQL Server ausführen.

Die Schlüsselwörter DELETE, INSERT und UPDATE spezifizieren die Art der Transact-SQL-Anweisung, die den Trigger auslösen soll. Es ist mindestens eine Option erforderlich. Wenn Sie mehrere Optionen angeben, trennen Sie sie durch Kommas. Die Reihenfolge der Schlüsselwörter spielt keine Rolle.

In SQL Server können Sie für ein und dieselbe INSERT-, UPDATE- oder DELETE-Option auch mehrere Trigger definieren, zum Beispiel zwei Trigger für eine UPDATE-Anweisung.

Als *SQL_Anweisungen* können Sie beliebig viele Transact-SQL-Anweisungen angeben. Allerdings sind folgende Anweisungen nicht zulässig:

- die CREATE-Anweisungen für DATABASE, DEFAULT, INDEX, PROCEDURE, RULE, SCHEMA, TABLE, TRIGGER und VIEW,

- die ALTER-Anweisungen für DATABASE, PROCEDURE, TABLE, TRIGGER und VIEW,
- die DROP-Anweisungen für DATABASE, DEFAULT, INDEX, PROCEDURE, RULE, TABLE, TRIGGER und VIEW,
- die Berechtigungsanweisungen DENY, GRANT und REVOKE,
- die Datenträgeranweisungen DISK INIT, DISK RESIZE,
- die Ladeoperationen für Datenbanken LOAD DATABASE, LOAD LOG,
- die Wiederherstellungsanweisungen RESTORE DATABASE, RESTORE LOG,
- die Anweisung RECONFIGURE,
- die Anweisung UPDATE STATISTICS,
- die Anweisung TRUNCATE TABLE.

Die Anweisung TRUNCATE TABLE entspricht in ihrer Wirkung einer DELETE-Anweisung ohne WHERE-Klausel und löscht somit alle Zeilen einer Tabelle. Da SQL Server einen Trigger nur für protokollierte Operationen auslöst und die Anweisung TRUNCATE TABLE eine nicht protokollierte Aktion darstellt, wird auch kein DELETE-Trigger ausgelöst, wenn Sie eine TRUNCATE TABLE-Anweisung für eine Tabelle ausführen. Die Berechtigung zur Ausführung einer TRUNCATE TABLE-Anweisung hat allerdings nur der Tabellenbesitzer, und diese Berechtigung ist auch nicht übertragbar. Demzufolge kann nur der Tabellenbesitzer mit TRUNCATE TABLE einen DELETE-Trigger unbeabsichtigt umgehen - und gerade der Tabellenbesitzer dürfte sich dieser Tatsache bewußt sein.

Weiterhin sind folgende Punkte zu beachten:

- Trigger lassen sich nur für Tabellen, nicht aber für Sichten erstellen.
- Alle Einstellungen der Umgebung, die Sie in einem Trigger mit SET-Anweisungen vornehmen, sind nur für die Lebensdauer des Triggers wirksam und nehmen ihre alten Werte an, sobald der Trigger die Ausführung beendet hat.
- Die Aktualisierung von Spalten der Datentypen text, ntext und image mit der WRITETEXT-Anweisung löst keinen Trigger aus, wobei es keine Rolle spielt, ob die Aktualisierung protokolliert oder nicht protokolliert abläuft.
- Verzichten Sie in Triggern auf SELECT-Anweisungen, die Ergebnisse zurückgeben. Die Reaktion auf diese Ergebnisse würde in allen Client-Anwendungen, die auf die Triggertabelle zugreifen, eine spezielle Behandlung erforderlich machen.
- Wenn Sie in einem Trigger mit Variablenzuweisungen arbeiten, sehen Sie als erstes die Anweisung SET NOCOUNT ON vor, damit SQL Server keine Meldung bezüglich betroffener Zeilen zum Client sendet.

17.3.1 Die Tabellen inserted und deleted

SQL Server legt zwei spezielle temporäre Tabellen an, wenn eine Anweisung Daten einfügt, aktualisiert oder löscht. Die Tabelle inserted nimmt Kopien der Zeilen auf, die von einer INSERT- oder UPDATE-Anweisung betroffen sind. In der Tabelle deleted verzeichnet SQL Server die alten Daten, auf die sich eine UPDATE- oder DELETE-Anweisung bezieht. Die Struktur der beiden Tabellen entspricht der zugrundeliegenden Triggertabelle. Der Zugriff auf die Tabellen inserted und deleted ist nur innerhalb

des Triggers möglich.

17.3.2 Die Beispieltabellen

Nach diesen umfangreichen Vorbemerkungen zeigen nun die folgenden Beispiele einfache Anwendungsmöglichkeiten für INSERT-, UPDATE- und DELETE-Trigger. Wie in Kapitel 16 beziehen sich die Beispiele auf generische Tabellen, die allerdings um eine Datumsspalte erweitert wurden. Löschen Sie gegebenenfalls die im Verlauf von Kapitel 16 erstellte Datenbank Test, und legen Sie mit der folgenden Anweisung die neue Datenbank Test mit den Tabellen TabelleA und TabelleB an:

```
create table TabelleA (
Sp11 int not null primary key,
Sp12 int,
Sp13 varchar(40),
Sp14 datetime)
go

create table TabelleB (
Sp21 int,
Sp22 int,
Sp23 varchar(40),
Sp24 datetime)
go
```

17.3.3 INSERT- und UPDATE-Trigger

Als erstes erstellen Sie einen Trigger TrigA für die TabelleA. Der Trigger löst aus, wenn Sie in dieser Tabelle Daten modifizieren (einfügen oder aktualisieren):

```
create Trigger TrigA
on tabellea
for insert, update
as
  update tabellea
  set sp14=getdate()
  from tabellea, inserted
  where tabellea.sp11=inserted.sp11
go
```

Wenn Sie jetzt einen Datensatz in die Tabelle einfügen oder Werte aktualisieren, löst SQL Server den Trigger TrigA aus. Der Trigger trägt in Spalte Sp14 das aktuelle Datum für die eingefügte oder

aktualisierte Zeile ein. Damit das Datum nicht in allen bereits vorhandenen Zeilen durch das neue Datum ersetzt wird, muß der Trigger die aktuelle - d.h. die eingefügte oder aktualisierte - Zeile ermitteln. Die entsprechende Suchbedingung ist in der WHERE-Klausel der UPDATE-Anweisung formuliert. Wie bereits erwähnt, löst ein Trigger erst dann aus, wenn SQL Server alle Prüfungen abgeschlossen hat und die Daten in der Tabelle eingetragen sind. Um die eingefügte Zeile zu bestimmen, vergleicht die Suchbedingung in der WHERE-Klausel die Werte von Spalte Sp11 mit dem in diese Spalte eingefügten Wert. Woher bekommt man nun den neuen Wert? An einen Trigger lassen sich im Gegensatz zu einer »richtigen« gespeicherten Prozedur keine Parameter übergeben. Des Rätsels Lösung liefert die Tabelle inserted, in die SQL Server die eingefügten Daten einträgt.

Fügen Sie nach dem Muster der folgenden Anweisung einige Zeilen in die TabelleA ein:

```
insert into tabellea (sp11) values (1011)
```

Führen Sie Anweisungen einzeln und mit jeweils einem anderen Wert in der VALUES-Liste aus, damit Sie sich von der ordnungsgemäßen Aktualisierung des Datums in Spalte Sp14 überzeugen können. Kontrollieren Sie das Resultat mit der Anweisung:

```
select sp11, sp14 from tabellea
```

Die Ergebnismenge sieht zum Beispiel wie folgt aus:

```
sp11        sp14
----------- --------------------------
1003        1999-04-21 12:14:27.760
1009        1999-04-21 12:15:00.527
1010        1999-04-21 12:14:08.350
1011        1999-04-21 12:16:29.783
1023        1999-04-21 12:15:51.590
1051        1999-04-21 12:15:36.890
```

Die Zeilen wurden nacheinander mit den Werten 1010, 1003, 1009, 1051, 1023 und 1011 eingefügt. Die zeitliche Reihenfolge können Sie am besten kontrollieren, wenn Sie die Ergebnismenge mit der Anweisung

```
select sp11, sp14 from tabellea
order by sp14
```

nach der Datumsspalte Sp14 sortieren:

```
sp11        sp14
----------- --------------------------
1010        1999-04-21 12:14:08.350
1003        1999-04-21 12:14:27.760
1009        1999-04-21 12:15:00.527
1051        1999-04-21 12:15:36.890
1023        1999-04-21 12:15:51.590
1011        1999-04-21 12:16:29.783
```

In diesem Beispiel kam es nur darauf an, die Arbeitsweise eines Triggers zu demonstrieren. Das aktuelle Datum können Sie in der Praxis wesentlich einfacher einfügen, wenn Sie für die Spalte Sp14 den Standardwert getdate() definieren:

```
create table TabelleA (
Sp11 int not null primary key,
Sp12 int,
Sp13 varchar(40),
Sp14 datetime DEFAULT getdate())
go
```

17.3.4 Aktualisierungen weitergeben

Nicht umsonst haben Sie zwei Beispieltabellen erzeugt. Trigger eignen sich nämlich hervorragend, um sogenannte *kaskadierte* Aktualisierungen durchzuführen. Wenn der Benutzer in die eine Tabelle Daten eingibt, kann man einen Trigger auslösen, der eine weitere Tabelle aktualisiert.

Löschen Sie zunächst eventuell vorhandene Daten in den beiden Tabellen mit den folgenden Anweisungen:

```
delete from tabellea
delete from tabelleb
```

Ändern Sie die Triggerdefinition mit der folgenden Anweisung (die Zeilennummern gehören nicht zur Definition und wurden nur für die sich anschließenden Erläuterungen eingefügt):

```
1:  alter Trigger TrigA
2:  on tabellea
3:  for insert, update
```

```
4:  as
5:    declare @isp11 int  -- für Wert aus inserted-tabelle
6:    declare @dsp11 int  -- für Wert aus deleted-tabelle
7:    select @isp11 = inserted.sp11 from inserted
8:    select @dsp11 = deleted.sp11 from deleted
9:
10:   raiserror('INS: %d DEL: %d',1,1,@isp11,@dsp11)
11:            with nowait
12:
13:   update tabelleb
14:   set sp21=@isp11
15:   where sp21=@dsp11
16:
17: go
```

Auf die Anweisung ALTER TRIGGER geht der entsprechende Abschnitt weiter unten in diesem Kapitel ein.

Zunächst deklariert der Trigger in den Zeilen 5 und 6 zwei lokale Variablen, die die Werte für die Spalte Sp11 aus der inserted- und der deleted-Tabelle aufnehmen. Zeile 7 ruft den Wert aus der inserted-Tabelle ab und weist ihn der Variablen @isp11 zu, Zeile 8 führt die analoge Zuweisung mit der deleted-Tabelle und der Variablen @dsp11 aus.

Zeile 10 ist ausschließlich für Demonstrationszwecke vorgesehen. Die Anweisung RAISERROR gibt eine benutzerdefinierte Meldung aus, die den Inhalt der Variablen @isp11 und @dsp11 zeigt. Die Klausel WITH NOWAIT in Zeile 11 stellt lediglich sicher, daß die Meldung sofort an den Client geht.

Die Zeilen 13 bis 15 aktualisieren die Spalte Sp21 der TabelleB. Das Beispiel geht davon aus, daß in TabelleB nur Werte stehen, die auch in der TabelleA enthalten sind. Wenn der Benutzer in TabelleA einen Wert ändert und dieser Wert auch in TabelleB vorkommt, soll der Trigger den entsprechenden Wert in TabelleB aktualisieren, damit die referentielle Integrität gewährleistet bleibt.

Fügen Sie nun die folgenden Testdatensätze ein

```
insert into tabellea (sp11) values (1001)
insert into tabellea (sp11) values (1002)
insert into tabellea (sp11) values (1003)
insert into tabellea (sp11) values (1004)
insert into tabellea (sp11) values (1005)

insert into tabelleb (sp21) values (1003)
insert into tabelleb (sp21) values (1005)
```

und fragen Sie die beiden Tabellen zur Kontrolle ab:

```
select sp11 from tabellea
select sp21 from tabelleb
```

Das Ergebnis lautet:

```
sp11
-----------
1001
1002
1003
1004
1005

sp21
-----------
1003
1005
```

In Sp21 von TabelleB dürfen laut Vereinbarung nur Werte enthalten sein, die auch in Sp11 von TabelleA vorkommen, während TabelleA auch Werte enthalten darf, die nicht in TabelleB verzeichnet sind.

Der Wert 1003 ist sowohl in TabelleA als auch in TabelleB enthalten. Wenn Sie jetzt mit der Anweisung

```
update tabellea
set sp11 = 5003
where sp11 = 1003
```

den Wert 1003 in Sp11 von TabelleA in den Wert 5003 ändern, muß sich diese Änderung auch in TabelleB niederschlagen. Nach Ausführung dieser Anweisung zeigt SQL Server folgendes an:

```
Meldung 50000, Ebene 1, Status 50000:
INS: 5003 DEL: 1003
sp11
-----------
1001
1002
1004
1005
5003
```

```
sp21
-----------
5003
1005
```

Aus der vom Trigger generierten Meldung können Sie ablesen, was bei einer UPDATE-Anweisung passiert: Den neuen Wert übernimmt SQL Server in die inserted-Tabelle, der alte Wert aus TabelleA kommt in die deleted-Tabelle. Die Zwischenspeicherung dieser Werte in den lokalen Variablen hat den Vorteil, daß Sie in der UPDATE-Anweisung des Triggers (Zeilen 13 bis 15 der obigen Triggerdefinition) nur eine Tabelle referenzieren müssen und die Anweisungen verständlicher zu formulieren sind.

Die UPDATE-Anweisung des Triggers sucht also die Zeile mit dem alten Wert in TabelleB (siehe die WHERE-Klausel in Zeile 15) und ersetzt ihn durch den neuen Wert (SET-Klausel in Zeile 14).

17.3.5 DELETE-Trigger

Mit Triggern können Sie nicht nur Daten über mehrere Tabellen hinweg aktualisieren, sondern auch löschen. Nehmen wir an, daß Sie den im letzten Abschnitt aktualisierten Wert 5003 aus TabelleA löschen wollen. Um die Datenintegrität weiterhin aufrechtzuerhalten, müssen Sie diesen Wert auch in TabelleB entfernen, da andernfalls ein Wert der TabelleB keine Entsprechung in TabelleA hätte. Dieser Zustand ist aber nach der getroffenen Vereinbarung nicht zulässig.

Wie Sie sicherlich schon vermutet haben, stellen Sie die Datenintegrität mit einem DELETE-Trigger sicher. Die Definition des Triggers lautet:

```
create Trigger TrigADel
on TabelleA
for delete
as
  declare @isp11 int  -- für Wert aus inserted-tabelle
  declare @dsp11 int  -- für Wert aus deleted-tabelle
  select @isp11 = inserted.sp11 from inserted
  select @dsp11 = deleted.sp11 from deleted

  raiserror('INS: %d DEL: %d',1,1,@isp11,@dsp11) with nowait

  delete from tabelleb
  where sp21=@dsp11
go
```

Auch hier ist die RAISERROR-Anweisung nur für Demonstrationszwecke vorgesehen. Die damit generierte Meldung zeigt, daß SQL Server die gelöschten Daten in die deleted-Tabelle übernimmt. Damit

läßt sich die in TabelleB zu löschende Zeile anhand des alten - gelöschten - Werts aus TabelleA bestimmen, wie es die WHERE-Klausel in der DELETE-Anweisung des Triggers zeigt.

Löschen Sie nun die Zeile mit dem Wert 5003 in Spalte Sp11, und fragen Sie die beiden Tabellen zur Kontrolle ab:

```
delete from tabellea
where sp11=5003

select sp11 from tabellea
select sp21 from tabelleb
```

Das Ergebnis dürfte Sie nun nicht mehr überraschen:

```
Meldung 50000, Ebene 1, Status 50000:
INS: 0 DEL: 5003
sp11
-----------
1001
1002
1004
1005

sp21
-----------
1005
```

Wenn Sie einen Wert löschen, der nicht in TabelleB vorkommt, passiert in TabelleB überhaupt nichts - nur die entsprechende Zeile in TabelleA verschwindet.

17.3.6 Mehrzeilige Operationen

Ein Trigger wird in der Regel von einer einzelnen Anweisung und auch nur einmal pro Anweisung ausgelöst, kann sich aber auf viele Zeilen auswirken. Von einer INSERT-Anweisung ist normalerweise nur eine Zeile betroffen, bei UPDATE- und DELETE-Anweisungen kann es sich durchaus um mehrere Zeilen handeln. Dieser Tatsache sollten Sie sich bewußt sein, wenn Sie in der Triggerdefinition mit Werten arbeiten, die der Trigger selbst ändert oder geändert hat. Das trifft zum Beispiel auf die Berechnung von Summen zu.

Eine durchaus vernünftige und saubere Lösung ist es, wenn Sie im Trigger mittels der Funktion @@ROWCOUNT die Anzahl der betroffenen Zeilen ermitteln und alle Operationen rückgängig machen, die sich auf mehr als eine Zeile beziehen.

Da ein Trigger, angefangen bei der auslösenden Anweisung bis zum Abschluß der Trigger-Anweisungen, im Sinne einer Transaktion ausgeführt wird, können Sie die von der Anweisung ausgeführten Operationen innerhalb des Triggers mit ROLLBACK TRANSACTION rückgängig machen. Mehr zu Transaktionen erfahren Sie in Kapitel 18.

17.4 Trigger verwalten

Die Definition eines vorhandenen Triggers können Sie sowohl im Enterprise Manager als auch über verschiedene gespeicherte Prozeduren anzeigen.

Enterprise Manager

Führen Sie die folgenden Schritte aus, um Informationen zu einem Trigger im Enterprise Manager anzuzeigen:

1. Erweitern Sie die Konsolenstruktur bis zur gewünschten Datenbank, so daß die Einträge für diese Datenbank zu sehen sind. Klicken Sie im Detailbereich mit der rechten Maustaste auf die Tabelle, die den Trigger enthält. Wählen Sie aus dem Kontextmenü den Befehl **Alle Aufgaben** und dann **Trigger verwalten**. Damit öffnen Sie das Dialogfeld **Triggereigenschaften** (siehe Abbildung 17.1).

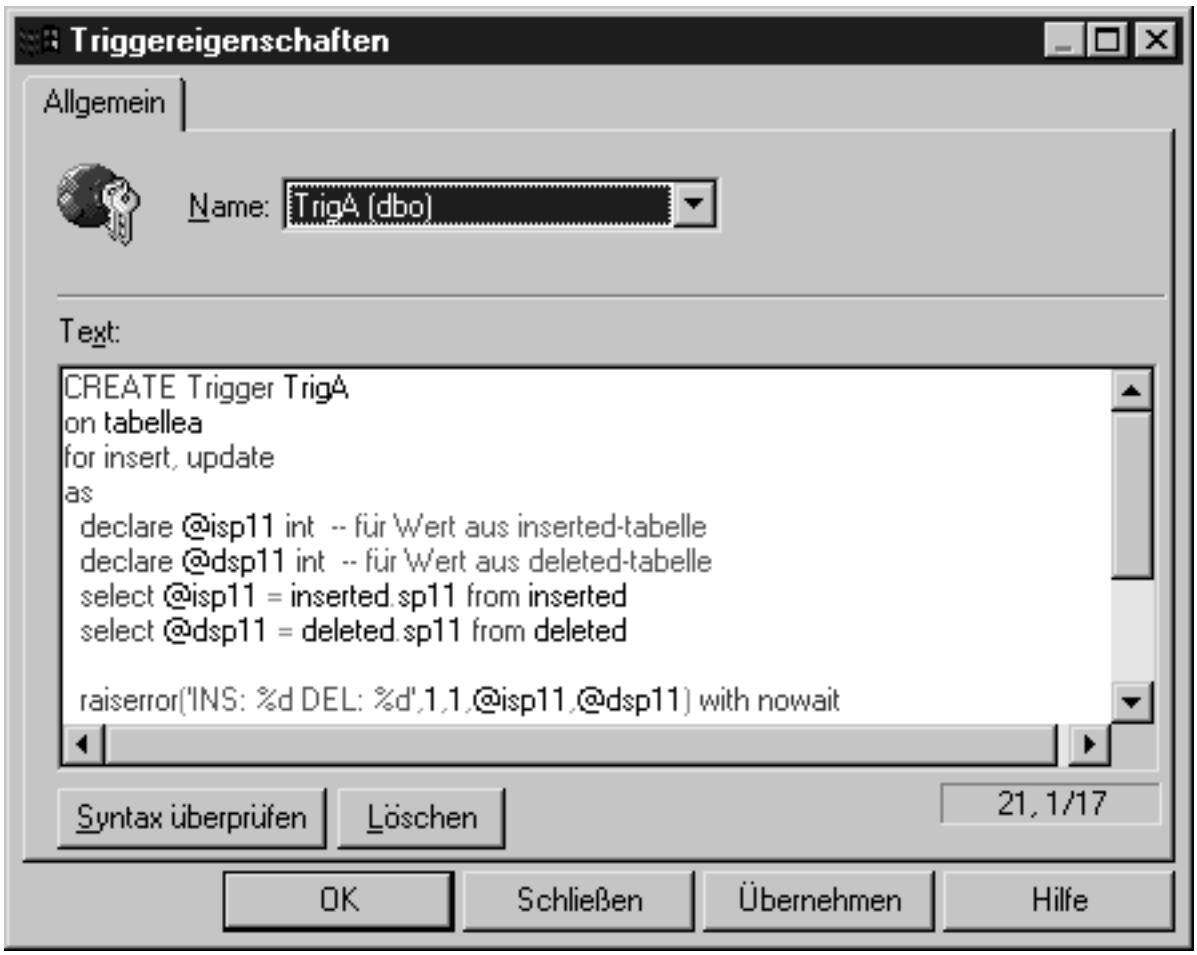


Abbildung 17.1: Das Dialogfeld Triggereigenschaften

2. Im Drop-down-Listenfeld **Name** wählen Sie den gewünschten Trigger aus. Die zugehörige

Definition erscheint im Bearbeitungsfeld **Text**. Hier können Sie auch die Anweisungen im Trigger bearbeiten. Die Schaltfläche **Syntax überprüfen** bewirkt genau das, was ihr Titel verspricht. Wenn Sie Änderungen am Trigger vorgenommen haben, wird die Schaltfläche **Übernehmen** aktiv, so daß Sie die geänderte Triggerdefinition in der Datenbank speichern können.

Die Systemtabelle sysobjects

Im Enterprise Manager lassen sich die vorhandenen Trigger für nur jeweils eine Tabelle auflisten. Wenn Sie aber wissen wollen, welche Trigger für die gesamte Datenbank definiert sind, können Sie diese Angaben aus der Systemtabelle sysobjects für die jeweilige Datenbank abrufen:

```
use test
select name from sysobjects
where type='TR'
```

Die Ergebnismenge sieht zum Beispiel wie folgt aus:

```
name
------------------------------------------------------------
TrigA
TrigADel

(2 row(s) affected)
```

Neben dem Namen lassen sich noch weitere Angaben aus der Systemtabelle sysobjects abrufen. Näheres dazu erfahren Sie in Kapitel 19.

Die gespeicherte Prozedur sp_help

Die Systemprozedur sp_help liefert Informationen zu verschiedenen Datenbankobjekten. Die Syntax lautet:

```
sp_help 'Objektname'
```

Als *Objektname* geben Sie den Namen des Triggers ein, um Informationen zu diesem Trigger abzurufen. Zum Beispiel liefern die Anweisungen

```
use test
exec sp_help 'TrigA'
```

folgendes Ergebnis:

```
Name      Owner     Type      Created_datetime
--------- --------- --------- --------------------------
TrigA     dbo       trigger   1999-04-21 13:59:07.570
```

Die gespeicherte Prozedur sp_depends

Die Abhängigkeiten eines Datenbankobjekts lassen sich mit der gespeicherten Prozedur sp_depends nach folgender Syntax anzeigen:

```
sp_depends 'Objektname'
```

Auch hier geben Sie wieder als Objektnamen den Namen des Triggers ein:

```
use test
exec sp_depends 'TrigA'
```

Im Beispiel erhalten Sie folgendes Ergebnis:

```
In der aktuellen Datenbank verweist das angegebene Objekt auf:
name            type              updated selected column
--------------- ----------------- ------- -------- -------------
dbo.TabelleB    user table        yes     yes      Sp21
```

Die gespeicherte Prozedur sp_helptext

SQL Server speichert die Definition von Objekten wie Regeln, Standardwerte, Sichten, gespeicherte Prozeduren und eben auch Trigger in der Systemtabelle syscomments. Mit der gespeicherten Prozedur sp_helptext rufen Sie den Text eines Triggers folgendermaßen ab:

```
use test
exec sp_helptext 'TrigA'
```

Im Ergebnis erhalten Sie den vollständigen Text der Triggerdefinition. Wenn jedoch die Triggerdefinition mit der Option WITH ENCRYPTION verschlüsselt wurde, müssen Sie sich an den Autor des Triggers wenden und den Quelltext anfordern, da sich der verschlüsselte Text nicht wieder dechiffrieren läßt.

17.5 Trigger ändern

Im letzten Abschnitt haben Sie bereits gesehen, wie sich die Triggerdefinition im Enterprise Manager ändern läßt. Natürlich gibt es auch eine Transact-SQL-Anweisung, mit der Sie den Trigger ändern können. Die vereinfachte Syntax lautet:

```
ALTER TRIGGER Triggername
ON Triggertabelle
FOR {[DELETE] [,] [INSERT] [,] [UPDATE]}
[WITH ENCRYPTION]
AS
  SQL_Anweisungen
```

Die Syntax ist bis auf das Schlüsselwort ALTER anstelle von CREATE identisch mit der Syntax der Anweisung CREATE TRIGGER. Ein Beispiel haben Sie bereits weiter vorn in diesem Kapitel im Abschnitt »Aktualisierungen weitergeben« kennengelernt.

17.6 Trigger umbenennen

Bevor Sie einen Trigger umbenennen, empfiehlt es sich, die Abhängigkeiten zu anderen Datenbankobjekten in Erfahrung zu bringen, beispielsweise mit der weiter vorn behandelten Systemprozedur sp_depends. Die allgemeine Transact-SQL zum Umbenennen eines Datenbankobjekts lautet:

```
sp_rename [@objname=] 'Objektname',
[@newname=] 'NeuerName'
[, [@objtype=] 'Objekttyp']
```

Als *Objektname* geben Sie den alten Namen des Triggers an, als *NeuerName* natürlich den neuen. Den *Objekttyp* können Sie bei einem Trigger weglassen oder als OBJECT spezifizieren. Die folgende Anweisung benennt den Trigger TrigA in NeuTrigA um:

```
use test
exec sp_rename 'TrigA', 'NeuTrigA'
```

SQL Server gibt daraufhin folgende Warnung aus:

```
Vorsicht: Wenn Sie Teile eines Objektnamens ändern, werden
Skripts und gespeicherte Prozeduren möglicherweise
funktionsunfähig.
object wurde in 'NeuTrigA' umbenannt.
```

17.7 Trigger löschen

Das Löschen eines Triggers ist eine einfache Angelegenheit. Führen Sie einfach die gespeicherte Prozedur DROP TRIGGER mit nachstehender Syntax aus:

```
DROP TRIGGER Triggername
```

Als *Triggername* lassen sich auch mehrere - durch Kommas getrennte - Trigger angeben. Optional können Sie den Besitzer des Triggers spezifizieren.

17.8 Geschachtelte und rekursive Trigger

Wenn die Anweisungen in einem Trigger dazu führen, daß ein anderer Trigger ausgelöst wird, spricht man von *geschachtelten Triggern*. Zum Beispiel könnte in den Tabellen, die in diesem Kapitel verwendet wurden, der Trigger der TabelleA bei einer Aktualisierung einen Trigger in TabelleB auslösen. Die Schachtelungstiefe von Triggern kann bis zu 32 Ebenen umfassen. Mit der Option nested triggers steuern Sie, ob das Schachteln von Triggern zulässig ist.

Ist die Option recursive triggers eingeschaltet, kann sich ein Trigger auch selbst aufrufen. Dabei ist zwischen direkter und indirekter Rekursion zu unterscheiden. Bei einer *direkten Rekursion* ruft sich der Trigger unmittelbar selbst wieder auf - zum Beispiel bei einer Aktualisierung, die das Auslösen desselben UPDATE-Triggers bewirkt. Im Szenario einer *indirekten Rekursion* ruft ein Trigger der Tabelle A einen Trigger der Tabelle B auf. Wenn der Trigger in Tabelle B eine Aktualisierung der Tabelle A veranlaßt, kommt es wieder zur Ausführung des Triggers in Tabelle A. Dieser Vorgang kann sich endlos fortsetzen. SQL Server begrenzt allerdings die Anzahl der rekursiven Ebenen auf 32 und bricht damit automatisch eine endlose Rekursion ab. Dennoch sollten Sie darauf achten, rekursive Triggerkonstruktionen zu vermeiden.

---

kapitel 18 transaktionen

18.1 alles oder nichts

nehmen wir an, sie müssen eine rechnung bezahlen. sie gehen zur bank und erteilen einen überweisungsauftrag (oder erledigen das im zeitalter des elektronischen zahlungsverkehrs von zu hause aus). die bank ihrerseits leitet den fälligen betrag von ihrem konto zum konto bei der bank des empfängers weiter. von ihrem konto wird der betrag abgebucht, auf dem konto des empfängers als haben gebucht. diese folge von geldbewegungen bezeichnet man als transaktion. wenn sie eine falsche kontonummer angegeben haben oder ihr konto nicht gedeckt ist, was ja vorkommen soll, wird die überweisung vom geldinstitut des vermeintlichen empfängers zurückgewiesen oder von ihrer bank gar nicht erst ausgeführt. eine nur zum teil abgewickelte transaktion gibt es nicht. genauso verhält es sich mit transaktionen in sql server.

18.2 merkmale von transaktionen

als *transaktion* bezeichnet man eine als logische arbeitseinheit ausgeführte folge von operationen. allerdings muß diese arbeitseinheit die vier acid-eigenschaften aufweisen, um als transaktion zu gelten.

das akronym acid steht für atomicity (unteilbarkeit), consistency (konsistenz), isolation und durability (beständigkeit).

- *unteilbarkeit*: diese eigenschaft wurde bereits im eingangs geschilderten beispiel beschrieben. entweder werden alle vorgesehenen datenänderungen (im beispiel abbuchen und zubuchen) ausgeführt oder gar keine.
- *konsistenz*: wenn zum beispiel gleichzeitig zu ihrer überweisung noch eine gutschrift auf ihrem konto eintrifft, wird ihr konto von zwei unterschiedlichen vorgängen bearbeitet. auf welchen kontostand bezieht sich nun welche transaktion? die arbeitseinheit muß absichern, daß sich alle daten am ende einer transaktion in einem konsistenten zustand befinden. insbesondere in einer relationalen datenbank gehört dazu auch, daß alle regeln auf die änderungen der transaktion angewandt werden, damit die integrität der daten erhalten bleibt.
- *isolation*: transaktionen müssen eigenständig sein. sie dürfen weder eine wirkung auf andere - gleichzeitig ablaufende - transaktionen ausüben noch von anderen transaktionen beeinflußt werden. einer transaktion stehen die daten entweder im zustand vor der änderung durch eine andere transaktion oder im zustand nach abschluß der zweiten transaktion zur verfügung. zwischenzustände sind nicht erlaubt.
- *beständigkeit*: eine transaktion hat ihr ziel erreicht, sobald alle operationen erfolgreich abgeschlossen sind. es gibt dann keinen grund mehr, die operationen rückgängig zu machen. die änderungen werden dauerhaft in das system übernommen. wichtige abläufe sollte man deshalb in einer großen transaktion einbetten. damit ist sichergestellt, daß entweder alle oder gar keine

änderungen an der datenbank vorgenommen werden.

für die dauer einer transaktion müssen alle änderungen aufgezeichnet und die von der transaktion betroffenen datenbankobjekte gesperrt werden können.

18.3 transaktionsprotokolle

sql server zeichnet alle änderungen an einer datenbank in einem zur datenbank gehörenden transaktionsprotokoll auf. hiervon ausgenommen sind die sogenannten *nicht protokollierten operationen*, beispielsweise das schnelle massenkopieren mit dem dienstprogramm bcp oder das abschneiden einer tabelle mit truncate table. beispiele für protokollierte änderungen an datenbanken sind modifikationen der daten über die transact-sql-anweisungen update, insert und delete, alle arten der objekterstellung und alle änderungen, die die sicherheit betreffen.

die bezeichnung *transaktions*protokoll geht vor allem auf die tatsache zurück, daß sql server alle informationen aufzeichnet, die innerhalb einer transaktion stattfinden. dazu gehört unter anderem die information, welche transaktion die jeweilige änderung durchgeführt hat, an welcher stelle eine transaktion beginnt und wie eine transaktion beendet wurde. mit speziellen transact-sql-anweisungen (wie begin transaction und commit transaction) haben sie die möglichkeit, den anfangspunkt und den endpunkt für eine gruppe von datenänderungen festzulegen.

sql server arbeitet mit einem sogenannten write-ahead-protokoll. das bedeutet, daß sql server alle änderungen an der datenbank zuerst in das transaktionsprotokoll schreibt und dann erst - bei fehlerfreier ausführung der anweisungen - die datenänderungen dauerhaft in die datenbank bzw. auf den datenträger übernimmt oder - bei fehlern - anhand der informationen im transaktionsprotokoll rückgängig macht.

zu jeder datenbank gehört mindestens ein transaktionsprotokoll, das sql server beim erstellen einer datenbank automatisch anlegt. sql server vergrößert das transaktionsprotokoll automatisch, falls es nicht mehr ausreichend platz bietet. damit das transaktionsprotokoll nicht irgendwann den gesamten platz auf der festplatte beansprucht, läßt es sich an einem bestimmten punkt kürzen.

bei jedem neustart führt sql server eine automatische wiederherstellung der datenbanken durch, um die datenbanken gegen systemabstürze zu schützen. dabei wird auch das transaktionsprotokoll überprüft. enthält das protokoll abgeschlossene transaktionen, die noch nicht in die datenbank geschrieben wurden, führt sql server die betreffenden transaktionen erneut aus. diesen vorgang bezeichnet man als *rollforward* oder *anwenden der transaktion*. bei einem *rollback* oder *zurücksetzen der transaktion* entfernt sql server alle unvollständigen transaktionen aus dem transaktionsprotokoll.

18.4 transaktionsarten

je nachdem, wie eine transaktion begonnen und abgeschlossen wird, unterscheidet man zwischen folgenden transaktionsarten:

- explizite transaktionen
- implizite transaktionen
- automatische transaktionen

18.4.1 explizite transaktionen

bei *expliziten transaktionen* legen sie den start- und endpunkt der transaktion explizit mit reservierten schlüsselwörtern fest. tabelle 18.1 führt die in transact-sql definierten schlüsselwörter zusammen mit einer kurzen beschreibung auf.

schlüsselwort	beschreibung
begin transaction	markiert den anfang einer expliziten lokalen transaktion.
begin distributed transaction	definiert den beginn einer verteilten transact-sql-transaktion. eine derartige transaktion läuft unter steuerung des microsoft distributed transaction coordinators (ms dtc).
commit transaction	markiert das ende einer erfolgreichen transaktion.
commit work	markiert das ende einer transaktion. die funktionsweise ist identisch mit commit transaction. allerdings erlaubt commit work nicht die angabe eines transaktionsnamens.
rollback transaction	macht die aktionen einer transaktion bis zum anfang der transaktion oder bis zu einem sicherungspunkt rückgängig.
rollback work	macht die aktionen einer transaktion rückgängig. die funktionsweise ist identisch mit rollback transaction. allerdings erlaubt rollback work keine angabe eines transaktionsnamens.
save transaction	legt einen sicherungspunkt in einer transaktion fest.

tabelle 18.1: schlüsselwörter für explizite transaktionen

das schlüsselwort transaction können sie in allen anweisungen mit tran abkürzen.

in den anweisungen begin transaction, commit transaction rollback transaction und save transaction läßt sich optional ein name für die transaktion spezifizieren. diese namen haben nur im zusammenhang mit geschachtelten transaktionen eine bedeutung. auf sicherungspunkte und geschachtelte transaktionen gehen die entsprechenden abschnitte weiter unten in diesem kapitel ein.

18.4.2 implizite transaktionen

bei einer expliziten transaktion müssen sie sowohl den start- als auch den endpunkt der transaktion selbst bestimmen. wenn sich eine verbindung im *impliziten transaktionsmodus* befindet, leitet sql server automatisch eine neue transaktion ein, sobald sie für die laufende transaktion eine commit- oder rollback-anweisung ausgeführt haben. genaugenommen legen sie sogar den startpunkt einer impliziten transaktion fest, nur verwenden sie dazu keine schlüsselwörter wie begin transaction. wenn sie nämlich eine der in tabelle 18.2 gezeigten anweisungen erstmalig ausführen, starten diese implizit eine neue transaktion. da nach abschluß einer transaktion sofort eine neue transaktion beginnt, entsteht eine fortlaufende transaktionskette.

alter table	create	delete

drop	fetch	grant
insert	open	revoke
select	truncate table	update

tabelle 18.2: anweisungen, die im impliziten modus eine transaktion starten

mit der set-anweisung

```
set implicit_transactions on | off
```

schalten sie den impliziten transaktionsmodus für eine verbindung ein (on) bzw. aus (off).

nachdem sich eine verbindung im impliziten transaktionmodus befindet, müssen sie darauf achten, eine transaktion auch abzuschließen. wenn sie das nicht tun, bleiben die laufenden transaktionen geöffnet. somit bestehen auch alle sperren unnötig lange weiter.

18.4.3 automatische transaktionen

sql server führt eigentlich immer transaktionen aus, auch wenn es auf den ersten blick nicht so aussieht. allerdings sind die einzelnen transact-sql-anweisungen in sich abgeschlossene transaktionen und bilden keine gruppe, die entweder als ganzes oder gar nicht ausgeführt wird. jede anweisung markiert damit gleichzeitig den beginn einer transaktion. der endpunkt der transaktion ist erreicht, wenn die anweisung beendet wird. sql server führt sozusagen hinter den kulissen eine commit transaction-anweisung aus, wenn die transact-sql-anweisung keine fehler geliefert hat, und andernfalls eine rollback transaction-anweisung.

die online-dokumentation bezeichnet automatische transaktionen als autocommit-transaktionen. der autocommitmodus ist immer standardmäßig eingestellt, sofern sie keine explizite transaktion definieren oder die verbindung in den impliziten transaktionsmodus setzen.

18.5 arbeitsweise von transaktionen

dieser abschnitt zeigt, wie sie anweisungen als transaktion ausführen und wie sich sql server bei fehlerhaften anweisungen verhält. in den beispielen kommen die bereits aus kapitel 16 und 17 bekannten tabellen der datenbank test zum einsatz. die tabellen können sie hier unverändert übernehmen - auch inklusive der definierten trigger - oder neu erstellen (wobei sie auf die trigger verzichten können).

die folgenden beispiele demonstrieren, wie sich fehlerhafte anweisungen im standardmodus sowie im expliziten und impliziten transaktionsmodus auswirken. um definierte ausgangsverhältnisse zu schaffen, löschen sie zunächst den inhalt von tabellea:

```
delete from tabellea
```

wenn sie jetzt die anweisungen

```
insert into tabellea (sp11)
              values (1003)
insert into tabellea (sp11,sp12,sp13)
              values (1004,'viertausend','4-4-4-4')
insert into tabellea (sp11,sp12,sp13)
              values (1007, 7777, '7-7-7-7')
```

ausführen, erhalten sie folgendes ergebnis:

```
server: nachr.-nr. 245, schweregrad 16, status 1, zeile 1
syntaxfehler beim konvertieren des varchar-wertes 'viertausend'
 in eine spalte vom datentyp int.
sp11        sp12        sp13
----------- ----------- --------------------
1003        null        null
```

sql server konnte die erste insert-anweisung fehlerfrei ausführen. deshalb erscheint auch die zeile mit dem wert 1003 für spalte sp11 in der tabelle. die angezeigte fehlermeldung geht auf die zweite insert-anweisung zurück. die spalte sp12 ist mit dem datentyp int deklariert. sql server kann die zeichenfolge 'viertausend' nicht in eine zahl konvertieren und damit auch nicht die zeile in die tabelle einfügen.

die dritte insert-anweisung ist zwar korrekt, wird aber dennoch nicht ausgeführt, weil bei der zweiten anweisung ein fehler aufgetreten ist und sql server die weitere bearbeitung des stapels abbricht.

löschen sie jetzt wieder den inhalt von tabellea, um den ausgangszustand herzustellen. führen sie dann die obigen anweisungen als explizite transaktion aus:

```
begin transaction
insert into tabellea (sp11)
              values (1003)
insert into tabellea (sp11,sp12,sp13)
              values (1004,'viertausend','4-4-4-4')
insert into tabellea (sp11,sp12,sp13)
              values (1007, 7777, '7-7-7-7')
if @@error>0
  rollback transaction
else
  commit transaction
```

```
go
```

das ergebnis lautet:

```
server: nachr.-nr. 245, schweregrad 16, status 1, zeile 1
syntaxfehler beim konvertieren des varchar-wertes 'viertausend'
 in eine spalte vom datentyp int.
sp11        sp12        sp13
----------- ----------- --------------------
```

der fehler tritt hier genauso auf und führt zur angegebenen fehlermeldung, die tabelle ist aber am ende der abarbeitung leer. sql server hat alle drei anweisungen im sinne einer transaktion ausgeführt, d.h. entweder alle anweisungen fehlerfrei oder gar nicht. durch die schlüsselwörter begin transaction und rollback transaction bzw. commit transaction sind die anweisungen in eine transaktion eingebettet. begin transaction leitet die transaktion ein, commit transaction schließt eine fehlerfreie transaktion ab, und rollback transaction macht alle nach begin transaction vorgenommenen änderungen rückgängig.

die funktion @@error liefert einen wert größer 0, wenn bei der abarbeitung ein fehler aufgetreten ist. im beispiel ist das der fall, so daß die anweisung rollback transaction im if-zweig zur ausführung kommt. wenn sich die anweisungen fehlerfrei ausführen lassen, schließt die anweisung commit transaction im else-zweig die transaktion ab, und sql server übernimmt die datenänderungen in die tabelle.

wenn sie die anweisung begin transaction aus der obigen anweisungsfolge entfernen und die verbindung mit

```
set implicit_transactions on
```

vor ausführung der anweisungen in den impliziten transaktionsmodus setzen, läuft das beispiel genauso ab - nur eben als implizite transaktion.

18.6 prüfpunkte

der abschluß einer transaktion mit commit schreibt die geänderten datenseiten einer datenbank noch nicht auf den datenträger. die änderungen sind zunächst einmal im transaktionsprotokoll verzeichnet. mit dem konzept der *prüfpunkte* stellt sql server sicher, daß alle datensätze des transaktionsprotokolls und alle geänderten datenbankseiten auf den datenträger geschrieben werden. daraus folgt, daß sql server im rahmen der automatischen wiederherstellung nur für diejenigen transaktionen einen rollforward ausführen muß, die noch nicht auf den datenträger geschrieben wurden.

wenn sie mit prüfpunkten arbeiten, können sie die zeit für die automatische wiederherstellung verringern. alle abgeschlossenen transaktionen, die im transaktionsprotokoll vor dem prüfpunkt verzeichnet sind,

kann sql server übergehen, da sie bereits auf dem datenträger stehen.

sql server wendet prüfpunkte automatisch an, wobei sie die zeitspanne zwischen zwei prüfpunkten mit der datenbankoption recovery interval in minuten einstellen können. der standardwert 0 für diese option bedeutet, daß sql server das intervall selbst bestimmt - in der praxis etwa 1 minute. darüber hinaus führt sql server eine automatische prüfpunktoperation aus, wenn sie

- mit der gespeicherten prozedur sp_dboption eine datenbankeinstellung ändern,
- sql server herunterfahren (indem sie den transact-sql-befehl shutdown ausführen oder den dienst mssqlserver beenden).

mit der transact-sql-anweisung checkpoint können sie manuell einen prüfpunkt erzwingen.

18.7 nicht erlaubte operationen

innerhalb einer transaktion können sie nicht alle transact-sql-anweisungen ausführen. das hängt einfach damit zusammen, daß sich bestimmte operationen nicht oder nur mit großem aufwand rückgängig machen lassen. tabelle 18.3 zeigt die anweisungen, die in transaktionen unzulässig sind.

alter database	backup log	create database
disk init	drop database	dump transaction
load database	load transaction	reconfigure
restore database	restore log	update statistics

tabelle 18.3: unzulässige transact-sql-anweisungen in transaktionen

18.8 geschachtelte transaktionen

um es gleich vorwegzunehmen: geschachtelte transaktionen sind keine selbständigen transaktionen. wenn sie in einer - äußeren - transaktion eine zweite - innere - transaktion einbetten, müssen sie zwar die innere transaktion mit commit oder rollback abschließen, was aber nicht heißt, daß sie die innere transaktion vollkommen unabhängig von der äußeren annehmen oder zurücksetzen können. sehen sie sich folgende konstruktion an:

```
1:  begin transaction aeussere
2:    anweisungen_1
3:    begin transaction innere
4:      anweisungen_2
5:    commit transaction innere
6:    anweisungen_3
7:  rollback transaction aeussere
```

zeile 1 leitet die transaktion mit dem namen aeussere ein. diese transaktion erstreckt sich über die zeilen 2 bis 6 und wird in zeile 7 abgeschlossen. zeile 3 markiert den beginn der geschachtelten transaktion innere. zu dieser transaktion gehört der anweisungsblock anweisungen_2. die commit-anweisung in zeile 5 markiert den abschluß der inneren transaktion.

im beispiel sollen sich die anweisungsblöcke anweisungen_1 und anweisungen_2 fehlerfrei ausführen lassen, während der block anweisungen_3 einen fehler liefert, so daß die rollback-anweisung in zeile 7 die transaktion aeussere rückgängig macht.

man könnte nun annehmen, daß die innere transaktion den anweisungsblock anweisungen_2 aufgrund der commit-anweisung (zeile 5) in die datenbank übernimmt und nur die anweisungsblöcke anweisungen_1 und anweisungen_3 der äußeren transaktion rückgängig gemacht werden. weit gefehlt! ein rollback am ende der äußeren transaktion macht auch die anweisungen der inneren transaktion rückgängig, und zwar unabhängig davon, ob die innere transaktion mit commit abgeschlossen wurde.

drehen wir das beispiel um. jetzt soll der innere block mit anweisungen_2 fehlerhaft sein, während sich die anweisungsblöcke anweisungen_1 und anweisungen_3 fehlerfrei ausführen lassen:

```
1:  begin transaction aeussere
2:    anweisungen_1
3:    begin transaction innere
4:      anweisungen_2
5:    rollback transaction innere
6:    anweisungen_3
7:  commit transaction aeussere
```

der transaktionsname für die innere transaktion ist hier lediglich angegeben, um die struktur deutlich zu machen. sql server registriert nur die namen der äußersten transaktion. folglich gibt es aus sicht von sql server keine transaktion innere, so daß die rollback-anweisung in zeile 6 zur folgenden fehlermeldung führt:

```
rollback für innere kann nicht durchgeführt werden. keine
transaktion und kein sicherungspunkt dieses namens wurde
gefunden.
```

in einer derartigen konstruktion müssen sie also die transaktionsnamen (hier in den zeilen 3 und 5) entfernen.

wenn sie dieses beispiel mit realen anweisungen ausführen würden, erhalten sie die fehlermeldung:

```
server: nachr.-nr. 3902, schweregrad 16, status 1, zeile 7
die commit transaction-anforderung hat keine entsprechende
begin transaction.
```

allerdings hat sql server den anweisungsblock anweisungen_3 ordnungsgemäß ausgeführt. was ist hier passiert? die rollback-anweisung in zeile 5 setzt nicht nur die innere transaktion, sondern auch die äußere transaktion zurück. damit ist sowohl die innere als auch die äußere transaktion abgeschlossen. die commit-anweisung in zeile 7 »hängt in der luft«, weil es keine offene transaktion mehr gibt. der anweisungsblock anweisungen_3 läuft damit außerhalb einer expliziten transaktion.

wenn sich die verbindung im impliziten transaktionsmodus befindet, erhalten sie die angegebene fehlermeldung nicht, weil sql server nach dem rollback in zeile 5 sofort wieder eine transaktion beginnt, die die commit-anweisung in zeile 7 abschließen kann.

die schachtelungsebene einer transaktion vermerkt sql server in einem transaktionszähler, den sie mit der funktion @@trancount abfragen können. der transaktionszähler wird mit jeder begin transaction-anweisung um 1 inkrementiert und mit jeder commit-anweisung um 1 dekrementiert. eine rollback-anweisung setzt den transaktionszähler auf 0, läßt ihn aber unverändert, wenn sich die anweisung auf einen sicherungspunkt (siehe nächster abschnitt) bezieht.

die commit-anweisung in einer inneren transaktion ist quasi nur schmückendes beiwerk - sql server ignoriert sie einfach. allerdings ist die anweisung erforderlich, um den transaktionszähler zu dekrementieren.

nachdem sie nun wissen, was geschachtelte transaktionen *nicht* können, stellt sich die frage, wofür sie eigentlich gut sind. sql server erlaubt geschachtelte transaktionen, damit man die anweisungen innerhalb einer gespeicherten prozedur als explizite transaktion ausführen kann.

wenn sie eine derartige gespeicherte prozedur im autocommit-modus aufrufen, werden die anweisungen der prozedur als in sich abgeschlossene transaktion ausgeführt und entweder angenommen oder rückgängig gemacht. findet der aufruf der gespeicherten prozedur dagegen in einer - äußeren - expliziten transaktion statt, entscheidet der abschluß der äußeren transaktion, ob die anweisungen der gespeicherten prozedur - d.h. die innere transaktion - in die datenbank übernommen oder rückgängig gemacht werden.

weitere informationen zu geschachtelten prozeduren und ein ausführlich erläutertes beispiel finden sie in der online-dokumentation. gehen sie im index zum eintrag **transaktionen / geschachtelt**, und wählen sie das thema **schachteln von transaktionen**.

18.9 sicherungspunkte

wenn bestimmte anweisungen innerhalb einer transaktion mit hoher wahrscheinlichkeit nicht zu einem fehler führen und sich deshalb aus einer im weiteren verlauf der transaktion eventuell ausgeführten rollback-operation ausklammern lassen, können sie mit der transact-sql-anweisung

```
save tran[saction] sicherungspunktname
```

einen sicherungspunkt definieren. wenn sie in einer nachfolgenden rollback-anweisung diesen

sicherungspunkt angeben, werden die anweisungen der transaktion von diesem punkt an rückgängig gemacht. die von begin transaction bis zum sicherungspunkt ausgeführten anweisungen bleiben von der rollback-operation verschont.

das etwas abgeänderte beispiel aus dem abschnitt »arbeitsweise von transaktionen« weiter vorn in diesem kapitel soll die wirkung eines sicherungspunktes demonstrieren:

```
1:  begin transaction
2:    insert into tabellea (sp11)
3:                  values (1003)
4:
5:  save transaction sicherungspunkt
6:    -- zeile nur zur probe einfügen und anzeigen,
7:    -- aber nicht dauerhaft übernehmen:
8:    insert into tabellea (sp11,sp12,sp13)
9:                  values (1004,'4000','4-4-4-4')
10:   select sp11, sp12, sp13 from tabellea
11:
12:   rollback transaction sicherungspunkt
13:
14:   insert into tabellea (sp11,sp12,sp13)
15:                 values (1007, 7777, '7-7-7-7')
16:
17: commit transaction
18: go
19: select sp11,sp12,sp13 from tabellea
```

die ergebnisse der insert-anweisungen in den zeilen 2/3 und 14/15 sind im rahmen der transaktion auf jeden fall in die datenbank zu übernehmen. dagegen soll die insert-anweisung in den zeilen 8/9 die daten nur zur probe in die tabelle eintragen. die select-anweisung in zeile 10 ruft die daten zur kontrolle ab. um die änderungen rückgängig zu machen, führt zeile 12 einen rollback zum sicherungspunkt aus. als ergebnis erhalten sie:

```
sp11        sp12        sp13
----------- ----------- --------------------
1003        null        null
1004        4000        4-4-4-4

sp11        sp12        sp13
----------- ----------- --------------------
1003        null        null
1007        7777        7-7-7-7
```

die obere ergebnismenge stammt von der select-anweisung in zeile 10, die untere von der select-anweisung in zeile 19. der ablauf dieser transaktion findet in folgenden schritten statt:

1. anfangspunkt der transaktion mit begin transaction markieren.
2. ausführen der insert-anweisung in den zeilen 2/3.
3. sicherungspunkt der transaktion festlegen (zeile 5).
4. ausführen der insert-anweisung in den zeilen 8/9.
5. abrufen der bisher in die tabelle eingetragenen daten mit der select-anweisung in zeile 10.
6. rückgängigmachen der insert-anweisung aus zeile 8/9, indem ein rollback zum sicherungspunkt in zeile 5 ausgeführt wird. die bis zu diesem punkt ausgeführten anweisungen (hier die insert-anweisung in den zeilen 2/3) sind von diesem rollback nicht betroffen.
7. ausführen der insert-anweisung in den zeilen 14/15.
8. abschließen der gesamten transaktion mit der commit-anweisung in zeile 17.
9. anzeigen der endgültigen ergebnisse mit der select-anweisung in zeile 19.

18.10 sperren

wenn sie mit ihrem auto gemütlich auf einer einsamen landstraße unterwegs sind, machen sie sich sicherlich keine gedanken darüber, daß ihnen jemand »ihre« straße streitig machen könnte. in einer größeren ortschaft treffen sie aber bestimmt auf eine rote ampel - der deutsche amtmann spricht von lichtsignalanlage - und müssen so lange warten, bis der verkehr in der anderen richtung gesperrt wird. eine kreuzung können nun mal nicht alle gleichzeitig benutzen, selbst wenn es hin und wieder versuche gibt.

zu einer datenbank können tausende von benutzern eine verbindung herstellen. würde man nur einem benutzer erlauben, mit einer datenbank zu arbeiten, hieße das übertragen auf das obige szenario, eine ganz stadt zu sperren, nur um einem auto die durchfahrt zu gewähren. es ist eben praktischer, den verkehr abschnittsweise zu regeln. und genau das passiert in einer datenbank auch. sql server sperrt nicht die gesamte datenbank, sondern nur einzelne datenbankobjekte. je feiner die abschnitte sind, desto mehr benutzer können gleichzeitig datenänderungen vornehmen - d.h. desto besser ist die parallelität der datenbankoperationen. den feinheitsgrad der sperren bezeichnet man als *granularität*.

18.10.1 arten von sperren

sql server 7.0 realisiert das sperren von ressourcen im gegensatz zu früheren versionen vollkommen dynamisch, je nachdem, welche art der sperre für eine bestimmte operation am besten geeignet ist. tabelle 18.4 zeigt die verschiedenen objektarten, oder ressourcen, die sql server sperren kann.

ressource (kurzzeichen)	beschreibung
zeilen-id (rid)	sperre auf zeilenebene. es wird jeweils nur eine zeile gesperrt. (rid steht für row identifier - zeilenbezeichner.)
schlüssel (key)	sperre auf zeilenebene in einem index. die sperre bezieht sich auf einen schlüsselwert oder einen bereich von schlüsseln.

seite (pag)	sperre auf seitenebene. die seite mit einer größe von 8 kbyte ist die standardspeichereinheit in sql server. eine seite kann mehrere tabellenzeilen oder indexeinträge enthalten.
block (ext)	sperre für eine gruppe von 8 zusammenhängenden daten- oder indexseiten.
tabelle (tab)	sperre auf tabellenebene. in die sperre eingeschlossen sind alle daten und indizes.
datenbank (db)	sperre auf datenbankebene

tabelle 18.4: ressourcen, die sql server sperren kann

innerhalb von tabelle 18.4 nimmt die granularität von oben nach unten zu.

für die in tabelle 18.4 genannten ressourcen kann sql server verschiedene arten von sperren anwenden. die sperrmodi sind in tabelle 18.5 zusammengefaßt.

sperrmodus (kurzzeichen)	**beschreibung**
gemeinsam (s)	auch als nur-lese-sperre bezeichnet. eine sperre, die beim lesen von daten verhindert, daß andere benutzer die daten ändern.
exklusiv (x)	eine exklusivsperre verwenden sie, wenn sie daten modifizieren wollen. die sperre verhindert das abrufen oder ändern von daten durch andere benutzer, bis sie die sperre freigeben.
aktualisierung (u)	die aktualisierungssperre verwenden sie wie eine exklusivsperre. allerdings verhindert die aktualisierungssperre einen deadlock, wenn mehrere verbindungen die ressource lesen, sperren und dann vielleicht aktualisieren.
beabsichtigt (i)	dieser sperrmodus wird auf ein objekt einer höheren ebene angewendet und zeigt an, daß sql server eine gemeinsame oder exklusive sperre für ressourcen weiter unten in der hierarchie plant.
schema (sch-s, sch-m)	diese sperren beziehen sich auf änderungen des datenbankschemas durch anweisungen der datendefinitionssprache. der typ sch-s steht für schemastabilität, der typ sch-m für schemamodifikation.
massenaktualisierung (bu)	wenn mehrere prozesse gleichzeitig daten in eine tabelle massenkopieren, verhindern diese sperren, daß andere prozesse auf diese tabelle zugreifen.

tabelle 18.5: arten der sperren in sql server

da sql server automatisch den passenden sperrmodus und die erforderliche granularität wählt, brauchen sie sich in der regel nicht selbst darum zu kümmern. treten allerdings bei bestimmten transaktionen

probleme auf, können sie das sperrverhalten für eine abfrage beeinflussen. dazu brauchen sie natürlich informationen, wie und welche sperren sql server anwendet.

18.10.2 sperrinformationen anzeigen

die im folgenden beschriebene anzeige von sperrinformationen mit der gespeicherten prozedur sp_lock und mit dem enterprise manager bezieht sich auf die laufenden aktivitäten in einer datenbank. es handelt sich also nicht um protokolle, die sie sich im nachhinein ansehen können.

wenn sie die anweisung

```
sp_lock
go
```

ausführen, erhalten sie zum beispiel folgende informationen:

```
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
8    5    117575457 1      pag  1:96            is       grant
8    5    0         0      db                   s        grant
8    5    117575457 0      tab                  is       grant
8    5    117575457 1      key  (3ff584f3b49a)  s        grant
9    1    0         0      db                   s        grant
9    2    0         0      db                   s        grant
9    5    0         0      db                   s        grant
9    1    117575457 0      tab                  is       grant
```

aus dieser übersicht lassen sich folgende informationen ablesen:

- in der spalte spid ist die systemprozeß-id des jeweiligen sql-server-prozesses angegeben.
- die spalte dbid zeigt die id der datenbank, die die sperre anfordert.
- unter objid finden sie die id des objekts, das die sperre anfordert.
- indid gibt die id des index an.
- der gesperrte ressourcentyp ist in der spalte type verzeichnet. die bedeutung der kurzzeichen können sie tabelle 18.4 entnehmen. nicht in der tabelle enthalten sind fil für dateisperre und idx für indexsperre.
- die in der spalte resource wiedergegebenen informationen hängen vom typ der jeweiligen sperre ab:

- rid (zeilensperre): die gesperrte zeile wird in der form *dateinummer*:*seitennummer*:*zeilenbezeichner* angegeben.
- key (schlüsselsperre): die angegebene hexadezimalzahl wird von sql server intern verwendet und läßt sich nicht durch den benutzer interpretieren.
- pag (seitensperre): die gesperrte seite wird in der form *dateinummer*:*seitennummer* angegeben. im obigen beispiel ist also die seite 96 in der datei nummer 1 mit einer seitensperre belegt.
- ext (blocksperre): gibt die erste seitennummer der 8 seiten eines gesperrten blocks an. die kennzeichnung der seite erfolgt in der form *dateinummer*:*seitennummer*.
- für die ressourcentypen tab und db ist die objekt-id bereits in der spalte objid bzw. dbid verzeichnet, so daß unter resource keine weiteren angaben erfolgen.
- die spalte mode gibt auskunft über den sperrmodus. die kurzzeichen können sie tabelle 18.5 entnehmen. beabsichtigte sperren (i) sind durch den jeweiligen untertyp gekennzeichnet (im obigen beispiel is für intent share, d.h. beabsichtigt/gemeinsam).
- in der spalte status finden sie die wahrscheinlich für sie interessanteren informationen. hier ist angegeben, ob die sperre momentan erteilt ist (grant), auf einen anderen vorgang warten muß (wait) oder zu einer anderen sperre konvertiert wird (cnvt). das konvertieren zu einer anderen sperre bedeutet, daß auf den übergang zu einem restriktiveren sperrmodus gewartet wird. unter dem aspekt der blockierung kann man wait mit cnvt gleichsetzen.

sperrinformationen lassen sich auch im enterprise manager anzeigen. erweitern sie die konsolenstruktur für den gewünschten server bis zum ordner **verwaltung**, und wählen sie hier den eintrag **aktuelle aktivität**. der detailbereich enthält die einträge für prozeßinformationen und sperren (siehe abbildung 18.1).

bild

abbildung 18.1: der ordner aktuelle aktivität im enterprise manager

der ordner **sperren / prozess-id** zeigt die laufenden prozesse. wenn sie die jeweilige prozeß-id erweitern, erscheinen im detailbereich informationen zum gesperrten objekt, zum sperrentyp, zum modus und zum status (siehe abbildung 18.2). die kurzzeichen sind die gleichen, die sie bei der behandlung der gespeicherten prozedur sp_lock kennengelernt haben.

bild

abbildung 18.2: der ordner sperren / prozess-id

im ordner **sperren / objekte** können sie sperren für die einzelnen objekte verfolgen (siehe abbildung 18.3).

Bild

abbildung 18.3: der ordner sperren / objekte

18.10.3 deadlocks

bei einem *deadlock* blockieren sich zwei transaktionen gegenseitig. der konflikt läßt sich nur lösen, indem eine transaktion abgebrochen wird. wenn eine transaktion t1 eine tabelle aktualisieren will, die gerade von einer anderen transaktion t2 gesperrt wird, handelt es sich noch nicht um einen deadlock. t1 wartet lediglich, bis t2 die tabelle wieder freigibt, und kann dann die aktualisierung ausführen.

nehmen wir nun an, daß eine transaktion t1 die tabelle a (exklusiv) sperrt und eine transaktion t2 die tabelle b. will jetzt t1 auf die tabelle b zugreifen und diese sperren, muß transaktion t1 warten, bis transaktion t2 ein commit ausgeführt hat und die sperre für tabelle b aufhebt. wenn allerdings die transaktion t2 ihrerseits auf die von t1 gesperrte tabelle a zugreifen will, muß transaktion t2 ebenfalls warten, bis t1 die sperre für die tabelle a aufhebt. da t1 die sperre für a nicht aufhebt, solange kein zugriff auf b möglich ist, t2 aber die sperre für b nicht aufhebt, weil sie auf die freigabe von a wartet, kommt es zu einer gegenseitigen blockierung - einem deadlock - der beiden transaktionen. dieser teufelskreis läßt sich nur verlassen, wenn eine der beiden transaktionen »nachgibt«, d.h. wenn sie durch äußeren eingriff abgebrochen wird.

18.10.4 isolationsstufen

im zusammenhang mit transaktionen und sperren tauchen die begriffe *isolationsstufen* und *serialisierbarkeit* auf. gleichzeitig ablaufende transaktion sind serialisierbar, wenn man die transaktionen auch einzeln nacheinander, d.h. seriell und vollständig voneinander isoliert, in beliebiger reihenfolge ausführen kann und in jedem fall derselbe zustand der datenbank erreicht wird. diese eigenschaften stellen sicher, daß die daten immer korrekt sind.

allerdings ist nicht immer eine vollständige isolation der transaktionen erforderlich. je geringer die isolation der transaktionen, desto besser ist die parallelität. allerdings muß man dafür eine mögliche inkonsistenz der daten in kauf nehmen. der sql-92-standard definiert vier isolationsstufen, die sql server ebenfalls unterstützt:

- read uncommitted: sperren werden nur für update-befehle ausgelöst. diese stufe bietet zwar die beste parallelität, erlaubt aber dirty reads, phantomwerte und nicht wiederholtes lesen. vor einem lesevorgang ist kein commit erforderlich.
- read committed: vor jedem lesevorgang muß ein commit ausgeführt worden sein. diese isolationsstufe ist die standardeinstellung von sql server.
- repeatable read: sperrt die daten bei lesevorgängen. damit ist wiederholtes lesen sichergestellt. andere benutzer können die daten erst aktualisieren, wenn die transaktion abgeschlossen ist.
- serializable: die transaktionen sind vollständig voneinander isoliert und damit serialisierbar.

die isolationsstufe legen sie mit der folgenden set-anweisung fest:

```
set transaction isolation level {read committed |
read uncommitted | repeatable read | serializable }
```

die eingestellte isolationsstufe gilt für die gesamte verbindung, sofern sie keinen anderen wert festlegen.

18.10.5 sperrverhalten steuern

während die einstellungen der isolationsstufen für die gesamte verbindung gelten, läßt sich das sperrverhalten auch in select-, insert-, update- und delete-anweisungen mit sperrhinweisen auf tabellenebene steuern. die aktuelle isolationsstufe für transaktionen wird damit für die laufende sitzung außer kraft gesetzt.

der für sperrhinweise relevante syntaxausschnitt sieht bei diesen anweisungen folgendermaßen aus:

```
select auswahlliste
from tabellenliste with (tabellenhinweis)
insert [into] tabellenname with (eingeschrtabellenhinweis)
update tabellenname with (eingeschrtabellenhinweis)
```

bzw.

```
update tabellenname
set spaltenname
from quelltabelle with (tabellenhinweis)
delete [from] tabellenname with (eingeschrtabellenhinweis)
```

bzw.

```
delete [from] tabellenname
from tabellenname with (tabellenhinweis)
```

als *tabellenhinweis* lassen sich die in tabelle 18.6 genannten werte angeben.

tabellenhinweis	beschreibung
holdlock	sperre gilt nur für die dauer der transaktion. gleichbedeutend mit serializable.
nolock	es wird keine sperre verwendet. dirty reads sind möglich. eine abfrage kann daten lesen, die für exklusive verwendung gesperrt sind. folglich können geänderte, aber noch nicht angenommene daten in der abfrage enthalten sein. gleichbedeutend mit readuncommitted.
paglock	fordert eine gemeinsame seitensperre statt einer gemeinsamen tabellensperre an.
readcommitted	die sperren entsprechen der isolationsstufe read committed.

readpast	gesperrte zeilen werden übersprungen.
readuncommitted	gleichbedeutend mit nolock.
repeatableread	die sperren entsprechen der isolationsstufe repeatable read.
rowlock	fordert eine gemeinsame zeilensperre statt einer gemeinsamen seiten- oder tabellensperre an.
serializable	gleichbedeutend mit holdlock.
tablock	fordert eine gemeinsame tabellensperre an.
tablockx	fordert eine exklusive tabellensperre an.
updlock	fordert beim lesen aktualisierungssperren statt gemeinsamer sperren an.

tabelle 18.6: tabellenhinweise

readpast, nolock, readuncommitted und updlock sind als *eingeschrtabellenhinweis* bei insert, update und delete nicht zulässig.

setzen sie die genannten sperrhinweise nur dann ein, wenn sie sich über die auswirkungen genau im klaren sind. normalerweise sollten sie sql server die automatische anwendung der sperren überlassen.

18.11 verteilte transaktionen

verteilte transaktionen sind transaktionen, die sich über mehrere server erstrecken. ein server - der *transaktionsurheber* - löst eine transaktion mit der anweisung begin distributed transaction aus und steuert den fortgang dieser transaktion. selbst eine transaktion auf ein und demselben server ist eine verteilte transaktion, sofern sie sich über mindestens zwei datenbanken erstreckt.

die konfiguration, mit der die beispiele für diesen abschnitt entstanden sind, besteht aus den computern eins (windows nt 4.0 server) und zwei (windows 98). die angegebenen schritte beziehen sich auf die ausführung der befehle mit dem sql server query analyzer auf zwei.

1. starten sie sql server query analyzer auf zwei.
2. öffnen sie zwei abfragefenster. im ersten stellen sie eine verbindung zu eins, im zweiten eine verbindung zu zwei her.
3. erstellen sie für beide verbindungen die datenbank trana mit der tabellea:

```
create database trana
go
use trana
go
create table tabellea (
spa1 int,
spa2 int,
spa3 char(20))
```

```
go
```

4. mit diesem schritt verfügen sie über eine datenbank trana auf dem computer eins und eine identische datenbank trana auf dem computer zwei.

die datenbanken und tabellen müssen nicht identisch sein. für diese einfache demonstration hat es lediglich den vorteil, daß sie sowohl für eins als auch für zwei mit jeweils denselben skripts die datenbanken erstellen und die daten einfügen können.

5. die aktualisierung der werte in der tabellea auf eins soll über die gespeicherte prozedur updatespa1 erfolgen. erstellen sie diese prozedur mit der folgenden anweisung im abfragefenster für computer eins (für zwei ist diese prozedur nicht erforderlich):

```
use trana
go
create procedure updatespa1 (@spa1 int, @spa3 char(20))
as
  update tabellea
  set spa3=@spa3
  where spa1=@spa1
go
```

6. fügen sie mit der folgenden anweisung in tabellea - jetzt wieder für beide computer - daten ein, und rufen sie die daten zur kontrolle ab:

```
use trana
insert into tabellea values (1001,11001,'eins')
insert into tabellea values (1002,11002,'zwei')
insert into tabellea values (1003,11003,'drei')
insert into tabellea values (1004,11004,'vier')
insert into tabellea values (1005,11005,'fünf')
select * from tabellea

spa1        spa2        spa3
----------- ----------- --------------------
1001        11001       eins
1002        11002       zwei
1003        11003       drei
1004        11004       vier
1005        11005       fünf

(5 row(s) affected)
```

auf beiden computern sind jetzt die gleichen daten vorhanden. im sql server query analyzer können sie nun das abfragefenster für eins schließen und damit diese verbindung trennen.

ändern sie zum test einen datensatz in tabellea von zwei und einen datensatz in tabellea von eins:

```
update tabellea
set spa3 = 'neuer wert für 1003' where spa1=1003
execute eins.trana.dbo.updatespa1 1004,'neuer wert für 1004'
```

rufen sie die ergebnisse ab:

```
select * from tabellea
select * from eins.trana.dbo.tabellea
```

tabellea auf zwei enthält jetzt die werte:

```
spa1        spa2        spa3
----------- ----------- --------------------
1001        11001       eins
1002        11002       zwei
1003        11003       neuer wert für 1003
1004        11004       vier
1005        11005       fünf
```

das ergebnis für tabellea auf eins lautet:

```
spa1        spa2        spa3
----------- ----------- --------------------
1001        11001       eins
1002        11002       zwei
1003        11003       drei
1004        11004       neuer wert für 1004
1005        11005       fünf
```

die folgende anweisung provoziert einen fehler bei der ausführung der gespeicherten prozedur auf eins:

```
update tabellea
set spa3 = 'neuer wert für 1002' where spa1=1002
```

```
execute eins.trana.dbo.updatespa1 'x1005','neuer wert für 1004'
```

wie sieht nun das ergebnis aus? in tabellea von zwei wurde der neue wert ordnungsgemäß eingetragen:

```
spa1        spa2        spa3
----------- ----------- --------------------
1001        11001       eins
1002        11002       neuer wert für 1002
1003        11003       neuer wert für 1003
1004        11004       vier
1005        11005       fünf
```

tabellea auf eins hat natürlich aufgrund des fehlers keine änderung erfahren:

```
spa1        spa2        spa3
----------- ----------- --------------------
1001        11001       eins
1002        11002       zwei
1003        11003       drei
1004        11004       neuer wert für 1004
1005        11005       fünf
```

dieser test sollte demonstrieren, daß sich anweisungen problemlos auch für mehrere computer ausführen lassen. allerdings hat die fehlerhafte ausführung der anweisung für eins keinerlei einfluß auf das ergebnis für zwei. aus diesem grund faßt man derartige anweisungen in einer transaktion zusammen. ergänzen sie die schlüsselwörter begin transaction und commit transaction, und führen sie die folgende anweisung aus:

```
use trana
go

begin transaction
  update tabellea
  set spa3 = 'neuer wert für 1001' where spa1=1001
  execute eins.trana.dbo.updatespa1 1005,'neuer wert für 1005'
commit transaction
go
```

sql server liefert daraufhin folgende fehlermeldung:

```
server: nachr.-nr. 8501, schweregrad 16, status 3, zeile 5
msdtc auf server 'zwei' ist nicht verfügbar.
```

bisher konnten sie die anweisungen über beide computer ausführen, weil es sich gar nicht um eine (explizite) transaktion gehandelt hat. sobald sie aber die anweisungen tatsächlich als transaktion ausführen wollen, erkennt sql server, daß es sich um eine verteilte transaktion handelt, und weigert sich, diese transaktion ohne koordinierung durch eine »höhere instanz« - nämlich durch den microsoft distributed coordinator - abzuwickeln.

18.11.1 distributed transaction coordinator

verteilte transaktionen müssen genau wie »normale« transaktionen den acid-regeln genügen. der mit der version 6.5 von sql server eingeführte distributed transaction coordinator (dtc) ist das werkzeug, das die entsprechende funktionalität unterstützt.

der distributed transaction coordinator besteht aus zwei komponenten: dem ressource-manager und dem transaktions-manager. während sich der ressource-manager um die von einer verteilten transaktion beanspruchten ressourcen kümmert, ist der transaktions-manager für die abwicklung der transaktion - sprich commit oder rollback - verantwortlich.

da an einer verteilten transaktion mehrere ressource-manager beteiligt sind, muß der abschlußvorgang einer transaktion nach einem speziellen schema - dem sogenannten zweiphasencommit - erfolgen. wenn sich nämlich nicht alle teile einer transaktion fehlerfrei ausführen lassen, ist die gesamte transaktion rückgängig zu machen. das wäre aber äußerst schwierig zu realisieren, wenn bereits einige ressource-manager ihren teil der verteilten transaktion mit commit abgeschlossen hätten. deshalb fordert der transaktions-manager in der vorbereitungsphase jeden ressource-manager auf, alle für einen commit erforderlichen arbeiten zu erledigen. erhält der transaktions-manager in der zweiten phase - der commit-phase von allen ressource-managern positive rückmeldungen, sendet der transaktions-manager schließlich an alle ressource-manager den endgültigen commitbefehl, andernfalls einen rollbackbefehl.

18.11.2 remoteserver einrichten

bevor die einzelnen server an einer verteilten transaktion teilnehmen können, muß man sie als remoteserver konfigurieren. führen sie dazu im enterprise manager folgende schritte aus:

1. erweitern sie die konsolenstruktur bis zum ordner **sicherheit**, erweitern sie diesen ordner, und klicken sie mit der rechten maustaste auf den eintrag **remoteserver**.
2. wählen sie aus dem kontextmenü den befehl **neuer remoteserver**. daraufhin wird das dialogfeld **remoteservereigenschaften** geöffnet (siehe abbildung 18.4).

abbildung 18.4: das dialogfeld remoteservereigenschaften

3. im feld **name** geben sie den namen des remoteservers ein. abbildung 18.4 zeigt das dialogfeld für computer eins, auf dem der remoteserver zwei festzulegen ist. schalten sie das kontrollkästchen rpc (für remote procedure call) ein. das ist unbedingt erforderlich, weil dtc über diese methode mit den remoteservern kommuniziert. die anmeldeinformationen lassen sich am einfachsten einstellen, wenn sie die erste option wählen und die vorgegebene einstellung <gleicher name> beibehalten. bei der zweiten option können sie den benutzernamen des remotecomputers einem lokalen benutzernamen zuordnen. der benutzer muß sowohl auf dem remotecomputer als auch dem lokalen benutzer gültig sein.
4. klicken sie auf **ok**, um den remoteserver zu übernehmen. wiederholen sie diese schritte für alle computer, die an einer verteilten transaktion per dtc teilnehmen sollen.

wenn sie zum beispiel verteilte transaktionen mit zwei computern ausführen, muß auf jedem der beiden computer der jeweils andere computer als remoteserver definiert sein. haben sie auf zwei den computer eins als remoteserver eingerichtet, aber auf eins noch nicht den computer zwei als remoteserver festgelegt, erhalten sie folgende fehlermeldung, falls sich eine transaktion über beide computer erstreckt:

```
verbindung zum server 'eins' konnte nicht hergestellt werden,
da 'zwei' nicht als remoteserver definiert ist.
```

18.11.3 dtc starten, anhalten und konfigurieren

der microsoft distributed transaction coordinator wird im rahmen der installation von sql server automatisch installiert. in der programmgruppe **microsoft sql server 7.0** finden sie den eintrag **msdtc admin-konsole**. wenn sie diesen befehl ausführen, erscheint das dialogfeld **dtc-verwaltungskonsole** (siehe abbildung 18.5).

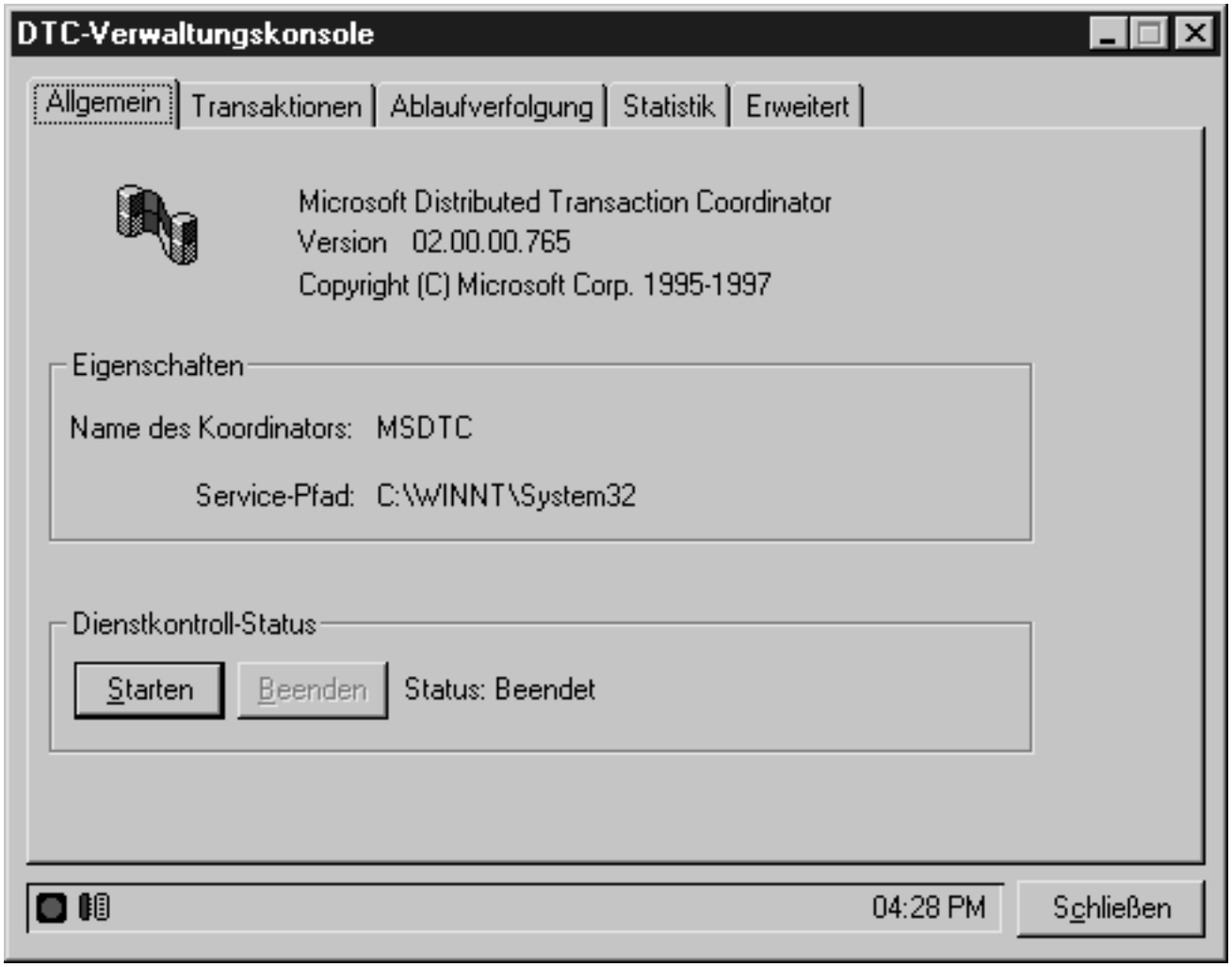


abbildung 18.5: das dialogfeld dtc-verwaltungskonsole

auf der registerkarte **allgemein** können sie im abschnitt **dienstkontroll-status** den msdtc-dienst starten, falls er nicht beim start des systems ohnehin automatisch gestartet wird.

der dienst msdtc läßt sich auch über den sql server dienst-manager starten. wählen sie dazu den dienst msdtc aus, und klicken sie auf **starten/fortsetzen**. weitere hinweise zum dienst-manager finden sie in kapitel 2.

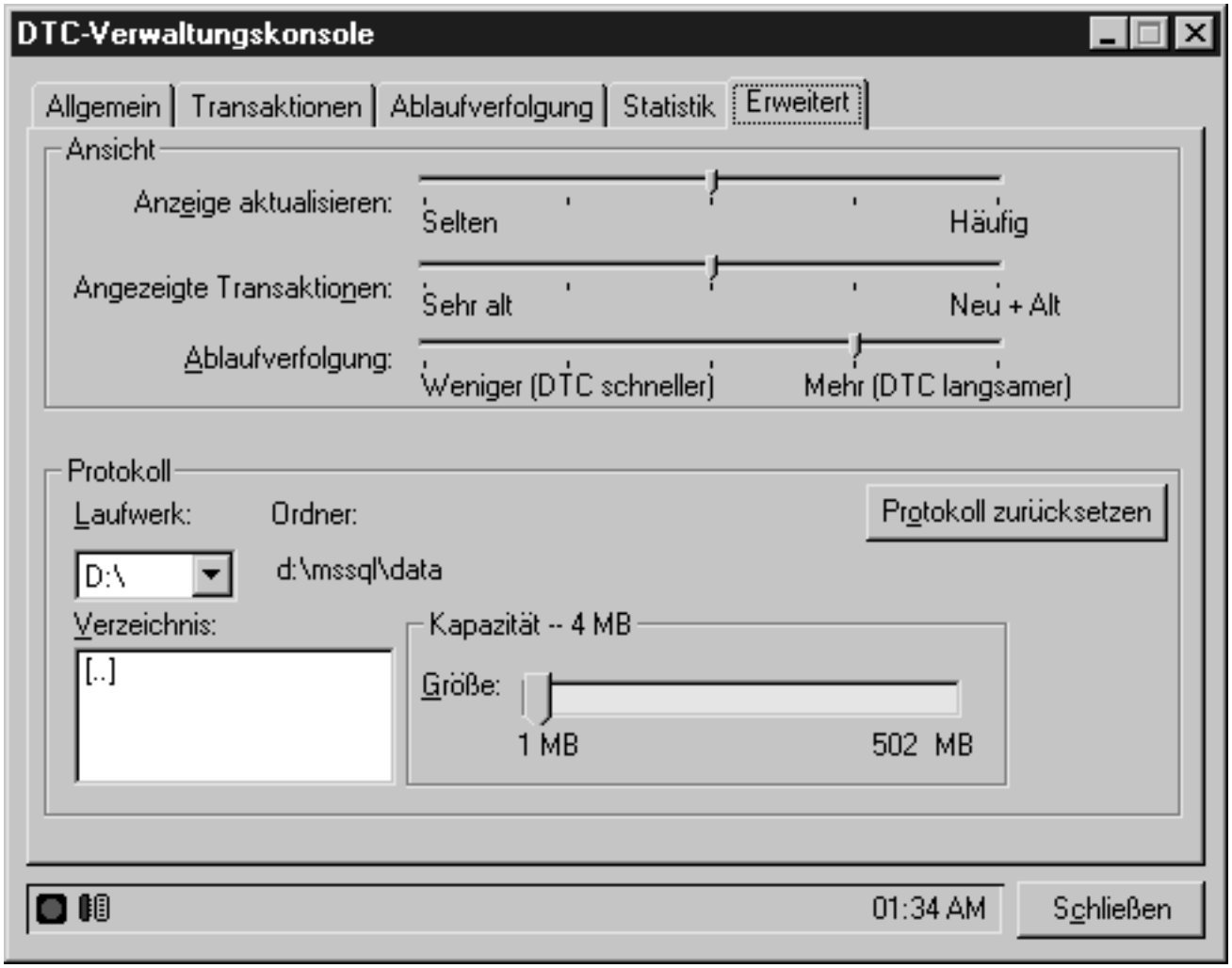


abbildung 18.6: die registerkarte erweitert der dtc-verwaltungskonsole

das dialogfeld **dtc-verwaltungskonsole** enthält außerdem folgende registerkarten:

- **transaktionen**: zeigt informationen zu allen offenen transaktionen an.
- **ablaufverfolgung**: zeigt meldungen an, die der dienst msdtc generiert.
- **statistik**: diese registerkarte zeigt statistische informationen zu einer aktiven transaktion an.
- **erweitert**: auf dieser registerkarte (siehe abbildung 18.6) lassen sich verschiedene dtc-optionen einstellen, die sich auf die überwachung des servers beziehen.

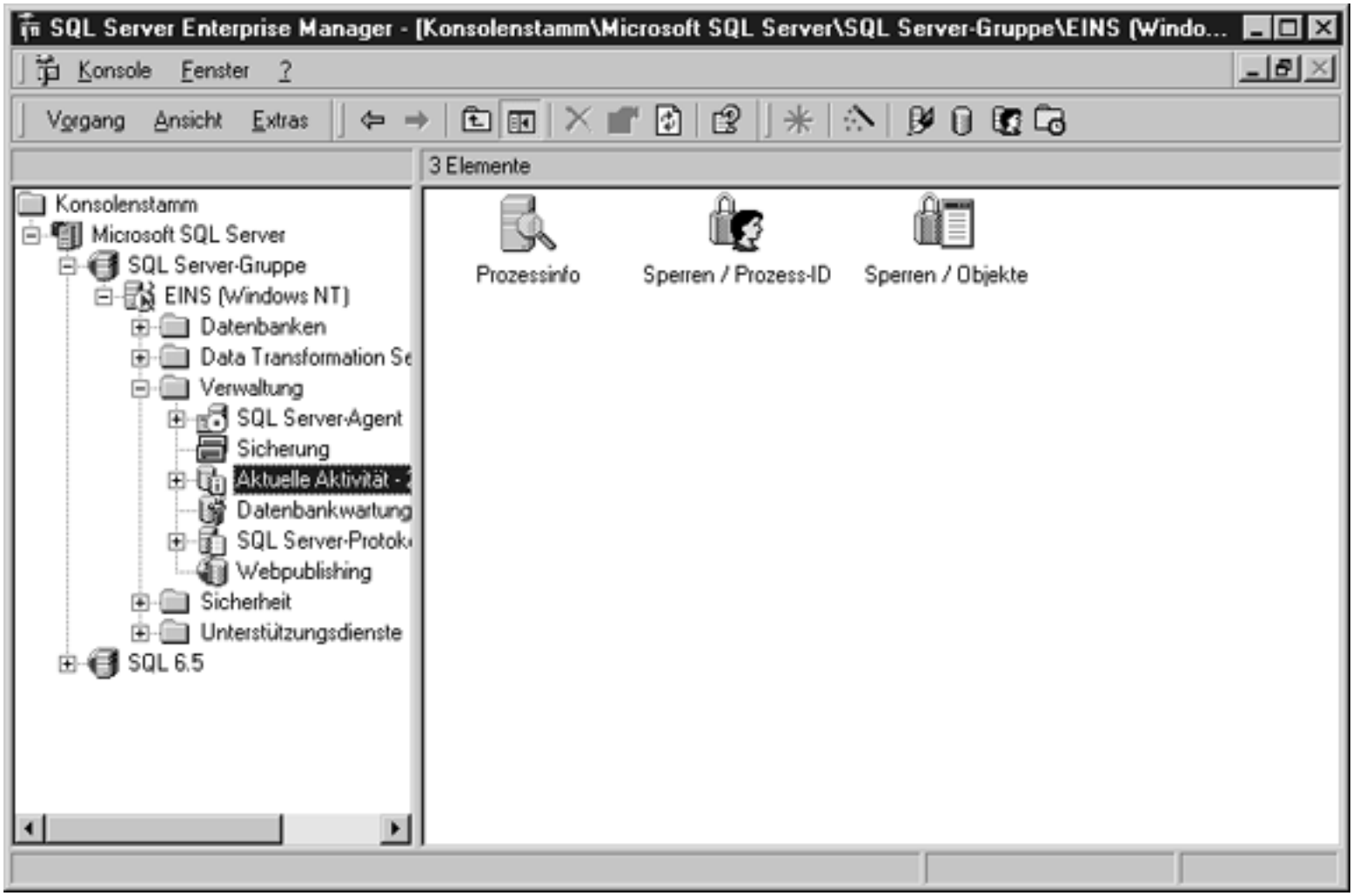
18.11.4 verteilte transaktionen ausführen

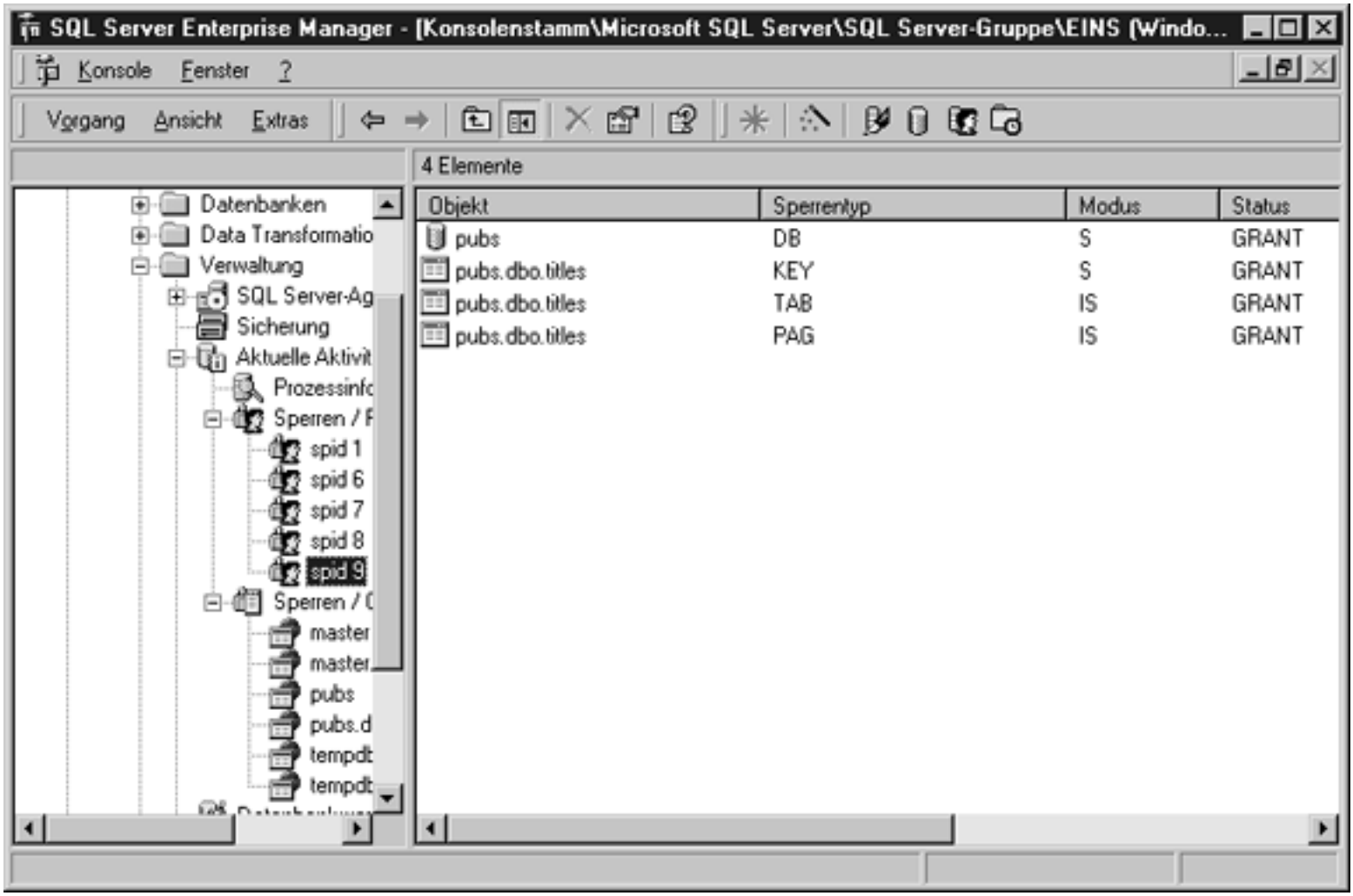
abschließend führen sie die am beginn dieses abschnitts angegebene anweisung als verteilte transaktion aus. wie bereits erwähnt, steuert derjenige server die verteilte transaktion, der sie mit begin distributed transaction einleitet. ergänzen sie also im beispiel das schlüsselwort distributed:

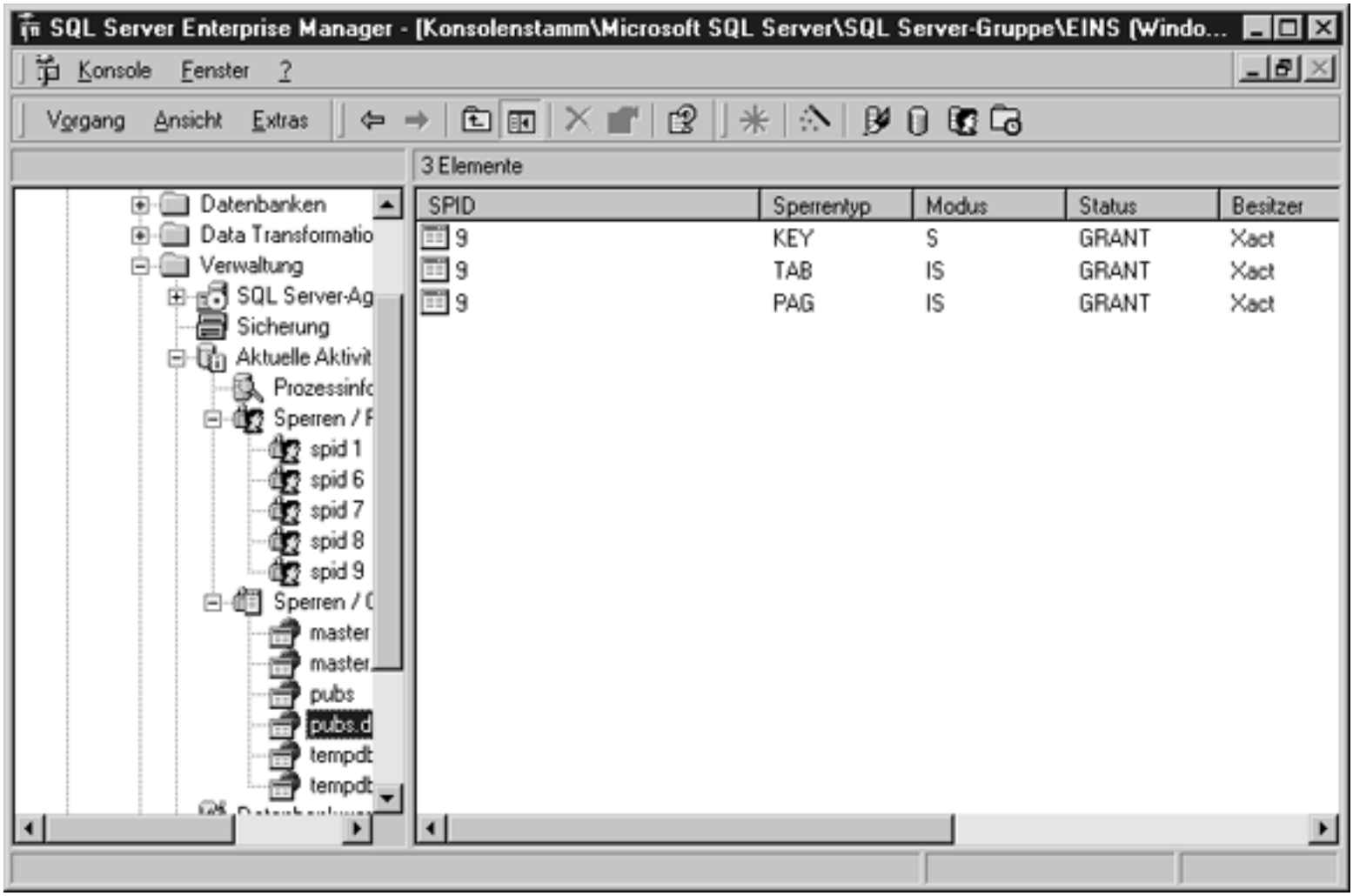
```
use trana
go

begin distributed transaction
  update tabellea
```

```
  set spa3 = 'neuer wert für 1001' where spa1=1001
  execute eins.trana.dbo.updatespa1 1005,'neuer wert für 1005'
commit transaction
go
```

normalerweise sieht man auch bei einer verteilten transaktion einen test vor, ob die anweisungen fehlerfrei ausgeführt werden konnten. im fehlerfall führen sie dann eine rollback-anweisung für die gesamte transaktion aus.


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windo...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
3 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation Se
Verwaltung
SQL Server-Agent
Sicherung
Aktuelle Aktivität -
Datenbankwartung
SQL Server-Protok
Webpublishing
Sicherheit
Unterstützungsdienste
SQL 6.5
Prozessinfo
Sperren / Prozess-ID
Sperren / Objekte


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windo...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
4 Elemente
Datenbanken
Data Transformatio
Verwaltung
SQL Server-Ag
Sicherung
Aktuelle Aktivit
Prozessinfo
Sperren / P
spid 1
spid 6
spid 7
spid 8
spid 9
Sperren / D
master
master
pubs
pubs.d
tempdb
tempdb
Objekt
Sperrentyp
Modus
Status
pubs
DB
S
GRANT
pubs.dbo.titles
KEY
S
GRANT
pubs.dbo.titles
TAB
IS
GRANT
pubs.dbo.titles
PAG
IS
GRANT


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windo...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
3 Elemente
Datenbanken
Data Transformatio
Verwaltung
SQL Server-Ag
Sicherung
Aktuelle Aktivit
Prozessinfo
Sperren / P
spid 1
spid 6
spid 7
spid 8
spid 9
Sperren / D
master
master
pubs
pubs.d
tempdb
tempdb
SPID
Sperrentyp
Modus
Status
Besitzer
9
KEY
S
GRANT
Xact
9
TAB
IS
GRANT
Xact
9
PAG
IS
GRANT
Xact

Kapitel 19 Der Systemkatalog

19.1 Mit System

In relationalen Datenbanken sind alle Informationen in Tabellen gespeichert. Nach der 4. Regel von Codd (siehe Kapitel 1) ist auch die Datenbankstruktur in Tabellen abzulegen. Neben dem RDBMS selbst sind die Informationen der Systemkataloge auch für Sichten und dynamische Anwendungen erforderlich. Es sind gegebenenfalls folgende Informationen gefragt:

- Anzahl und Namen der Tabellen und Ansichten in einer Datenbank.
- Anzahl der Spalten in einer Tabelle oder Ansicht, dazu die Spaltennamen, Datentypen, Wertebereiche und Genauigkeiten.
- Die auf einer Tabelle definierten Bedingungen.
- Die für eine Tabelle definierten Indizes und Schlüssel.

Die in den Systemkatalogen enthaltenen Daten sind sogenannte *Metadaten*, die die Attribute von Objekten in einem System beschreiben. Die Systemkataloge von SQL Server bestehen aus einem Satz von Systemtabellen mit Metadaten, die Informationen über Datenbanken und Datenbankobjekte liefern. Auf die Informationen in den Systemkatalogen können Anwendungen nach verschiedenen Methoden zugreifen:

- Informationsschemasichten
- Gespeicherte Systemprozeduren und -funktionen.
- OLE-DB-Schema-Rowsets
- ODBC-Katalogfunktionen

Die beiden ersten Verfahren stellt SQL Server selbst bereit. Generell empfiehlt es sich nicht, direkt auf die Systemtabellen zuzugreifen. Erstens kann sich die Struktur dieser Tabellen in zukünftigen Versionen von SQL Server ändern, zweitens können unpassende oder fehlerhafte Änderungen das gesamte Datenbanksystem unbrauchbar machen. Wenn Sie mit den oben genannten Verfahren auf die Systemtabellen zugreifen, wirken sich strukturelle Änderungen der zugrundeliegenden Tabellen nicht auf Ihre Anwendungen aus, und die Anwendungen sind zwischen heterogenen SQL-92-kompatiblen Datenbanksystemen portabel.

19.2 Systemtabellen

Die Tabellen des Systemkatalogs sind für die gesamte Funktion von SQL Server lebenswichtig. Deshalb weist die Online-Dokumentation immer wieder darauf hin, daß man die Tabellen nicht direkt - insbesondere nicht mit den Anweisungen DELETE, UPDATE oder INSERT sowie benutzerdefinierten Triggern - ändern sollte.

Während der Entwicklung einer Datenbank ist es manchmal aber hilfreich oder unumgänglich, Informationen ohne Umwege aus einer Systemtabelle abzurufen. Wie Kapitel 17 gezeigt hat, kann man zum Beispiel die für eine Datenbank definierten Trigger mit Hilfe der Systemtabelle sysobjects auflisten.

Nach dem Ort der Speicherung lassen sich die Systemtabellen folgenden Kategorien zuordnen:

- Systemtabellen nur in der master-Datenbank
- Systemtabellen in jeder Datenbank
- SQL-Server-Agent-Tabellen in der msdb-Datenbank
- Tabellen in der msdb-Datenbank
- für Replikationsinformationen verwendete Tabellen in der master-Datenbank, in der distribution-Datenbank und in der Datenbank des Benutzers

Insgesamt sind das über 100 Tabellen. Die nächsten Abschnitte konzentrieren sich auf einige wichtige Systemtabellen aus diesem umfangreichen Fundus. Die meisten dieser Tabellen bilden auch die Grundlage für die Informationsschemasichten, die im Anschluß behandelt werden.

19.2.1 Datenbankobjekte - sysobjects

Die Systemtabelle sysobjects gehört zu den wichtigsten und am meisten abgefragten Tabellen des Systemkatalogs. Sie gibt Auskunft über die Objekte, die in einer bestimmten Datenbank enthalten sind.

Tabelle 19.1 zeigt die wichtigsten Spalten der Systemtabelle sysobjects.

Spalte	Datentyp	Beschreibung
name	sysname	Objektname
Id	int	Objekt-ID
xtype	char(2)	Objekttyp mit den Werten: C - CHECK-Einschränkung D - Standardeinschränkung F - Fremdschlüsseleinschränkung L - Protokoll P - gespeicherte Prozedur PK - Primärschlüsseleinschränkung RF - gespeicherte Replikationsfilterprozedur S - Systemtabelle TR - Trigger U - Benutzertabelle UQ - UNIQUE-Einschränkung V - Sicht X - erweiterte gespeicherte Prozedur
uid	smallint	Benutzername des Besitzerobjekts
crdate	datetime	Datum der Objekterstellung

type	char(2)	ähnlich wie xtype, allerdings mit zum Teil abweichenden Kennbuchstaben

Tabelle 19.1: Die Systemtabelle sysobjects

Das folgende Beispiel listet Name und Erstellungsdatum aller Trigger der Datenbank pubs auf:

```
use pubs
select name, crdate from sysobjects
where xtype='tr'
```

Die Ergebnismenge sieht etwa wie folgt aus (zur »jungfräulichen« Datenbank pubs gehört nur der Trigger employee_insupd):

```
name                 crdate
-------------------- --------------------------
employee_insupd      1999-04-15 04:07:59.927
Webseite Verleger_1  1999-04-27 14:56:55.973
Webseite Verleger_2  1999-04-27 14:56:55.983
Webseite Verleger_4  1999-04-27 14:56:55.983

(4 row(s) affected)
```

19.2.2 Installierte Datenbanken - sysdatabases

In der Tabelle sysdatabases sind die in SQL Server gespeicherten Datenbanken erfaßt. Tabelle 19.2 zeigt die wesentlichen Spalten. Die Tabelle ist nur in der Systemdatenbank master vorhanden.

Spalte	Datentyp	Beschreibung
name	sysname	Name der Datenbank
dbid	smallint	ID der Datenbank
sid	varbinary(149)	System-ID des Datenbankerstellers
status status2	int	Statusbits, zum Teil mit der gespeicherten Prozedur sp_dboption einstellbar
crdate	datetime	Erstellungsdatum

category	int	Bitmuster mit Informationen für die Replikation: 1 - publiziert 2 - abonniert 4 - Mergepublikation 8 - Mergeabonnement
filename	nvarchar(260)	Betriebssystempfad und -name für die primäre Datei der Datenbank
suid	smallint	ID des SQL-Server-Benutzernamens, der die Datenbank besitzt

Tabelle 19.2: Die Systemtabelle sysdatabases

Mit der Anweisung

```
use master
select name, crdate, filename from sysdatabases
```

rufen Sie Name, Erstellungsdatum und Pfad/Name der primären Datei für die installierten Datenbanken ab.

Die Ergebnismenge sieht etwa folgendermaßen aus (die Ausgabe wurde aus satztechnischen Gründen gekürzt):

```
name           crdate                  filename
-------------- ----------------------- --------------------------
distribution 1999-04-27 15:39:40.790 d:\MSSQL7\data\distribu...
lotto        1999-06-11 19:51:19.100 d:\MSSQL7\data\lotto.mdf
master       1998-11-13 03:00:18.500 d:\MSSQL7\DATA\MASTER.MDF
model        1999-03-10 12:05:18.013 d:\MSSQL7\DATA\model.mdf
msdb         1999-03-10 12:05:19.007 d:\MSSQL7\DATA\msdbdata...
Northwind    1999-03-10 12:05:20.157 d:\MSSQL7\DATA\northwnd...
pubs         1999-04-15 04:07:56.700 d:\MSSQL7\data\pubs.mdf
tempdb       1999-06-14 19:00:32.750 d:\MSSQL7\DATA\TEMPDB.MDF

(8 row(s) affected)
```

Systemdatentypen - systypes

Jede Datenbank enthält die Tabelle systypes, in der alle Datentypen und benutzerdefinierten Datentypen erfaßt sind. Tabelle 19.3 zeigt die wichtigsten Spalten. In systypes ist unter anderem auch der Datentyp sysname verzeichnet, den viele Systemtabellen selbst verwenden.

Spalte	Datentyp	Beschreibung
name	sysname	Name des Datentyps
xtype	tinyint	physischer Speichertyp
xusertype	smallint	erweiterter Benutzertyp
length	smallint	physische Länge des Datentyps
tdefault	int	ID der gespeicherten Prozedur, die eine Integritätsprüfung für diesen Datentyp vornimmt
domain	int	ID der gespeicherten Prozedur, die eine Integritätsprüfung für diesen Datentyp vornimmt
uid	smallint	Benutzername des Datentyperstellers
usertype	smallint	Benutzertyp-ID
variable	bit	1 bei Datentypen variabler Länge, 0 bei Datentypen fester Länge
allownulls	bit	NULL-Zulässigkeit des Datentyps. Dieser Wert hat Vorrang gegenüber der Standard-NULL-Zulässigkeit des Datentyps, die Sie mit den Anweisungen CREATE TABLE oder ALTER TABLE festlegen.
type	tinyint	physischer Datenspeichertyp
prec	smallint	Genauigkeit für diesen Datentyp
scale	tinyint	Dezimalstellen für diesen Datentyp

Tabelle 19.3: Die Systemtabelle systypes

Die folgende Anweisung liefert einen Überblick über Name, Länge, Längentyp (variabel, fest), Speichertyp, Genauigkeit und Anzahl der Dezimalstellen für alle Datentypen, die in der Datenbank pubs definiert sind. Diese Informationen sind unter anderem bei der Fehlersuche in einer Datenbank nützlich, etwa um festzustellen, warum ein benutzerdefinierter Datentyp für den vorgesehenen Zweck nicht ausreichend ist.

```
use pubs
select name, length, variable, type, prec, scale from systypes

name                        length variable type prec   scale
--------------------------- ------ -------- ---- ------ -----
binary                      8000   0        45   8000   NULL
bit                         1      0        50   1      0
char                        8000   0        47   8000   NULL
datetime                    8      0        61   23     3
decimal                     17     0        55   38     38
empid                       9      0        47   9      NULL
```

```
...
uniqueidentifier          16     0        36   16     NULL
varbinary                 8000   1        37   8000   NULL
varchar                   8000   1        39   8000   NULL

(27 row(s) affected)
```

Tabellenspalten - syscolumns

Die in jeder Datenbank vorhandene Systemtabelle syscolumns enthält Einträge für die Spalten aller Tabellen und Sichten sowie die Parameter einer Systemprozedur. Tabelle 19.4 zeigt die wichtigsten Spalten dieser Systemtabelle.

Spalte	Datentyp	Beschreibung
name	sysname	Spaltenname oder Parametername einer Prozedur
id	int	Objekt-ID der Tabelle (bei Spalten) oder ID der gespeicherten Prozedur (bei Parametern)
xtype type	tinyint	physischer Speichertyp entsprechend der Systemtabelle systypes (siehe Tabelle 19.3)
length	smallint	maximale physische Speicherlänge aus systypes
colid	smallint	ID der Spalte oder des Paramters
cdefault	int	ID des Standardwertes für die Spalte
domain	int	ID der Regel oder der CHECK-Einschränkung für die Spalte
number	smallint	Nummer der Unterprozedur bei gruppierten Prozeduren (siehe Kapitel 12, »Prozeduren gruppieren«) 0 bei Einträgen, die sich nicht auf eine Prozedur beziehen
status	tinyint	Bitmuster, das folgende Eigenschaften der Spalte oder des Parameters beschreibt: 0x08 - NULL-Werte sind zulässig 0x10 - ANSI-Zeichenauffüllung war beim Hinzufügen von varchar- oder varbinary-Spalten aktiviert 0x40 - Parameter ist OUTPUT-Parameter 0x80 - Spalte ist Identitätsspalte
prec	smallint	Genauigkeit der Spalte
scale	int	Dezimalstellen der Spalte
iscomputed	int	1 - berechnete Spalte 0 - nicht berechnete Spalte

isoutparam	int	1 - Parameter der Prozedur ist Ausgabeparameter 0 - kein Ausgabeparameter
isnullable	int	1 - Spalte läßt NULL-Werte zu 0 - Spalte läßt keine NULL-Werte zu

Tabelle 19.4: Die Systemtabelle syscolumns

Informationen zur Systemtabelle syscolumns lassen sich mit der gespeicherten Prozedur sp_columns abrufen.

Definitionstabelle - syscomments

Die Systemtabelle syscomments verzeichnet den Text der Definition verschiedener Datenbankobjekte. Die Tabelle enthält Einträge für alle Sichten, Regeln, Standardwerte, Trigger, CHECK-Einschränkungen, DEFAULT-Einschränkungen und gespeicherte Prozeduren. Insbesondere schreibt SQL Server in diese Systemtabelle die Definitionen von Prozeduren und Triggern, und zwar auf Wunsch verschlüsselt. Tabelle 19.5 zeigt die wesentlichen Spalten von syscomments.

Spalte	Datentyp	Beschreibung
id	int	ID des Objekts, auf das sich die Spalte text bezieht
number	smallint	Nummer der Unterprozedur bei gruppierten Prozeduren (siehe Kapitel 12, »Prozeduren gruppieren«) 0 bei Einträgen, die sich nicht auf eine Prozedur beziehen
ctext	varbinary(8000)	tatsächlicher Text der SQL-Definitionsanweisungen
texttype	smallint	0 - Benutzerkommentar 1 - Systemkommentar 4 - verschlüsselter Kommentar
encrypted	bit	0 - Prozedur nicht verschlüsselt 1 - Prozedur verschlüsselt
compressed	bit	0 - Prozedur nicht komprimiert 1 - Prozedur komprimiert
text	nvarchar(4000)	tatsächlicher Text der SQL-Definitionsanweisung

Tabelle 19.5: Die Systemtabelle syscomments

Konfigurationsoptionen - sysconfigures

Die in der Datenbank master gespeicherte Systemtabelle sysconfigures verzeichnet alle vom Benutzer festgelegten Konfigurationsoptionen. Das betrifft alle vor dem letzten Start von SQL Server sowie alle seitdem dynamisch festgelegten Optionen. Tabelle 19.6 zeigt die Spalten dieser Systemtabelle.

Spalte	Datentyp	Beschreibung

value	int	Der vom Benutzer veränderbare Wert der jeweiligen Variablen. SQL Server verwendet diesen Wert erst nach Ausführung von RECONFIGURE.
config	smallint	Nummer der Konfigurationsvariablen
comment	nvarchar(255)	Erläuterung der Konfigurationsoption
status	smallint	Bitmuster für den Status der Option: 0 - statisch (beim Neustart von SQL Server wirksam) 1 - dynamisch (nach Ausführen von RECONFIGURE wirksam) 2 - erweitert (Anzeige nur, wenn show advanced option eingeschaltet) 3 - dynamisch und erweitert

Tabelle 19.6: Die Systemtabelle sysconfigures

Berechtigungen - sysprotects

Die Systemtabelle sysprotects ist in jeder Datenbank vorhanden und liefert Informationen zu Berechtigungen, die mit GRANT- oder DENY-Anweisungen auf Sicherheitskonten angewandt wurden. Tabelle 19.7 zeigt die Spalten dieser Systemtabelle.

Spalte	Datentyp	Beschreibung
id	int	ID des Objekts, auf das sich diese Berechtigungen beziehen
uid	smallint	ID des Benutzers oder der Gruppe
action	tinyint	Gibt die Berechtigungen an: 26 - REFERENCES 193 - SELECT 195 - INSERT 196 - DELETE 197 - UPDATE 198 - CREATE TABLE 203 - CREATE DATABASE 207 - CREATE VIEW 222 - CREATE PROCEDURE 224 - EXECUTE 228 - BACKUP DATABASE 233 - CREATE DEFAULT 235 - BACKUP LOG 236 - CREATE RULE
protecttype	tinyint	Art des Schutzes: 204 - GRANT_W_GRANT 205 - GRANT 206 - REVOKE

columns	varbinary(4000)	Bitmuster der Spalten, denen die Berechtigungen zugeordnet sind: 0x00 - alle Spalten 0x01 - Berechtigungen treffen für diese Spalte zu NULL - keine Angaben
grantor	smallint	Benutzername des Benutzers, der die GRANT- oder REVOKE-Anweisung für die Berechtigung ausgegeben hat

Tabelle 19.7: Die Systemtabelle sysprotects

Abhängigkeiten - sysdepends

Die Systemtabelle sysdepends speichert alle Abhängigkeiten, die zwischen Objekten einer Datenbank bestehen, d.h. die Abhängigkeit der Sichten, Prozeduren und Trigger von den in ihrer Definition genannten Tabellen, Sichten und Prozeduren. Tabelle 19.8 gibt die relevanten Spalten dieser Systemtabelle an.

Spalte	Datentyp	Beschreibung
id	int	Objekt-ID
depid	int	ID eines abhängigen Objekts
number	smallint	Prozedurnummer
depnumber	smallint	Nummer einer abhängigen Prozedur
selall	bit	1, wenn Objekt in einer Anweisung SELECT * verwendet wird
resultobj	bit	1, wenn das Objekt aktualisiert wird
readobj	bit	1, wenn das Objekt gelesen wird

Tabelle 19.8: Die Systemtabelle sysdepends

Einschränkungen - sysconstraints

Die Systemtabelle sysconstraints ist in jeder Datenbank vorhanden und verzeichnet die Zuordnung von Einschränkungen zu Objekten. Die wesentlichen Spalten zeigt Tabelle 19.9.

Spalte	Datentyp	Beschreibung
constid	int	Nummer der Einschränkung
id	int	ID der Tabelle, für die diese Einschränkung gilt
colid	smallint	ID der Spalte, für die diese Einschränkung gilt. 0, wenn sich die Einschränkung auf die gesamte Tabelle bezieht

status	int	Art der Einschränkung: 1 - PRIMARY KEY 2 - UNIQUE KEY 3 - FOREIGN KEY 4 - CHECK 5 - DEFAULT 16 - auf Spaltenebene 32 - auf Tabellenebene

Tabelle 19.9: Die Systemtabelle sysconstraints

Beziehungen - sysreferences

Die in jeder Datenbank gespeicherte Systemtabelle sysreferences enthält Zuordnungen von FOREIGN KEY-Einschränkungen zu den referenzierten Spalten. Tabelle 19.10 zeigt die Spalten dieser Systemtabelle.

Spalte	Datentyp	Beschreibung
constid	int	ID der FOREIGN KEY-Einschränkung
fkeyid	int	ID der verweisenden Tabelle
rkeyid	int	ID der referenzierten Tabelle
rkeyindid	smallint	Index-ID des eindeutigen Index auf der referenzierten Tabelle, der die referenzierten Schlüsselspalten abdeckt
keycnt	smallint	Anzahl der Spalten im Schlüssel
fkey1 bis fkey16	smallint	Spalten-ID der verweisenden Spalte

Tabelle 19.10: Die Systemtabelle sysreferences

Indizes - sysindexes

Die in jeder Datenbank vorhandene Systemtabelle sysindexes enthält Angaben zu allen Indizes und Tabellen in der Datenbank. Tabelle 19.11 zeigt die wichtigsten Spalten dieser Systemtabelle.

Spalte	Datentyp	Beschreibung
id	int	ID der Tabelle (bei indid = 0 oder 255), sonst ID der Tabelle, zu der der Index gehört

status	int	interne Statusinformationen des Systems: 1 - Befehl abbrechen beim Versuch, doppelten Schlüssel einzufügen 2 - eindeutiger Index 4 - Befehl abbrechen beim Versuch, doppelte Zeilen einzufügen 16 - gruppierter Index 64 - Index erlaubt doppelte Zeilen 2048 - Index für PRIMARY KEY-Einschränkung verwendet 4096 - Index für UNIQUE-Einschränkung verwendet
indid	smallint	ID des Index: 1 - gruppierter Index >1 - nicht gruppierter Index 255 - Eintrag für Tabellen, die text- oder image-Spalten enthalten
keycnt	smallint	Anzahl der Schlüssel
dpages	int	bei indid = 0 oder 1: Anzahl der verwendeten Datenseiten bei indid = 255 wird dpages auf 0 gesetzt sonst: Anzahl der Indexseiten
keys	varbinary(816)	IDs der Spalten, aus denen der Indexschlüssel besteht
name	sysname	bei indid = 0 oder 255: Name der Tabelle sonst: Name des Index

Tabelle 19.11: Die Systemtabelle sysindexes

Benutzer - sysusers

Vor allem der Datenbankadministrator dürfte an den Informationen der Systemtabelle sysusers interessiert sein. Sie enthält Einträge für Windows-NT-Benutzer, Windows-NT-Gruppen, SQL-Server-Benutzer und SQL-Server-Rollen in der Datenbank. Tabelle 19.12 gibt die wichtigsten Spalten an.

Spalte	Datentyp	Beschreibung
uid	smallint	in dieser Datenbank eindeutiger Benutzername: 1 - Besitzer der Datenbank
name	sysname	in dieser Datenbank eindeutiger Benutzer- oder Gruppenname
createdate	datetime	Erstellungsdatum des Kontos
updatedate	datetime	letzte Aktualisierung des Kontos
suid	smallint	aus syslogins kopierter Serverbenutzername 1 - Systemadministrator -1 - Gastkonto

gid	smallint	ID der Gruppe, zu der dieser Benutzer gehört bei uid=gid definiert dieser Eintrag eine Gruppe -2 bei public-Rolle bei allen anderen Rollen ist suid = -gid
hasdbaccess	int	1, wenn das Konto Datenbankzugriff hat
islogin	int	1, wenn das Konto eine Windows-NT-Gruppe, ein Windows-NT-Benutzer oder SQL-Server-Benutzer mit einem Anmeldekonto ist
isntname	int	1, wenn das Konto eine Windows-NT-Gruppe oder ein Windows-NT-Benutzer ist
isntgroup	int	1, wenn das Konto eine Windows-NT-Gruppe ist
isntuser	int	1, wenn das Konto ein Windows-NT-Benutzer ist
issqluser	int	1, wenn das Konto ein SQL-Server-Benutzer ist
isaliased	int	1, wenn das Konto als Alias für einen anderen Benutzer vorgesehen ist
issqlrole	int	1, wenn das Konto eine SQL-Server-Rolle ist
isapprole	int	1, wenn das Konto eine Anwendungsrolle ist

Tabelle 19.12: Die Systemtabelle sysusers

Ein Beispiel für den Zugriff auf die Systemtabelle sysusers finden Sie im Skript Instpubs.sql in den Zeilen 912 bis 934 (siehe Anhang B).

Sprachen - syslanguages

Die in SQL Server verfügbaren Sprachen sind in der Systemtabelle syslanguages zeilenweise abgelegt. US-Englisch ist nicht in der Tabelle enthalten, aber jederzeit in SQL Server verfügbar. Interessant dürften in dieser Systemtabelle vor allem die Einstellungen für Datum und Uhrzeit sein.

Spaltenname	Datentyp	Beschreibung und Beispiel
langid	smallint	Eindeutige Sprachen-ID (*0* bis *23*)
dateformat	nchar(3)	Reihenfolge von Tag, Monat und Jahr Deutsch: *dmy*
datefirst	tinyint	Erster Tag der Woche. 1 - Montag bis 7 - Sonntag. Deutsch: *7*
name	sysname	Offizieller Name der Sprache Französisch: *Français*
alias	sysname	Alternativer Name der Sprache Französisch: *French*

months	nvarchar(372)	Liste der Monatsnamen in voller Länge Deutsch: *Januar, Februar, März, ...*
shortmonths	varchar(132)	Liste der abgekürzten Montatsnamen Deutsch: *Jan, Feb, Mär, ...*
days	nvarchar(217)	Liste der Tagesnamen Deutsch: *Montag, Dienstag, Mittwoch, ...*
lcid	int	Gebietsschema-ID von Windows NT Deutsch: *1031*
mslangid	smallint	SQL-Server-Nachrichtengruppe-ID Deutsch: *1031*

Tabelle 19.13: Spalten der Systemtabelle syslanguages

Bezeichnung	NT LCID bzw. SQL-Server-Nachrichtengruppen-ID
Englisch	1033
Deutsch	1031
Französisch	1036
Japanisch	1041
Dänisch	1030
Spanisch	1034
Italienisch	1040
Niederländisch	1043
Norwegisch	2068
Portugiesisch	2070
Finnisch	1035
Schwedisch	1053
Tschechisch	1029
Ungarisch	1038
Polnisch	1045
Rumänisch	1048
Kroatisch	1050
Slowakisch	1051
Slowenisch	1060

Griechisch	1032
Bulgarisch	1026
Russisch	1049
Türkisch	1055

Tabelle 19.14: In SQL Server verfügbare Sprachen

Fehlermeldungen - sysmessages

SQL Server speichert Fehlermeldungen und Warnungen in der Systemtabelle sysmessages. Für jede Meldung ist eine Zeile vorgesehen. Die Tabelle sysmessages der Datenbank master enthält alle Fehlermeldungen und Warnungen des Systems. Tabelle 19.15 zeigt die Spalten dieser Systemtabelle.

Spalte	Datentyp	Beschreibung
error	int	Eindeutige Fehlernummer
severity	smallint	Schweregrad
description	nvarchar(255)	Erläuterung des Fehlers. Die Parameter sind in Form von Platzhaltern (z.B. %1) enthalten.
mslangid	smallint	ID der Sprachversion (siehe Tabelle 19.14)

Tabelle 19.15: Spalten der Systemtabelle sysmessages

Symbol	Bedeutung
%d, %ld oder %D	Dezimale Ganzzahl
%x oder %X	Hexadezimalzahl (klein bzw. groß geschrieben)
%ls oder %.*ls	Zeichenfolge
%S_Typ	Von SQL Server definierte Struktur
%c	Einzelnes Zeichen
%Lf	Gleitkommazahl doppelter Genauigkeit

Tabelle 19.16: Symbole für Platzhalter in der Spalte description

Mit der folgenden Anweisung zeigen Sie den Schweregrad und die Beschreibung für den Fehler 21 an:

```
use master
select severity, description from sysmessages
where error = 21 and msglangid=1031
```

Als Ergebnis erhalten Sie:

```
severity description
-------- --------------------------------------------------
10       Warnung: Fataler Fehler %1! am %2! aufgetreten.
Notieren Sie Fehler und Zeitpunkt, und wenden Sie sich an den
Systemadministrator.

(1 row(s) affected)
```

Benutzerdefinierte Meldungen - sysmessages

In der Systemtabelle sysmessages können Sie benutzerdefinierte Fehlermeldungen bzw. Warnungen ablegen. Dafür stellt SQL Server die gespeicherte Prozedur sp_addmessage bereit:

```
sp_addmessage[@messagenum =] Meldungsnummer,
[@severity =] Schweregrad,
[@msgtext =] 'Meldungstext'
[, [@lang =] 'Sprache']
[, [@with log =] 'MitProtokoll']
[, [@replace =] 'REPLACE']
```

Die Parameter haben folgende Bedeutung:

- @messagenum: Die ID der Meldung. Die Werte für benutzerdefinierte Fehlermeldungen beginnen bei 50001.
- @severity: Schweregrad, Standardwert NULL, zulässige Werte 1 bis 25. Fehlermeldungen mit einem Schweregrad zwischen 19 und 25 kann nur der Systemadministrator hinzufügen.
- @msgtext: Text der Fehlermeldung, Standardwert NULL.
- @lang: Die Sprache für die Meldung. Fehlt diese Angabe, verwendet SQL Server die Standardsprache der Sitzung.
- @with log: Gibt an, ob die Meldung in das Windows-NT-Anwendungsprotokoll geschrieben werden soll (bei true). Standardwert ist false.
- @replace: Wenn eine Meldung unter der Fehlernummer und dem Schweregrad bereits vorhanden ist, müssen Sie die Zeichenfolge REPLACE angeben, um die alte Meldung durch die neue zu ersetzen.

Bevor Sie eine lokalisierte Fehlermeldung - d.h. in einer anderen Sprache als US_ENGLISH - eintragen können, verlangt SQL Server zuerst die englische Version. In der lokalisierten Version ist nach den ersetzbaren Parametern ein Ausrufezeichen zu schreiben. Zwischen Prozentzeichen und Formatcode geben Sie die Nummer des ersetzbaren Parameters an. Die Zahlen müssen von 1 beginnend in ununterbrochener Folge vorkommen. Maximal 20 Parameter sind möglich. Die Reihenfolge der

Parameter kann in der lokalisierten Version von der englischen abweichen.

Das folgende Beispiel fügt eine benutzerdefinierte Fehlermeldung mit der Fehlernummer 50001 und dem Schweregrad 16 in die Systemtabelle sysmessages ein:

```
exec sp_addmessage 50001,16,'dec: %1d; str: %2s; char:%3c; hex:
%4X',us_english,false,replace
exec sp_addmessage 50001,16,'str: %2s!; char:%3c!; dez: %1d!;
hex: %4X!',deutsch,false,replace
```

Das Beispiel geht davon aus, daß bereits eine Meldung mit der Fehlernummer 50001 und dem Schweregrad 16 vorhanden war, so daß die Meldung zu ersetzen ist (replace). Wenn Sie nicht mit benannten Parametern arbeiten, ist auch ein Wert (true oder false) an der Position @with log anzugeben, da das Schlüsselwort REPLACE erst danach folgt. Beachten Sie, daß die Reihenfolge der ersetzbaren Parameter in der lokalisierten (deutschen) Version von der englischen abweichen darf.

Wenn Sie die Meldung des obigen Beispiels anzeigen, können Sie vier Parameter ersetzen: eine Ganzzahl, einen String, ein einzelnes Zeichen und eine Hexadezimalzahl.

Die Anweisung FORMATMESSAGE gibt eine benutzerdefinierte Meldung aus und ersetzt dabei die übergebenen Parameter:

```
formatmessage(Meldungsnummer, Parameterliste)
```

In der Parameterliste müssen die Parameter in der Reihenfolge erscheinen wie sie in der englischen Version der Meldung numeriert sind, selbst wenn die Parameter in der lokalisierten Meldung an einer anderen Stelle im Meldungstext vorkommen. Im Meldungstext des obigen Beispiels sind vier Parameter zu ersetzen. Beachten Sie, daß die Parameter entsprechend der englischen Version zu übergeben sind, auch wenn im deutschen Meldungstext die übergebenen Werte an anderer Stelle erscheinen. Die Anweisung

```
print formatmessage(50001,1E5,'abc',65+4,255)
```

liefert demnach die Ausgabe:

```
str: abc; char:  E; dez: 100000; hex:   FF
```

Eine benutzerdefinierte Meldung löschen Sie mit der gespeicherten Prozedur sp_dropmessage:

```
sp_dropmessage [@msgnum =] Meldungsnummer
[, [@lang =] 'Sprache']
```

Als Sprache können Sie eine bestimmte Sprache wie deutsch angeben oder mit 'all' alle Sprachversionen löschen. Bevor Sie die englische Version löschen, müssen Sie alle anderen lokalisierten Meldungen löschen.

Die Anweisung

```
exec sp_dropmessage 50001,'all'
```

löscht alle Sprachversionen der benutzerdefinierten Fehlermeldung mit der Nummer 50001.

19.3 Informationsschema-Sichten

SQL Server schirmt die Systemtabellen mit einer Reihe von Schemasichten ab, die mit dem SQL-92-Standard kompatibel sind. Tabelle 19.17 gibt einen Überblick über diese Schemasichten und nennt auch die zugrundeliegenden Systemtabellen.

Sicht	Beschreibung	Basiert auf
CHECK_ CONSTRAINTS	Liefert Informationen zu CHECK-Einschränkungen.	sysobjects syscomments
COLUMN_ DOMAIN_USAGE	Liefert Informationen zu benutzerdefinierten Datentypen.	sysobjects syscolumns systypes
COLUMN_ PRIVILEGES	Liefert Informationen zu Benutzerberechtigungen.	sysprotects sysobjects syscolumns
COLUMNS	Liefert Informationen zu den Spalten, auf die der Benutzer in der aktuellen Datenbank zugreifen kann.	sysobjects spt_data type_info systypes syscolumns syscomments sysconfigures syscharsets

CONSTRAINT_ COLUMN_USAGE	Liefert Informationen zu den Spalten, die der Benutzer in der aktuellen Datenbank besitzt und für die Einschränkungen definiert sind.	sysobjects syscolumns systypes
CONSTRAINT_ TABLE_USAGE	Liefert Informationen zu den Tabellen, die der Benutzer in der aktuellen Datenbank besitzt und für die Einschränkungen definiert sind.	sysobjects
DOMAIN_ CONSTRAINTS	Liefert Informationen zu benutzerdefinierten Datentypen, auf die der aktuelle Benutzer in der aktuellen Datenbank zugreifen kann und die mit Regeln verbunden sind.	sysobjects systypes
DOMAINS	Liefert Informationen zu benutzerdefinierten Datentypen, auf die der aktuelle Benutzer in der aktuellen Datenbank zugreifen kann.	spt_data type info systypes syscomments sysconfigures syscharsets
KEY_COLUMN_ USAGE	Liefert Informationen zu Spalten mit Schlüsseleinschränkungen, die der aktuelle Benutzer in der aktuellen Datenbank besitzt.	sysobjects syscolumns sysreferences spt_values sysindexes
REFERENTIAL_ CONSTRAINTS	Liefert Informationen zu Fremdeinschränkungen, die der aktuelle Benutzer in der aktuellen Datenbank besitzt.	sysreferences sysindexes sysobjects
SCHEMATA	Liefert Informationen zu den Datenbanken, für die der aktuelle Benutzer Berechtigungen besitzt.	sysdatabases sysconfigures syscharsets
TABLE_ CONSTRAINTS	Liefert Informationen zu Tabelleneinschränkungen, die der aktuelle Benutzer in der aktuellen Datenbank besitzt.	sysobjects

TABLE_ PRIVILEGES	Liefert Informationen zu Tabellenprivilegien, die dem aktuellen Benutzer in der aktuellen Datenbank zugewiesen oder von diesem erteilt wurden.	sysprotects sysobjects
TABLES	Liefert Informationen zu den Tabellen in der aktuellen Datenbank, für die der aktuelle Benutzer Berechtigungen besitzt.	sysobjects
VIEW_COLUMN_USAGE	Liefert Informationen zu den Spalten, die der aktuelle Benutzer in der aktuellen Datenbank besitzt und die in Sichtdefinitionen verwendet werden.	sysobjects sysdepends
VIEW_TABLE_ USAGE	Liefert Informationen zu den Tabellen, die der aktuelle Benutzer in der aktuellen Datenbank besitzt und die in einer Sicht verwendet werden.	sysobjects sysdepends
VIEWS	Liefert Informationen zu den Sichten, auf die der aktuelle Benutzer in der aktuellen Datenbank zugreifen kann.	sysobjects syscomments

Tabelle 19.17: Informationsschema-Sichten

Auf die Daten in Informationsschemasichten greifen Sie nach folgendem Muster (am Beispiel der Sicht CHECK_CONSTRAINTS) zu:

```
use pubs
select * from information_schema.check_constraints
```

19.4 Gespeicherte Systemprozeduren und -funktionen

Transact-SQL definiert gespeicherte Systemprozeduren für den Server und Systemfunktionen, die Kataloginformationen zurückgeben. Obwohl diese gespeicherten Prozeduren und Funktionen speziell auf SQL Server zugeschnitten sind, isolieren sie die Benutzer von der Struktur der zugrundeliegenden Tabellen des Systemkatalogs. Man sollte die Tabellen des Systemkatalogs niemals direkt ändern, sondern mit den gespeicherten Systemprozeduren arbeiten.

Im Verlauf dieses Kapitels haben Sie bereits einige Prozeduren kennengelernt. Einen Überblick über die gespeicherten Systemprozeduren finden Sie im Anhang C.

---

kapitel 20 wartung

20.1 hege und pflege

eine datenbank, um die sich niemand kümmert, wird über kurz oder lang nicht mehr ihren zweck erfüllen können. beispielsweise kann das transaktionsprotokoll einen umfang annehmen, der die ganze festplatte belegt, oder die datenbank ist nach einem systemabsturz überhaupt nicht mehr zugänglich. die pflege der datenbestände umfaßt deshalb datenbanksicherungen, transaktionsprotokollsicherungen, konsistenzprüfungen und ähnliche aufgaben, die nicht direkt mit dem abrufen und der verarbeitung von daten im zusammenhang stehen.

20.2 wartungspläne

in den aufgabenbereich des datenbankadministrators fallen auch routinemäßige wartungsarbeiten. dabei muß er aber nicht alles manuell erledigen. mit dem datenbankwartungsplanungs-assistenten kann man viele aufgaben automatisieren, die zum täglichen brot des datenbankadministrators gehören. insbesondere können sie folgende wartungsaufgaben planen und zu beliebigen zeitpunkten ausführen lassen:

- sicherung von datenbanken und transaktionsprotokollen
- arbeiten, die sich auf die indizierung beziehen
- optimieren des speicherplatzes für daten- und indexseiten
- konsistenzprüfungen
- statistiken aktualisieren

20.3 datenbankwartungsplanungs-assistent

den assistenten können sie nach verschiedenen methoden aufrufen:

- im **taskpad für 'erste schritte'** wählen sie **sql server verwalten** und dann **wartungsplan erstellen**.
- wenn die konsolenstruktur bis zur datenbank erweitert ist, für die sie einen wartungsplan erstellen wollen, klicken sie mit der rechten maustaste auf die datenbank und wählen **alle aufgaben / wartungsplan**.
- in der symbolleiste klicken sie auf **assistenten auswählen**.
- im menü **extras** wählen sie **assistenten**.

im dialogfeld **assistent auswählen** (siehe abbildung 20.1) erweitern sie den zweig **verwaltung**, markieren den eintrag **datenbankwartungsplanungs-assistent** und klicken auf **ok**.

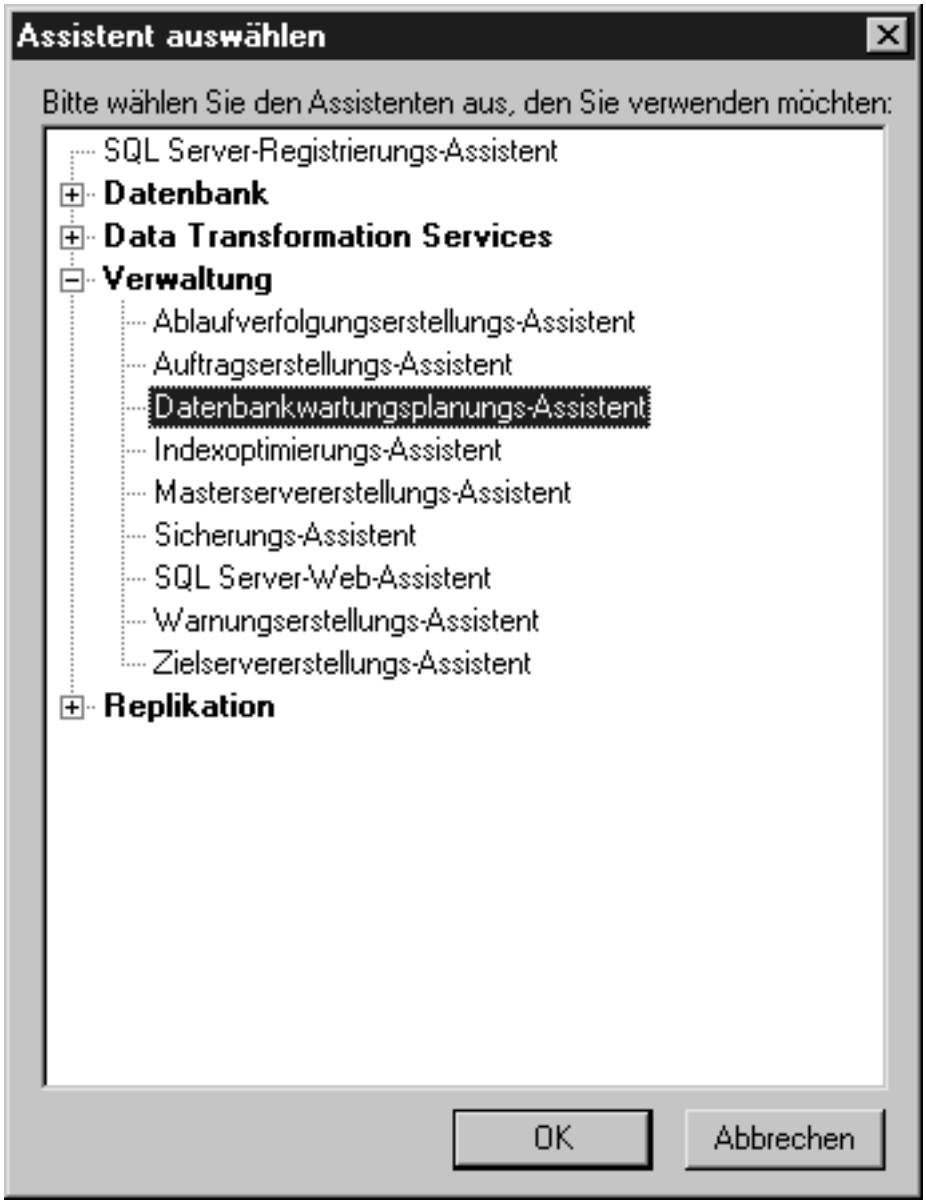


abbildung 20.1: den datenbankwartungs-planungs-assistenten über das dialogfeld assistent auswählen aufrufen

nachdem sie den assistenten gestartet haben, führen sie die folgenden schritte aus, um einen wartungsplan zu erstellen:

1. klicken sie im startdialogfeld des assistenten auf **weiter**. im zweiten dialogfeld des assistenten wählen sie die datenbank(en) aus, die sie in den wartungsplan einbeziehen möchten (siehe abbildung 20.2). für datenbanksicherungen sollten sie auf jeden fall die systemdatenbanken einschließen. klicken sie auf **weiter**.

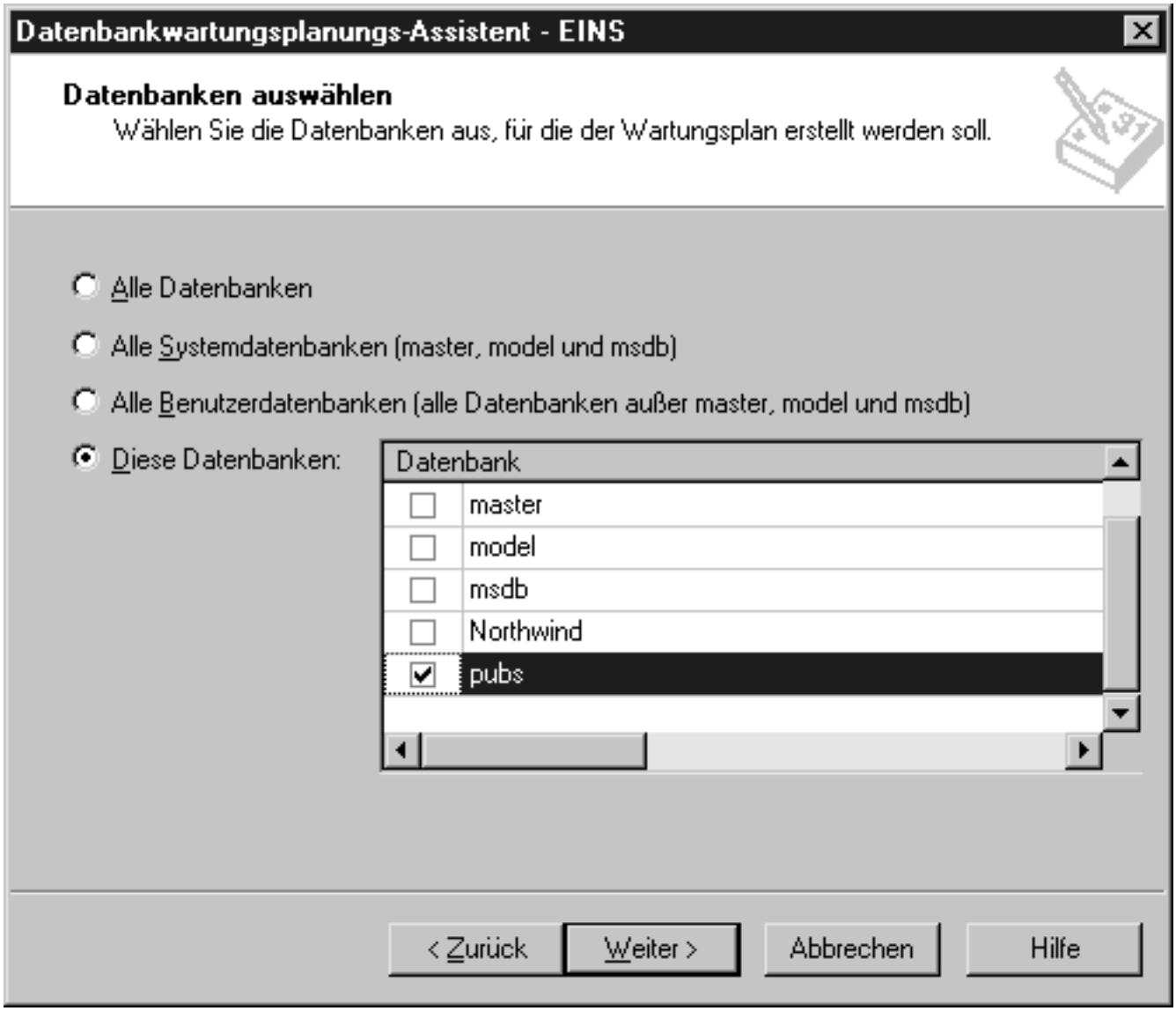


abbildung 20.2: datenbanken auswählen, für die ein wartungsplan zu erstellen ist

2. im dialogfeld **parameter für datenoptimierung aktualisieren** (siehe abbildung 20.3) legen sie fest, auf welche weise der platz in einer datenbank zu optimieren und zu reorganisieren ist. die einzelnen optionen haben folgende bedeutung:

daten- und indexseiten neu organisieren: die reorganisation der daten- und indexseiten trägt insbesondere bei umfangreichen datenbeständen zur leistungssteigerung bei, da dann die aktualisierung im laufenden betrieb schneller vonstatten geht. wenn sie dieses kontrollkästchen aktivieren, werden die indizes der datenbanktabellen gelöscht und mit dem ursprünglichen (bei der ersten option) bzw. einem neuen füllfaktor (zweite option) neu erstellt. wenn sie die datenbank vorwiegend abfragen, legen sie einen niedrigen füllfaktor fest, da sql server dann weniger indexseiten einlesen muß. dagegen sollten sie bei häufigen einfügeoperationen einen höheren füllfaktor wählen, damit eine umstrukturierung der indexseiten nicht allzu oft erforderlich ist (siehe dazu kapitel 14).

statistik für den abfrageoptimierer aktualisieren: anhand von verteilungsstatistiken zu den indizes in den benutzertabellen kann sql server die bewegung in den tabellen während der verarbeitung von transact-sql-anweisungen optimieren. bei der automatischen aktualisierung der verteilungsstatistiken tastet sql server einen bestimmten prozentsatz der daten für einen tabellenindex ab. je höher der prozentsatz, desto langsamer die aktualisierung.

nicht verwendeten speicherplatz aus datenbankdatei entfernen: aktivieren sie dieses kontrollkästchen, wenn sie nicht genutzten speicherplatz bei der aktualisierung automatisch freigeben wollen. der prozentwert für die menge des verbleibenden speichers nach der verkleinerung bezieht sich auf den ungenutzten speicherplatz und nicht auf die gesamtgröße der datenbank.

unter der rubrik *terminplan* können sie festlegen, wie oft die mit dem sql server-agenten geplante datenoptimierung stattfinden soll. wenn sie nicht mit dem vorgabewert zufrieden sind, klicken sie auf die schaltfläche **ändern** und legen neue termine fest.

klicken sie auf **weiter**.

bild

abbildung 20.3: das dialogfeld parameter für datenoptimierung aktualisieren

3. im dialogfeld **datenintegritätsprüfung** (siehe abbildung 20.4) legen sie fest, ob bei der wartung eine prüfung auf inkonsistente daten stattfinden soll. wenn sie das kontrollkästchen **datenbankintegrität überprüfen** einschalten, führt sql server die transact-sql-anweisung dbcc checkdb aus, um etwaige integritätsprobleme in der datenbank zu melden. es empfiehlt sich, die indizes einzuschließen und auch das kontrollkästchen **kleinere probleme beheben** einzuschalten.

wenn sie das kontrollkästchen **diese tests vor dem sichern durchführen** einschalten, sichert sql server die datenbank und das transaktionsprotokoll nicht, falls die integritätstests inkonsistente daten ergeben.

auch für die datenintegritätsprüfung können sie den vorgegebenen terminplan ändern. wenn sie alle optionen festgelegt haben, klicken sie auf **weiter**.

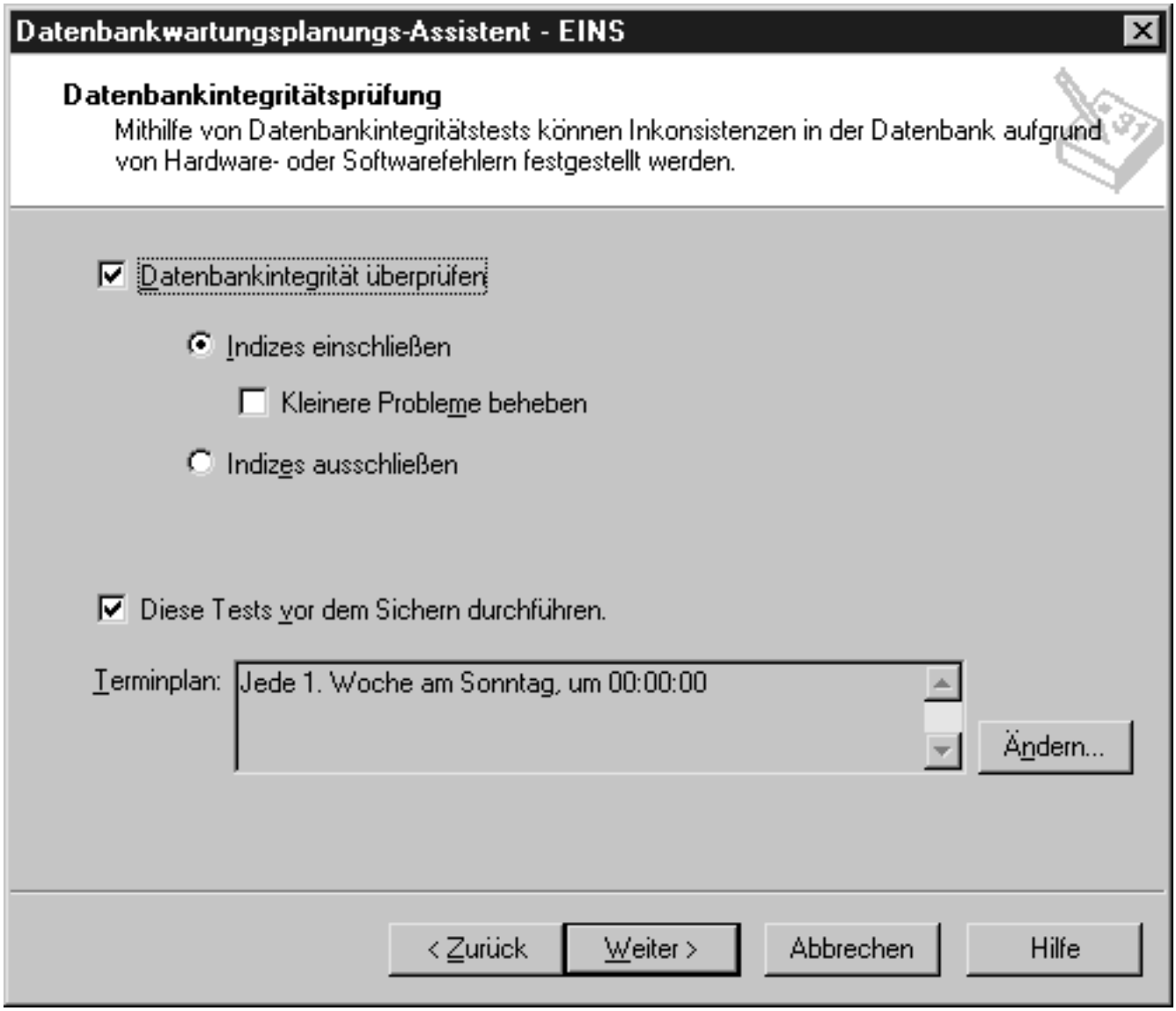


abbildung 20.4: das dialogfeld datenbankintegritätsprüfung

4. für die erwähnte datenbanksicherung legen sie im nächsten dialogfeld (siehe abbildung 20.5) einen sicherungsplan fest. es empfiehlt sich, das kontrollkästchen **datenbank als teil des wartungsplanes sichern** einzuschalten, damit sich bei hard- oder software-fehlern eine sicherung der datenbank wiederherstellen läßt. ist das kontrollkästchen **integrität der sicherung nach beendigung der sicherung überprüfen** eingeschaltet, überprüft sql server per transact-sql-anweisung restore verifyonly, ob die sicherung vollständig und der zugriff auf die datenträger möglich ist.

wenn ein bandlaufwerk am server angeschlossen ist, können sie eine sicherung auf band spezifizieren. bei der zweiten option (festplatte) wird die sicherung auf demselben computer abgelegt, auf dem auch die zu sichernde datenbank gespeichert ist. das verzeichnis legen sie im nächsten dialogfeld fest.

schließlich können sie den terminplan für die datensicherung ändern.

klicken sie auf **weiter**, wenn sie alle einstellungen vorgenommen haben.

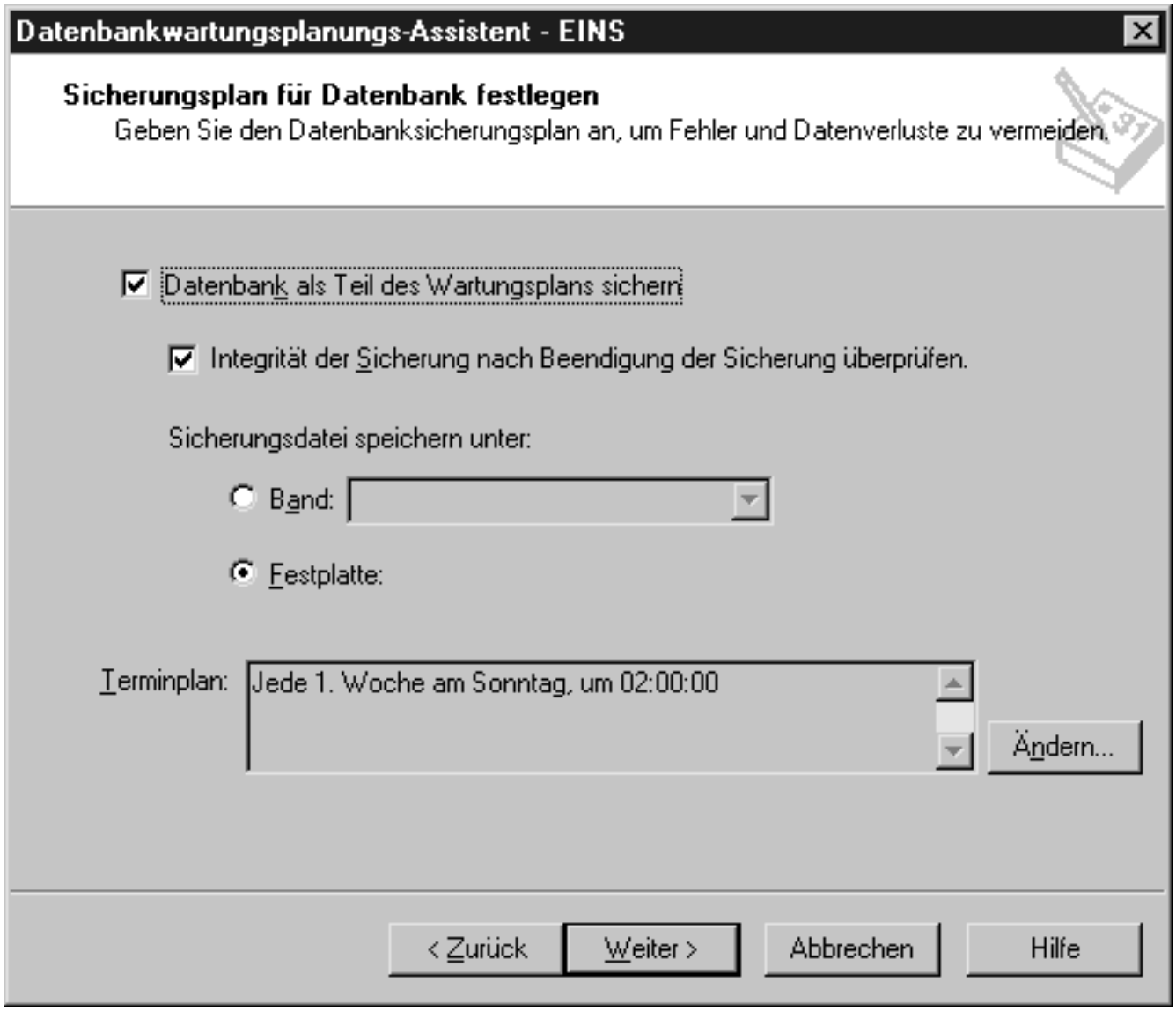


abbildung 20.5: das dialogfeld sicherungsplan für datenbank festlegen

5. wenn sie im dialogfeld für den sicherungsplan die festplatte als sicherungsmedium gewählt haben, gelangen sie als nächstes zum dialogfeld **festplattenverzeichnis für sicherung angeben** (siehe abbildung 20.6). als standardverzeichnis ist \backup unterhalb des bei der installation festgelegten pfades (in der regel \mssql7) vorgegeben. wählen sie die option *dieses verzeichnis verwenden*, wenn sie ein anderes verzeichnis (vorzugsweise auf einer anderen festplatte) festlegen wollen.

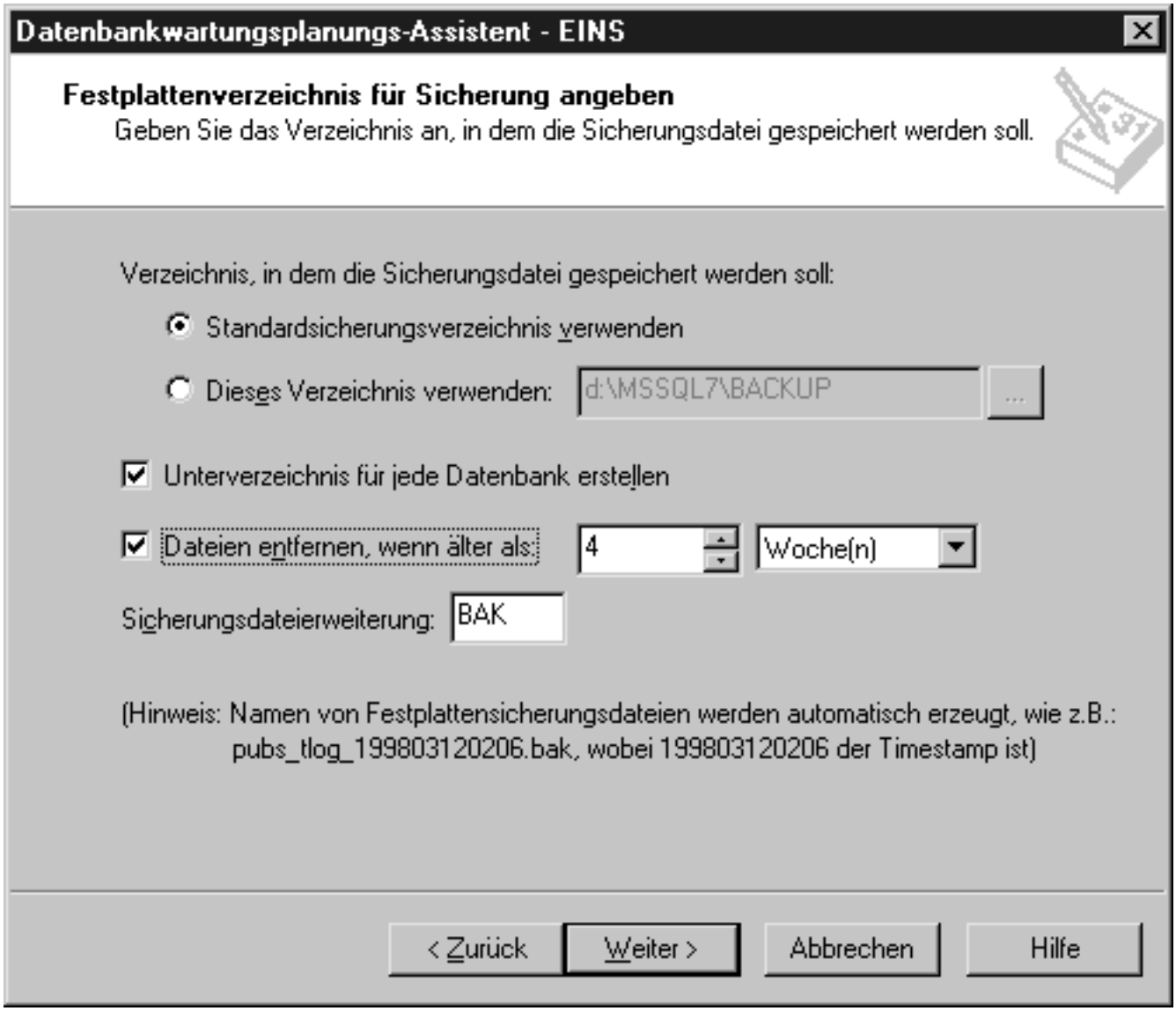


abbildung 20.6: das dialogfeld festplattenverzeichnis für sicherung angeben

schalten sie das kontrollkästchen **unterverzeichnis für jede datenbank erstellen** ein, wenn sie die einzelnen datenbanksicherungen in einem der datenbank zugeordneten verzeichnis unterbringen wollen.

wenn sie das kontrollkästchen **dateien entfernen, wenn älter als** einschalten, werden die entsprechenden sicherungsdateien von der festplatte gelöscht. sollte es der platz auf der platte zulassen, wählen sie einen möglichst großen zeitraum, um mehrere zurückliegende sicherungen - und nicht nur die unmittelbar letzte - aufzubewahren. manchmal bleibt ein fehler eine ganze zeitlang unbemerkt und kann somit einige sicherungen unbrauchbar machen. wenn sie in diesem fall auf eine ältere version zurückgreifen können, läßt sich der fehler möglicherweise erkennen und beseitigen.

sql server versieht die dateinamen automatisch mit einem zeitstempel und der dateierweiterung bak. im betreffenden bearbeitungsfeld können sie bei bedarf eine andere erweiterung festlegen.

klicken sie auf **weiter**.

6. die beiden nächsten dialogfelder (abbildung 20.7 und abbildung 20.8) beziehen sich auf die sicherung des transaktionsprotokolls. die optionen und einstellungen entsprechen weitgehend denen bei der sicherung der datendateien (siehe erläuterung zu abbildung 20.5 und abbildung 20.6). das zweite dialogfeld erscheint auch hier nur dann, wenn sie eine sicherung auf festplatte festgelegt haben.

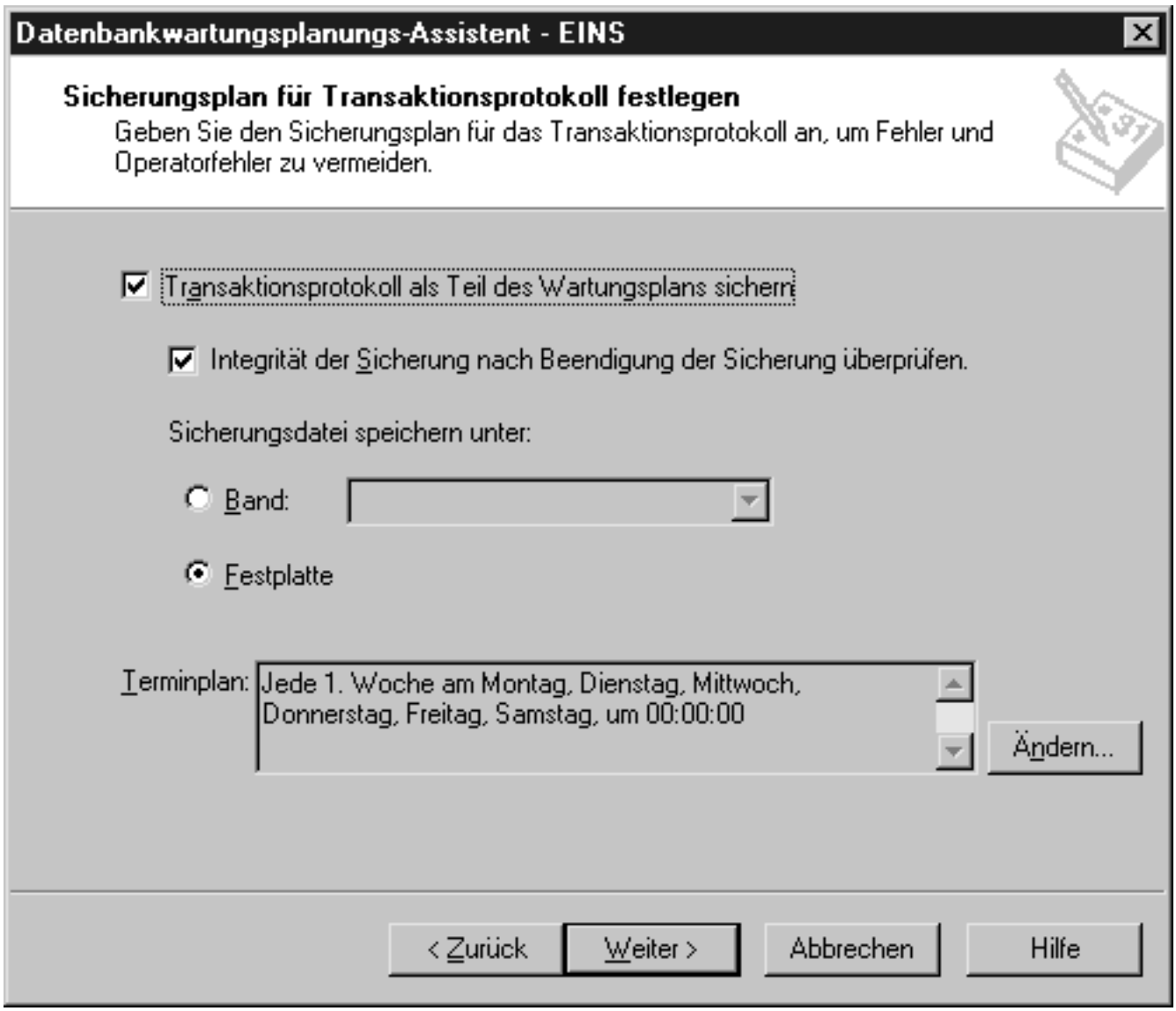


abbildung 20.7: das dialogfeld sicherungsplan für transaktionsprotokoll festlegen

7. im dialogfeld **erzeugte berichte speichern** (siehe abbildung 20.9) legen sie fest, ob und wo die vom wartungsplan erstellten berichte auf der festplatte abzulegen sind.

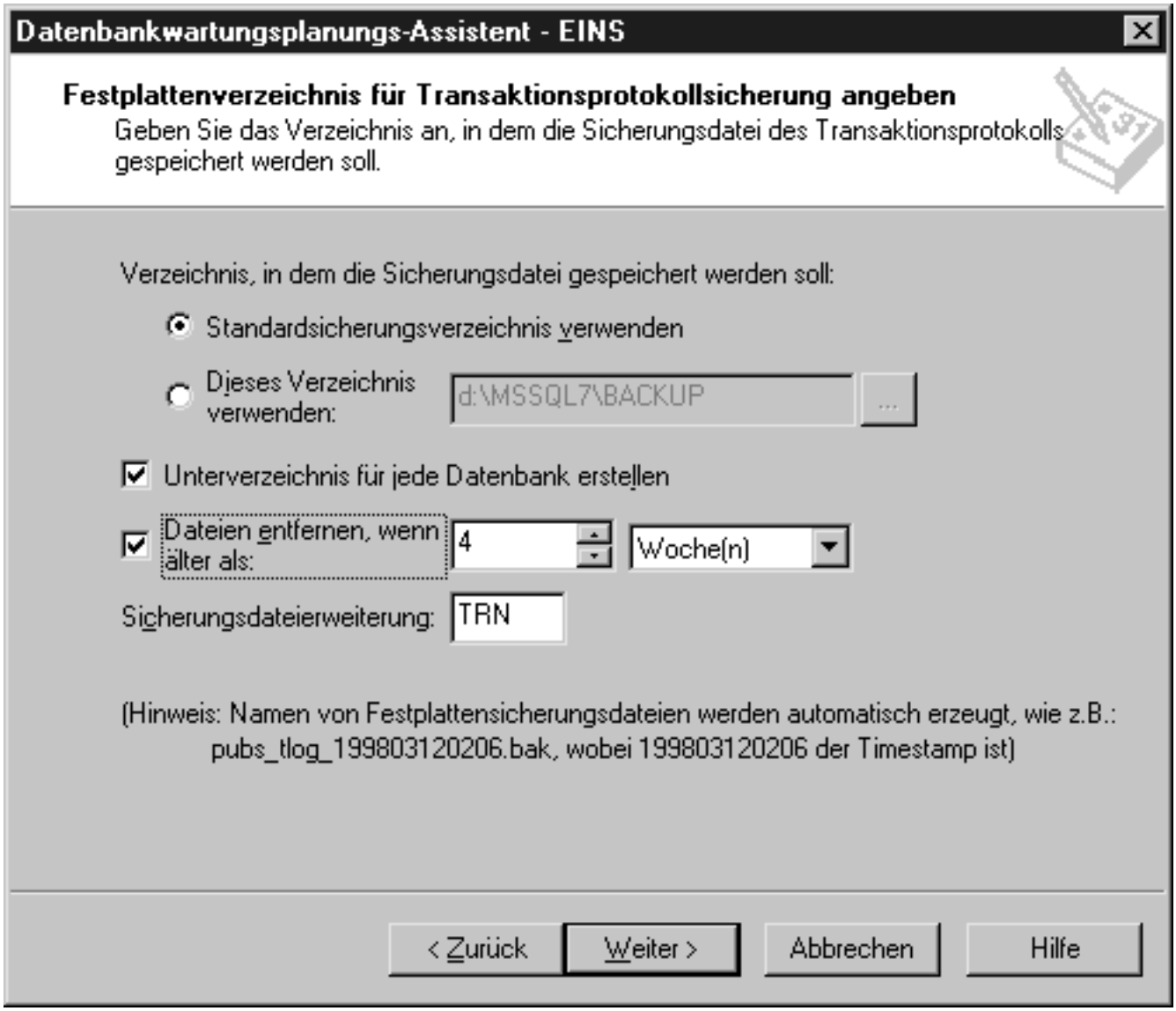


abbildung 20.8: das dialogfeld festplattenverzeichnis für transaktionsprotokollsicherung angeben

falls bereits operatoren eingerichtet sind, ist das kontrollkästchen **e-mail-bericht an operator senden** aktiviert, und sie können einen operator aus dem zugeordneten drop-down-listenfeld auswählen. über die schaltfläche **neu** gelangen sie zu einem dialogfeld, in dem sie eigenschaften für neue operatoren festlegen können.

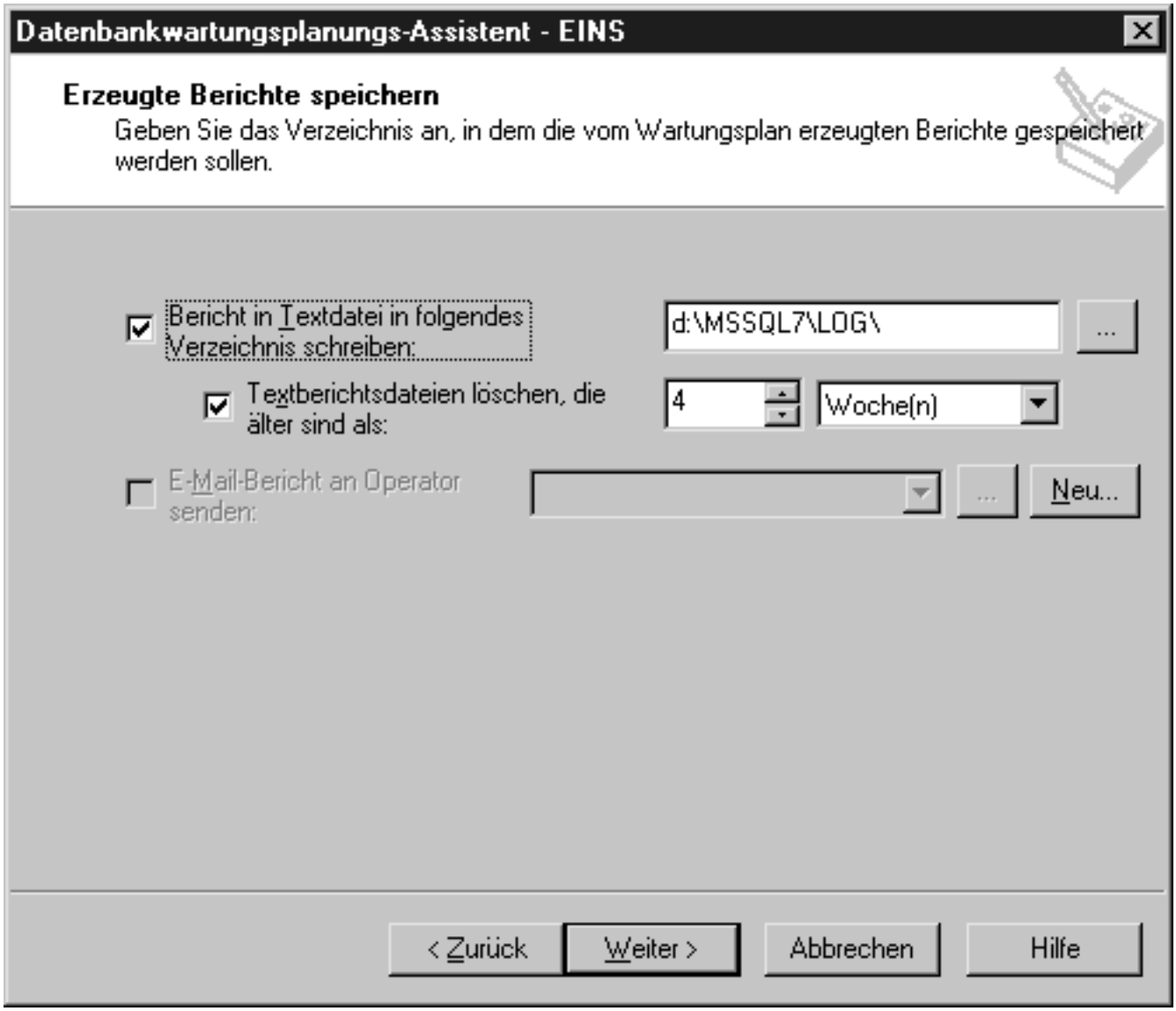


abbildung 20.9: das dialogfeld erzeugte berichte speichern

klicken sie auf **weiter**.

bild

abbildung 20.10: das letzte dialogfeld des datenbankwartungsplanungs-assistenten

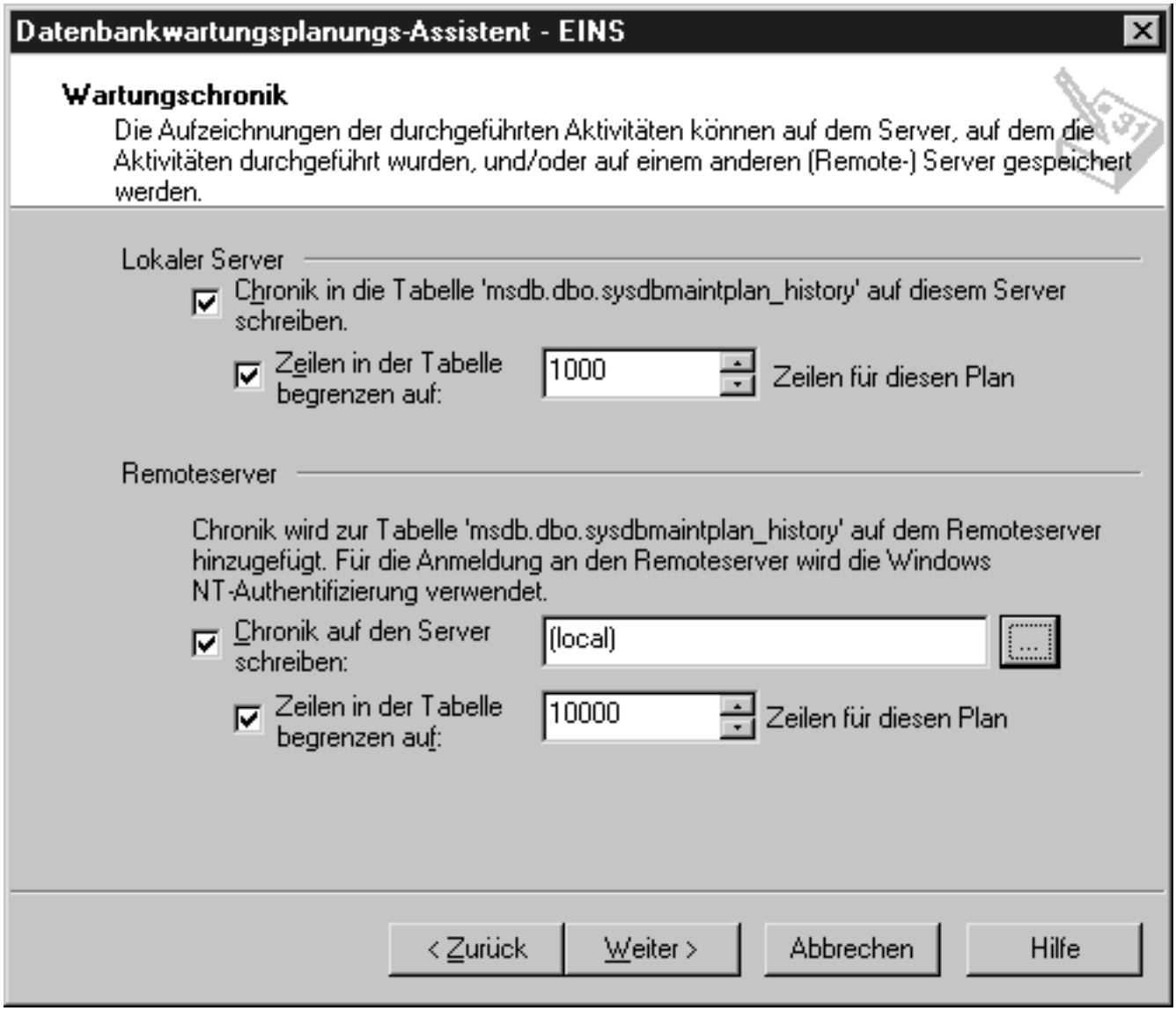


abbildung 20.11: das dialogfeld wartungschronik

8. im dialogfeld **wartungschronik** (siehe abbildung 20.10) legen sie fest, ob und wie der erstellte bericht in die genannten systemtabellen auf dem lokalen server bzw. einem remoteserver zu schreiben ist.
9. das letzte dialogfeld des assistenten (siehe abbildung 20.11) zeigt eine übersicht des spezifizierten wartungsplans an. im bearbeitungsfeld **planname** können sie den vorgegebenen namen für den wartungsplan überschreiben (im beispiel wartungsplan pubs). klicken sie auf **fertigstellen**, um den wartungsplan erstellen zu lassen.

abschließend erscheint ein meldungsfeld, daß der wartungsplan erfolgreich erstellt wurde.

20.4 wartungspläne anzeigen und bearbeiten

die erstellten wartungspläne können sie im enterprise manager bearbeiten. erweitern sie dazu die konsolenstruktur des servers, dann den ordner **verwaltung**, und klicken sie auf **datenbankwartungspläne**. im rechten fensterausschnitt des enterprise managers erscheinen die vorhandenen wartungspläne.

die zugehörigen aufträge sind im ordner **sql server-agent** unter **aufträge** zu finden (siehe abbildung 20.12).

Bild

abbildung 20.12: die vom datenbankwartungsplanungs-assistenten erzeugten aufträge

kapitel 23 geht im rahmen des sql server-agenten näher auf aufträge ein.

gehen sie nach diesem kleinen exkurs wieder zum ordner **datenbankwartungspläne** zurück. klicken sie mit der rechten maustaste auf den wartungsplan, den sie anzeigen oder bearbeiten möchten. wählen sie aus dem kontextmenü den befehl **eigenschaften**, um das in abbildung 20.13 dargestellte dialogfeld **datenbankwartungsplan** zu öffnen.

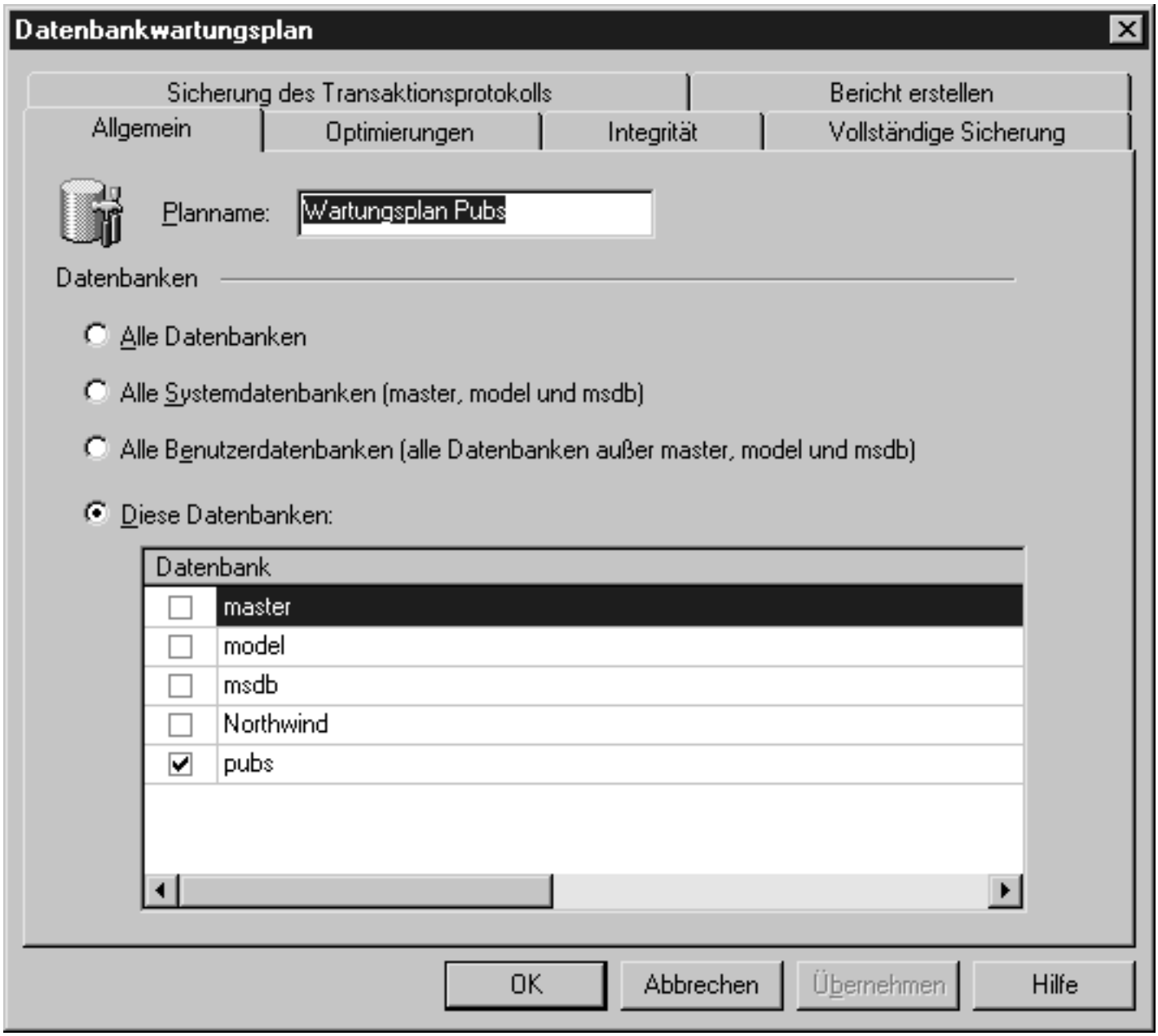


abbildung 20.13: das dialogfeld datenbankwartungsplan

die registerkarten im dialogfeld **datenbankwartungsplan** fassen alle optionen zusammen, die sie mit

dem datenbankwartungsplanungs-assistenten festgelegt haben. tabelle 20.1 zeigt die zuordnung zwischen den registerkarten des dialogfelds **datenbankwartungsplan** und den dialogfeldern des datenbankwartungsplanungs-assistenten. die bedeutung der optionen auf den registerkarten sind bereits bei den einzelnen schritten des assistenten erläutert worden.

registerkarte im dialogfeld datenbankwartungsplan	dialogfeld des datenbankwartungsplaungs-assistenten	erläuterung siehe bei abbildung
allgemein	datenbanken auswählen	abbildung 20.2
optimierungen	parameter für die datenoptimierung aktualisieren	abbildung 20.3
integrität	datenintegritätsprüfung	abbildung 20.4
vollständige sicherung	sicherungsplan für datenbank festlegen festplattenverzeichnis für sicherung angeben	abbildung 20.5 abbildung 20.6
sicherung des transaktionsprotokolls	sicherungsplan für transaktionsprotokoll festlegen festplattenverzeichnis für transaktionsprotokollsicherung angeben	abbildung 20.7 abbildung 20.8
bericht erstellen	erzeugte berichte speichern wartungschronik	abbildung 20.9 abbildung 20.10

tabelle 20.1: registerkarten und dialogfelder für wartungspläne

wenn sie änderungen des wartungsplans im dialogfeld **datenbankwartungsplan** vornehmen, wird die schaltfläche **übernehmen** aktiv. klicken sie auf **übernehmen**, um die änderungen zu akzeptieren, und dann auf **ok**, um das dialogfeld zu schließen.

20.5 sqlmaint und xp_sqlmaint

mit dem dienstprogramm sqlmaint können sie ebenfalls wartungsaufgaben wahrnehmen. es gibt sogar die möglichkeit, einen mit dem datenbankwartungsplanungs-assistenten erstellten plan auszuführen. das dienstprogramm sqlmaint starten sie von der befehlszeile. wenn sie mit dem sql server query analyzer arbeiten wollen, können sie die prozedur xp_sqlmaint aufrufen, die ihrerseits sqlmaint startet und dabei einen bereitgestellten string mit den schaltern für sqlmaint übergibt.

die syntax des dienstprogramms sqlmaint lautet:

```
sqlmaint
[-?] |
[
  [-s <servername>]
  [-u <benutzername> [-p <kennwort>]]
  {
    [ -d <datenbankname> | -planname <name> | -planid <guid>]
    [-rpt <ausgabedatei> [-deltxtrpt <zeitabschnitt>] ]
    [-to <operatorname>]
    [-htmlrpt <html-datei> [-delhtmlrpt <zeitabschnitt>] ]
    [-rmunusedspace <mindestgröße der db in mb>
                    <% freier speicherplatz>]
    [-ckdb | -ckdbnoidx]
    [-ckal | -ckalnoidx]
    [-cktxtal]
    [-ckcat]
    [-updsts]
    [-updoptistats <% der datenbank in stichprobe>]
    [-rebldidx <% freier speicherplatz>]
    [-writehistory]
    [
      {-bkupdb [<sicherungspfad>]
      | -bkuplog [<sicherungspfad>]
      }
      {-bkupmedia
        {festplatte [ [-delbkups <zeitabschnitt>]
                      [-crbksubdir ] [ -usedefdir ]
                    ]
        | band
        }
      }
      [-bkuponlyifclean]
      [-vrfybackup]
    ]
  }
]
```

die klausel *zeitabschnitt* ist folgendermaßen aufgebaut:

```
zahl[minutes | hours | days | weeks | months]
```

die *zahl* gibt einen zeitraum an, nach dem eine bestimmte datei (siehe die beschreibung der argumente)

als veraltet gilt und gelöscht werden kann. dabei steht minutes für minuten, hours für stunden, days für tage, weeks für wochen und month für monate. ohne spezifikation gilt die voreinstellung wochen.

tabelle 20.2 erläutert die argumente des programms sqlmaint.

argument	beschreibung
-?	gibt die syntax von sqlmaint aus.
-s <servername>	bezeichnet den zielserver. ohne diese angabe gilt der lokale computer als standardwert.
-u <benutzername>	der für die anmeldung erforderliche name des benutzers. ohne diesen wert versucht sqlmaint eine anmeldung per windows nt-authentifizierung.
-p <kennwort>	das zum benutzernamen gehörende kennwort, wenn -u angegeben ist. enthält das kennwort sonderzeichen, ist es in doppelte anführungszeichen zu schreiben.
-d <datenbankname>	name der datenbank, auf die sich die wartung bezieht.
-planname <name> -planid <guid>	der name und der global eindeutige bezeichner eines wartungsplans, den sie mit dem datenbankwartungsplanungs-assistenten erstellt haben.
-rpt <ausgabedatei>	vollständiger pfad und name der textdatei, in die sqlmaint den bericht schreibt.
-deltxtrpt <zeitabschnitt>	löscht die textberichte, die älter als in <zeitabschnitt> angegeben sind.
-to <operatorname>	spezifiziert den operator, der den bericht per sql mail erhalten soll.
-htmlrpt <html-datei>	vollständiger pfad und name einer html-datei, in die sqlmaint den bericht schreibt.
-delhtmlrpt <zeitabschnitt>	löscht html-berichtsdateien, die älter als in <zeitabschnitt> angegeben sind.
-rmunusedspace <mindestgröße der db in mb> <% freier speicherplatz>	entfernt freien speicherplatz aus der datenbank, wenn diese die angegebene mindestgröße in mbyte erreicht hat. der prozentwert gibt an, wieviel freier speicher bezogen auf die endgröße in der datenbank verbleibt.
-ckdb -ckdbnoidx	führt eine dbcc checkdb-anweisung bzw. eine dbcc checkdb-anweisung mit der option noindex aus.
-ckal -ckalnoidx	führt eine dbcc newalloc-anweisung bzw. eine dbcc newalloc-anweisung mit der option noindex aus.
-cktxtal	führt eine dbcc textall-anweisung aus.

-ckcat	führt eine dbcc checkcatalog-anweisung aus.
-updsts	führt für jede tabelle der datenbank die anweisung update statistics aus.
-updoptistats <% der datenbank in stichprobe>	führt für jede tabelle der datenbank die anweisung update statistics *tabelle* with sample *stichprobeprozent* percent aus.
-rebldidx <% freier speicherplatz>	erstellt die indizes der tabellen in der zieldatenbank neu, wobei der angegebene prozentsatz die differenz vom füllfaktor zu 100% darstellt.
-writehistory	schreibt einen eintrag für jede wartungsoperation in die systemtabelle msdb.dbo.sysdbmaintplan_history.
-bkupdb [<sicherungspfad]>	sichert die gesamte datenbank.
-bkuplog [<sicherungspfad]>	sichert nur das transaktionsprotokoll.
-bkupmedia	medientyp der sicherung (festplatte oder band).
-delbkups <zeitabschnitt>	löscht die sicherungsdateien, die älter als in <zeitabschnitt> angegeben sind.
-crbksubdir	erstellt die sicherung im eigenen unterverzeichnis (unterhalb von <sicherungspfad> oder vom standardverzeichnis).
-usedefdir	erstellt die sicherung im standardverzeichnis (c:\mssql7\backup bei einer standardinstallation von sql server).
-bkuponlyifclean	sicherung nur durchführen, wenn die datenprüfungen mit den -ck-optionen keine fehler geliefert haben.
-vrfybackup	führt restore verifyonly unmittelbar nach abschluß der sicherung aus.

tabelle 20.2: argumente des dienstprogramms sqlmaint

wenn *benutzername*, *kennwort*, *datenbankname* oder die pfadangaben sonderzeichen enthalten, sind die jeweiligen werte in doppelte anführungszeichen zu schreiben. andernfalls sind die anführungszeichen optional.

achten sie darauf, daß nach den schaltern *ein leerzeichen* stehen muß. während sie zum beispiel beim dienstprogramm osql die befehlszeile

```
osql -usa -seins
```

für eine anmeldung mit dem benutzernamen sa für den server eins schreiben können, ist bei sqlmaint folgendes zu formulieren:

```
sqlmaint -u sa -s eins
```

andernfalls gibt sqlmaint beharrlich die syntax aus (analog zum schalter -?).

die gespeicherte prozedur xp_sqlmaint rufen sie mit folgender syntax auf:

```
xp_sqlmaint 'schalterzeichenfolge'
```

in der *schalterzeichenfolge* verwenden sie das gleiche format, die gleichen argumente und die gleichen werte wie beim aufruf des programms sqlmaint von der befehlszeile. lediglich der schalter -? ist nicht erlaubt.

das erste beispiel zeigt einen einfachen aufruf, der zwar keine wartungsaufgaben wahrnimmt, den sie aber in ähnlicher form zum test der syntax und der verbindung verwenden können. im beispiel erfolgt der aufruf vom computer zwei aus.

beim start von der befehlszeile lautet die anweisung:

```
sqlmaint -s eins -d pubs -rpt "\\zwei\d\sqlkomp\wartg.txt"
```

die funktionell identische anweisung läßt sich mit der gespeicherten prozedur über sql server query analyzer wie folgt ausführen:

```
exec xp_sqlmaint '-s eins -d pubs -rpt "\\zwei\d\wartg.txt"'
```

im namen der berichtsdatei (schalter -rpt) müssen sie den vollständigen unc-namen angeben, wenn die datei auf einem remoteserver gespeichert werden soll. im obigen beispiel findet der aufruf der prozedur xp_sqlmaint zwar auf zwei statt, bezieht sich aber auf die datenbank pubs auf eins. mithin liegt das ziel der berichtsdatei wartg.txt aus sicht des servers eins auf dem remoteserver zwei, und zwar dort im stammverzeichnis d:\.

die anmeldung (bei zwei) erfolgte nicht über windows-nt-, sondern sql-server-authentifizierung. in diesem fall setzt die prozedur xp_sqlmaint die parameter -u mit dem benutzernamen und -p mit dem kennwort vor die schalterzeichenfolge und übergibt alles zusammen an sqlmaint. der name des zielservers (hier eins) ist teil der zeichenfolge. bei windows-nt-authentifizierung wird die zeichenfolge unverändert weitergereicht.

das ergebnis lautet:

```
output of sqlmaint.exe
------------------------------------------------------------

microsoft (r) sqlmaint-dienstprogramm (unicode), version
 7.00.623
copyright (c) microsoft corporation, 1995 - 1996

an computer mit sql server 'eins' angemeldet als 'sa' (nicht
 vertraut)
startet wartung der datenbank 'pubs' auf fri apr 30 22:01:54
 1999

beendet wartung der datenbank 'pubs' auf fri apr 30 22:01:54
 1999

sqlmaint.exe prozessexitcode: 0 (erfolg)

(10 row(s) affected)
```

die ergebnisse erscheinen sowohl auf dem bildschirm als auch in der angegebenen berichtsdatei. die mit -rpt bzw. -htmlrpt erstellten ausgabedateien erhalten automatisch eine datumskennzeichnung im format _yyyymmddhhmm. diese kennzeichnung wird zwischen den eigentlichen dateinamen und den punkt vor der dateierweiterung (falls vorhanden) eingeschoben. der name der erzeugten berichtsdatei lautet demnach:

```
wartg_199904302201.txt
```

das nächste beispiel zeigt, wie sie einen bereits vorhandenen wartungsplan im aufruf von sqlmaint verwenden können.

die global eindeutigen bezeichner und die namen aller wartungspläne sind in der systemdatenbank msdb.dbo.sysdbmaintplans verzeichnet. diese werte können sie mit der anweisung

```
select plan_id, plan_name from msdb..sysdbmaintplans
```

abrufen und erhalten etwa das folgende ergebnis:

```
plan_id                              plan_name
------------------------------------ --------------------------
00000000-0000-0000-0000-000000000000 all ad-hoc plans
```

```
ab3a14e3-4614-11d3-b285-0000b434c3be wartungsplan pubs

(2 row(s) affected)
```

im beispiel zeigt die zweite zeile den wartungsplan, den sie in diesem kapitel mit dem datenbankwartungsplanungs-assistenten erstellt haben. die folgende anweisung verwendet den wert aus der spalte plan_id, um den wartungsplan mit hilfe der gespeicherten prozedur xp_sqlmaint auszuführen. außerdem sind die schalter zur ausführung der anweisungen dbcc checkdb und dbcc newalloc angegeben:

```
exec xp_sqlmaint '-planid ab3a14e3-4614-11d3-b285-0000b434c3be
 -rpt "d:\sqlkomp\wartg.txt" -ckdb -ckal'
```

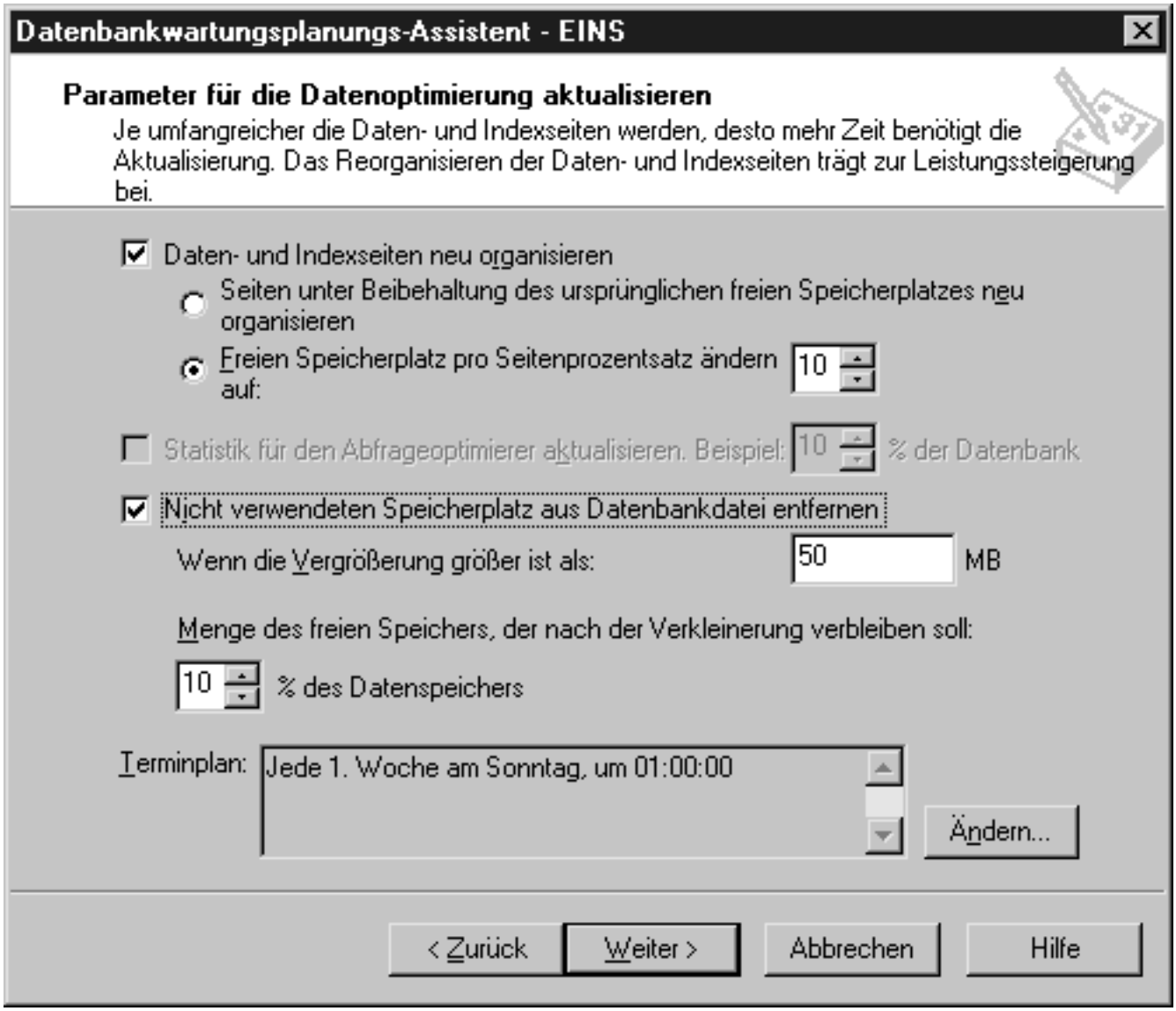
die anweisung wurde von eins aus und mit einer anmeldung per windows-nt-authentifizierung ausgeführt. das ergebnis lautet:

```
output of sqlmaint.exe
------------------------------------------------------------

microsoft (r) sqlmaint-dienstprogramm (unicode), version
 7.00.623
copyright (c) microsoft corporation, 1995 - 1996

an computer mit sql server 'eins' angemeldet als
 'eins\administrator' (vertraut)
startet wartungsplan 'wartungsplan pubs' für fri apr 30
 18:06:58 1999

[1] datenbank pubs: daten- und indexverknüpfung überprüfen...

    ** ausführungsdauer: 0 std, 0 min, 3 sek **

[2] datenbank pubs: daten- und indexreservierung überprüfen...

    ** ausführungsdauer: 0 std, 0 min, 1 sek **

ende des wartungsplans 'wartungsplan pubs' für fri apr 30
 18:07:02 1999

sqlmaint.exe prozessexitcode: 0 (erfolg)

(18 row(s) affected)
```

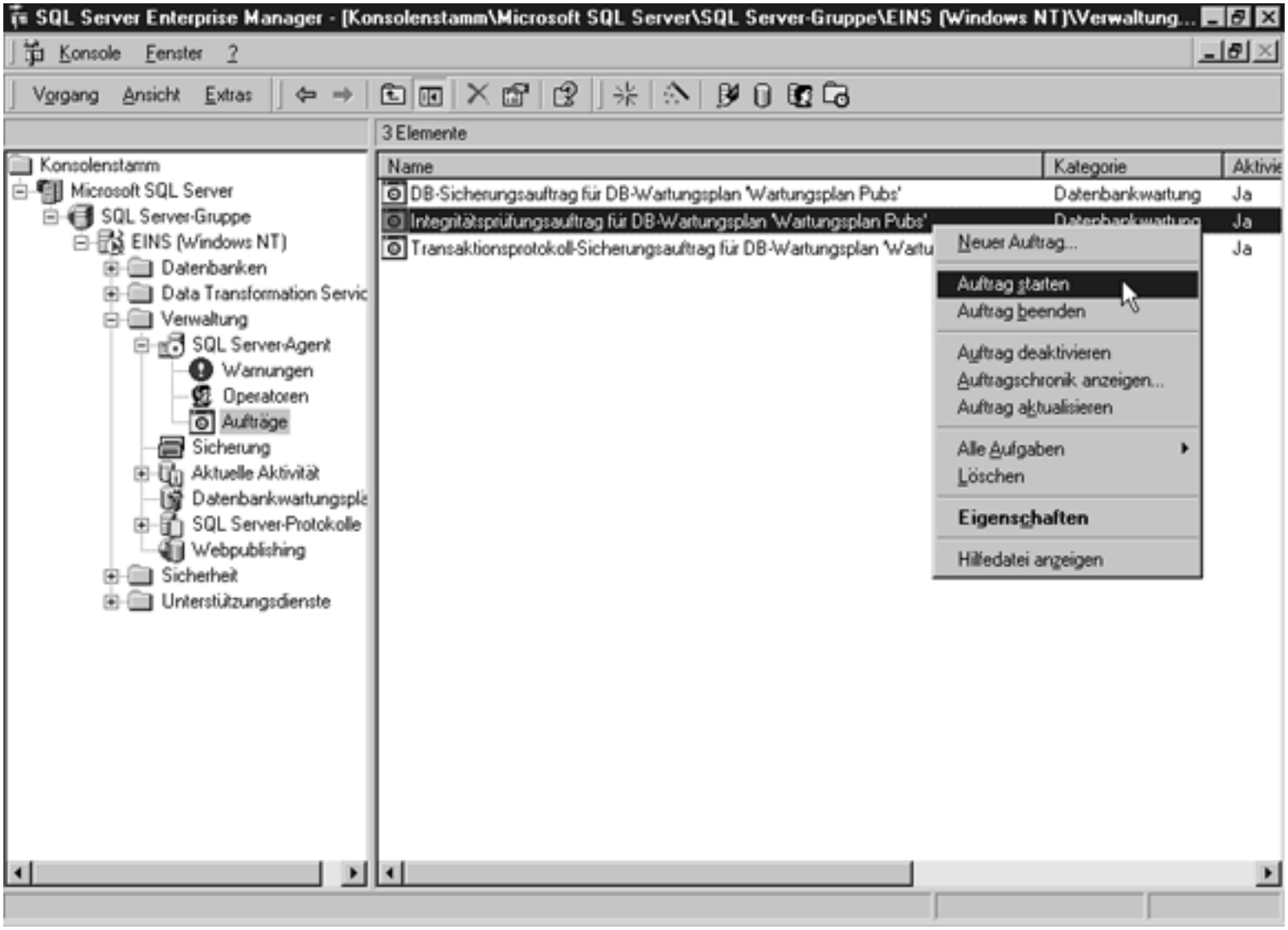
die hier für die bildschirmausgabe dargestellten ergebnisse hat sqlmaint parallel in die datei wartg_199904301806.txt im verzeichnis d:\sqlkomp auf server eins geschrieben.

weitere beispiele zum dienstprogramm sqlmaint finden sie in der online-dokumentation.

es sei noch einmal darauf hingewiesen, daß die dienstprogramme der befehlszeile durchaus ihre berechtigung haben. wenn sich zum beispiel der enterprise manager nicht mehr starten läßt und ihnen damit den zugang zum datenbankwartungsplanungs-assistenten (oder irgendeinem anderen teil dieser umgebung) verwehrt, bleiben ihnen die befehlszeilenprogramme als rettungsanker. in diesem - leider nicht konstruierten - fall war noch der sql server query analyzer funktionsfähig, so daß sich eine wartung der dateien über die prozedur xp_sqlmaint mit einem zum glück (oder in weiser voraussicht?) angelegten wartungsplan problemlos durchführen ließ. das dienstprogramm sqlmaint kann man aber genauso einfach aufrufen, sofern der name des passenden wartungsplans bekannt ist.


Datenbankwartungsplanungs-Assistent - EINS
Parameter für die Datenoptimierung aktualisieren
Je umfangreicher die Daten- und Indexseiten werden, desto mehr Zeit benötigt die Aktualisierung. Das Reorganisieren der Daten- und Indexseiten trägt zur Leistungssteigerung bei.
Daten- und Indexseiten neu organisieren
Seiten unter Beibehaltung des ursprünglichen freien Speicherplatzes neu organisieren
Freien Speicherplatz pro Seitenprozentsatz ändern auf:
10
Statistik für den Abfrageoptimierer aktualisieren. Beispiel:
10
% der Datenbank
Nicht verwendeten Speicherplatz aus Datenbankdatei entfernen
Wenn die Vergrößerung größer ist als:
50
MB
Menge des freien Speichers, der nach der Verkleinerung verbleiben soll:
10
% des Datenspeichers
Terminplan:
Jede 1. Woche am Sonntag, um 01:00:00
Ändern...
< Zurück
Weiter >
Abbrechen
Hilfe


Datenbankwartungsplanungs-Assistent - EINS
Datenbankwartungsplanungs-Assistenten beenden
Sie haben die erforderlichen Schritte zum Erstellen eines Datenbankwartungsplans abgeschlossen. Eine Kurzbeschreibung des Plans wird nachstehend angezeigt.
Planname:
Wartungsplan Pubs
DATENBANKEN
pubs
Server
(local)
INTEGRITÄTSPRÜFUNGEN
Jede 1. Woche am Sonntag, um 00:00:00
Integritätsprüfungen vor der Sicherung der Datenb
< Zurück
Fertig stellen
Abbrechen
Hilfe


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Verwaltung...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
3 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation Servic
Verwaltung
SQL Server-Agent
Warnungen
Operatoren
Aufträge
Sicherung
Aktuelle Aktivität
Datenbankwartungspla
SQL Server-Protokolle
Webpublishing
Sicherheit
Unterstützungsdienste
Name
Kategorie
Aktivie
DB-Sicherungsauftrag für DB-Wartungsplan 'Wartungsplan Pubs'
Datenbankwartung
Ja
Integritätsprüfungsauftrag für DB-Wartungsplan 'Wartungsplan Pubs'
Ja
Transaktionsprotokoll-Sicherungsauftrag für DB-Wartungsplan 'Wartu
Ja
Neuer Auftrag...
Auftrag starten
Auftrag beenden
Auftrag deaktivieren
Auftragschronik anzeigen...
Auftrag aktualisieren
Alle Aufgaben
Löschen
Eigenschaften
Hilfedatei anzeigen

kapitel 21 sicherheit

21.1 nomen est omen

mit einem datenbank-managementsystem sind oftmals tausende von benutzern gleichzeitig verbunden. in einem unternehmen können bestimmte daten allen mitarbeitern oder sogar der öffentlichkeit zugänglich sein, während sensible daten nur einem ausgewählten personenkreis zur verfügung stehen sollen. der begriff sicherheit ist nicht nur im zusammenhang mit einer mißbräuchlichen verwendung von daten zu sehen, sondern auch im hinblick auf schäden, die sich aus einer unbeabsichtigten fehlbedienung ergeben.

die in einem netzwerk angeschlossenen benutzer arbeiten oftmals an denselben dateien und mit denselben programmen. dabei taucht das problem auf, daß ein benutzer nicht in eine datei schreiben darf, während ein anderer benutzer die datei liest. außerdem sollen oftmals bestimmte dokumente nicht allen benutzern im netzwerk uneingeschränkt zugänglich sein. die zusammenarbeit der benutzer in einem netzwerk und der zugriff auf die daten muß also nach bestimmten regeln erfolgen. diese regeln werden durch die sicherheitsmechanismen des betriebssystems und des datenbank-managementsystems durchgesetzt.

21.2 sicherheitsarchitektur

der terminus »sicherheit« deutet bereits an, daß wir es mit einem sensiblen bereich zu tun haben, in dem experimente fehl am platz sind. während man einfache `select`-abfragen noch intuitiv ausführen kann, sind für die realisierung von schutzmechanismen grundlegende kenntnisse der sicherheitsarchitektur erforderlich.

21.2.1 sicherheitsebenen

hinter dem allgemeinen begriff sicherheit verbergen sich verschiedene mechanismen, die sich in folgende ebenen gliedern:

- betriebssystem
- sql server
- datenbanken
- datenbankobjekte

betriebssystem

die verbindung zum server, auf dem sql server läuft, erfordert eine anmeldung, die vom konkreten betriebssystem des servers abhängig ist. das gilt auch dann, wenn der benutzer an dem computer arbeitet, auf dem auch sql server installiert ist. das bevorzugte betriebssystem für sql server ist windows nt. unter diesem betriebssystem sind die sicherheitsmechanismen am strengsten realisiert. bei einer installation

von sql server unter windows 95/98 ist diese ebene der sicherheit nicht vorhanden.

sql server

für die verbindung zu sql server muß der benutzer über eine gültige sql-server-anmeldung verfügen. unter windows nt läßt sich die überprüfung - die sogenannte authentifizierung - des benutzers aus der windows-nt-anmeldung übernehmen. im betriebssystem windows 95/98 ist sql server allein für die authentifizierung verantwortlich.

datenbanken

die anmeldung beim betriebssystem und bei sql server allein genügt noch nicht, um auf die datenbanken von sql server zugreifen zu können. der benutzer muß auch die passenden berechtigungen für die gewünschten datenbanken besitzen.

datenbankobjekte

damit ein benutzer mit den objekten einer datenbank (tabellen, sichten, gespeicherte prozeduren) arbeiten kann, muß er über die erforderlichen berechtigungen für das jeweilige objekt verfügen.

21.2.2 benutzerebenen

benutzer ist nicht gleich benutzer. nicht jeder, der die hürden der anmeldung hinter sich gebracht hat, kann nach belieben in sql server schalten und walten. das ist nur dem »allmächtigen« - d.h. dem systemadministrator - möglich. angefangen beim systemadministrator mit uneingeschränkten rechten bis hin zum otto-normal-benutzer mit nahezu keinen rechten sind die benutzerebenen innerhalb von sql server folgendermaßen klassifiziert:

- systemadministrator
- datenbankbesitzer
- datenbankobjektbesitzer
- andere benutzer

systemadministrator

der systemadministrator und alle mitglieder der festen server-rolle sysadmin (siehe den abschnitt zu rollen weiter unten in diesem kapitel) haben uneingeschränkten zugriff auf sql server. dieser personenkreis kann alle administrativen aufgaben wahrnehmen, beispielsweise datenbanken sichern und wiederherstellen, sql server installieren oder anderen benutzern berechtigungen erteilen und entziehen.

datenbankbesitzer

jede datenbank hat in sql server genau einen *datenbankbesitzer* (dbo - database owner). das ist der benutzer, der die datenbank erstellt hat. dieser benutzer besitzt systemadministratorrechte für die datenbank und kann bestimmen, welchen benutzern welcher zugriff auf die datenbank gewährt wird.

datenbankobjektbesitzer

der *datenbankobjektbesitzer* ist der benutzer, der ein datenbankobjekt (tabelle, sicht, index, trigger oder gespeicherte prozedur) erstellt. der benutzer muß vorher vom datenbankbesitzer oder vom systemadministrator die berechtigung erhalten haben, einen bestimmten objekttyp zu erstellen. wenn der benutzer über die berechtigung verfügt, das objekt zu erstellen, erhält er automatisch von sql server alle auf dieses objekt zutreffenden berechtigungen (beispielsweise bei tabellen und sichten die objektberechtigungen select, update und delete oder bei gespeicherten prozeduren die berechtigung execute).

andere benutzer

alle nicht zu den drei ersten kategorien gehörenden benutzer müssen objektberechtigungen erhalten, damit sie in der datenbank arbeiten können. darüber hinaus kann der systemadministrator diesen benutzern anweisungsberechtigungen gewähren, damit sie objekte in der datenbank erzeugen und löschen können.

21.3 sicherheitskonzepte von windows nt

unter windows nt server kann sql server auf ein ausgefeiltes sicherheitssystem zurückgreifen. das zeigt sich vor allem bei den anmeldeprozeduren, die für den zugang zu sql server von jedem benutzer zu durchlaufen sind. die folgenden abschnitte beschäftigen sich mit den wichtigsten elementen von windows nt, die unmittelbaren einfluß auf die sicherheitsmechanismen von sql server haben.

21.3.1 benutzername und kennwort

als erstes muß sich ein benutzer gegenüber dem system legitimieren. dazu meldet er sich unter einem namen und gegebenenfalls mit einem kennwort an. für die verschiedenen objekte eines systems (zum beispiel ordner, dateien, drucker oder datenbanken) lassen sich dem benutzer bestimmte berechtigungen zuweisen. der benutzername ist die visitenkarte und legitimation des benutzers in einem windows-nt-netzwerk. ohne benutzername kein zugang zum netz.

21.3.2 konten

die angaben zu einem netzwerkteilnehmer werden in einem sogenannten konto zusammengefaßt. dazu gehören:

- anmeldename
- kennwort
- zugriffsberechtigungen für das system und dessen ressourcen
- zugehörigkeit zu gruppen

21.3.3 gruppen

in der regel erhalten mehrere benutzer für gleiche objekte die gleichen berechtigungen. um diese zugriffsrechte einfacher organisieren zu können, lassen sich netzwerkteilnehmer mit gleichen

berechtigungen zu *gruppen* zusammenfassen.

21.3.4 domänen

eine *domäne* ist eine zusammenstellung von computern, die der administrator eines windows-nt-server-netzwerks definiert. in einer domäne werden benutzer- und gruppenkonten zentral verwaltet.

21.3.5 benutzer-manager

in windows nt verwalten sie benutzer über den benutzer-manager für domänen. wählen sie dazu **start** / **programme** / **verwaltung (allgemein)** / **benutzer-manager für domänen**.

bild

abbildung 21.1: der benutzer-manager von windows nt

um einen neuen benutzer einzurichten, wählen sie aus dem menü **datei** den befehl **neuer benutzer**. im dialogfeld **neuer benutzer** geben sie den benutzernamen genauso an, wie ihn der benutzer bei der anmeldung an sql server über die windows-nt-authentifizierung verwenden soll. über die schaltfläche **gruppen** im unteren teil des dialogfelds gelangen sie zum dialogfeld **gruppenmitgliedschaften**. hier legen sie fest, in welchen gruppen der neue benutzer mitglied sein soll. klicken sie auf **ok**, um das dialogfeld **gruppenmitgliedschaften** zu schließen und im dialogfeld **neuer benutzer** auf **hinzufügen**, um das konto für den neuen benutzer erstellen zu lassen. abbildung 21.1 zeigt den benutzer-manager mit den geöffneten dialogfeldern **neuer benutzer** und **gruppenmitgliedschaften**.

wenn sie weitere benutzer einrichten möchten, führen sie die genannten schritte erneut aus. klicken sie am ende dieser prozedur auf die schaltfläche **schliessen**.

21.4 das sicherheitsmodell von sql server

sql server ist primär auf das betriebssystem windows nt ausgerichtet. das dokumentiert sich auch in der sicherheitsarchitektur. grundsätzlich ist aber zwischen den sicherheitssystemen von windows nt und sql server zu unterscheiden, obwohl es viele parallelen und wechselwirkungen gibt. beispielsweise lassen sich gruppen von windows nt den sicherheitskonten von sql server zuordnen.

21.4.1 anmeldung

stellen sie sich den server einmal als gebäude vor, für das sie einen ausweis brauchen, um es betreten zu können. weiterhin soll im hinteren teil des gebäudes ein datenbankzentrum angeschlossen sein, das ebenfalls eine legitimation von ihnen verlangt. zum datenbankzentrum müssen sie durch das hauptgebäude gehen. damit sie der pförtner überhaupt einläßt, brauchen sie ihren ausweis und müssen gegebenenfalls noch ein vereinbartes kennwort nennen. erst wenn sie diese erste hürde genommen haben, können sie sich zum datenbankzentrum begeben. hier kann sich nun das gleiche wiederholen: sie müssen einen zweiten ausweis vorzeigen und eventuell noch ein kennwort angeben. vielleicht genügt es aber auch, wenn sie der pförtner vom hauptgebäude beim einlaßdienst des datenbankzentrums anmeldet.

übertragen auf windows nt und sql server bedeutet das: die anmeldung bei windows nt ist unumgänglich. aber wenn sie mit sql server arbeiten möchten, kann dafür bereits ihre anmeldung bei windows nt genügen, oder sie müssen sich bei sql server separat anmelden.

die überprüfung eines benutzers bezeichnet man als *authentifizierung*. sql server unterscheidet zwei arten der identifizierung: *sql-server-authentifizierung* und *windows-nt-authentifizierung*. unter windows nt kann der benutzer einen der beiden folgenden authentifizierungsmodi wählen:

- windows-nt-authentifizierung
- gemischter modus (windows-nt-authentifizierung oder sql-server-authentifizierung)

windows-nt-authentifizierung

wenn sie der pförtner im eingangs geschilderten szenario beim einlaßdienst des datenbankzentrums anmeldet, entspricht das der windows-nt-authentifizierung von sql server. in diesem sicherheitsmodus übernimmt sql server den benutzernamen und das zugehörige kennwort aus der windows-nt-anmeldung, so daß sie eine separate anmeldung bei sql server umgehen können. als benutzer haben sie also den vorteil, daß sie sich nur ein kennwort merken müssen.

kommen wir noch einmal zum pförtner-szenario zurück: nicht jeder besucher, der das hauptgebäude betreten darf, hat gleichzeitig zugang zum datenbankzentrum. der in frage kommende benutzerkreis muß im rahmen dieser internen regelung bereits vorab von »höherer stelle« bestätigt worden sein. sinngemäß gilt das auch für die windows-nt-authentifizierung. der datenbankadministrator muß die möglichkeiten schaffen, damit sich ein benutzer per windows-nt-authentifizierung bei sql server anmelden kann.

gemischter modus

im gemischten modus hängt es vom benutzer ab, wie er sich anmelden will. gibt er beim herstellen der verbindung zu sql server einen sql-server-benutzernamen an, tritt die sql-server-authentifizierung in kraft. wenn der benutzer keinen sql-server-benutzernamen bereitstellt, kommt der mechanismus der windows-nt-authentifizierung zum tragen. und wenn keiner der angegebenen namen gültig ist? ganz klar: kein zugang zu sql server.

21.4.2 benutzer

datenbankbenutzer

wenn sie die hürden der anmeldung entweder per windows-nt-authentifizierung oder per sql-server-authentifizierung genommen haben, sind sie vom objekt ihrer begierde - einer bestimmten datenbank - immer noch ein stück entfernt. unabhängig vom sicherheitsmodus prüft sql server nämlich erst, ob sie überhaupt für den zugriff auf die datenbank zugelassen sind. andernfalls liefert sql server eine fehlermeldung zurück. für jede datenbank ist ein entsprechender benutzername erforderlich.

um die namensvielfalt nicht ausufern zu lassen, wird der anmeldename auf einen benutzernamen für die jeweilige datenbank abgebildet. wer erteilt ihnen nun den zugriff auf die datenbank? das können nur mitglieder der festen datenbankrolle db_accessadmin (der datenbankadministrator) oder db_owner (der datenbankbesitzer selbst) erledigen.

tip

benutzer, benutzername, benutzerkonto

gemeinhin bezeichnet man als *benutzer* diejenige person, die am computer mit einer anwendung arbeitet. im sprachgebrauch von sql server ist damit - zum teil - etwas ganz anderes gemeint.

der benutzer im herkömmlichen sinn meldet sich bei sql server an, um auf eine datenbank zuzugreifen. die anmeldung erfolgt unter dem *anmeldenamen*, den sql server auf einen *benutzernamen* abbildet. diesem benutzernamen wird der zugriff auf eine datenbank gewährt. ein *benutzerkonto*, das für den benutzernamen eingerichtet wird, steuert die berechtigungen zum durchführen von aktivitäten in der datenbank. das benutzerkonto bezeichnet die online-dokumentation von sql server auch als *sicherheitskonto* oder schlicht als *benutzer*.

sobald also der begriff »benutzer« auftaucht, müssen sie aus dem kontext erkennen, ob die person oder aber das benutzerkonto gemeint ist.

um das ganze auf die spitze zu treiben: ein *benutzer* kann sich bei sql server unter einem *anmeldenamen* (gleichbedeutend mit *benutzername*) anmelden und auf sql server zugreifen, ohne daß für ihn ein *benutzer* - sprich *benutzerkonto* - eingerichtet ist (siehe dazu den nächsten abschnitt zum benutzer guest).

der benutzer guest

das benutzerkonto guest (gast) nimmt eine sonderstellung ein. wenn man einer datenbank das benutzerkonto guest hinzufügt, kann jeder benutzer mit einer gültigen sql-server-anmeldung - unabhängig vom sicherheitsmodus - auf die datenbank zugreifen, ohne daß der benutzer über ein konto für diese datenbank verfügen muß.

sql server führt dabei folgende schritte durch:

- ist dem anmeldenamen ein gültiger benutzername oder ein alias für die datenbank zugeordnet, gewährt sql server dem benutzer unter diesem namen oder alias den zugriff auf die datenbank.
- wenn der benutzer kein konto für die datenbank hat, aber ein benutzer guest für die datenbank definiert ist, nimmt der benutzername die identität des benutzers guest an.
- trifft weder die erste noch die zweite bedingung zu, verweigert sql server den zugriff auf die datenbank.

nach der installation von sql server ist der benutzer guest für die datenbanken master und tempdb definiert. für alle anderen datenbanken müssen sie den benutzer guest selbst hinzufügen. das läßt sich mit der gespeicherten prozedur sp_grantdbaccess realisieren:

```
if not exists(select * from sysusers
                      where name='guest'
                      and hasdbaccess=1)
  exec sp_grantdbaccess 'guest'
```

die anweisung prüft zuerst, ob der benutzer guest bereits in der datenbank existiert und das konto datenbankzugriff hat. sollte das nicht der fall sein, wird der benutzer guest mit sp_grantdbaccess hinzugefügt.

analog dazu können sie den benutzer guest aus einer datenbank mit der gespeicherten prozedur sp_revokedbaccess entfernen:

```
exec sp_revokedbaccess 'guest'
```

wenn sie verhindern möchten, daß beliebige benutzer mit einer gültigen sql-server-anmeldung auf eine datenbank zugreifen können, entfernen sie einfach den benutzer guest aus den betreffenden datenbanken.

aliasnamen

aliasnamen verwendet man als verweis auf einen datenbankbenutzernamen, der von mehreren benutzernamen gemeinsam verwendet wird. sql server 7.0 ersetzt die mechanismen der aliasnamen durch die leistungsfähigeren rollen und stellt sie nur aus gründen der abwärtskompatibilität zu früheren sql-server-versionen zur verfügung.

21.4.3 berechtigungen

bisher konnten sie ohne weiteres auf datenbanken zugreifen, daten abrufen, ändern und löschen. irgendwelche berechtigungen waren nicht erforderlich - oder doch? wozu braucht man berechtigungen? und wer erteilt wem welche berechtigungen?

zunächst ist festzustellen, daß jedes objekt in sql server einen besitzer hat.

wenn sie beispielsweise mit

```
create database test
create table testtabelle (
spalte1 as int,
spalte2 as int)
```

die neue datenbank test erstellen und dann im enterprise manager den eintrag **db-benutzernamen** für die datenbank öffnen, erscheint dort der benutzer dbo. diese abkürzung steht für database owner - datenbankbesitzer (siehe abbildung 21.2).

bild

abbildung 21.2: der eintrag für den datenbankbesitzer

hinter dem datenbankbesitzer verbirgt sich zunächst der benutzer, der die datenbank erstellt hat. wenn sie sich jetzt unter einem anderen namen - zum beispiel otto - bei sql server anmelden und versuchen, auf die

datenbank test zuzugreifen, erhalten sie folgende fehlermeldung:

```
serverbenutzer 'otto' ist kein gültiger benutzer in datenbank
 'test'.
```

der besitzer der datenbank (oder der datenbankadministrator) hätte dem benutzer otto die berechtigungen für den zugriff auf die datenbank erteilen müssen.

berechtigungen lassen sich erteilen (grant), verweigern (deny) und entfernen (revoke). die nächsten abschnitte zeigen, welche berechtigungen es gibt und wie man sie gewährt.

sql server unterscheidet zwischen objektberechtigungen und anweisungsberechtigungen.

objektberechtigungen

objektberechtigungen steuern den zugriff auf objekte von sql server. zu diesen objekten gehören tabellen, spalten, sichten und gespeicherte prozeduren. ein benutzer muß über die passenden berechtigungen verfügen, damit er eine aktion in bezug auf ein objekt ausführen kann. das gilt nicht nur für änderungen von objekten, sondern bereits für das abrufen von daten. die erteilte objektberechtigung bezieht sich immer auf *ein* konkretes objekt, beispielsweise auf die tabelle titles in der datenbank pubs, und nicht auf alle tabellen der datenbank.

sql server unterscheidet die objektberechtigungen select, insert, update, delete und references. tabelle 21.1 zeigt die objekte und die berechtigungen, die sich dafür erteilen lassen.

objekt	berechtigungen
gesamte tabelle oder sicht	select, insert, update, delete, references (nur bei tabellen)
spalten einer tabelle oder sicht	select, update
zeilen einer tabelle oder sicht	insert, update
gespeicherte prozeduren	`execute`

tabelle 21.1: objektberechtigungen

das folgende beispiel geht davon aus, daß die datenbank von sa erstellt wurde, aber auch für den benutzer guest eingerichtet ist. wenn sie sich unter einem anderen namen als sa - zum beispiel frank - anmelden, können sie die anweisung

```
use lotto
```

problemlos ausführen, weil frank ein gültiger benutzername ist und über das gastkonto auch zugriff auf die datenbank lotto hat. weitere berechtigungen sind aber nicht erteilt worden. deshalb liefert die anweisung

```
select * from ldaten
```

die fehlermeldung:

```
select-berechtigung für objekt 'ldaten', datenbank 'lotto',
 besitzer 'dbo' verweigert.
```

der benutzer frank kann zwar über das gastkonto auf die datenbank zugreifen, hat dort aber keine weiteren berechtigungen.

anweisungsberechtigungen

anweisungsberechtigungen beziehen sich nicht auf ein bestimmtes objekt, sondern legen fest, wer objekte im administrativen sinn manipulieren darf, zum beispiel datenbanken erstellen oder sichern. tabelle 21.2 zeigt eine übersicht der anweisungsberechtigungen.

art der berechtigung	berechtigt zum erstellen von
`create database`	datenbanken. diese berechtigung kann nur der systemadministrator sa und nur für benutzer in der datenbank master erteilen.
`create procedure`	gespeicherten prozeduren.
`create table`	tabellen.
`create default`	standardwerten für eine tabellenspalte.
`create rule`	regeln für eine tabelle.
`create view`	sichten.
`backup database`	sicherungen der datenbank.
`backup log`	sicherungen des transaktionsprotokolls.

tabelle 21.2: anweisungsberechtigungen

wie sie objektberechtigungen und anweisungsberechtigungen erteilen und entziehen, erfahren sie im entsprechenden abschnitt weiter unten in diesem kapitel.

21.5 rollen

die in früheren versionen von sql server vorhandenen gruppen wurden in der version 7.0 durch die leistungsfähigeren *rollen* ersetzt. mit rollen lassen sich logische gruppen von benutzern mit bestimmten berechtigungen bilden.

analog zu den benutzernamen sind windows-nt-gruppen und sql-server-gruppen (bzw. -rollen) zwei separate mechanismen.

wenn man einer rolle eine berechtigung erteilt, verweigert oder aufhebt, wirkt sich das auf alle benutzer aus, die mitglieder dieser rolle sind. ein benutzer in sql server 7.0 kann auch mitglied mehrerer rollen sein.

sql server unterscheidet zwei arten von rollen: server-rollen und datenbankrollen.

21.5.1 server-rollen

server-rollen steuern die berechtigungen für den gesamten server. eine liste der festen server-rollen läßt sich mit der gespeicherten prozedur sp_helpsrvrole erhalten.

rolle	beschreibung	berechtigungen
`sysadmin`	systemadministratoren	kann alle arten von aktivitäten in sql server ausführen.
`securityadmin`	sicherheitsadministratoren	kann anmeldungen an sql server steuern und datenbankberechtigungen erteilen.
`serveradmin`	server-administratoren	kann einstellungen von sql server modifizieren und sql server herunterfahren.
`setupadmin`	setup-administratoren	kann die replikation installieren und erweiterte gespeicherte prozeduren steuern.
`processadmin`	prozeßadministratoren	kann sql-server-prozesse steuern.
`diskadmin`	festplattenadministratoren	kann datenträgerdateien verwalten.
`dbcreator`	datenbankersteller	kann datenbanken erstellen und modifizieren.

tabelle 21.3: feste server-rollen von sql server

die feste server-rolle sysadmin schließt die berechtigungen für alle anderen festen server-rollen ein.

die server-rollen können sie auch im enterprise manager anzeigen. erweitern sie dazu in der konsolenstruktur des servers den zweig **sicherheit**, und klicken sie auf den eintrag **serverrollen**. der detailbereich listet daraufhin die verfügbaren server-rollen mit namen und beschreibungen auf (siehe abbildung 21.3).

bild

abbildung 21.3: anzeigen der server-rollen im enterprise manager

21.5.2 datenbankrollen

datenbankrollen fassen benutzer auf der ebene einzelner datenbanken zusammen. damit lassen sich berechtigungen für bestimmte benutzer oder gruppen von benutzern für datenbankspezifische objekte erteilen. man kann rollen auch verschachteln und auf diese weise den anmeldungen hierarchische gruppen von berechtigungen zuweisen.

sql server unterscheidet drei arten von datenbankrollen:

- feste datenbankrollen
- benutzerdefinierte datenbankrollen
- implizite rollen

feste datenbankrollen

die von sql server bei der installation vordefinierten rollen für eine datenbank bezeichnet man als *feste datenbankrollen*. obwohl die festen datenbankrollen in verschiedenen datenbanken unter dem gleichen namen existieren, bleibt der gültigkeitsbereich einer rolle auf eine bestimmte datenbank beschränkt.

tabelle 21.4 führt die festen datenbankrollen auf.

rolle	beschreibung	berechtigungen
`db_owner`	datenbankbesitzer	uneingeschränkter zugriff auf alle datenbankobjekte. kann objekte löschen und neu erstellen. kann anderen benutzern objektberechtigungen erteilen. ein benutzer dieser rolle kann wartungs- und konfigurationsaktivitäten in der datenbank ausführen. diese rolle schließt die funktionalität aller anderen festen datenbankrollen ein.
`db_accessadmin`	administratoren für den datenbankzugriff	steuert den zugriff auf die datenbank, indem windows-nt-gruppen, windows-nt-benutzer und sql-server-benutzer der datenbank hinzugefügt bzw. entfernt werden.
`db_securityadmin`	administratoren für die datenbanksicherheit	verwaltet die sicherheit, d.h. anweisungs-/objektberechtigungen und rollen in der datenbank.
`db_ddladmin`	datenbank-ddl-administratoren	kann objekte in der datenbank erstellen, ändern und löschen.

db_backupoperator	administratoren für die datenbanksicherung	kann die datenbank sichern.
db_datareader	datenbankdatenleser	uneingeschränkter select-zugriff auf die daten aller tabellen in der datenbank. diese rolle gewährt keine insert-, delete- und update-berechtigungen für die datenbanktabellen.
db_datawriter	datenbankdatenschreiber	kann insert-, delete- und update-anweisungen für alle tabellen der datenbank ausführen. diese rolle gewährt keine select-berechtigung für die datenbanktabellen.
db_denydatareader	datenbank-verweigerungsdatenleser	verweigert select-berechtigungen für alle datenbanktabellen. allerdings erlaubt es diese rolle den benutzern, vorhandene tabellenschemas zu modifizieren. vorhandene tabellen können weder erstellt noch gelöscht werden.
db_denydatawriter	datenbank-verweigerungsdatenschreiber	verweigert die ausführung von insert-, select- und update-anweisungen für alle datenbanktabellen.
public	öffentliche rolle	jeder datenbankbenutzer ist mitglied der public-rolle. der benutzer wird automatisch mitglied der öffentlichen rolle, wenn er zugriffsrechte für die datenbank erhält.

tabelle 21.4: feste datenbankrollen

die feste datenbankrolle db_owner schließt die berechtigungen für alle anderen festen server-rollen ein.

im enterprise manager können sie die datenbankrollen wie folgt anzeigen: erweitern sie in der konsolenstruktur die gewünschte datenbank, und klicken sie auf den eintrag **rollen**. im detailbereich werden die für die jeweilige datenbank definierten rollen aufgelistet (siehe abbildung 21.4).

bild

abbildung 21.4: datenbankrollen für die datenbank pubs

jeder benutzer in einer datenbank ist automatisch mitglied der public-rolle. wenn allen benutzern einer datenbank bestimmte berechtigungen erteilt werden sollen, kann man diese berechtigungen für die public-rolle vereinbaren.

mit *benutzerdefinierten datenbankrollen* lassen sich benutzer für eine bestimmte sicherheitsfunktion gruppieren. während server-rollen für den gesamten server gelten, beziehen sich benutzerdefinierte datenbankrollen auf einzelne datenbanken.

bei benutzerdefinierten datenbankrollen ist zwischen den typen standardrolle und anwendungsrolle zu unterscheiden.

standardrollen

standardrollen bieten eine datenbankspezifische methode zum erstellen benutzerdefinierter rollen, mit denen sich die sicherheit durchsetzen und verwalten läßt. vom konzept her entspricht die standardrolle den datenbankgruppen der früheren versionen von sql server, bietet aber zusätzliche funktionalität.

anwendungsrollen

sql server implementiert die sicherheitsmechanismen in der datenbank selbst. das hat genau die gleichen vorteile, wie sie sie bei der durchsetzung von einschränkungen und geschäftsregeln auf der server-seite kennengelernt haben. die aspekte der sicherheit lassen sich in den meisten fällen auf der systemebene am besten kontrollieren. allerdings kann es bei bestimmten anwendungen erforderlich sein, spezielle sicherheitssteuerungen zu realisieren.

in einem unternehmen müssen zum beispiel alle mitarbeiter der buchhaltung auf die gehaltsdatenbank zugreifen können. normalerweise erfolgen die buchungen über eine anwendung, in der verschiedene sicherheitsprüfungen realisiert sind. die buchhaltung ist ein sensibler bereich, in dem schon geringste fehler zu einem chaos führen können. ein benutzer mit zugriff auf die gehaltsdatenbank könnte sich aber auch mit sql server query analyzer bei sql server anmelden, die daten direkt ändern und auf diese weise die prüfeinrichtungen der buchhaltungsanwendung unterlaufen.

sql server 7.0 bietet jetzt die möglichkeit, mit hilfe von anwendungsrollen den zugriff auf bestimmte daten nur durch eine festgelegte anwendung zuzulassen. die authentifizierung des benutzers beginnt somit bereits in der anwendung. der benutzer nimmt normalerweise gar keine notiz davon, daß sich die anmeldeprozedur geändert hat.

die anwendung kann den benutzer auffordern, sich mit seinem sql-server-benutzernamen oder vorzugsweise seiner windows-nt-anmeldung gegenüber der anwendung zu legitimieren. die anwendung übernimmt ihrerseits die anmeldung des benutzers bei sql server (mit dem beim start angeforderten benutzernamen) und aktiviert die anwendungsrolle. von diesem zeitpunkt an wirken die berechtigungen der anwendungsrolle. die dem benutzer ursprünglich erteilten berechtigungen werden außer kraft gesetzt. dieser zustand bleibt während der gesamten verbindungsdauer der anwendung mit sql server erhalten.

21.6 anmeldungen verwalten

die anmeldungen für sql server können sie mit einem anmeldungserstellungs-assistenten, dem enterprise manager und gespeicherten prozeduren verwalten. als erstes wird gezeigt, wie sie mit dem assistenten einen benutzer erstellen. der anmeldungserstellungs-assistent ist allerdings nicht uneingeschränkt zu empfehlen, da sich beispielsweise keine standarddatenbank festlegen läßt. in der voreinstellung ist ein benutzer nach der anmeldung mit der systemdatenbank master verbunden. dabei kann es durchaus passieren, daß der benutzer unbeabsichtigt daten in dieser datenbank verändert, was sich auf das gesamte system auswirkt. arbeiten sie mit dem assistenten, wenn sie nur selten einen benutzer einrichten müssen, und greifen sie dann auf die anderen werkzeuge von sql server zurück, um entsprechende anpassungen für den jeweiligen benutzer vorzunehmen.

21.6.1 anmeldungen erstellen

anmeldungserstellungs-assistent

um einen neuen benutzer einzurichten, vorhandene benutzer zu löschen oder kennwörter zu ändern, müssen sie sich als systemadministrator bei sql server anmelden (oder mitglied der festen server-rolle securityadmin sein).

mit dem anmeldungserstellungs-assistenten können sie einen neuen benutzer einrichten. führen sie dazu folgende schritte aus:

1. erweitern sie die konsolenstruktur im enterprise manager bis zum server, für den sie einen benutzer einrichten möchten.
2. wählen sie **extras / assistenten**. erweitern sie im dialogfeld **assistenten auswählen** den eintrag **datenbank**, und markieren sie **anmeldungserstellungs-assistent**. klicken sie auf **ok**.
3. klicken sie im startdialogfeld des anmeldungserstellungs-assistenten auf **weiter**.
4. im zweiten dialogfeld legen sie den authentifizierungsmodus für den benutzernamen fest (siehe abbildung 21.5). klicken sie auf **weiter**.

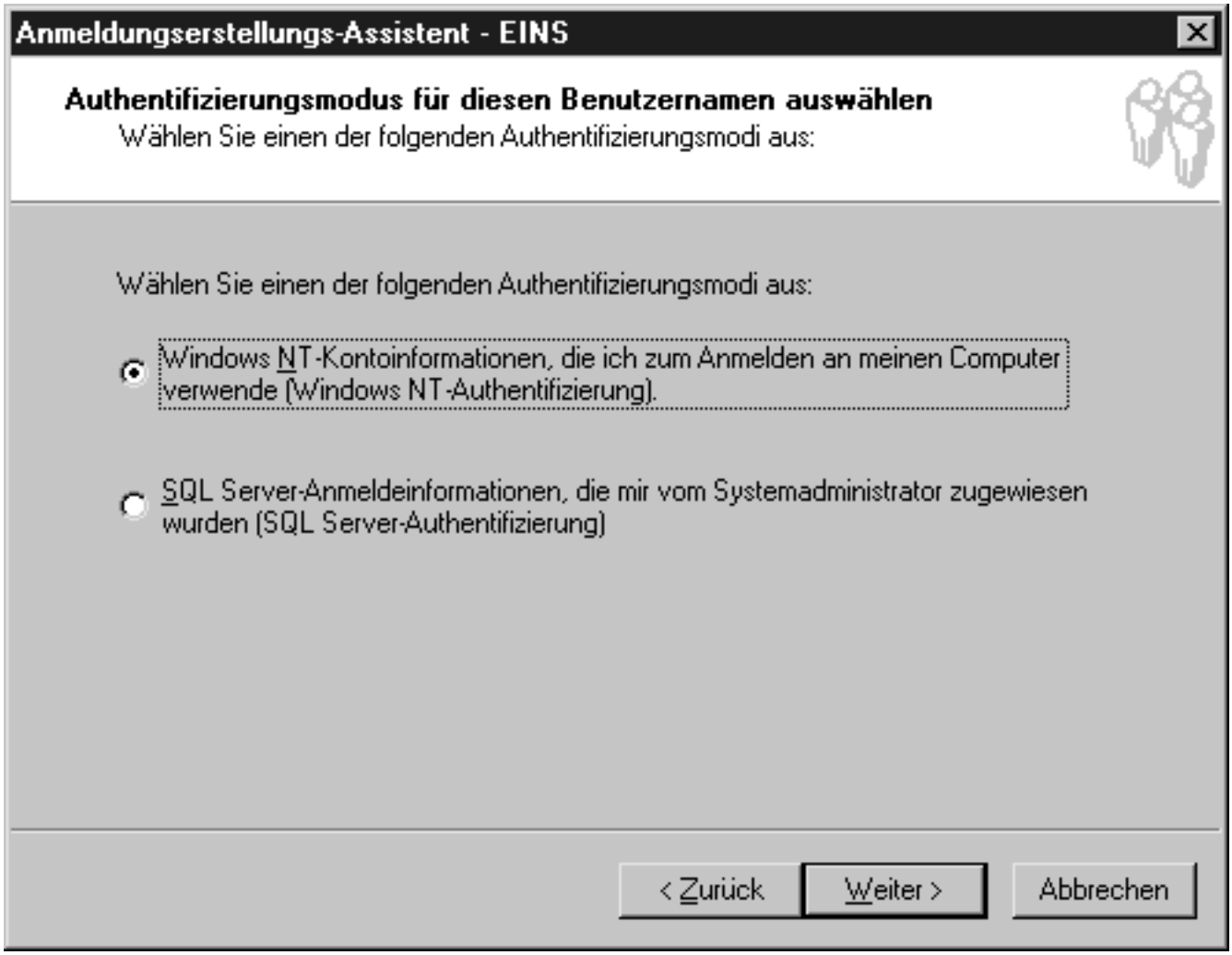


abbildung 21.5: authentifizierungsmodus festlegen

5. wenn sie die windows-nt-authentifizierung gewählt haben, legen sie im dritten dialogfeld des assistenten das windows-nt-konto fest (siehe abbildung 21.6). bei sql-server-authentifizierung geben sie den benutzernamen und ein kennwort im dialogfeld gemäß abbildung 21.7 ein. klicken sie auf **weiter**.

bei windows-nt-authentifizierung muß der benutzer oder die gruppe von windows nt bereits vorhanden sein, bevor sie das konto in sql server übernehmen können. außerdem muß der benutzername derselbe sein wie für die windows-nt-domäne.

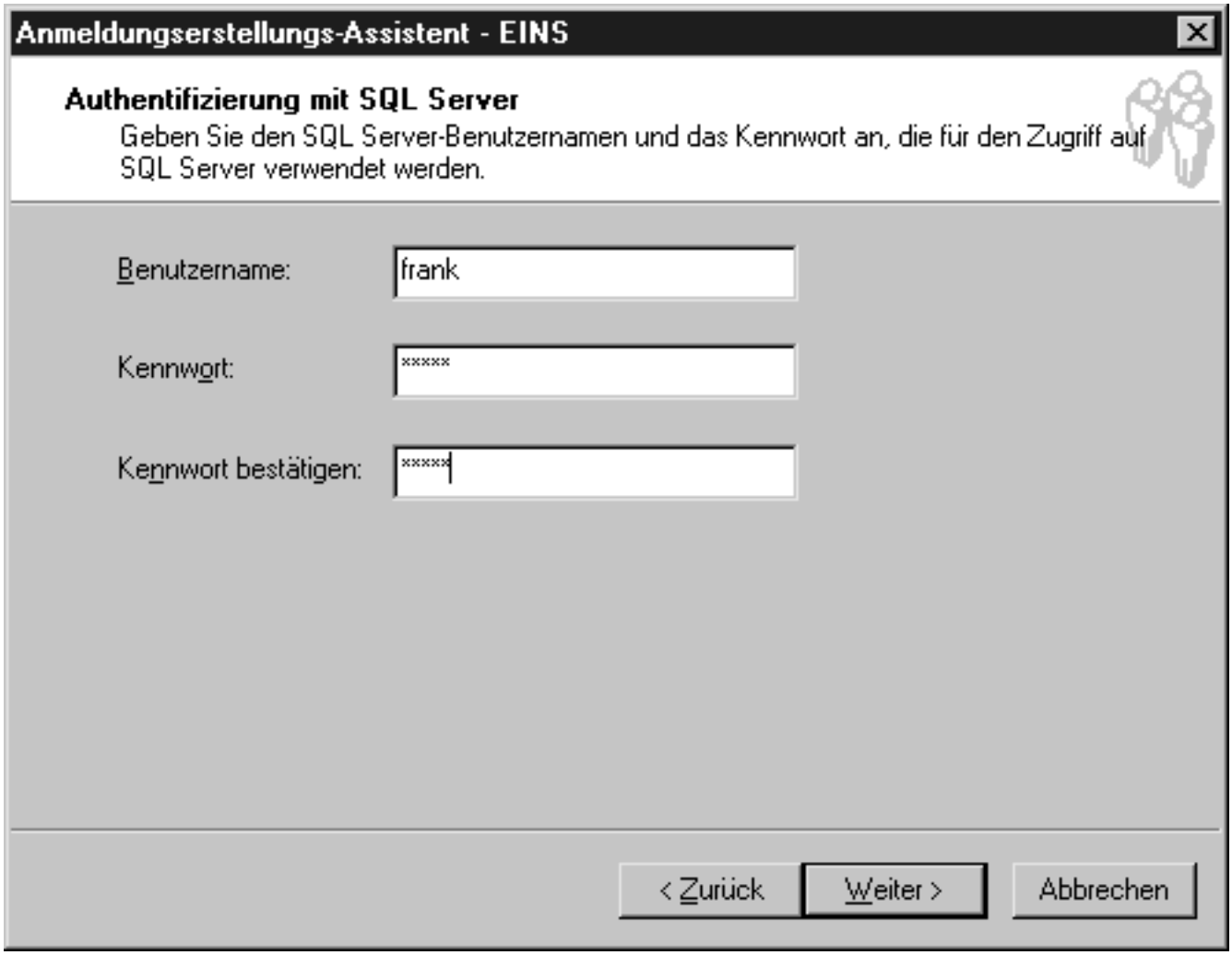


abbildung 21.6: benutzername für sql-server-authentifizierung festlegen

abbildung 21.7: windows-nt-konto für windows-nt-authentifizierung festlegen

6. unabhängig vom authentifizierungsmodus gelangen sie zum vierten dialogfeld, in dem sie den zugriff auf sicherheitsrollen gewähren können (siehe abbildung 21.8). klicken sie auf **weiter**.

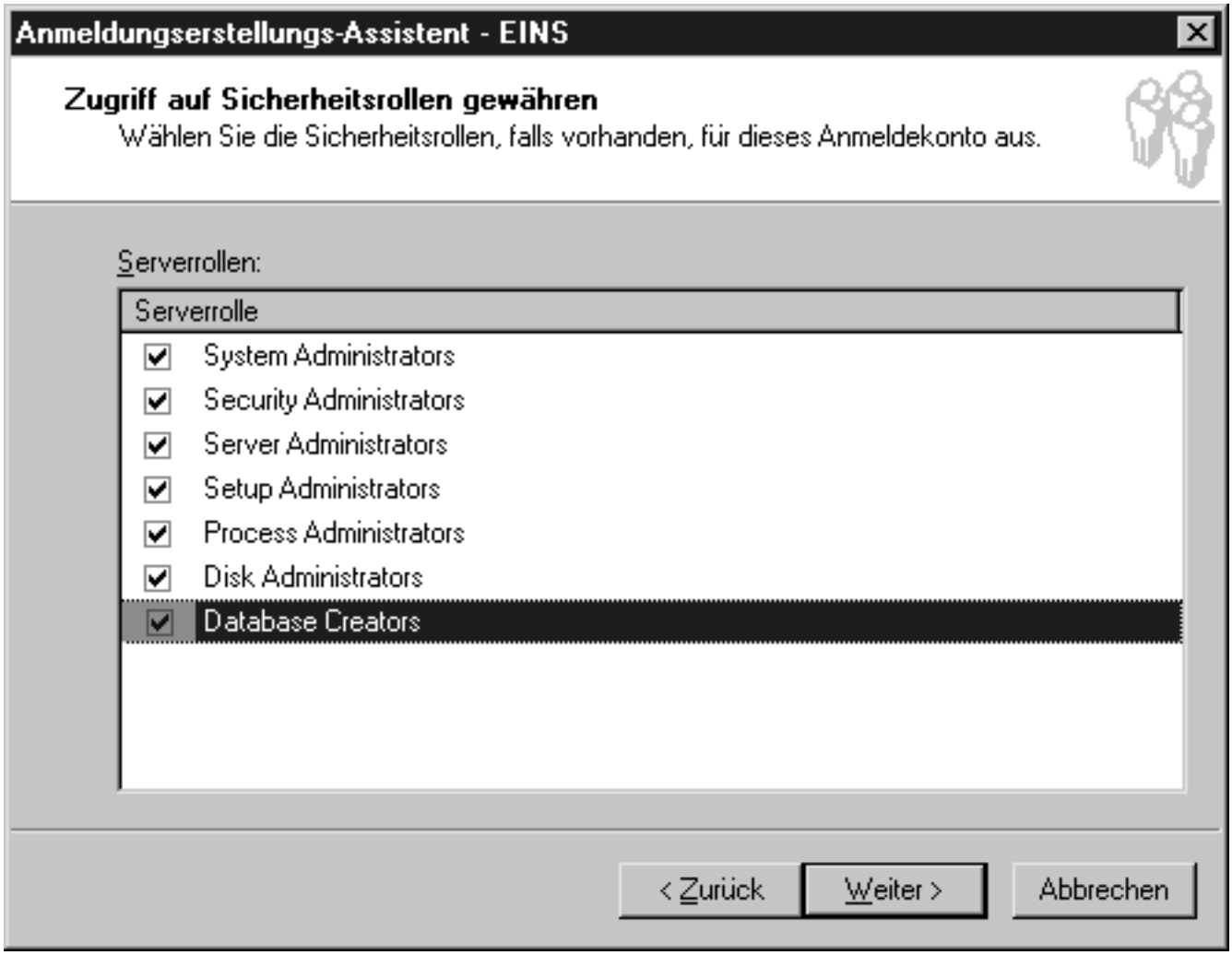


abbildung 21.8: auswahl der server-rollen, die dem benutzer gewährt werden sollen

7. im fünften dialogfeld des anmeldungserstellungs-assistenten können sie die datenbanken auswählen, auf die der benutzer zugreifen darf (siehe abbildung 21.9). klicken sie auf **weiter**.

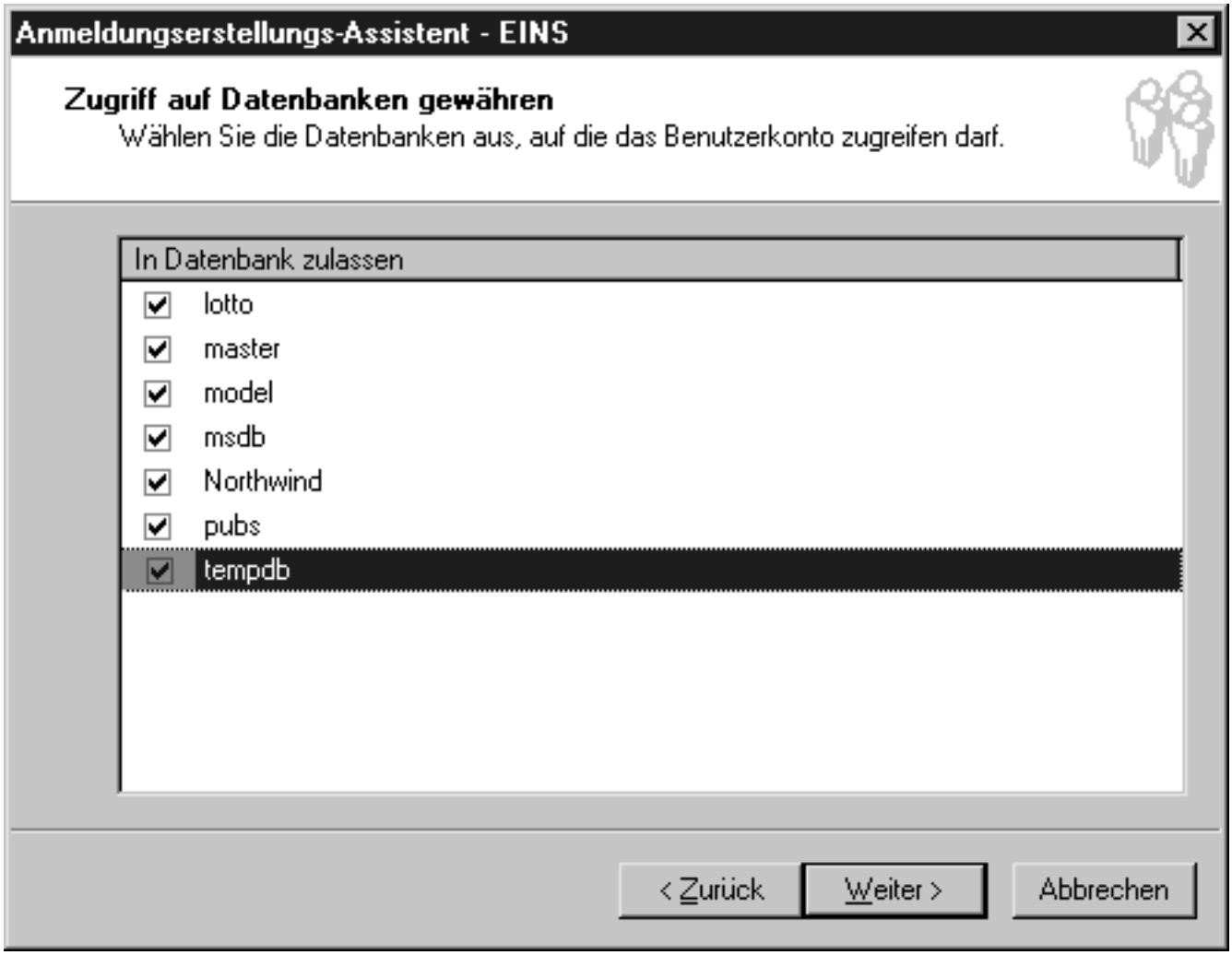


abbildung 21.9: hier legen sie den zugriff auf einzelne oder alle datenbanken fest

8. schließlich zeigt der assistent im letzten dialogfeld eine zusammenfassung der von ihnen gewählten einstellungen an (siehe abbildung 21.10). klicken sie auf **fertigstellen**, um das konto für den neuen benutzernamen einrichten zu lassen. nach kurzer zeit erscheint das dialogfeld **assistent beendet!** mit einer erfolgsmeldung.

bild

abbildung 21.10: letztes dialogfeld des anmeldungserstellungs-assistenten

enterprise manager

mit dem enterprise manager erstellen sie eine anmeldung in folgenden schritten:

1. erweitern sie in der konsolenstruktur den zweig des servers, für den sie eine anmeldung erstellen wollen. erweitern sie dann den ordner **sicherheit**.
2. klicken sie mit der rechten maustaste auf den eintrag **benutzernamen**. wählen sie aus dem kontextmenü den befehl **neuer benutzername**.
3. im dialogfeld **sql-server-anmeldungseigenschaften - neuer benutzername** (siehe abbildung 21.11) können sie auf der registerkarte **allgemein** neben den einstellungen, die sie vom anmeldungserstellungs-assistenten kennen, vor allem die standarddatenbank festlegen.

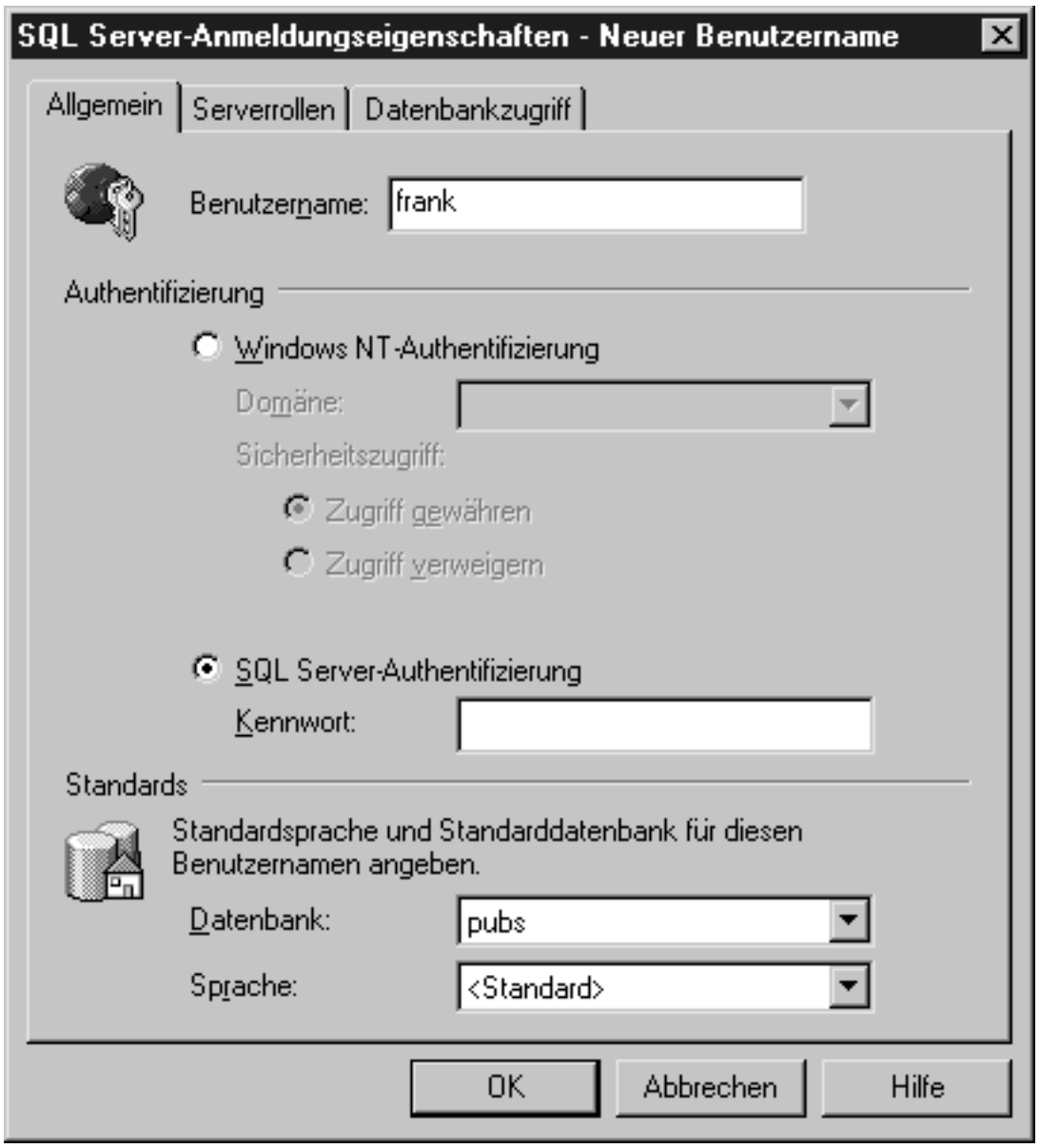


abbildung 21.11: das dialogfeld sql-server-anmeldungseigenschaften

mit dem enterprise manager können sie auch die eigenschaften eines vorhandenen benutzers ändern. klicken sie dazu mit der rechten maustaste im detailbereich auf den jeweiligen benutzernamen, und wählen sie aus dem kontextmenü den befehl **eigenschaften**. das dialogfeld **sql-server-anmeldungseigenschaften** - *benutzer* entspricht dem in abbildung 21.11 gezeigten dialogfeld, außer daß sie den authentifizierungsmodus nicht ändern können.

wenn sie die anmeldung an sql server modifizieren wollen, klicken sie in der konsolenstruktur mit der rechten maustaste auf den betreffenden server und wählen **eigenschaften** aus dem kontextmenü. im dialogfeld **sql-server-eigenschaften** - *servername* können sie auf der registerkarte **sicherheit** (siehe abbildung 21.12) zwischen gemischtem modus (option *sql server und windows nt*) oder windows-nt-authentifizierung (option *nur windows nt*) wählen.

bild

abbildung 21.12: im dialogfeld sql-server-eigenschaften können sie den authentifizierungsmodus ändern

transact-sql

bei den gespeicherten prozeduren, mit denen sich anmeldungen verwalten lassen, ist zwischen prozeduren für windows-nt-authentifizierung und prozeduren für sql-server-authentifizierung zu unterscheiden.

einen neuen benutzernamen erstellen sie mit der gespeicherten prozedur sp_addlogin.

syntax:

```
sp_addlogin [@loginame =] 'benutzername'
[,[@passwd=] 'kennwort']
[,[@defdb=]  'standarddatenbank']
[,[@deflanguage = ] 'sprache'
[,[@sid = ] 'sid']
[,[@encryptopt = ] 'verschlüsselung']
```

argumente:

benutzername und *kennwort* dürften selbsterklärend sein. die *standarddatenbank* ist diejenige datenbank, mit der der benutzer nach der anmeldung verbunden ist. in der voreinstellung gilt die systemdatenbank master.

beispiel:

die anweisung

```
sp_addlogin 'frank'
```

erstellt eine neue anmeldung für den benutzernamen frank.

wenn sie die gespeicherte prozedur sp_addlogin von einer anwendung aus aufrufen, sollten sie darauf achten, daß das kennwort zu keinem zeitpunkt im klartext zu lesen ist. das betrifft auch den kompilierten maschinencode der anwendung, da es relativ leicht ist, etwa mit einem hex-editor die ausführbare datei oder eine zugehörige dll nach einem kennwort auszuspähen.

den zugriff auf sql server erlauben sie einem benutzer- oder gruppenkonto von windows nt mit der gespeicherten prozedur sp_grantlogin:

```
sp_grantlogin [@loginame =] 'anmeldung'
```

als *anmeldung* geben sie das hinzuzufügende benutzer- oder gruppenkonto von windows nt im format domäne\benutzer an, zum beispiel:

```
sp_grantlogin 'eins\frank'
```

21.6.2 anmeldungen löschen

wie werden sie nun einen benutzer wieder los, der vielleicht aus dem unternehmen ausgeschieden ist oder den sie nur zum test eingerichtet haben? so etwas wie einen abmeldungserstellungs-assistenten oder einen anmeldungsaufhebungs-assistenten gibt es nicht. dennoch läßt sich ein benutzer über den enterprise manager wieder entfernen. führen sie dazu folgende schritte aus:

1. erweitern sie die konsolenstruktur bis zu den untereinträgen des jeweiligen servers, und erweitern sie dann den eintrag **sicherheit**.
2. klicken sie auf **benutzernamen**. markieren sie den »unliebsamen« benutzer im rechten fensterbereich. drücken sie die **ç**-taste. (wenn sie lieber mit der maus arbeiten: klicken sie mit der rechten maustaste, und wählen sie **löschen** aus dem kontextmenü.) bestätigen sie, daß dieser benutzername gelöscht werden soll (siehe abbildung 21.13).

bild

abbildung 21.13: im enterprise manager einen benutzer löschen

einen benutzernamen können sie nur dann entfernen, wenn dieser keine objekte mehr besitzt. andernfalls müssen sie erst die betreffenden objekte entfernen. allerdings können sie die objekte auch mittels der gespeicherten prozedur sp_changeobjectowner an einen vorhandenen benutzer übertragen. eine ganze datenbank läßt sich mit der gespeicherten prozedur sp_changedbowner an einen neuen besitzer übertragen.

den besitzer eines objekts in der aktuellen datenbank ändern sie mit der gespeicherten prozedur sp_changeobjectowner:

```
sp_changeobjectowner [@objname =] 'objekt',
                     [@newowner =] 'besitzer'
```

als *objekt* geben sie den namen einer in der aktuellen datenbank vorhandenen tabelle, sicht oder gespeicherten prozedur an. als besitzer können sie einen benutzer oder eine rolle von sql server oder einen benutzer oder eine gruppe von windows nt spezifizieren. die gespeicherte prozedur sp_changeobjectowner können nur mitglieder der festen datenbankrollen db_owner, db_ddladmin und db_securityadmin ausführen.

die syntax der gespeicherten prozedur sp_changedbowner lautet:

```
sp_changedbowner [@loginame =] 'anmeldename' [,[@map =]
remap_alias_flag]
```

der *anmeldename* gibt den neuen besitzer der aktuellen datenbank an. der neue benutzer kann nicht zum besitzer der aktuellen datenbank werden, wenn er bereits über einen vorhandenen alias oder das sicherheitskonto des benutzers innerhalb der datenbank zugriff auf die datenbank hat. gegebenenfalls ist zuerst der alias oder der benutzer innerhalb der aktuellen datenbank zu verwerfen (sp_dropalias bzw. sp_dropuser). eine liste der verfügbaren werte für *anmeldename* erhalten sie über die gespeicherte prozedur sp_helplogins.

wenn *remap_alias_flag* gleich true ist, werden die aliasnamen für den alten datenbankbesitzer auf den neuen besitzer der aktuellen datenbank abgebildet, bei false verworfen. der standardwert null ist gleichbedeutend mit true.

die gespeicherte prozedur sp_changedbowner können nur mitglieder der festen server-rolle sysadmin oder der besitzer der aktuellen datenbank ausführen.

die anweisung

```
exec sp_changedbowner 'frank'
```

oder

```
exec sp_changedbowner @loginame = 'frank'
```

macht frank zum besitzer der aktuellen datenbank und bildet alle aliasnamen zum alten datenbankbesitzer auf frank ab.

transact-sql

einen benutzernamen löschen sie mit der gespeicherten prozedur sp_droplogin. der gelöschte benutzer kann damit nicht mehr auf sql server unter diesem namen zugreifen:

```
sp_droplogin [@loginame =] 'benutzername'
```

der zu entfernende *benutzername* muß bereits in sql server existieren.

beispiel:

die anweisung

```
sp_droplogin 'frank'
```

entfernt den im letzten abschnitt hinzugefügten benutzernamen frank.

wenn der benutzername einem sicherheitskonto (d.h. einem benutzer) in einer datenbank zugeordnet ist, läßt sich der benutzername nicht entfernen. die online-dokumentation gibt an, den benutzer zunächst mit sp_dropuser zu entfernen. da aber die gespeicherte prozedur sp_dropuser nur aus gründen der abwärtskompatibilität in sql server 7.0 vorhanden ist und ohnehin die gespeicherte prozedur sp_revokedbaccess ausführt, sollten sie bei neuentwicklungen auf die gespeicherte prozedur sp_revokedbaccess zurückgreifen, um einen benutzer zu entfernen.

die gespeicherte prozedur sp_revokedbaccess entfernt ein sicherheitskonto (d.h. einen benutzer) aus der aktuellen datenbank:

```
sp_revokedbaccess [@name_in_db =] 'name'
```

name bezeichnet den kontonamen eines sql-server-benutzers oder einen benutzer bzw. eine gruppe von windows nt.

den eintrag für benutzer- oder gruppenkonten von windows nt entfernen sie mit der gespeicherten prozedur sp_revokelogin:

```
sp_revokelogin [@loginame =] 'anmeldung'
```

als *anmeldung* geben sie das zu entfernende benutzer- oder gruppenkonto von windows nt im format domäne\benutzer an, zum beispiel:

```
sp_revokelogin 'eins\frank'
```

wenn sie den benutzer bzw. die gruppe von windows nt nicht entfernen, jedoch am zugriff auf sql server hindern möchten, erledigen sie das mit der gespeicherten prozedur

```
sp_denylogin [@loginame =] 'anmeldung'
```

21.6.3 kennwort ändern

bei den bisherigen beispielen in diesem buch hat das thema sicherheit keine rolle gespielt. die verbindung zu sql server haben sie im gemischten modus (der standardeinstellung) mit dem benutzernamen sa und ohne kennwort hergestellt. der systemadministrator (sa) ist ein spezieller benutzername, der der festen server-rolle sysadmin zugeordnet ist und nicht geändert werden kann. die installation von sql server weist sa kein kennwort zu. wenn sie den gemischten modus für die anmeldung zu sql server gewählt haben, sollten sie als erstes das kennwort für sa ändern.

enterprise manager

das kennwort für einen sql-server-benutzernamen ändern sie mit dem enterprise manager wie folgt:

1. erweitern sie die konsolenstruktur bis zum gewünschten server, den server selbst und unter diesem server den ordner **benutzernamen**.
2. klicken sie auf **benutzernamen**, um die vorhandenen benutzer im detailbereich auf der rechten seite anzuzeigen (siehe abbildung 21.14).
3. klicken sie mit der rechten maustaste auf den zu ändernden benutzernamen (im beispiel sa), und wählen sie **eigenschaften** aus dem kontextmenü. (sie können auch den benutzernamen markieren und dann den befehl **eigenschaften** aus dem menü **vorgang** wählen.)

abbildung 21.14: auswählen des benutzers im enterprise manager

4. im dialogfeld **sql server-anmeldungseigenschaften** *benutzername* tragen sie auf der registerkarte **allgemein** das neue kennwort in das entsprechende eingabefeld ein (siehe abbildung 21.15).

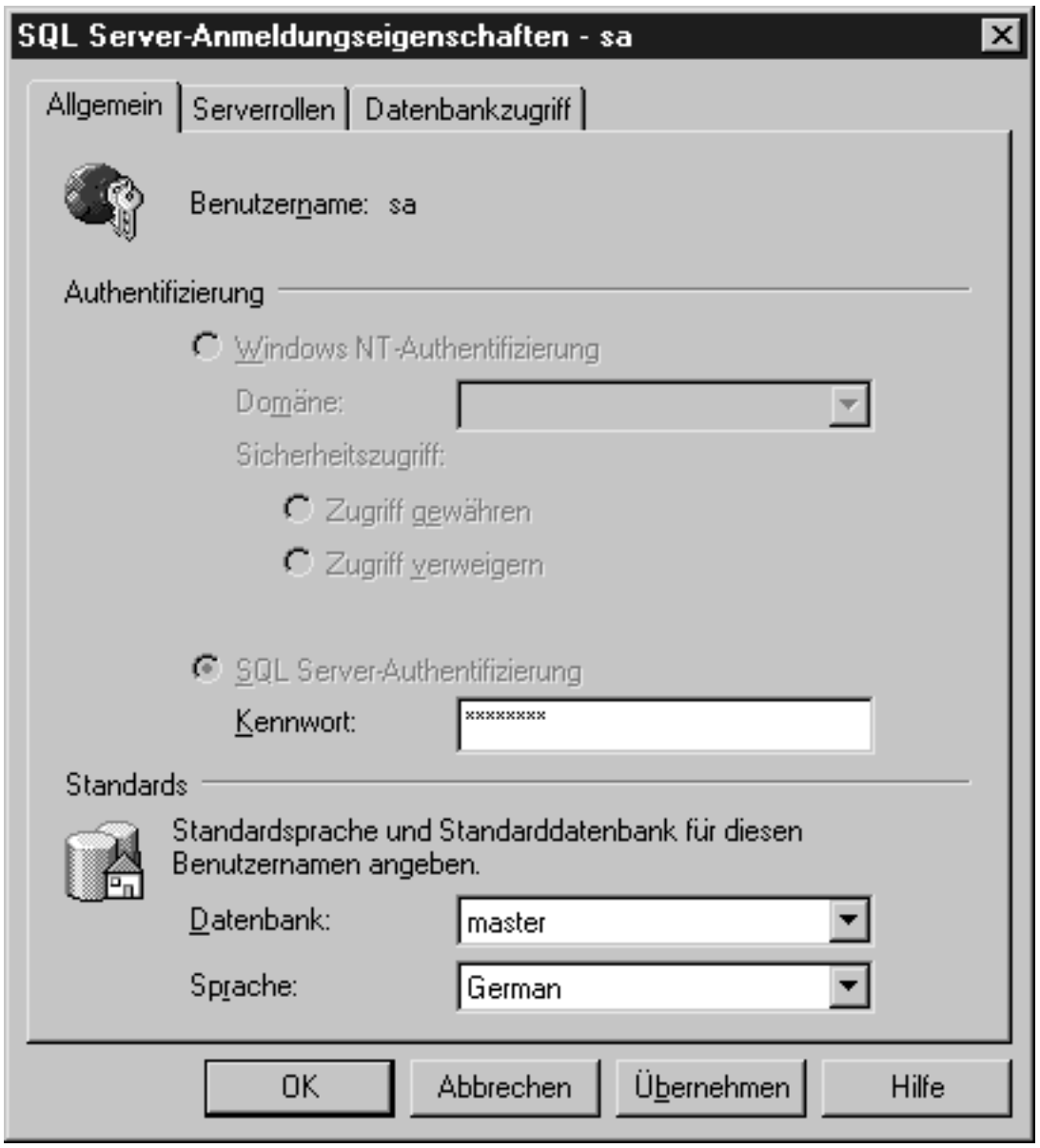


abbildung 21.15: kennwort für benutzernamen eingeben

5. klicken sie auf **übernehmen**. im dialogfeld **kennwort bestätigen** geben sie das neue kennwort noch einmal zur kontrolle ein (siehe abbildung 21.16).

abbildung 21.16: das dialogfeld kennwort bestätigen

6. klicken sie auf **ok**, um das kennwort zu bestätigen, und im dialogfeld **anmeldungseigenschaften** ebenfalls auf **ok**, um das dialogfeld zu schließen.

transact-sql

ein kennwort läßt sich mit der gespeicherten prozedur sp_password ändern.

syntax:

```
sp_password [[@old =] 'altes kennwort',]
{[@new =] 'neues kennwort'}
[,[@loginame =] 'anmeldename'
```

argumente:

mit dem argument *altes kennwort* geben sie das bisher verwendete kennwort an. der standardwert ist null. dem benutzer sa ist nach der installation das standardkennwort null zugewiesen.

als *neues kennwort* tragen sie das neue kennwort ein. wenn sie den benannten parameter *@new* nicht verwenden, müssen sie das alte kennwort angeben.

der *anmeldename* bezeichnet den namen, für den das neue kennwort gelten soll. der anmeldename muß bereits existieren.

beispiel:

```
exec sp_password null, 'ok', 'sa'
```

die anweisung legt das kennwort ok als neues kennwort für den benutzer sa fest. da nach der installation von sql server dem benutzernamen sa kein kennwort zugeordnet ist, müssen sie null als altes kennwort angeben. mit benannten parametern läßt sich die gleiche anweisung folgendermaßen formulieren:

```
exec sp_password @new ='ok', @loginame = 'sa'
```

sobald sie einen benannten parameter verwenden, müssen sie auch alle folgenden angaben mit benannten parametern schreiben. die anweisung

```
exec sp_password @new = 'ok', 'sa'
```

liefert eine fehlermeldung.

wenn sie in sql server query analyzer mit der standardanmeldung (sa ohne kennwort) die anweisung des obigen beispiels ausgeführt haben, sollten sie sich das kennwort notieren oder gut merken, da es ab sofort gültig ist. probieren sie es aus. rufen sie aus dem menü **datei** von query analyzer den befehl **verbinden** auf, um eine neue verbindung zu sql server herzustellen, wählen sie die option *sql server-authentifizierung*, und geben sie sa als benutzernamen ein. wenn sie das eingabefeld für das kennwort leer lassen, erhalten sie die fehlermeldung, daß die verbindung nicht hergestellt werden kann. bei sql server können sie sich nur noch mit dem neuen kennwort anmelden.

im fenster der noch bestehenden verbindung läßt sich das kennwort allerdings noch rückgängig machen. überschreiben sie einfach die anweisung, mit der sie das kennwort geändert haben, durch

```
exec sp_password 'ok', null, 'sa'
```

und führen sie die anweisung aus. damit ist alles wieder beim alten.

anmeldeinformationen anzeigen

mit der gespeicherten prozedur sp_helplogins können sie sich einen überblick über die auf dem server eingerichteten benutzer verschaffen.

syntax:

```
sp_helplogins [[@loginnamepattern =] 'anmeldename']
```

argumente:

wenn sie den optionalen parameter *anmeldename* nicht angeben, werden informationen zu allen benutzern angezeigt.

die bezeichnung des benannten parameters ist etwas irreführend. »pattern« weist auf ein muster hin. man könnte annehmen, daß es möglich ist, anmeldenamen mit platzhaltern wie etwa f* anzugeben, um alle benutzer anzuzeigen, die mit f beginnen. das ist allerdings nicht möglich.

beispiel:

führen sie die anweisung

```
exec sp_helplogins
```

im sql server query analyzer aus. abbildung 21.17 zeigt das ergebnis.

bild

abbildung 21.17: anmeldeinformationen im sql server query analyzer anzeigen

benutzer und gruppen von windows nt sind immer von benutzern und rollen in sql server zu unterscheiden. das drückt sich auch in den zu verwendenden gespeicherten prozeduren aus:

sp_grantlogin erlaubt windows-nt-benutzern, auf sql server zuzugreifen, während sp_addlogin einen benutzer in sql server hinzufügt bzw. eine anmeldung auf datenbankbenutzer abbildet.

21.7 rollen verwalten

im anmeldungserstellungs-assistenten und im enterprise manager haben sie bereits dialogfelder für rollen kennengelernt. die zuordnung eines benutzers zu einer rolle nehmen sie im enterprise manager wie folgt vor:

1. erweitern sie in der konsolenstruktur den zweig **sicherheit**. klicken sie auf **benutzernamen**. im detailbereich erscheinen die vorhandenen benutzer.
2. klicken sie mit der rechten maustaste auf den gewünschten benutzernamen, und wählen sie aus dem kontextmenü den befehl **eigenschaften**.

bild

abbildung 21.18: die registerkarte serverrollen

3. im dialogfeld **sql server-anmeldungseigenschaften** sind auf der registerkarte **serverrollen** die rollen aufgeführt, denen der gewählte benutzer angehört (siehe abbildung 21.18). klicken sie auf den namen einer server-rolle, um die zugehörige beschreibung im unteren teil des dialogfelds anzuzeigen.

wenn sie auf die schaltfläche **eigenschaften** klicken, erscheint das dialogfeld **serverrolle - eigenschaften** für die ausgewählte server-rolle. auf der registerkarte **allgemein** sind die benutzer verzeichnet, die momentan zu dieser server-rolle gehören (siehe abbildung 21.19).

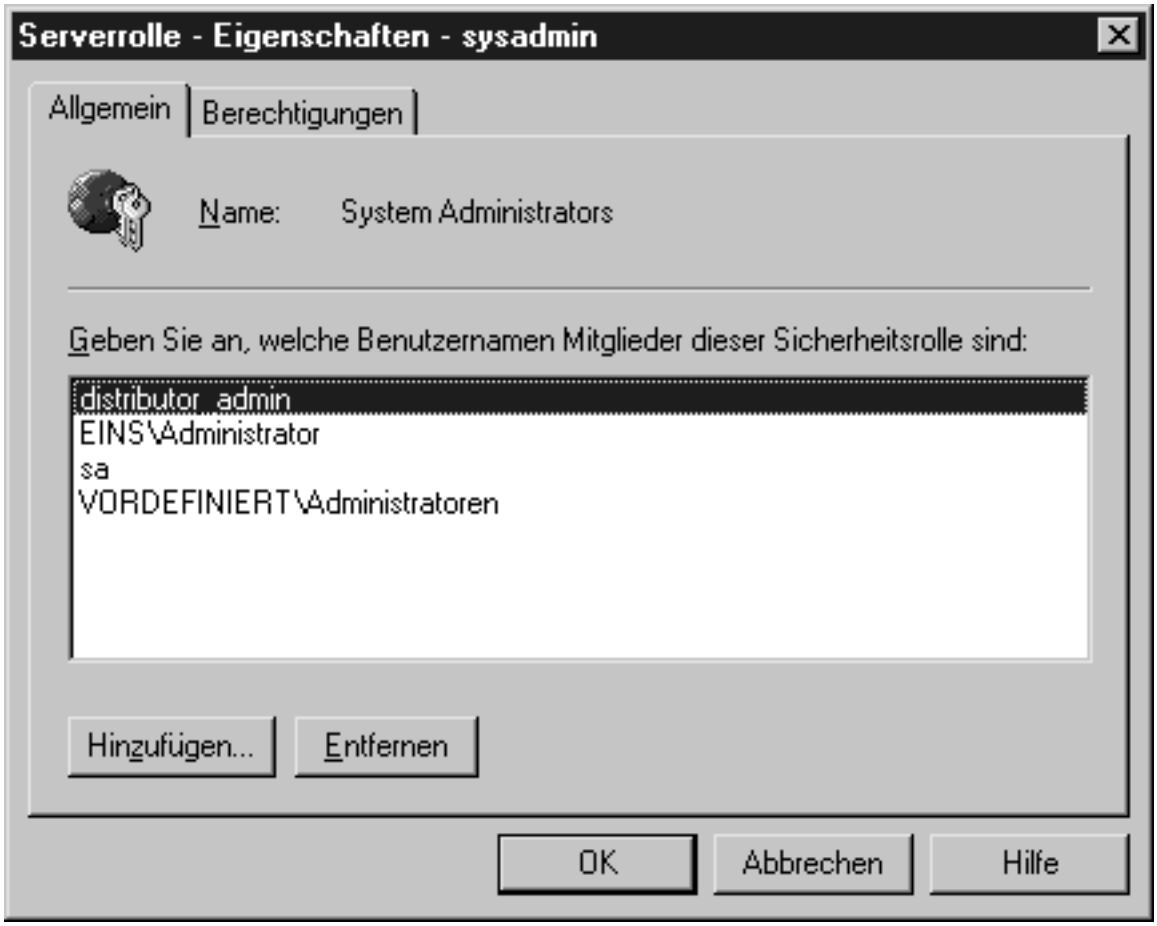


abbildung 21.19: verzeichnis der mitglieder in der ausgewählten server-rolle

auf der registerkarte **berechtigungen** sind die befehle aufgelistet, die mitglieder der gewählten server-rolle ausführen können (siehe abbildung 21.20).

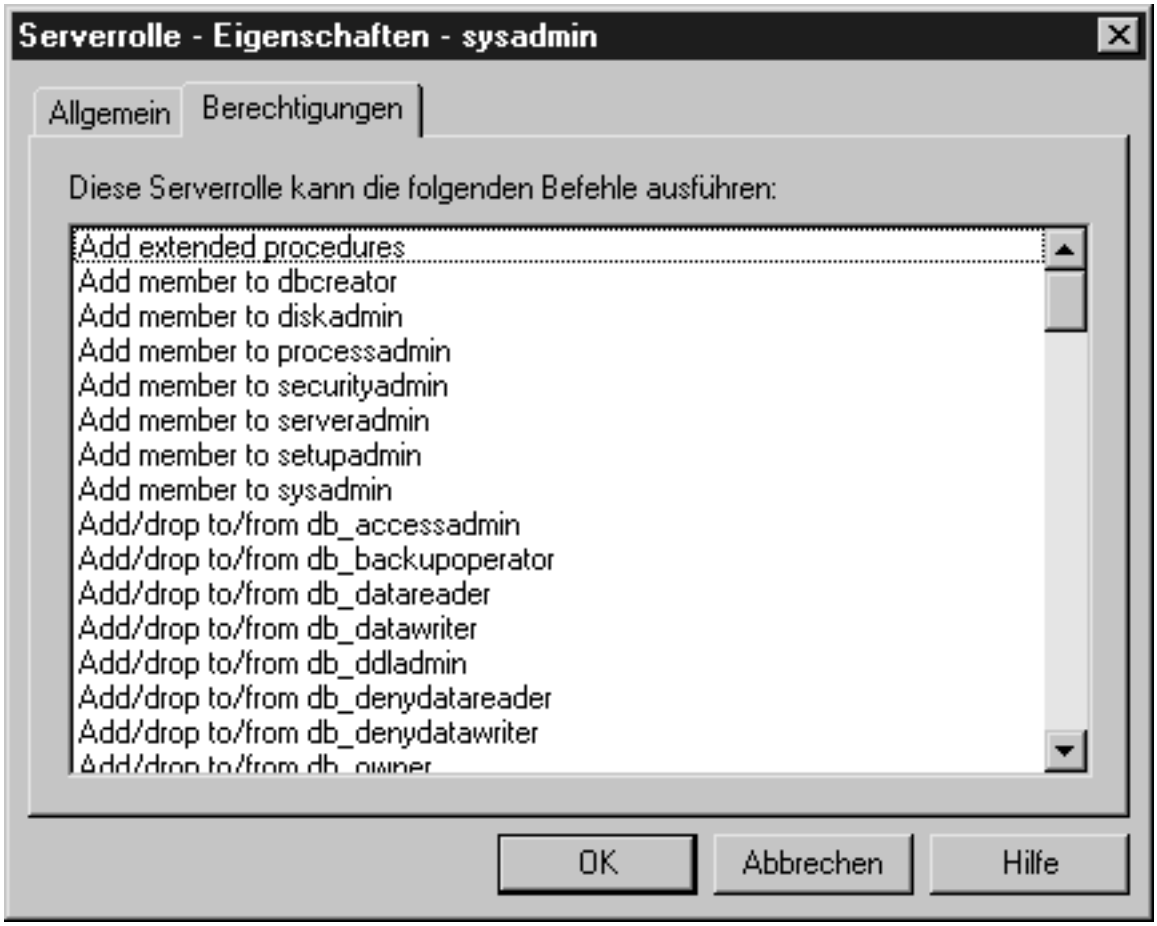


abbildung 21.20: liste der befehle, die die server-rolle ausführen kann

transact-sql

die verwaltung der server-rollen können sie auch mit den folgenden gespeicherten prozeduren vornehmen:

einen benutzer nehmen sie mit der gespeicherten prozedur

```
sp_addsrvrolemember [@loginame =] 'benutzername',
                    [@rolename =] 'serverrolle'
```

in die angegebene *server-rolle* auf und löschen ihn mit

```
sp_dropsrvrolemember [@loginame =] 'benutzername',
                     [@rolename =] 'serverrolle'
```

die gespeicherte prozedur

```
sp_helpsrvrolemember [[@srvrolename =] 'serverrolle']
```

zeigt informationen zu den mitgliedern der angegebenen *server-rolle* an und

```
sp_srvrolepermission [[@srvrolename=] 'serverrolle']
```

liefert die berechtigungen für eine feste server-rolle.

als *server-rolle* sind die folgenden werte möglich.

- sysadmin
- securityadmin
- serveradmin
- setupadmin
- processadmin
- diskadmin
- dbcreator

erläuterungen zu diesen server-rollen finden sie in tabelle 21.3 weiter oben in diesem kapitel.

21.8 datenbankzugriff und rollen

die verwaltung des datenbankzugriffs und der datenbankrollen nehmen sie im enterprise manager wie folgt vor:

1. erweitern sie in der konsolenstruktur den zweig **sicherheit**. klicken sie auf **benutzernamen**. im detailbereich erscheinen die vorhandenen benutzer.
2. klicken sie mit der rechten maustaste auf den gewünschten benutzernamen, und wählen sie aus dem kontextmenü den befehl **eigenschaften**.
3. gehen sie im dialogfeld **sql server-anmeldungseigenschaften** auf die registerkarte **datenbankzugriff** (siehe abbildung 21.21). schalten sie in der spalte **zulassen** das kontrollkästchen der gewünschten datenbank ein, um den zugriff auf die jeweilige datenbank durch den benutzernamen zu erlauben.

bild

abbildung 21.21: die registerkarte datenbankzugriff

4. im listenfeld **in datenbankrolle zulassen** schalten sie das kontrollkästchen für die datenbankrolle ein, der der benutzer angehören soll. in tabelle 21.4 weiter oben in diesem kapitel finden sie eine erläuterung der datenbankrollen.
5. klicken sie auf **eigenschaften**, um eine liste der benutzer anzuzeigen, die momentan mitglied in dieser rolle sind (siehe abbildung 21.22).

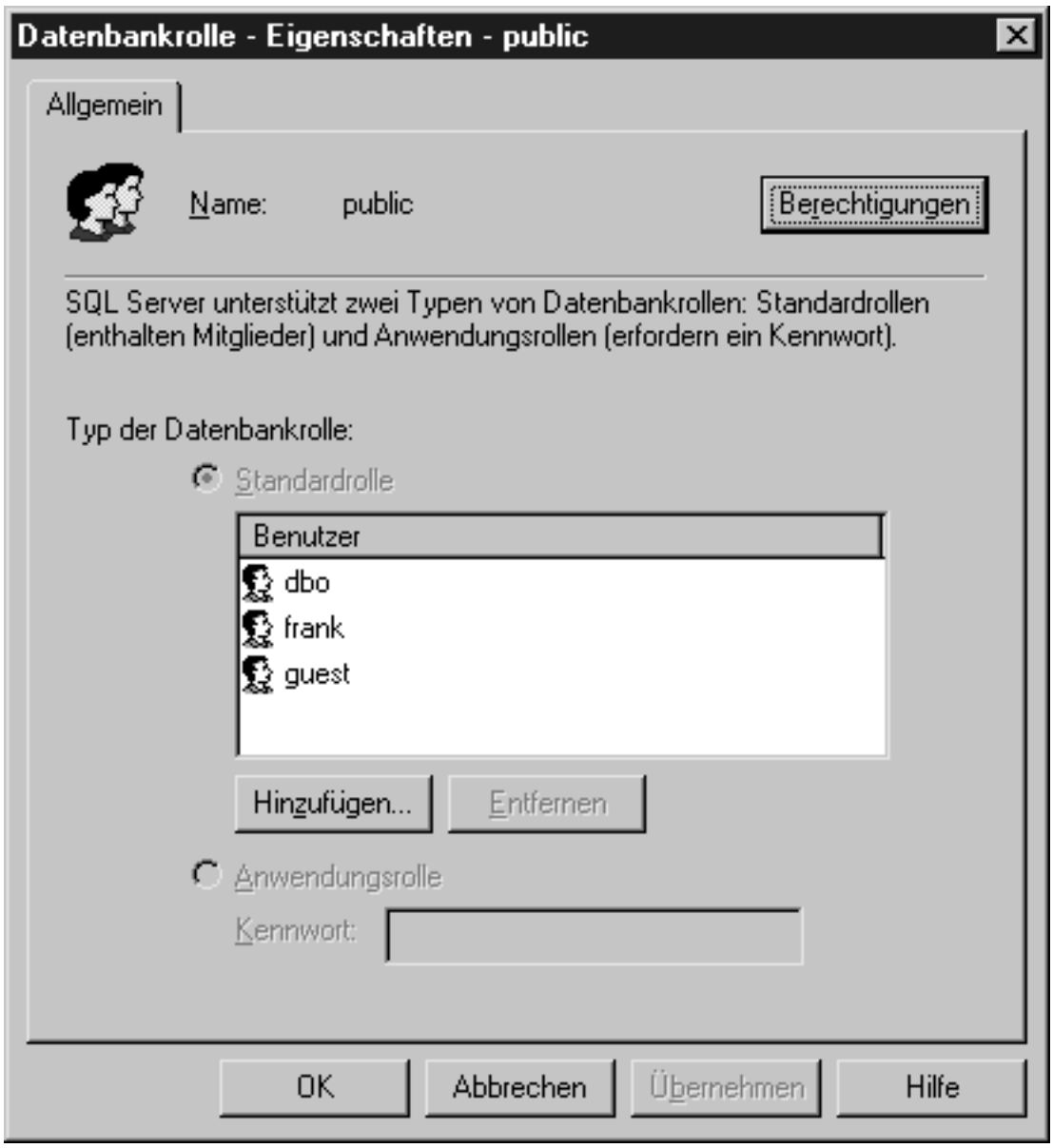


abbildung 21.22: mitglieder der datenbankrolle public

6. wenn sie im dialogfeld **datenbankrolle - eigenschaften** auf die schaltfläche **berechtigungen** klicken, gelangen sie zu dem in abbildung 21.23 dargestellten dialogfeld, in dem die berechtigungen für die jeweilige datenbankrolle aufgeführt sind. die schaltfläche **berechtigungen** ist nur für die public-rolle und benutzerdefinierte rollen verfügbar.

bild

abbildung 21.23: die berechtigungen für die datenbankrolle public

7. schließen sie die eigenschaftsdialogfelder, und klicken sie im dialogfeld **sql server-anmeldungseigenschaften** auf **übernehmen** und dann auf **ok**, um die geänderten zuordnungen zu den server-rollen zu übernehmen.

transact-sql

die folgende übersicht zeigt die gespeicherten prozeduren, mit denen sie die verwaltung des datenbankzugriffs und der datenbankrollen per transact-sql vornehmen können:

```
sp_grantdbaccess   [@loginame =] 'benutzername'
                   [,[@name_in_db =] 'nameindatenbank'
                   [output]]
sp_revokedbaccess  [@name_in_db =] 'nameindatenbank'
sp_helpuser        [[@name_in_db =] 'nameindatenbank']

sp_addrole         [@rolename =] 'rolle'
                   [,[@ownername =] 'besitzer']
sp_droprole        [@rolename =] 'rolle'
sp_helprole        [[@rolename =] 'rolle'

sp_addapprole      [@rolename =] 'rolle',
                   [@password =] 'kennwort'
sp_dropapprole     [@rolename =] 'rolle'
sp_setapprole      [@rolename =] 'rolle',
                   [@password =] {encrypt n'kennwort'}
                                 | 'kennwort'
                   [,[@encrypt =] 'verschlüsselungstyp'
sp_helprolemember  [[@rolename =] 'rolle']
sp_addrolemember   [@rolename =] 'rolle',
                   [@membername =] 'sicherheitskonto'
sp_helpdbfixedrole [[@rolename =] 'rolle']
sp_dbfixedrolepermission [[@rolename =] 'rolle']
```

21.9 objektberechtigungen

die auf datenbankobjekte bezogenen berechtigungen (siehe tabelle 21.1 weiter oben in diesem kapitel) weisen sie mit dem enterprise manager in folgenden schritten zu:

1. erweitern sie die konsolenstruktur bis zu der datenbank, in der die betreffenden datenbankobjekte enthalten sind, und erweitern sie den ordner dieser datenbank.
2. klicken sie auf das objekt, für das sie berechtigungen erteilen wollen: **tabellen**, **sichten** bzw. **gespeicherte prozeduren**. im detailbereich erscheint daraufhin eine liste der vorhandenen objekte.
3. im detailbereich klicken sie mit der rechten maustaste auf das jeweilige objekt und wählen aus dem kontextmenü den befehl **eigenschaften**. klicken sie im eigenschaftsdialogfeld des objekts auf die schaltfläche **berechtigungen**. abbildung 21.24 zeigt als beispiel das dialogfeld **objekteigenschaften** für die tabelle ldaten der datenbank lotto. hier zeigt sich, warum das beispiel im abschnitt »objektberechtigungen« weiter oben in diesem kapitel eine fehlermeldung geliefert hat. der objekteigentümer kann zwar alle operationen für das objekt ausführen, ein anderer benutzer benötigt aber die entsprechenden objektberechtigungen. diese sind aber für keinen der im dialogfeld aufgeführten datenbankbenutzer (bzw. die datenbankrollen) erteilt worden.

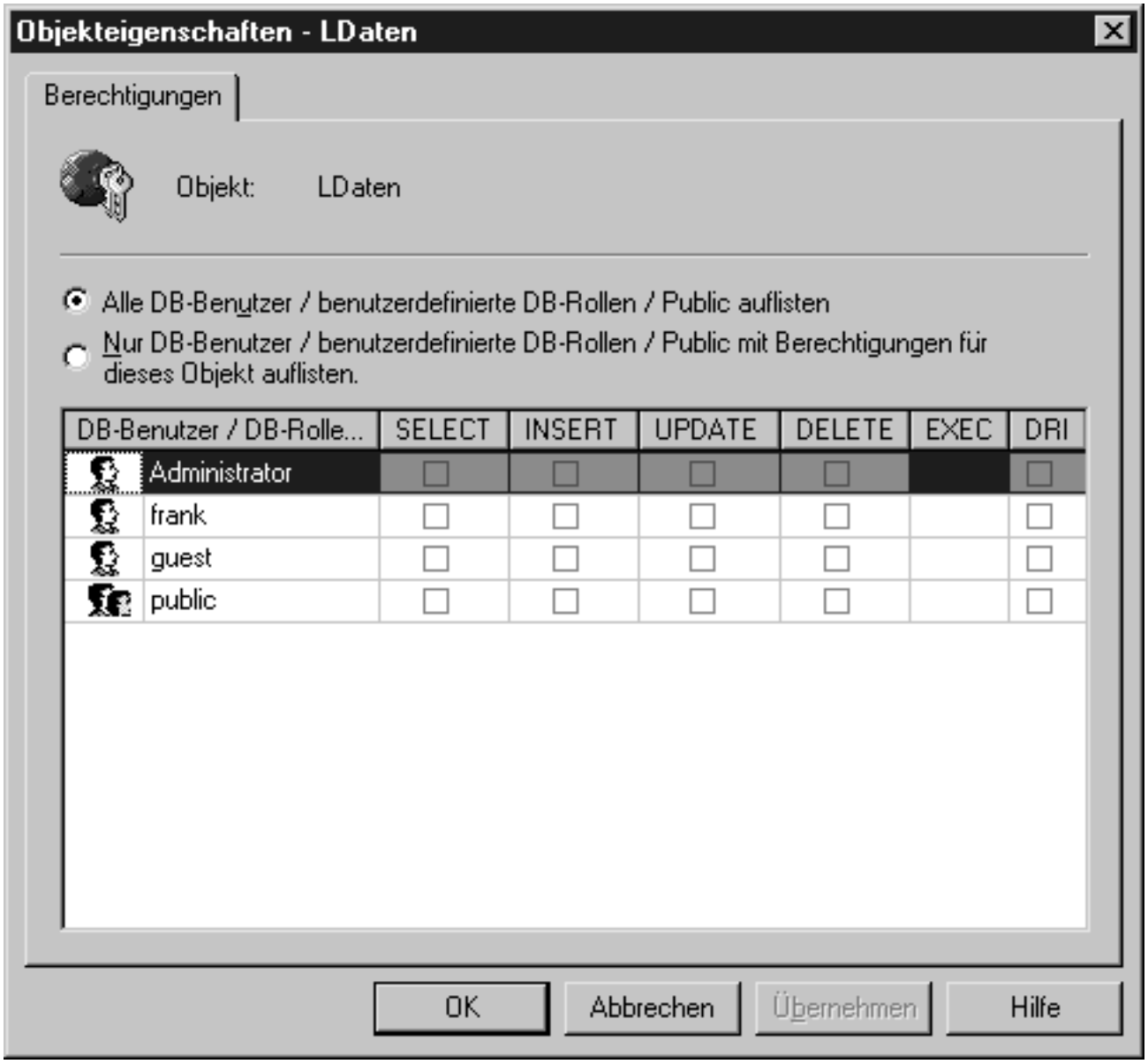


abbildung 21.24: das dialogfeld objekteigenschaften für die tabelle ldaten der datenbank lotto

4. schalten sie die kontrollkästchen der in der spaltenüberschrift angegebenen berechtigungen für den gewünschten benutzer ein, um dem benutzer die jeweilige berechtigung zu erteilen, oder aus, um die berechtigung zu entziehen. wenn sie im beispiel das kontrollkästchen der select-berechtigung für den benutzer frank einschalten, können sie die im schritt 3 erwähnte anweisung ohne fehler ausführen. durch wiederholtes klicken auf ein kontrollkästchen können sie zwischen berechtigung erteilen (schwarzes x), berechtigung verhindern (rotes x) und berechtigung entziehen (leeres kontrollkästchen) wechseln.
5. sobald sie auf **übernehmen** klicken, werden die erteilten, entzogenen oder verhinderten berechtigungen wirksam - auch für eine bereits bestehende verbindung zur jeweiligen datenbank. klicken sie dann auf **ok**, um die geöffneten dialogfelder zu schließen.

für einen bestimmten benutzer können sie in einem arbeitsgang auch für mehrere objekte berechtigungen erteilen:

1. klicken sie in der konsolenstruktur im erweiterten zweig der datenbank auf **db-benutzernamen** oder **rollen**. der detailbereich listet daraufhin die für die datenbank eingetragenen benutzer bzw.

rollen auf. klicken sie mit der rechten maustaste auf den benutzer bzw. die rolle, und wählen sie aus dem kontextmenü den befehl **eigenschaften**.

Bild

abbildung 21.25: im dialogfeld datenbankbenutzer können sie berechtigungen für mehrere objekte erteilen

2. im eigenschaftsdialogfeld **datenbankbenutzer** bzw. **datenbankrolle** klicken sie auf die schaltfläche **berechtigungen**. abbildung 21.25 zeigt als beispiel das eigenschaftsdialogfeld **datenbankbenutzer** für die datenbank pubs.
3. für das umschalten der berechtigungen gilt das gleiche wie für das dialogfeld **objekteigenschaften** (siehe abbildung 21.24 bzw. schritt 4 weiter oben in diesem abschnitt).
4. klicken sie auf **übernehmen**, um die änderungen wirksam werden zu lassen, und dann auf **ok**, um die dialogfelder zu schließen.

transact-sql

mit den transact-sql-anweisungen grant und revoke lassen sich sowohl objekt- als auch anweisungsberechtigungen erteilen bzw. entziehen. auch wenn sie normalerweise mit dem enterprise manager arbeiten - spaltenberechtigungen können sie nur per transact-sql erteilen.

die syntax der anweisung grant für das erteilen von objektberechtigungen sieht folgendermaßen aus:

```
grant {all [privileges] | berechtigung [,...n]}
{[(spaltenliste)] on {tabelle | sicht}
 | on {tabelle | sicht} [(spaltenliste)]
 | on {gespeicherteprozedur | erweiterteprozedur} }
to sicherheitskonto [,...n]
[with grant option]
[as {gruppe | rolle}]
```

die klausel with grant option erlaubt es dem angegebenen sicherheitskonto, die spezifizierten berechtigungen auf andere sicherheitskonten zu übertragen. wenn sie zum beispiel mit der anweisung

```
grant select on gehalt to peter with grant option
```

dem benutzer peter für die tabelle gehalt die select-berechtigung erteilen und die klausel with grant option verwenden, kann peter seinerseits diese berechtigung auf paul übertragen:

```
grant select on gehalt to paul
```

damit kann nun auch paul auf die tabelle gehalt zugreifen. wenn sich das weiter fortsetzt, kennen am ende alle mitarbeiter die gehaltsstruktur des betriebs. verwenden sie with grant option also mit vorsicht!

objektberechtigungen entziehen sie mit der anweisung revoke:

```
revoke [grant option for]
{all [privileges] | berechtigung [,...n]}
{[(spaltenliste)] on {tabelle | sicht}
 | on {tabelle | sicht} [(spaltenliste)]
 | on {gespeicherteprozedur | erweiterteprozedur} }
{to | from]
sicherheitskonto [,...n]
[cascade]
[as {gruppe | rolle}]
```

21.10 anweisungsberechtigungen

die in tabelle 21.2 weiter oben in diesem kapitel erläuterten anweisungsberechtigungen lassen sich mit dem enterprise manager folgendermaßen verwalten:

1. erweitern sie die konsolenstruktur bis zum ordner der gewünschten datenbank. klicken sie mit der rechten maustaste auf die datenbank, und wählen sie aus dem kontextmenü den befehl **eigenschaften**.
2. aktivieren sie im eigenschaftsdialogfeld der datenbank die registerkarte **berechtigungen** (siehe abbildung 21.26).

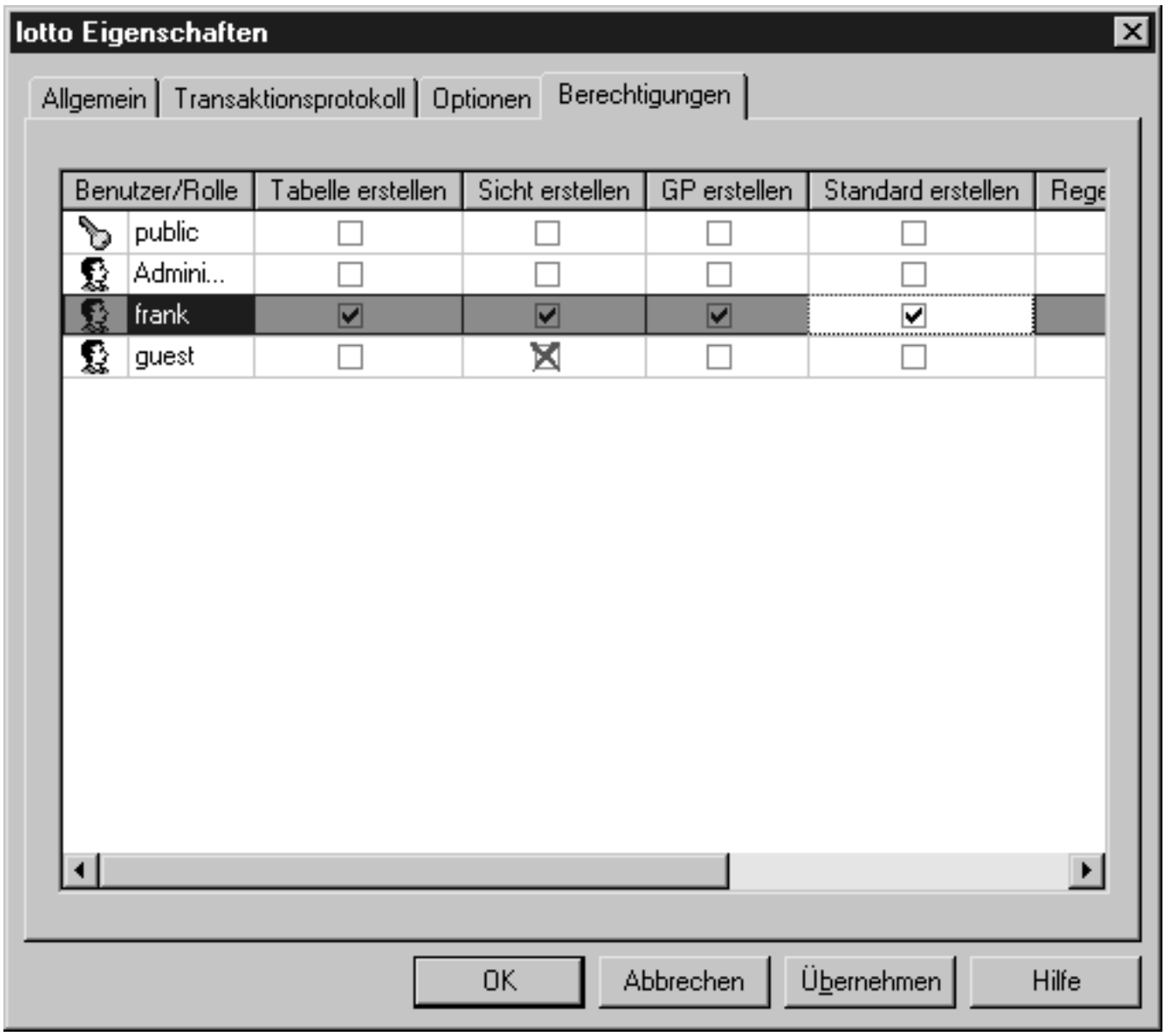


abbildung 21.26: im eigenschaftsdialogfeld der datenbank legen sie anweisungsberechtigungen fest

3. schalten sie die kontrollkästchen der anweisungsberechtigungen für den jeweiligen benutzer um (schwarzes x - berechtigung erteilen, rotes x - berechtigung verhindern, leeres kontrollkästchen - berechtigung entziehen). tabelle 21.5 zeigt die in sql server definierten anweisungsberechtigungen und die zugehörigen spaltenüberschriften im eigenschaftsdialogfeld.

art der berechtigung	spaltenüberschrift im eigenschaftsdialogfeld
create procedure	gp erstellen
create table	tabelle erstellen
create default	standard erstellen
create rule	regel erstellen
create view	sicht erstellen
backup database	sicherungs-db

backup log	sicherungsprotokoll

tabelle 21.5: anweisungsberechtigungen

sicher ist ihnen aufgefallen, daß das dialogfeld keine spalte für die anweisungsberechtigung create database enthält. diese berechtigung kann nur der systemadministrator (sa) oder ein mitglied der rolle sysadmin und nur für benutzer in der datenbank master erteilen.

4. klicken sie auf **übernehmen**, um die anweisungsberechtigungen wirksam werden zu lassen, und dann auf **ok**, um das dialogfeld zu schließen.

transact-sql

wie bereits erwähnt, dienen die transact-sql-anweisungen grant und revoke auch dazu, anweisungsberechtigungen zu erteilen bzw. zu entziehen. die syntax der anweisung grant für das erteilen von anweisungsberechtigungen lautet:

```
grant {all | anweisungsliste}
to sicherheitskonten
```

die syntax der anweisung revoke für das entziehen von anweisungsberechtigungen sieht ähnlich aus:

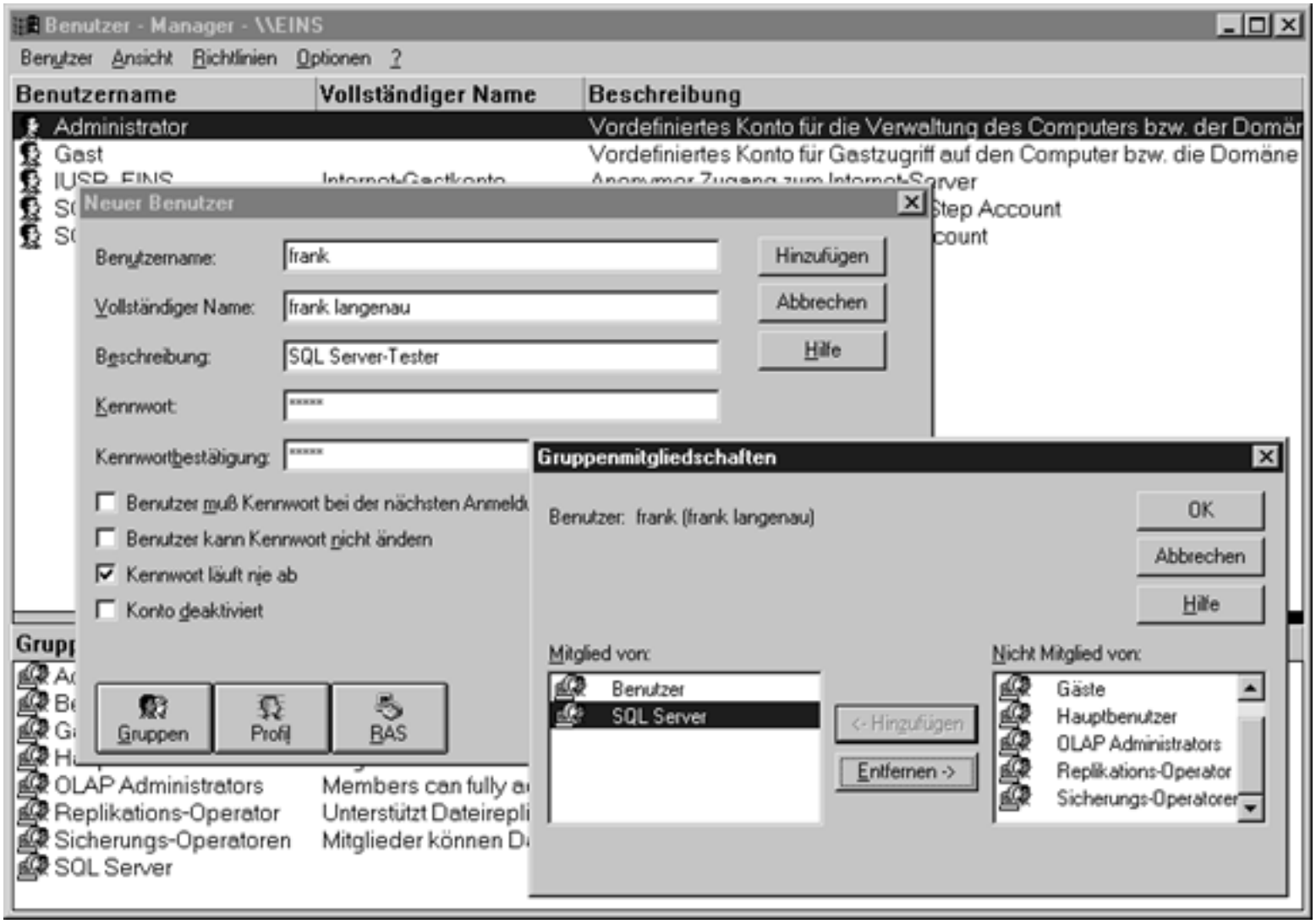
```
revoke {all | anweisungsliste}
from sicherheitskonten
```

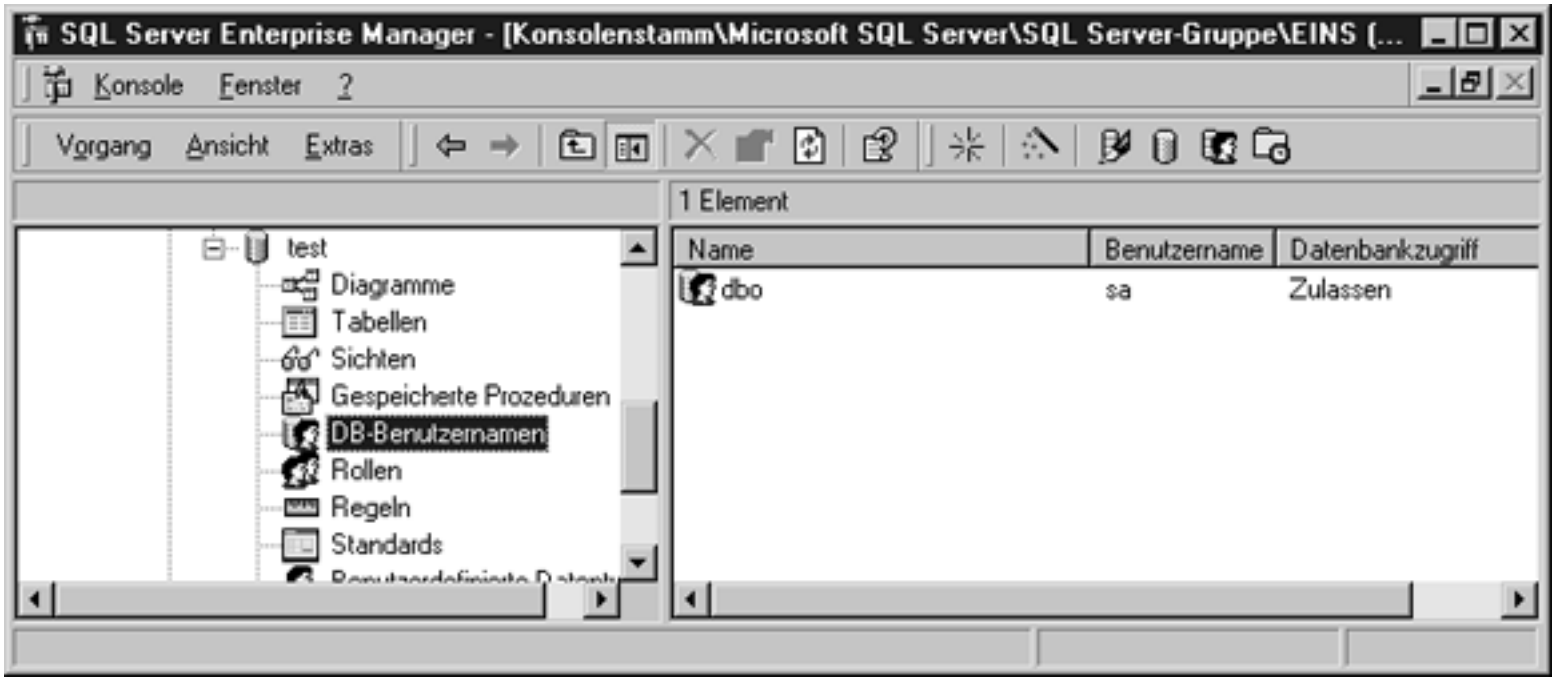
21.11 sicherheit mit sichten

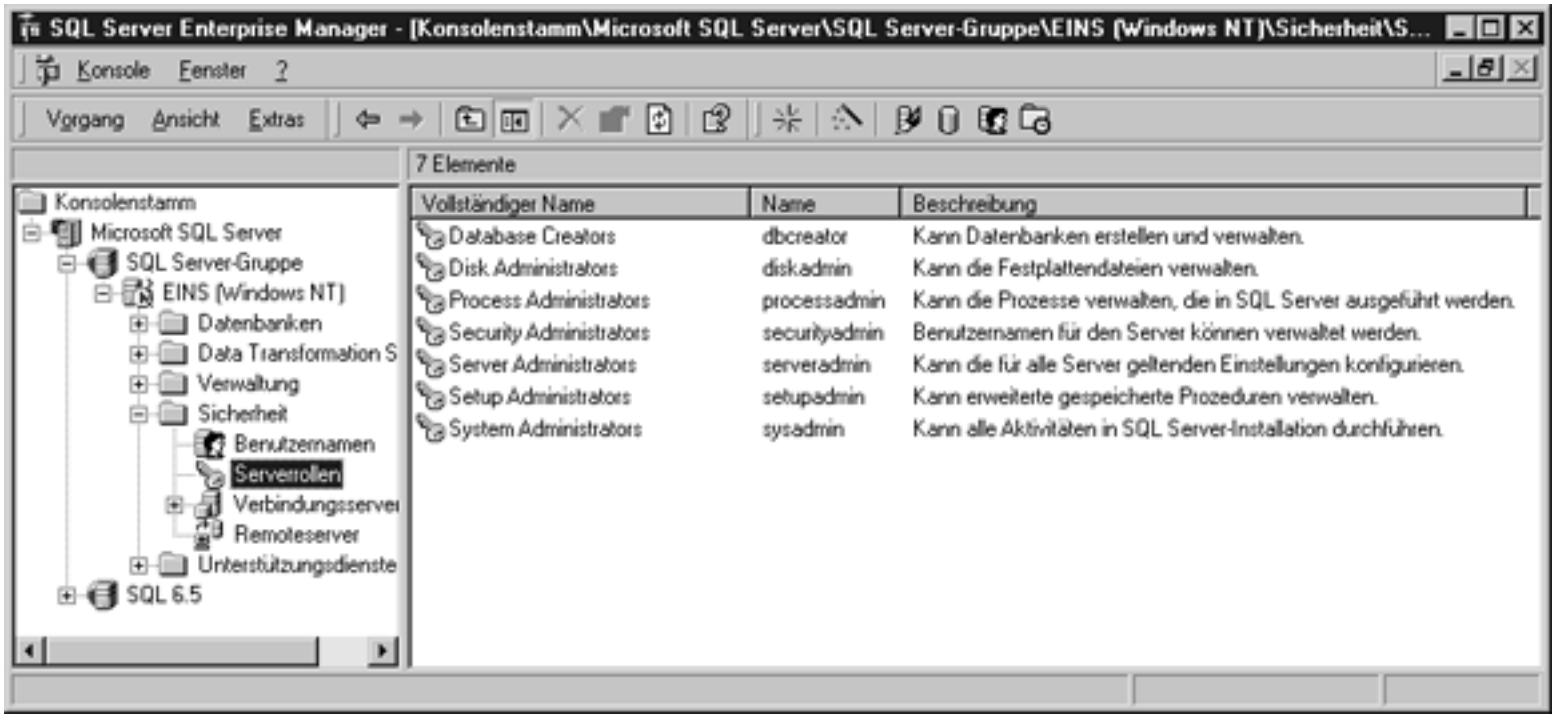
wenn sie in einer mitarbeitertabelle neben den adressen auch vertrauliche informationen wie etwa das gehalt speichern und eine sekretärin keinen einblick in die gehaltsspalte erhalten soll, wenn sie adressen von mitarbeitern aktualisieren muß, haben sie prinzipiell zwei möglichkeiten, um den zugriff auf die gehaltsspalte zu unterbinden:

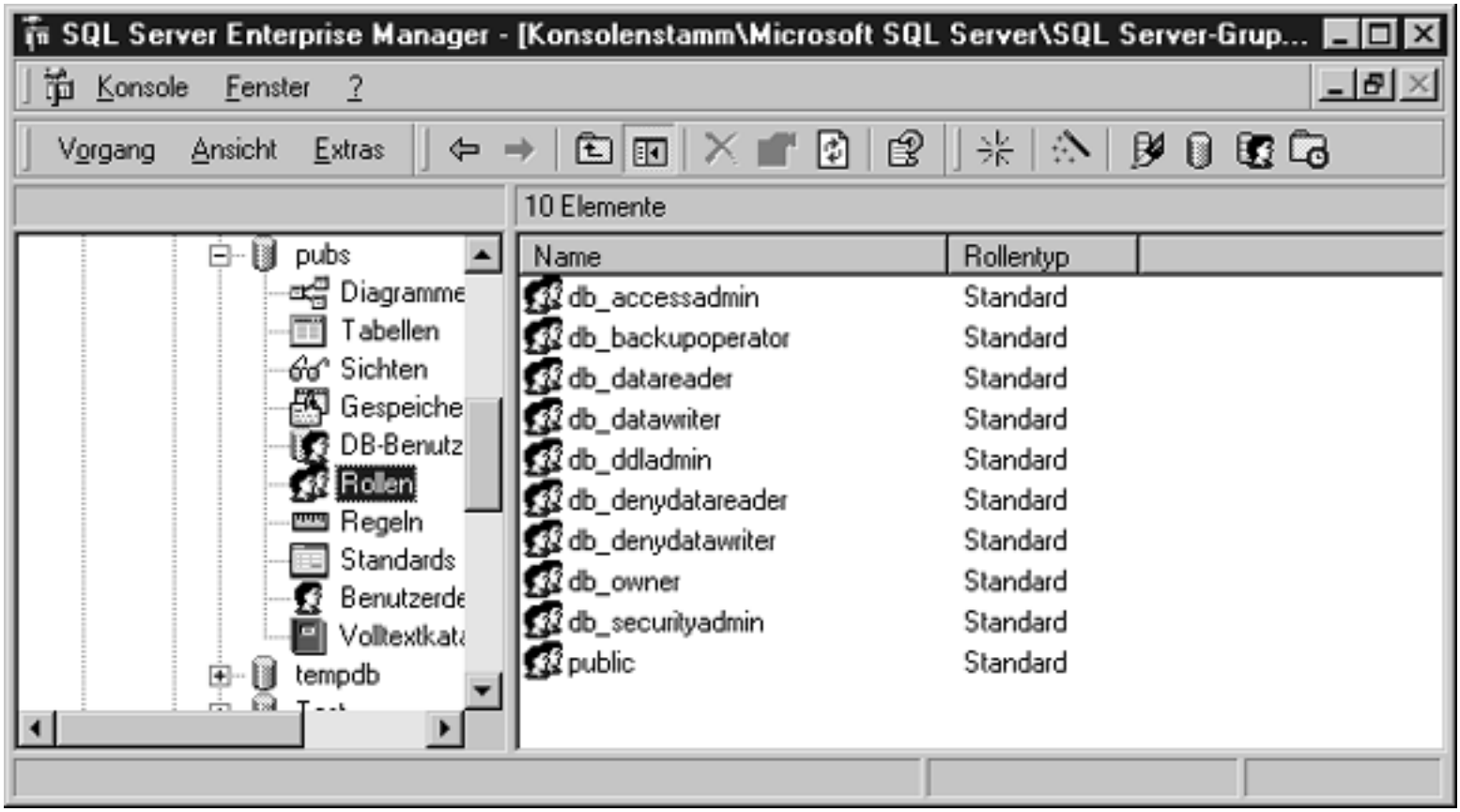
- sie vergeben spaltenorientierte objektberechtigungen für die mitarbeitertabelle.
- sie erstellen eine sicht mit den öffentlich zugänglichen spalten und erteilen nur für die sicht und nicht für die zugrundeliegende tabelle die entsprechenden berechtigungen.

wie bereits erwähnt, können sie spaltenberechtigungen nicht mit dem enterprise manager erstellen. außerdem kann es insbesondere bei komplexen datenbankstrukturen ziemlich mühsam sein, den überblick über die erteilten spaltenberechtigungen zu behalten. wesentlich einfacher ist es, wenn sie mit sichten arbeiten und die sicherheit auf dieser ebene realisieren.

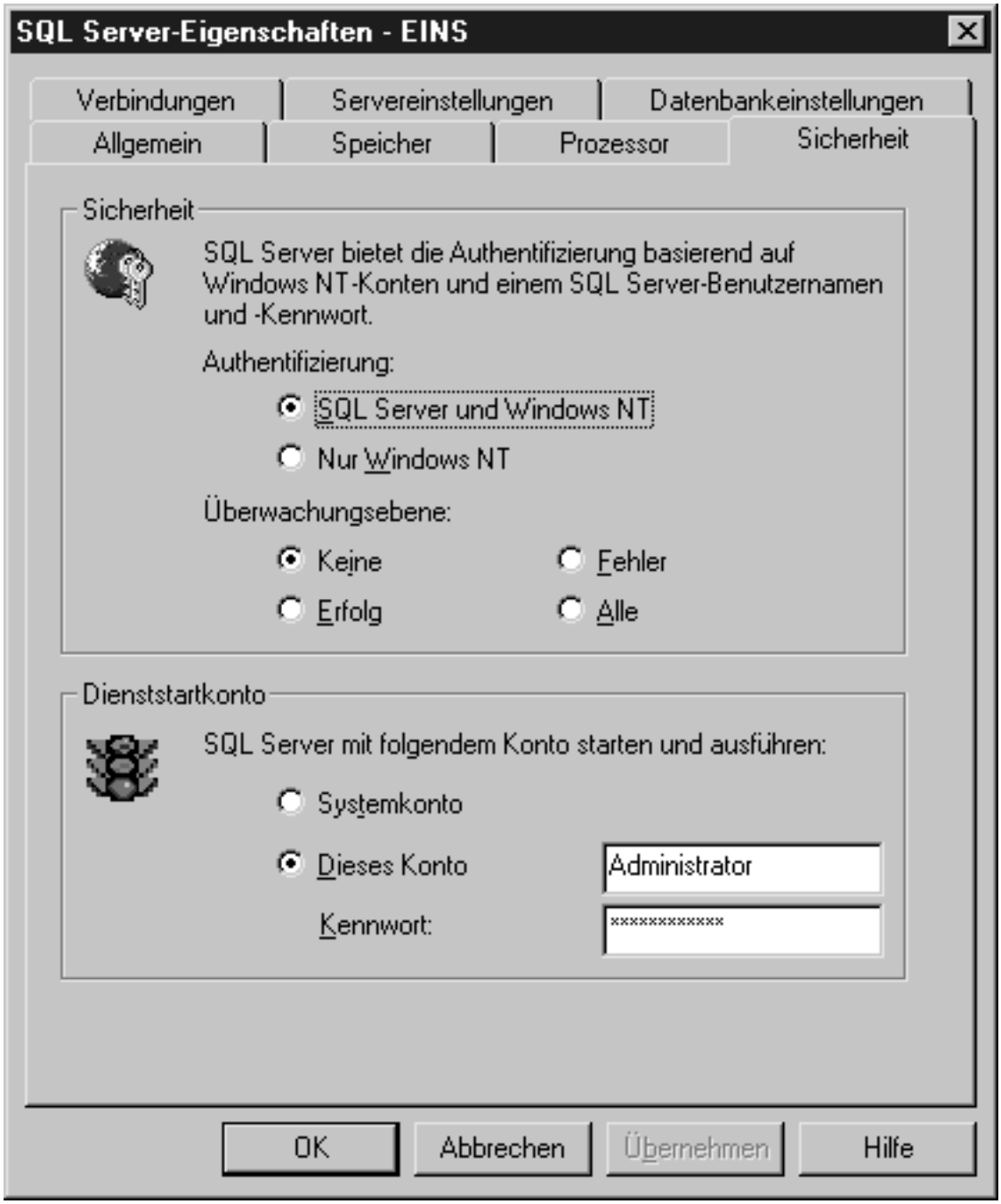

Benutzer - Manager - \\EINS
Benutzer Ansicht Richtlinien Optionen ?
Benutzername
Vollständiger Name
Beschreibung
Administrator
Vordefiniertes Konto für die Verwaltung des Computers bzw. der Domär
Gast
Vordefiniertes Konto für Gastzugriff auf den Computer bzw. die Domäne
Neuer Benutzer
Benutzername: frank
Vollständiger Name: frank langenau
Beschreibung: SQL Server-Tester
Kennwort:
Kennwortbestätigung:
Hinzufügen
Abbrechen
Hilfe
Benutzer muß Kennwort bei der nächsten Anmeld
Benutzer kann Kennwort nicht ändern
Kennwort läuft nie ab
Konto deaktiviert
Gruppen
Profil
BAS
Gruppenmitgliedschaften
Benutzer: frank (frank langenau)
OK
Abbrechen
Hilfe
Mitglied von:
Benutzer
SQL Server
<- Hinzufügen
Entfernen ->
Nicht Mitglied von:
Gäste
Hauptbenutzer
OLAP Administrators
Replikations-Operator
Sicherungs-Operatoren
OLAP Administrators
Replikations-Operator
Sicherungs-Operatoren
SQL Server

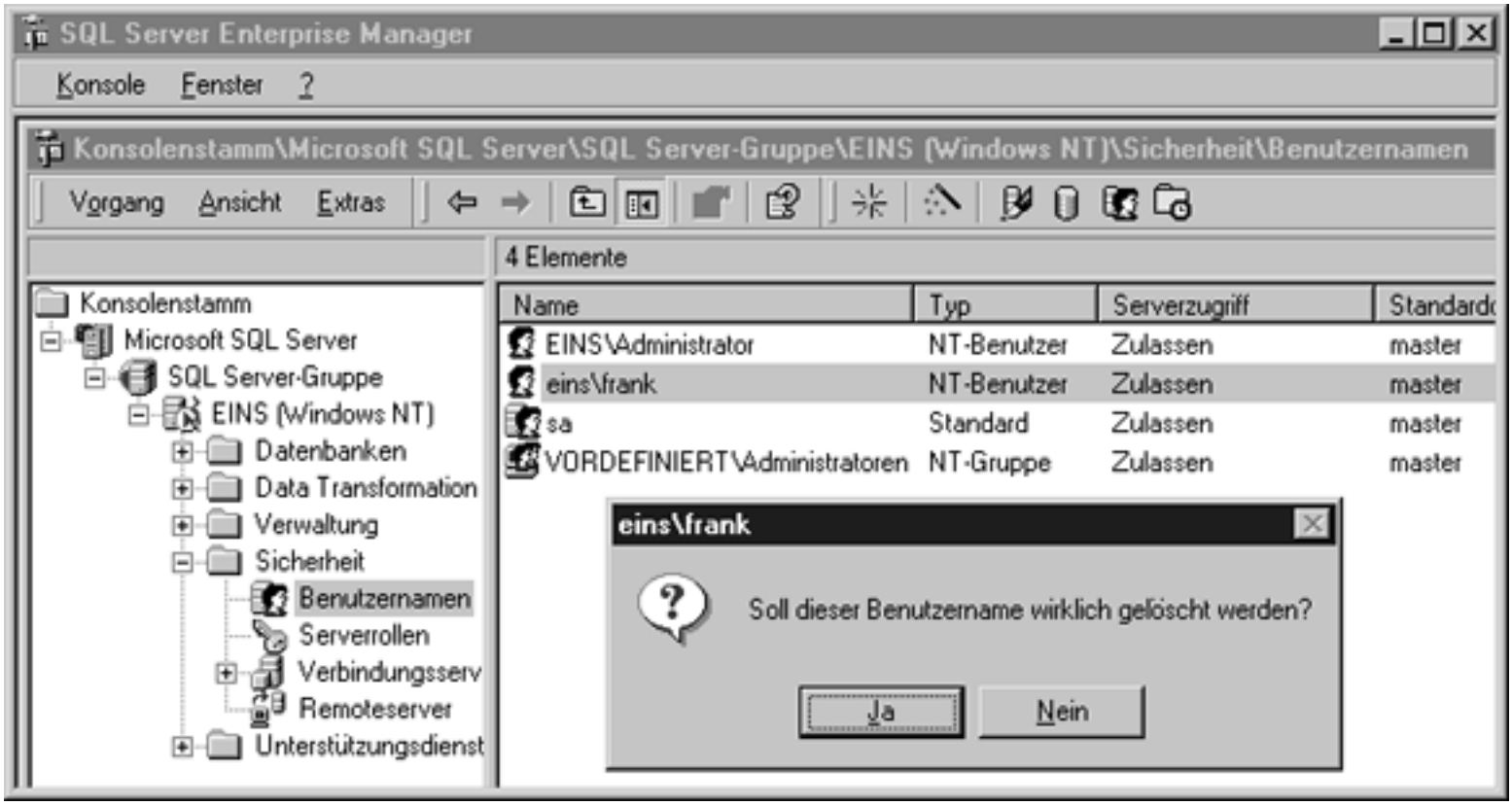

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
1 Element
test
Diagramme
Tabellen
Sichten
Gespeicherte Prozeduren
DB-Benutzernamen
Rollen
Regeln
Standards
Name
Benutzername
Datenbankzugriff
dbo
sa
Zulassen

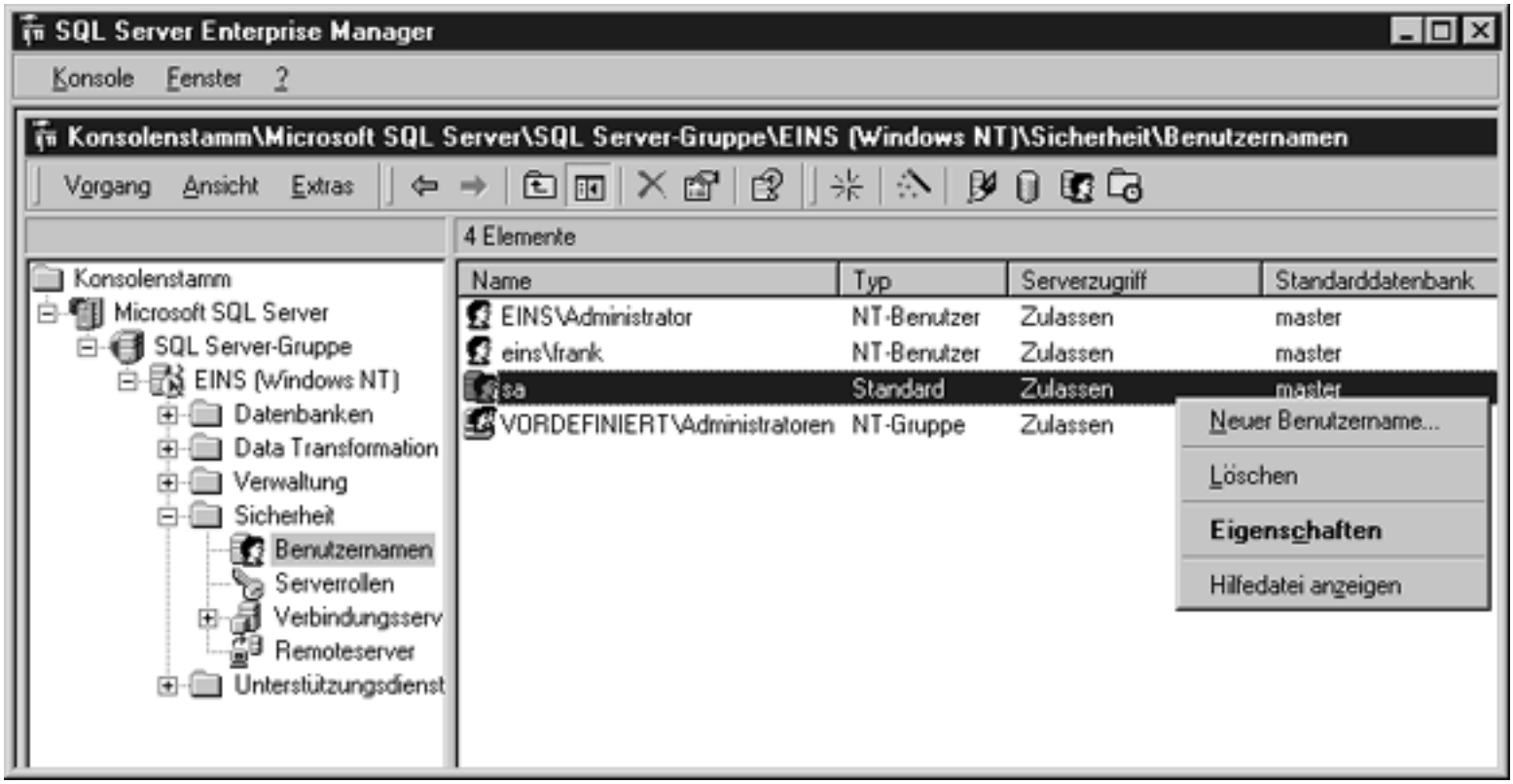

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Sicherheit\S...
Konsole Fenster ?
Vorgang Ansicht Extras
7 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation S
Verwaltung
Sicherheit
Benutzernamen
Serverrollen
Verbindungsserver
Remoteserver
Unterstützungsdienste
SQL 6.5
Vollständiger Name
Name
Beschreibung
Database Creators
dbcreator
Kann Datenbanken erstellen und verwalten.
Disk Administrators
diskadmin
Kann die Festplattendateien verwalten.
Process Administrators
processadmin
Kann die Prozesse verwalten, die in SQL Server ausgeführt werden.
Security Administrators
securityadmin
Benutzernamen für den Server können verwaltet werden.
Server Administrators
serveradmin
Kann die für alle Server geltenden Einstellungen konfigurieren.
Setup Administrators
setupadmin
Kann erweiterte gespeicherte Prozeduren verwalten.
System Administrators
sysadmin
Kann alle Aktivitäten in SQL Server-Installation durchführen.

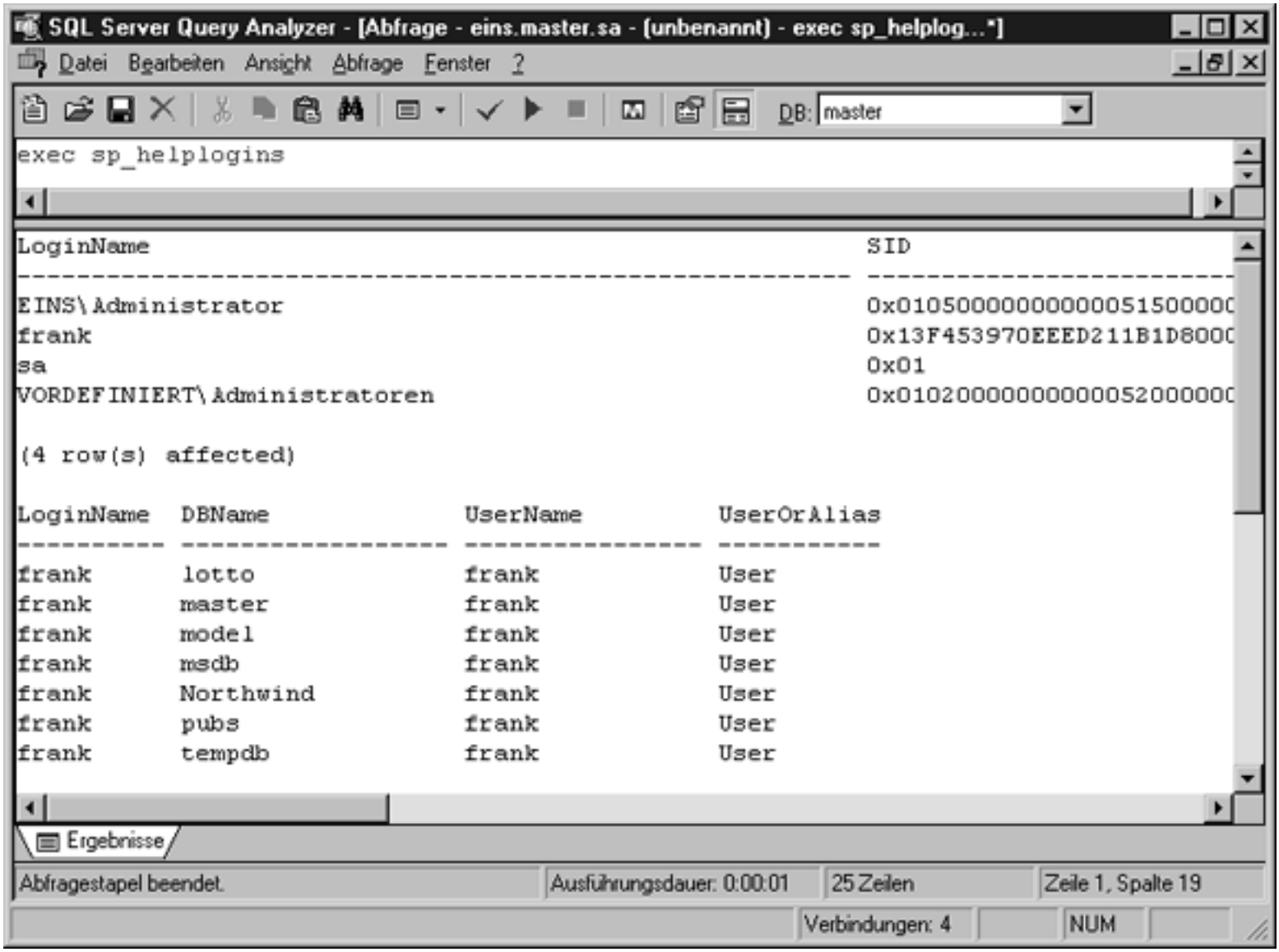

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Grup...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
10 Elemente
pubs
Diagramme
Tabellen
Sichten
Gespeiche
DB-Benutz
Rollen
Regeln
Standards
Benutzerde
Volltextkata
tempdb
Name
Rollentyp
db_accessadmin
Standard
db_backupoperator
Standard
db_datareader
Standard
db_datawriter
Standard
db_ddladmin
Standard
db_denydatareader
Standard
db_denydatawriter
Standard
db_owner
Standard
db_securityadmin
Standard
public
Standard

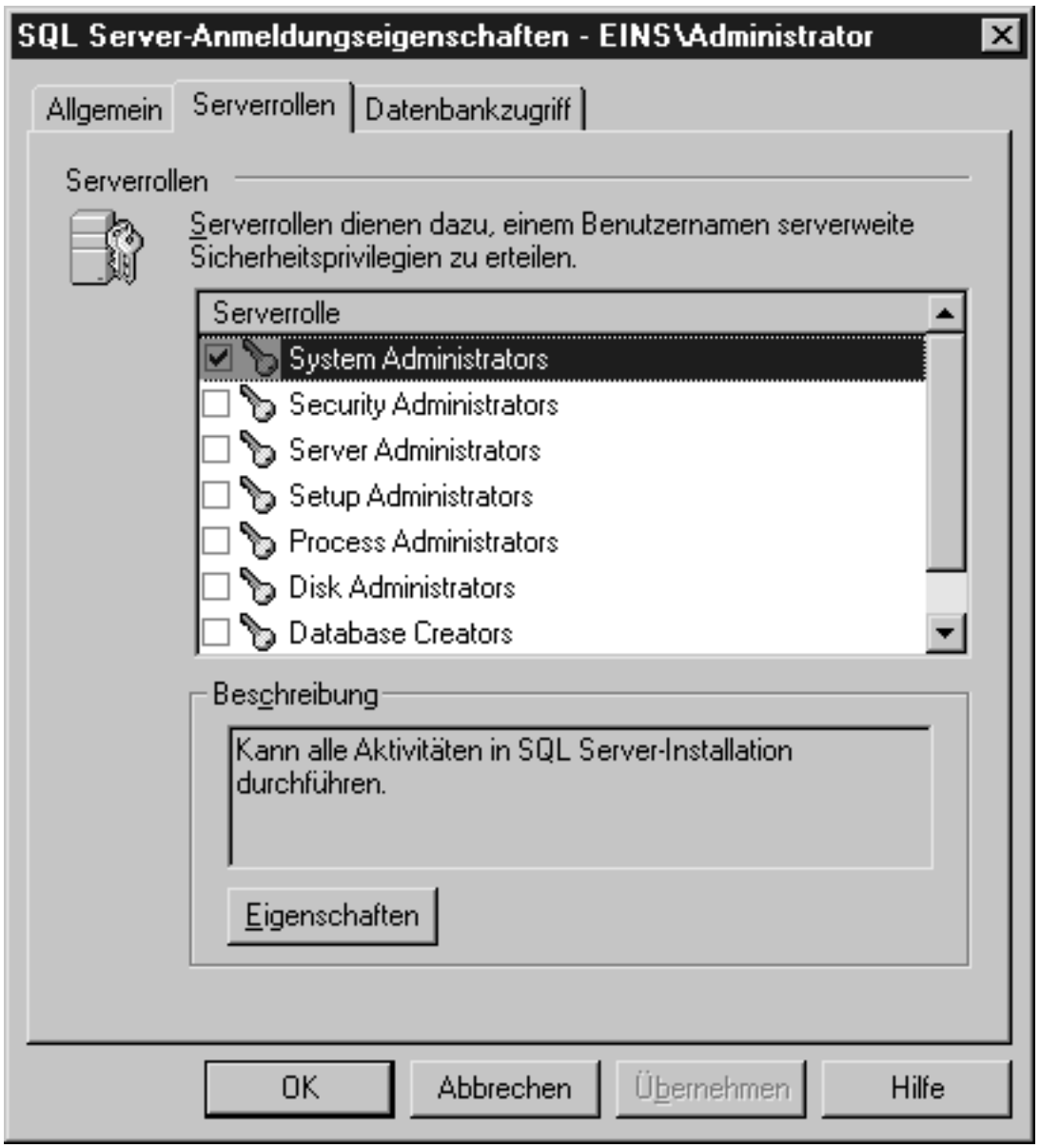

Anmeldungserstellungs-Assistent - EINS
Anmeldungserstellungs-Assistenten beenden
Sie haben die Schritte beendet, die zum Erstellen eines Benutzernamens erforderlich sind. Die Sicherheitsoptionen, die Sie erteilt haben, werden nachstehend angezeigt.
Authentifizierung: Windows NT
Erteilt an: eins\frank
Zugriff auf folgende Sicherheitsrollen gewährt:
System Administrators
Security Administrators
Server Administrators
Setup Administrators
Process Administrators
Disk Administrators
Database Creators
< Zurück
Fertig stellen
Abbrechen

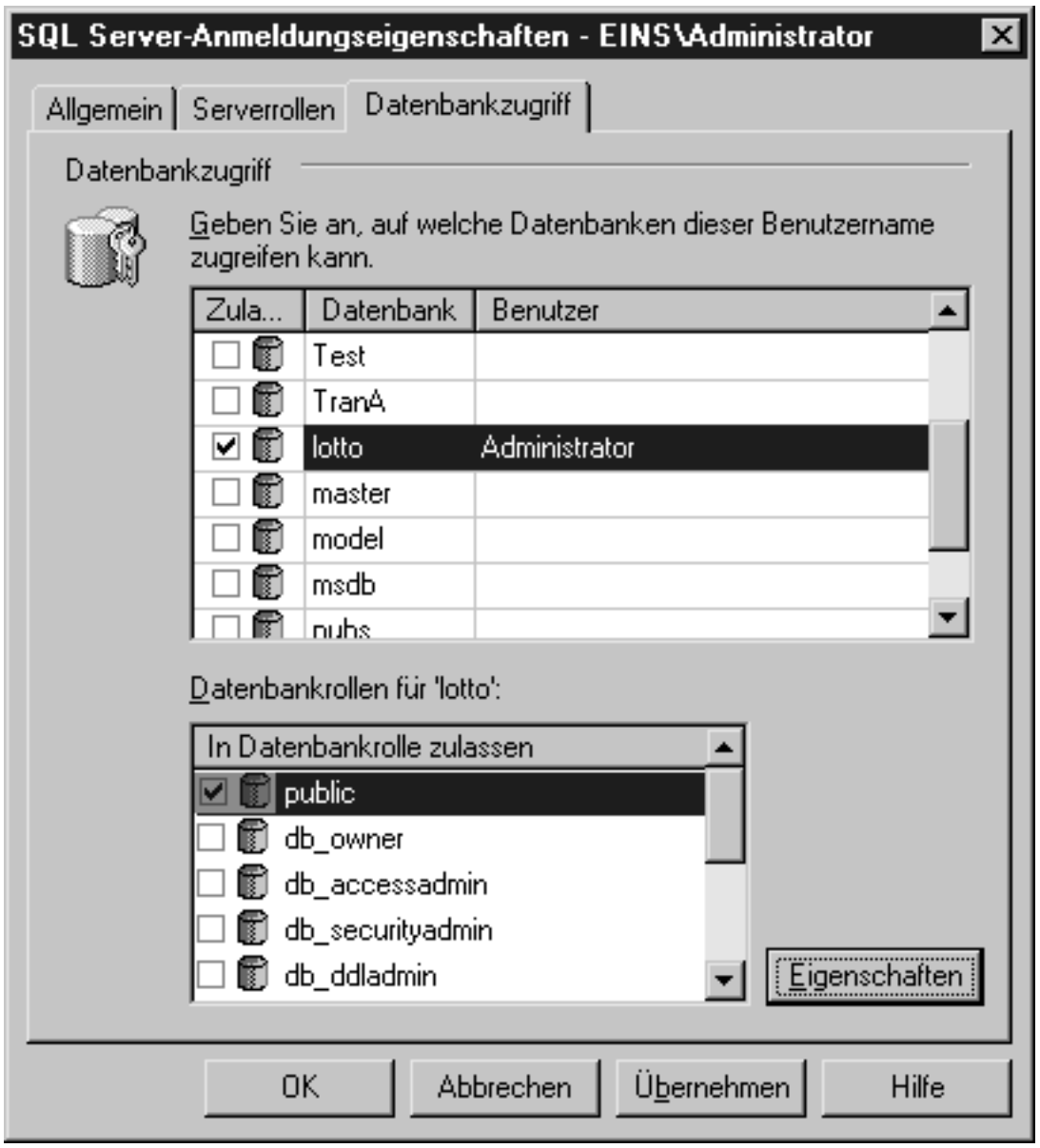

SQL Server-Eigenschaften - EINS
Verbindungen
Servereinstellungen
Datenbankeinstellungen
Allgemein
Speicher
Prozessor
Sicherheit
Sicherheit
SQL Server bietet die Authentifizierung basierend auf Windows NT-Konten und einem SQL Server-Benutzernamen und -Kennwort.
Authentifizierung:
SQL Server und Windows NT
Nur Windows NT
Überwachungsebene:
Keine
Fehler
Erfolg
Alle
Dienststartkonto
SQL Server mit folgendem Konto starten und ausführen:
Systemkonto
Dieses Konto
Administrator
Kennwort:
OK
Abbrechen
Übernehmen
Hilfe

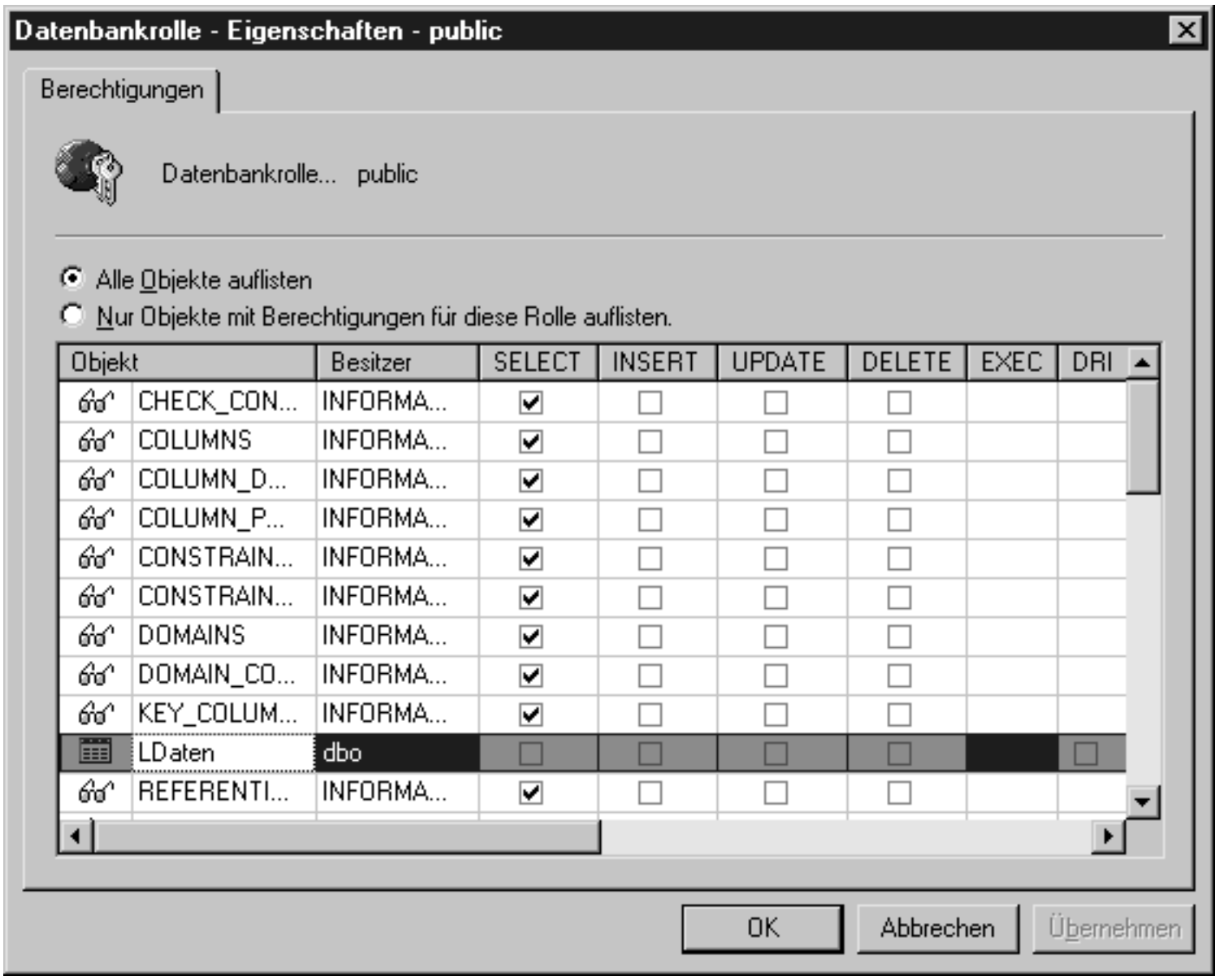

SQL Server Enterprise Manager
Konsole Fenster ?
Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Sicherheit\Benutzernamen
Vorgang Ansicht Extras
4 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation
Verwaltung
Sicherheit
Benutzernamen
Serverrollen
Verbindungsserv
Remoteserver
Unterstützungsdienst
Name
Typ
Serverzugriff
Standardd
EINS\Administrator NT-Benutzer Zulassen master
eins\frank NT-Benutzer Zulassen master
sa Standard Zulassen master
VORDEFINIERT\Administratoren NT-Gruppe Zulassen master
eins\frank
Soll dieser Benutzername wirklich gelöscht werden?
Ja
Nein

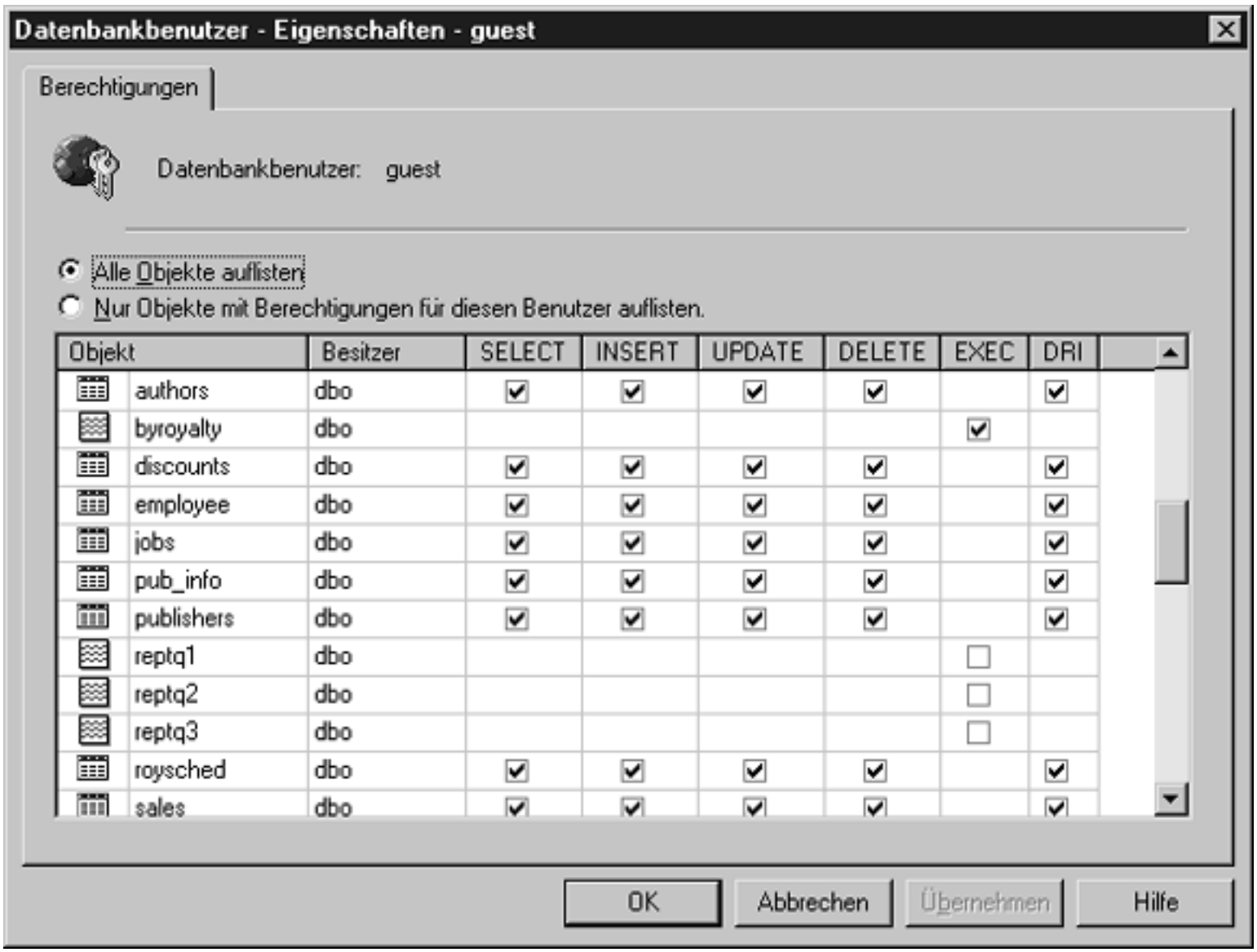

SQL Server Enterprise Manager
Konsole Fenster ?
Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Sicherheit\Benutzernamen
Vorgang Ansicht Extras
4 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation
Verwaltung
Sicherheit
Benutzernamen
Serverrollen
Verbindungsserv
Remoteserver
Unterstützungsdienst
Name
Typ
Serverzugriff
Standarddatenbank
EINS\Administrator
NT-Benutzer
Zulassen
master
eins\frank
NT-Benutzer
Zulassen
master
sa
Standard
Zulassen
master
VORDEFINIERT\Administratoren
NT-Gruppe
Zulassen
Neuer Benutzername...
Löschen
Eigenschaften
Hilfedatei anzeigen


SQL Server Query Analyzer - [Abfrage - eins.master.sa - (unbenannt) - exec sp_helplog...*]
Datei Bearbeiten Ansicht Abfrage Fenster ?
DB: master
exec sp_helplogins
LoginName SID
EINS\Administrator 0x0105000000000005150000
frank 0x13F453970EEED211B1D800
sa 0x01
VORDEFINIERT\Administratoren 0x0102000000000005200000
(4 row(s) affected)
LoginName DBName UserName UserOrAlias
frank lotto frank User
frank master frank User
frank model frank User
frank msdb frank User
frank Northwind frank User
frank pubs frank User
frank tempdb frank User
Ergebnisse
Abfragestapel beendet.
Ausführungsdauer: 0:00:01
25 Zeilen
Zeile 1, Spalte 19
Verbindungen: 4
NUM


SQL Server-Anmeldungseigenschaften - EINS\Administrator
Allgemein
Serverrollen
Datenbankzugriff
Serverrollen
Serverrollen dienen dazu, einem Benutzernamen serverweite Sicherheitsprivilegien zu erteilen.
Serverrolle
System Administrators
Security Administrators
Server Administrators
Setup Administrators
Process Administrators
Disk Administrators
Database Creators
Beschreibung
Kann alle Aktivitäten in SQL Server-Installation durchführen.
Eigenschaften
OK
Abbrechen
Übernehmen
Hilfe


SQL Server-Anmeldungseigenschaften - EINS\Administrator
Allgemein
Serverrollen
Datenbankzugriff
Datenbankzugriff
Geben Sie an, auf welche Datenbanken dieser Benutzername zugreifen kann.
Zula...
Datenbank
Benutzer
Test
TranA
lotto
Administrator
master
model
msdb
pubs
Datenbankrollen für 'lotto':
In Datenbankrolle zulassen
public
db_owner
db_accessadmin
db_securityadmin
db_ddladmin
Eigenschaften
OK
Abbrechen
Übernehmen
Hilfe


Datenbankrolle - Eigenschaften - public
Berechtigungen
Datenbankrolle... public
Alle Objekte auflisten
Nur Objekte mit Berechtigungen für diese Rolle auflisten.
Objekt
Besitzer
SELECT
INSERT
UPDATE
DELETE
EXEC
DRI
CHECK_CON... INFORMA...
COLUMNS INFORMA...
COLUMN_D... INFORMA...
COLUMN_P... INFORMA...
CONSTRAIN... INFORMA...
CONSTRAIN... INFORMA...
DOMAINS INFORMA...
DOMAIN_CO... INFORMA...
KEY_COLUM... INFORMA...
LDaten dbo
REFERENTI... INFORMA...
OK
Abbrechen
Übernehmen


Datenbankbenutzer - Eigenschaften - guest
Berechtigungen
Datenbankbenutzer: guest
Alle Objekte auflisten
Nur Objekte mit Berechtigungen für diesen Benutzer auflisten.
Objekt
Besitzer
SELECT
INSERT
UPDATE
DELETE
EXEC
DRI
authors dbo
byroyalty dbo
discounts dbo
employee dbo
jobs dbo
pub_info dbo
publishers dbo
reptq1 dbo
reptq2 dbo
reptq3 dbo
roysched dbo
sales dbo
OK
Abbrechen
Übernehmen
Hilfe

kapitel 22 replikation

22.1 überall griffbereit

auf die vorteile einer zentralen verwaltung der daten und die serverseitige realisierung der datenintegrität sowie die durchsetzung von geschäftsregeln wurde bereits mehrfach hingewiesen. in verschiedenen situationen ist es allerdings besser oder sogar erforderlich, die gesamte datenbank oder ausgewählte teile davon autonom auf dezentralen computern verfügbar zu haben.

stellen sie sich zum beispiel die datenbank für die verwaltung eines kaufhauses vor. in dieser datenbank sind unter anderem die preise der artikel und die lagerbestände gespeichert.

für die kassensysteme spielen die informationen zur verfügbarkeit eines artikels keine rolle. hier geht es in erster linie um preise, die sich zwar hin und wieder ändern, generell aber an allen kassen gleich sind und ständig aus der datenbank abgefragt werden. die kassensysteme belasten also die datenbank durch eine unzahl von lesevorgängen. viele kassen - viele gleichzeitige datenbankverbindungen und eine starke netzwerkbelastung.

auf der anderen seite in diesem vereinfachten szenario steht die lagerverwaltung. die angelieferten waren müssen in die datenbank aufgenommen werden, inventuren sind durchzuführen, preise sind zu aktualisieren, und bestellungen sind auszulösen. praktisch ist dieser teil der datenbank von starkem transaktionsverkehr geprägt (stichwort oltp - online-transaktionsverarbeitung), während der kassenbereich eher der analytischen online-verarbeitung - olap - zuzurechnen ist (auch wenn hier keine analyse im eigentlichen sinne stattfindet).

wenn man nun die preisinformationen aus der zentralen datenbank auf die kassensysteme dupliziert, können die zahlreichen kassen autonom arbeiten und verringern damit die netzwerkbelastung sowie die arbeitslast der zentralen datenbank. sobald sich preisänderungen ergeben, werden diese aus der zentralen datenbank auf alle kassensysteme übertragen. das kann in verkehrsschwachen zeiten erfolgen, beispielsweise am wochenende oder in der nacht. ebenso ist es möglich, die in den kassensystemen erfaßten verkaufszahlen an die zentrale datenbank zu übermitteln, um die bestandsführung zu aktualisieren.

noch ein beispiel: ein handelsvertreter braucht auf seinem laptop ein abbild der produktionspalette, die in der unternehmensdatenbank gespeichert ist. kehrt der vertreter vom außendienst zurück, muß er die neu aufgenommenen bestellungen in die unternehmensdatenbank übertragen und gegebenenfalls produktaktualisierungen auf den laptop überspielen.

die daten befinden sich quasi in einem kreislauf, in dem sie regelmäßig aktualisiert und synchron gehalten werden. der lateiner kennt dafür das wort *replicatio* - zu deutsch kreislauf oder wiederaufrollen. damit ist die bedeutung des begriffs *replikation* in bezug auf sql server zumindest im ansatz umrissen. mit den mitteln der replikation lassen sich datenbestände und gespeicherte prozeduren zu anderen standorten kopieren und automatisch synchronisieren.

22.2 replikation im überblick

grundlage der replikation ist das mit der version 6.0 von sql server eingeführte prinzip »publizieren und abonnieren«. eine echte replikation, die die daten sofort bei allen änderungen synchronisiert und bestätigungen im sinne eines zweiphasencommits wie bei verteilten transaktionen (siehe kapitel 18) einholt, setzt allerdings eine ständige verbindung zum server voraus. da dies nicht immer gegeben ist, kann es zwischenzeitlich zu abweichenden datenbeständen auf den einzelnen computern kommen. sobald aber die verbindung zu sql server wiederhergestellt ist, kommt die synchronisierung in gang.

22.2.1 das prinzip »publizieren und abonnieren«

in einer sql-server-replikation gibt es grundsätzlich zwei beteiligte: *publikationsserver* und *abonnementserver*. der publikationsserver - oder *verleger* - liefert die daten, der abonnementserver - der *abonnent* - bezieht sie. diese terminologie läßt parallelen zu einem zeitschriftenverlag erkennen: der verleger bringt eine zeitschrift heraus, der leser abonniert sie. nun wird aber der verleger sicherlich nicht jedem einzelnen leser die zeitschrift persönlich ins haus liefern. in der regel ist noch ein dritter im bunde: der *verteiler*. auf sql server übertragen heißt das, daß der verleger die publikationen an einen verteiler liefert, der sie seinerseits an die abonnenten weitergibt.

im replikationsprozeß kann sql server die rollen des verlegers, verteilers oder abonnenten übernehmen:

- der verleger verwaltet die zugrundeliegenden datenbanken, macht die geänderten daten für die replikation verfügbar und sendet sie zur weiterleitung an den verteiler.
- auf dem verteiler ist eine sogenannte verteilungsdatenbank installiert, die die daten vom verleger aufnimmt und von hier aus an die abonnenten weiterleitet. außerdem speichert der verteiler metadaten, chronologische daten und - bei transaktionsreplikation - transaktionen.
- der abonnent empfängt und speichert die veröffentlichten daten. neu in sql server 7.0 ist die möglichkeit, daß der abonnent die daten aktualisieren kann. dadurch wird er allerdings nicht zum verleger, denn die für die replikation verwendeten basistabellen sind nur auf dem eigentlichen verleger vorhanden. allerdings kann der abonnent zum verleger für weitere abonnenten avancieren.

die hierarchische struktur einer replikation erlaubt nur einen verleger, während es mehrere abonnenten geben kann. allerdings kann sql server auch mehrere rollen übernehmen, beispielsweise verleger und verteiler.

damit sind die teilnehmer bekannt. was ist nun unter einer publikation zu verstehen?

22.2.2 publikationen

genau wie eine zeitschrift besteht eine *publikation* in sql server aus verschiedenen artikeln. der *artikel* stellt die basiseinheit einer replikation dar und ist immer mit einer publikation verbunden. eine publikation kann folgende elemente enthalten:

- tabellen
- ausgewählte spalten einer tabelle
- ausgewählte zeilen einer tabelle

- ausgewählte zeilen und spalten einer tabelle
- gespeicherte prozeduren

die mit einem filter ausgewählten spalten einer tabelle bezeichnet man als *vertikal partitionierte tabelle*. dementsprechend spricht man bei ausgewählten zeilen von einer *horizontal partitionierten tabelle*.

folgenden elemente lassen sich nicht publizieren:

- die systemdatenbanken model, tempdb und msdb
- die systemtabellen der datenbank master

der abonnent konnte in der version 6.x von sql server auch einzelne artikel abonnieren. sql server 7.0 stellt diese möglichkeit via transact-sql nur aus gründen der abwärtskompatibilität bereit. deshalb sollten sie darauf verzichten und nur publikationen abonnieren.

22.2.3 abonnements

sinn und zweck einer replikation ist es, geänderte daten zum abonnenten weiterzugeben. dabei ist zu klären, wann die übertragung stattfindet und wer sie auslöst.

bei einem *pushabonnement* schickt der verleger die publikation unaufgefordert an den abonnenten. ein beispiel dafür sind börsenkurse, die nahezu in echtzeit an die broker weitergegeben werden müssen. die übertragung kann allerdings auch nach einem zeitplan erfolgen. auf jeden fall ist für diese form der replikation eine stabile verbindung zwischen verleger bzw. verteiler und den abonnenten erforderlich.

geht die initiative zur übertragung der geänderten daten vom abonnenten aus, spricht man von einem *pullabonnement*. diese methode eignet sich insbesondere für eine große zahl von abonnenten, die beispielsweise über das internet mit dem verleger in verbindung treten. auch für mobile benutzer, die ihren laptop nur gelegentlich an den server anschließen können, ist das pullabonnement die passende form der replikation.

22.2.4 replikationsarten

sql server unterstützt verschiedene arten der replikation, die sich vor allem hinsichtlich der funktionalität und der erreichbaren datenkonsistenz unterscheiden:

- snapshot (optional: snapshot mit sofort aktualisierbaren abonnenten)
- transaktion (optional: transaktion mit sofort aktualisierbaren abonnenten)
- merge

für die auswahl der geeigneten replikationsart sind die datenkonsistenz, die angestrebte autonomie der abonnenten und die netzwerkbelastung zu berücksichtigen. eine anwendung kann auch mehrere replikationsarten einsetzen.

snapshotreplikation

bei dieser einfachsten form der replikation, die bereits in sql server 6.x vorhanden war, erstellt der verleger eine momentaufnahme (snapshot - schnappschuß) des schemas und der daten zu einem

bestimmten zeitpunkt und repliziert diese informationen zu den abonnenten. auf den abonnentenservern werden die alten daten durch das replikat ersetzt. diese art der replikation verwendet man beispielsweise bei nachschlagetabellen, die sich kaum ändern.

snapshotreplikation mit sofort aktualisierbaren abonnenten

diese option der snapshotreplikation realisiert zusätzlich einen zweiphasencommit. neben den regelmäßigen aktualisierungen, die der verleger an den abonnenten sendet, kann der abonnent die replizierten daten auch lokal ändern. der dabei erforderliche zweiphasencommit stellt sicher, daß die änderungen sowohl beim verleger als auch beim abonnenten durchgeführt werden. andernfalls werden die aktualisierungen verworfen.

transaktionsreplikation

die ebenfalls in sql server 6.x implementierte transaktionsreplikation stützt sich auf das transaktionsprotokoll der verlegerdatenbank. der verleger kennzeichnet hier bestimmte transaktionen zur replikation. nur diese änderungen fließen in die publikation ein und gelangen zum abonnenten. dadurch verringert sich die netzwerkbelastung.

transaktionsreplikation mit sofort aktualisierbaren abonnenten

als option läßt sich die transaktionsreplikation mit einem zweiphasencommit kombinieren. im gegensatz zur snapshotreplikation mit zweiphasencommit läuft die aktualisierung bei der transaktionsreplikation wesentlich schneller ab, da nur einzelne transaktionen und nicht der gesamte snapshot der daten zu replizieren sind.

mergereplikation

sql server 7.0 führt die mergereplikation ein. dabei können verschiedene server unabhängig voneinander die replizierten daten ändern und später beim verleger zusammenführen (merge - mischen). das klingt zwar sehr verlockend, kann aber in der praxis zu konflikten führen. wenn zum beispiel mehrere abonnenten dieselbe zeile in derselben tabelle ändern, muß der verleger diese konflikte erkennen und auflösen.

22.2.5 replikationstopologien

wie bereits erwähnt, können die an einer replikation beteiligten server verschiedene rollen - verleger, verteiler, abonnent - übernehmen. während die im letzten abschnitt behandelten replikationsarten den logischen austausch der replizierten daten beschreiben, geben die replikationstopologien die physische realisierung der replikation an. grundsätzlich unterstützt sql server nur die sterntopologie, bei der die abonnenten an einen verleger oder verteiler angeschlossen sind und es keine direkten kommunikationswege zwischen den abonnenten gibt.

zentraler verleger

bei diesem modell ist der publizierende server gleichzeitig der verteiler. das bedeutet, daß sowohl die basisdatenbank als auch die verteilungsdatenbank auf ein und demselben server untergebracht sind. in

einer oltp-umgebung können die abonnenten den publikationsserver von den abfrageintensiven olap-diensten entlasten (entsprechend dem eingangs geschilderten beispiel eines kaufhauses).

zentraler verleger mit remote-verteiler

bei einer großen zahl von abonnenten installiert man den verteiler auf einem eigenen server, der aus sicht des verlegers einen remote-verteiler darstellt. dieses standardmodell eines sql-server-systems eignet sich ebenfalls für eine umgebung, in der die oltp-dienste auf dem publikationsserver laufen und die olap-dienste auf die abonnenten ausgelagert werden.

publizierender abonnent

wenn sich die abonnenten an standorten befinden, die der verleger nur über langsame oder teure netzwerkverbindungen erreichen kann, richtet man auf der seite der abonnenten einen speziellen server ein, der mit den eigentlichen abonnenten über schnelle bzw. kostengünstige leitungen verbunden ist. der verleger muß gleichzeitig verteiler sein, da zwischen verleger und verteiler eine schnelle verbindung erforderlich ist. der speziell eingerichtete server ist zunächst ein abonnent, der die daten über die langsame/teure verbindung vom verleger bezieht. gleichzeitig übernimmt dieser abonnent die rolle des verlegers und verteilers für die eigentlichen abonnenten.

zentraler abonnent

den bisher beschriebenen modellen ist gemeinsam, daß die zu replizierenden daten an einem zentralen standort anfallen und an abnehmer an verschiedenen orten zu verteilen sind. der umgekehrte fall ist aber genauso möglich. stellen sie sich ein versandhaus vor, das mehrere großlager unterhält, wobei jedes lager die bestandsführung autonom abwickelt. in diesem szenario kann ein zentraler abonnent als datensammelsystem arbeiten, während die server in den einzelnen lagern als verleger und verteiler fungieren. damit die konsolidierung der daten auf dem zentralen abonnenten nicht zu konflikten führt, wenn gleiche artikel aus verschiedenen lagern in derselben tabelle zusammengefaßt werden, unterscheidet man die lagerstandorte durch einen eindeutigen code, sieht in der konsolidierungstabelle eine entsprechende spalte vor und nimmt diese in den primärschlüssel der tabelle auf. hier bietet sich fast von selbst eine transaktionsreplikation an.

22.3 replikation einrichten

unabhängig von der art der replikation und der verwendeten topologie sind folgende schritte auszuführen, um sql server für eine replikation einzurichten und zu aktivieren:

- verteiler installieren
- verleger installieren
- abonnenten installieren
- publikationen erstellen
- publikationen abonnieren

alle genannten schritte können sie im enterprise manager ausführen. sql server erleichtert ihnen diese aufgaben mit einer reihe von assistenten und dem taskpad **daten replizieren**.

markieren sie in der konsolenstruktur den gewünschten server, und aktivieren sie im menü **ansicht** den befehl **taskpad**. im detailbereich erscheint daraufhin das **taskpad für 'erste schritte'**. klicken sie auf **daten replizieren**. der enterprise manager listet jetzt im detailbereich die im zusammenhang mit der replikation stehenden aufgaben auf (siehe abbildung 22.1). wenn sie den mauszeiger auf ein symbol setzen, erscheint eine erläuterung der jeweiligen aufgabe in form einer quickinfo.

abbildung 22.1: taskpad im enterprise manager für die aufgaben der replikation

22.3.1 replikation konfigurieren

führen sie die folgenden schritte aus, um einen verteilungsserver und einen publikationsserver zu installieren:

1. klicken sie im taskpad **daten replizieren** auf das symbol **replikation konfigurieren**. damit starten sie den publizierungs- und verteilungskonfigurations-assistenten. abbildung 22.2 zeigt das startdialogfeld des assistenten. der konfigurationsprozeß beginnt also nicht mit dem verleger, sondern dem verteiler. die verteilungsdatenbank ist dreh- und angelpunkt der replikation in sql server. klicken sie auf **weiter**.

den assistenten können sie auch über das menü **extras** des enterprise managers aufrufen. wählen sie den befehl **replikation** und im daraufhin erscheinenden untermenü **publizierung und abonnenten konfigurieren**.

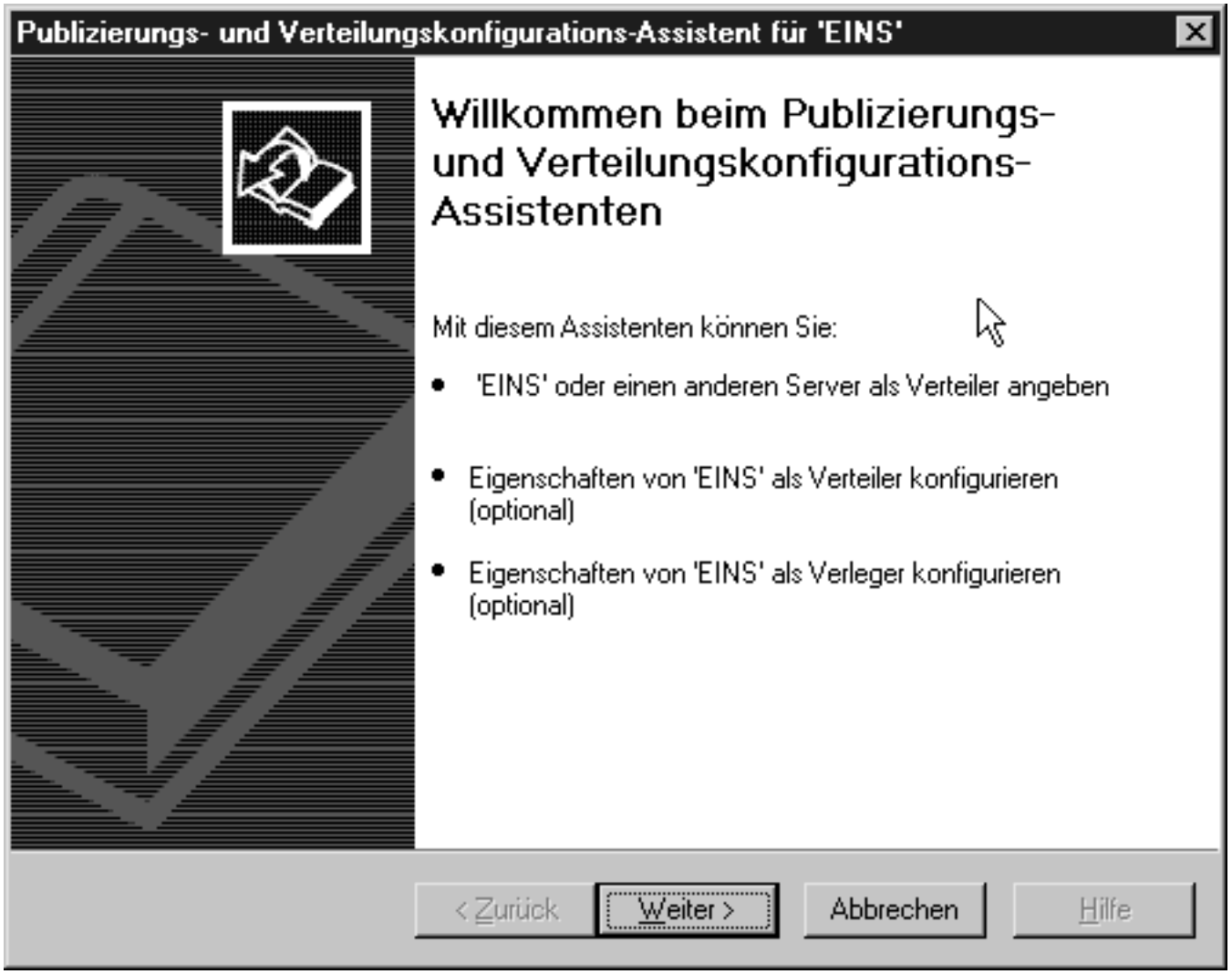


abbildung 22.2: startdialogfeld des publizierungs- und verteilungskonfigurations-assistenten

2. im dialogfeld **verteiler auswählen** (siehe abbildung 22.3) legen sie den verteiler für die replikation fest. voreingestellt ist die option für den lokalen computer, auf dem sie wahrscheinlich auch den verleger installieren wollen. einen anderen server können sie nur dann festlegen, wenn dieser bereits als verteiler konfiguriert ist. klicken sie auf **weiter**.

wenn sie verleger und verteiler auf demselben computer installieren, müssen sie den zusätzlichen platz einkalkulieren, den die verteilerdatenbank beansprucht. darüber hinaus beanspruchen die agenten zur verwaltung des verteilers zusätzliche prozessorzeit. bei oltp-systemen sollten sie also einen separaten verteiler vorsehen.

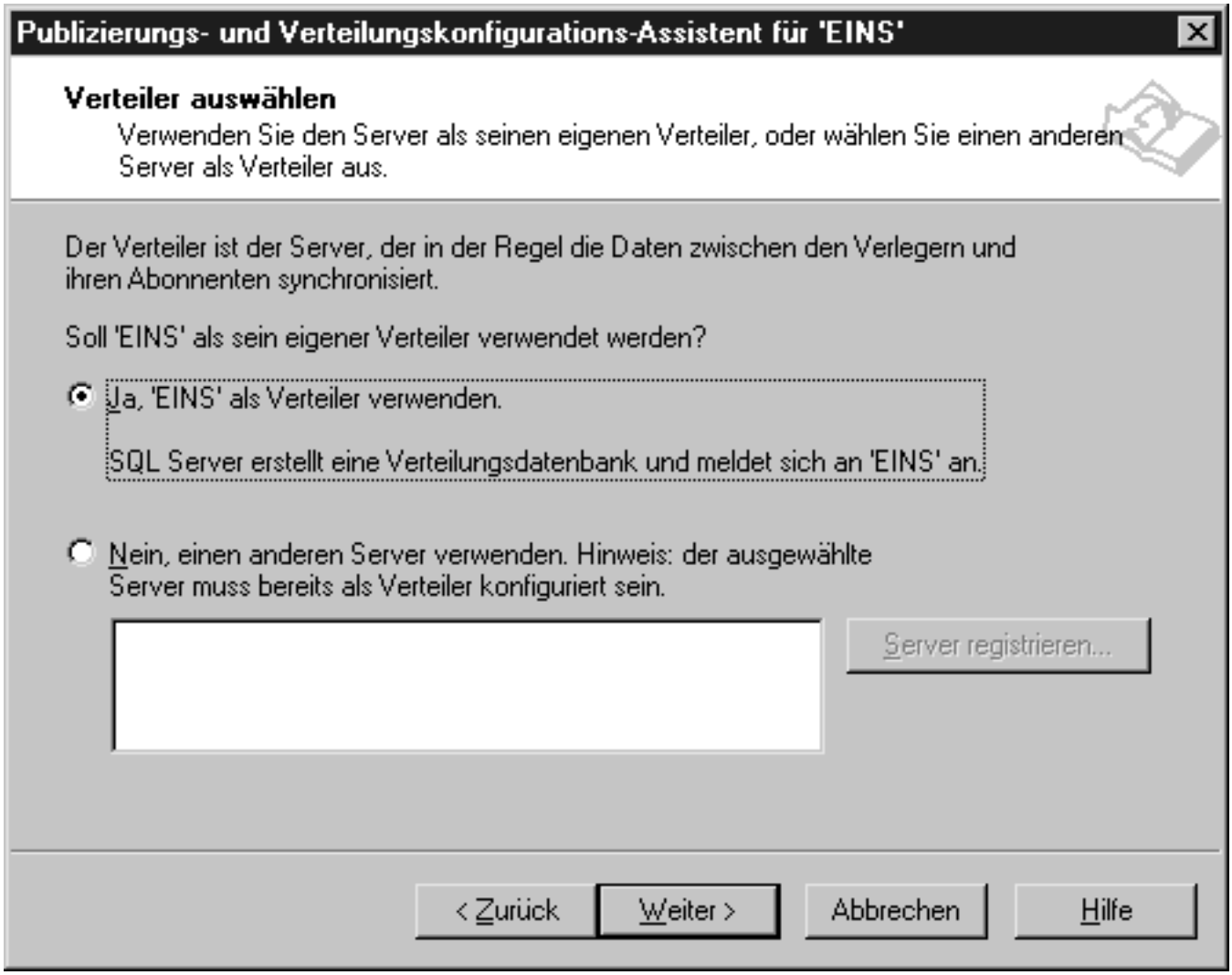


abbildung 22.3: das dialogfeld verteiler auswählen

3. im nächsten dialogfeld (siehe abbildung 22.4) legen sie die standardkonfiguration fest. voreingestellt ist die zweite option. wie das listenfeld zeigt, kann ein und derselbe server als verleger, dessen verteiler und gleichzeitig als sein abonnent agieren. außerdem wird die verteilungsdatenbank distribution im verzeichnis \mssql7\data\ dieses servers angelegt.

wählen sie für das beispiel die erste option *ja, ich möchte den namen ...* . klicken sie dann auf **weiter**.

bild

abbildung 22.4: das dialogfeld standardkonfiguration verwenden

4. im dialogfeld **verteilungsdatenbankinformationen bereit-stellen** (siehe abbildung 22.5) können sie einen anderen namen für die verteilungsdatenbank festlegen sowie die standorte der verteilungsdatenbank oder der dazugehörigen protokolldatei wählen. die verzeichnisse müssen lokal zum angegebenen server sein. das im beispiel verwendete laufwerk h: befindet sich auf einer zweiten festplatte des servers eins. klicken sie auf **weiter**.

bild

abbildung 22.5: das dialogfeld verteilungsdatenbankinformationen bereitstellen

5. im dialogfeld **verleger aktivieren** (siehe abbildung 22.6) können sie anderen servern die

berechtigung erteilen, diesen verteiler zu verwenden. wenn der gewünschte server nicht in der liste aufgeführt ist, klicken sie auf die schaltfläche **server registrieren** und geben im eigenschaftsdialogfeld die jeweiligen verbindungsoptionen an.

abbildung 22.6: das dialogfeld verleger aktivieren

markieren sie den gewünschten server in der liste, und klicken sie auf die schaltfläche mit den drei punkten, um die sicherheitsoptionen für diesen server einzustellen (siehe abbildung 22.7). schließen sie das dialogfeld, und klicken sie dann auf **weiter**.

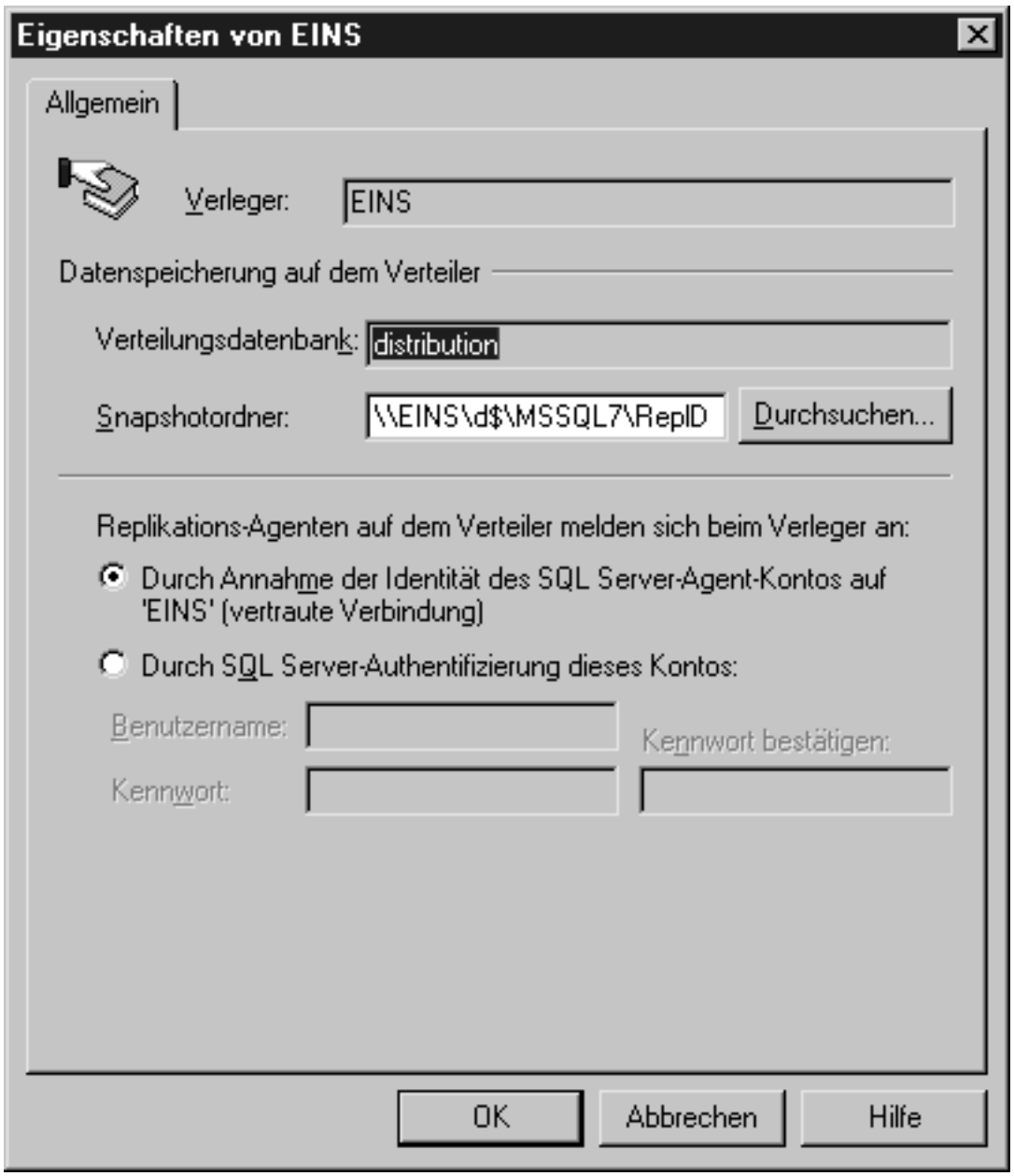


abbildung 22.7: legen sie hier die sicherheitsoptionen für den server fest

6. im dialogfeld **publikationsdatenbanken aktivieren** (siehe abbildung 22.8) wählen sie die zu replizierenden datenbanken aus und legen die gewünschte replikationsart - transaktionsreplikation und/oder mergereplikation - fest. die kontrollkästchen können sie separat einschalten, oder sie wählen über die schaltflächen im rechten teil des dialogfelds alle datenbanken für die betreffende replikationsart aus. klicken sie dann auf **weiter**.

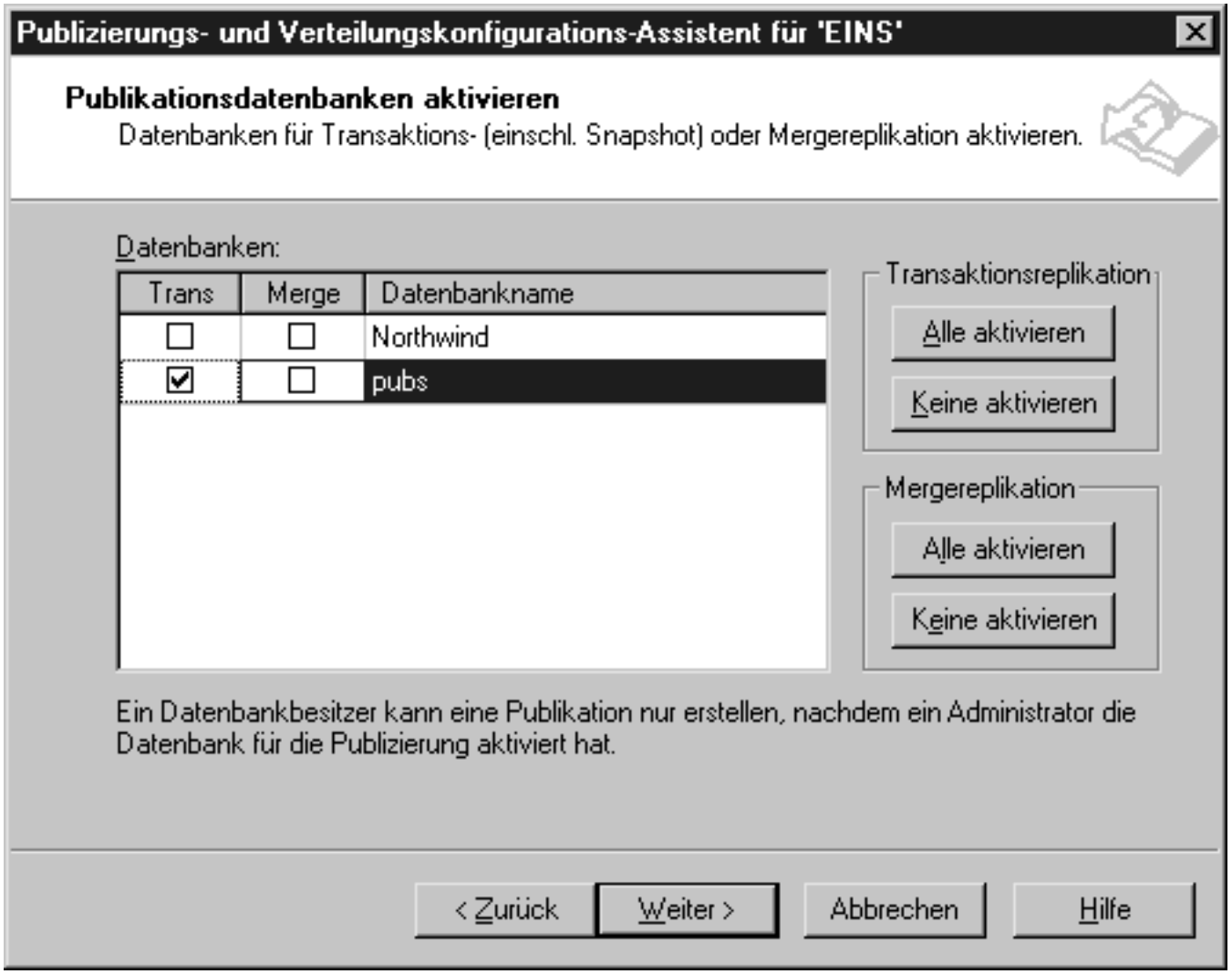


abbildung 22.8: in diesem dialogfeld wählen sie die zu publizierenden datenbanken und den typ der replikation aus

wenn sie eine datenbank für die publikation aktivieren wollen, müssen sie mitglied der festen server-rolle sysadmin sein. der datenbankbesitzer kann die datenbank nur dann publizieren, wenn sie zuvor für die publikation aktiviert wurde. der systemadministrator nimmt wie immer eine sonderstellung ein. er kann datenbanken publizieren, ohne daß sie zuvor aktiviert wurden.

7. im nächsten dialogfeld aktivieren sie abonnenten, die publikationen von diesem verleger beziehen können (siehe abbildung 22.9). schalten sie die kontrollkästchen neben den servern ein, die sie aktivieren möchten. auch hier können sie wieder zusätzliche server registrieren.

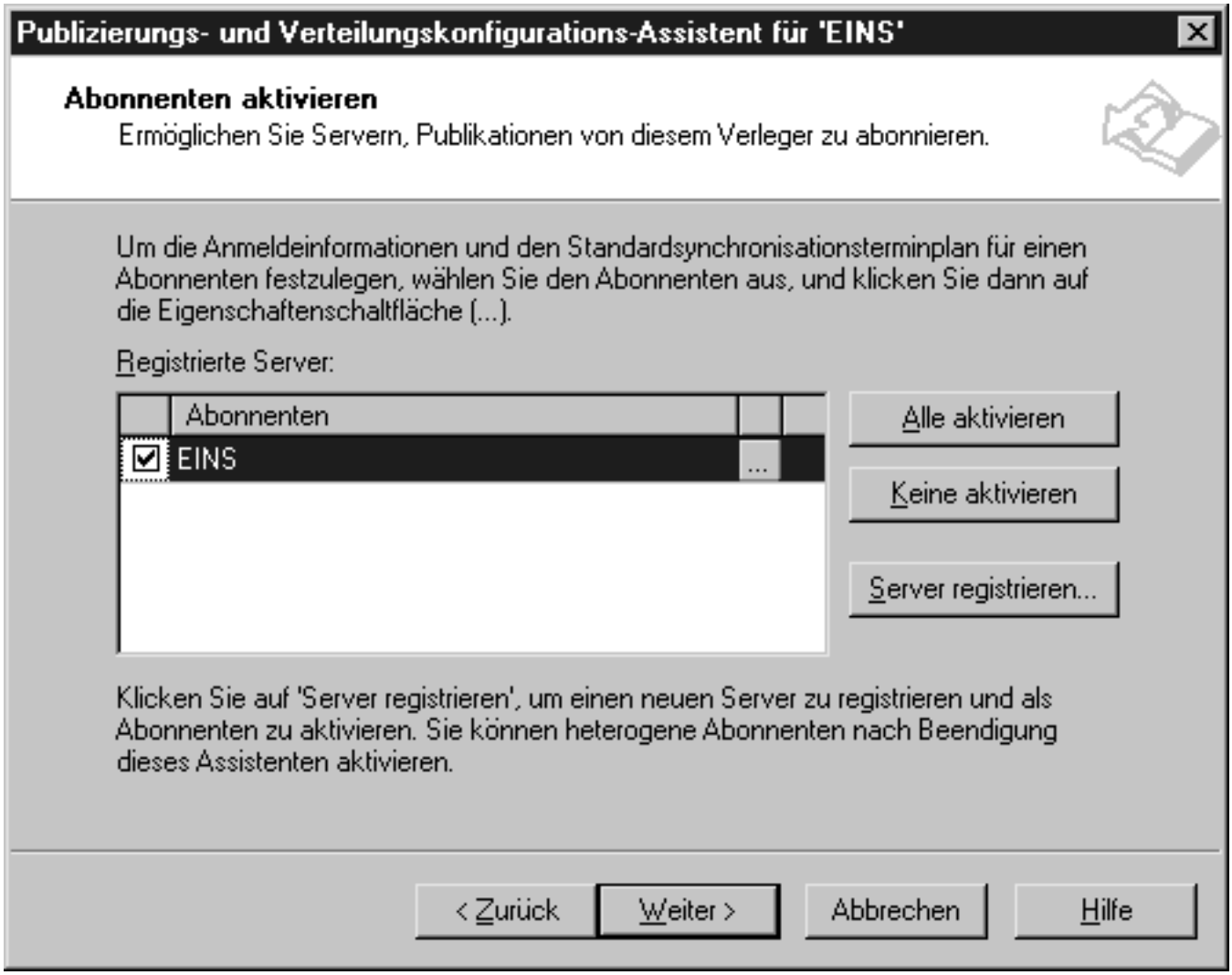


abbildung 22.9: das dialogfeld abonnenten aktivieren

über die schaltfläche **server registrieren** können sie nur microsoft sql server für die replikation hinzufügen. wenn sie mit anderen servern arbeiten, müssen sie diese außerhalb des assistenten registrieren.

8. über die schaltfläche mit den drei punkten gelangen sie zum eigenschaftsdialogfeld des jeweiligen servers. auf der registerkarte **allgemein** (siehe abbildung 22.10) legen sie die üblichen verbindungsinformationen - anmeldung per windows-nt- oder sql-server-authentifizierung - fest. die registerkarte **terminpläne** (siehe abbildung 22.11) ist für die planung von pushabonnements vorgesehen. hier geben sie die zeitpunkte an, wann die verteilungs- und merge-agenten aktiv sein sollen.

bild

abbildung 22.10: auf der registerkarte terminpläne legen sie fest, wann die verteilungs- und merge-agenten aktiv sein sollen

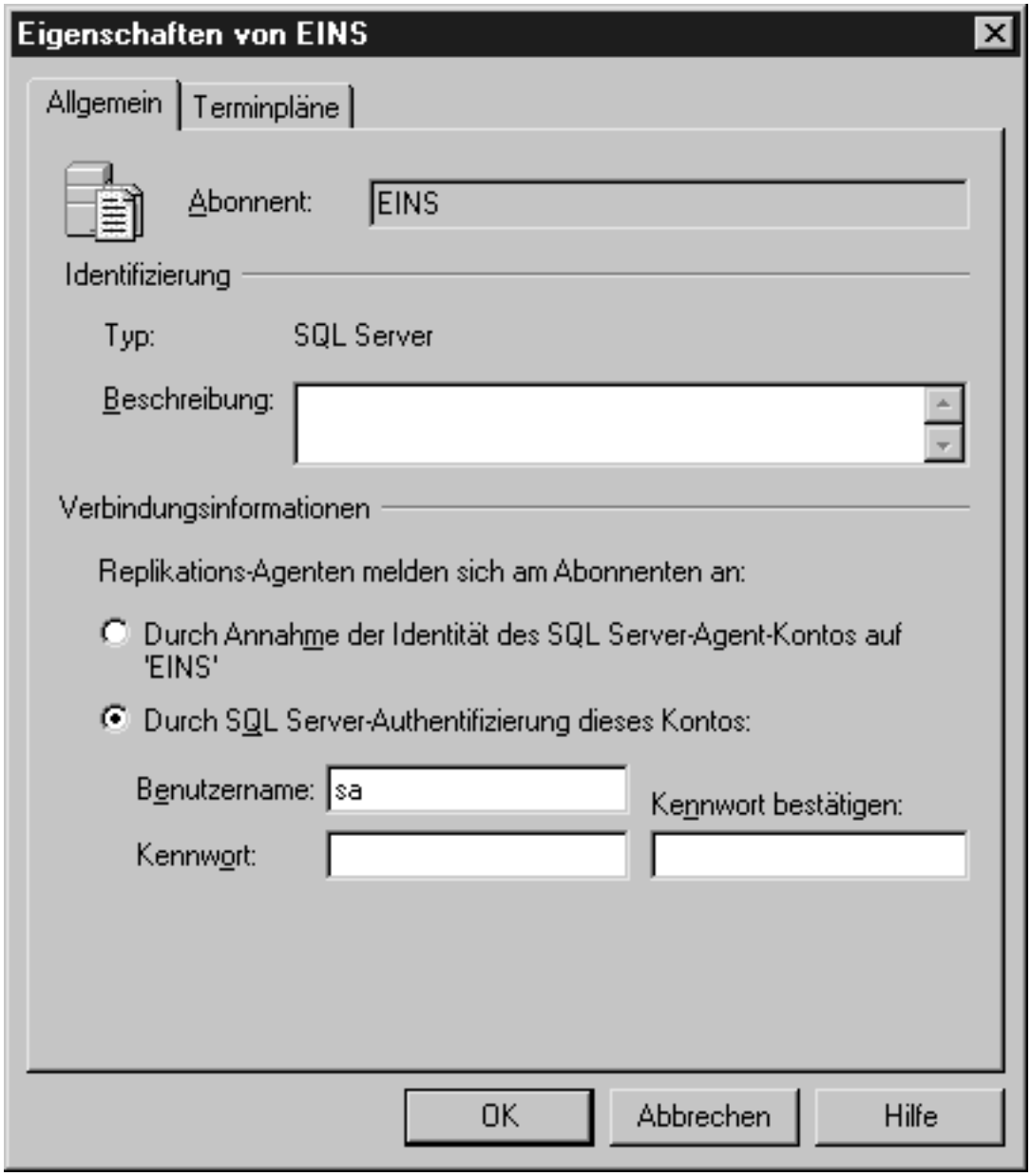


abbildung 22.11: auf der registerkarte allgemein legen sie die art der anmeldung eines abonnenten fest

denken sie daran, daß auch agenten zeit beanspruchen. wenn sie zum beispiel verleger und verteiler auf ein und demselben server installieren und die agenten fortlaufend ausgeführt werden, sollten sie diesen server nicht für eine datenbank mit hohem transaktionsaufkommen einsetzen.

9. das letzte dialogfeld des assistenten (siehe abbildung 22.12) faßt alle einstellungen zusammen. klicken sie auf **fertigstellen**.

bild

abbildung 22.12: letztes dialogfeld des publizierungs- und verteilungskonfigurations-assistenten

sql server zeigt nun den fortschritt beim implementieren der von ihnen gewählten konfigurationseinstellungen an (siehe abbildung 22.13).

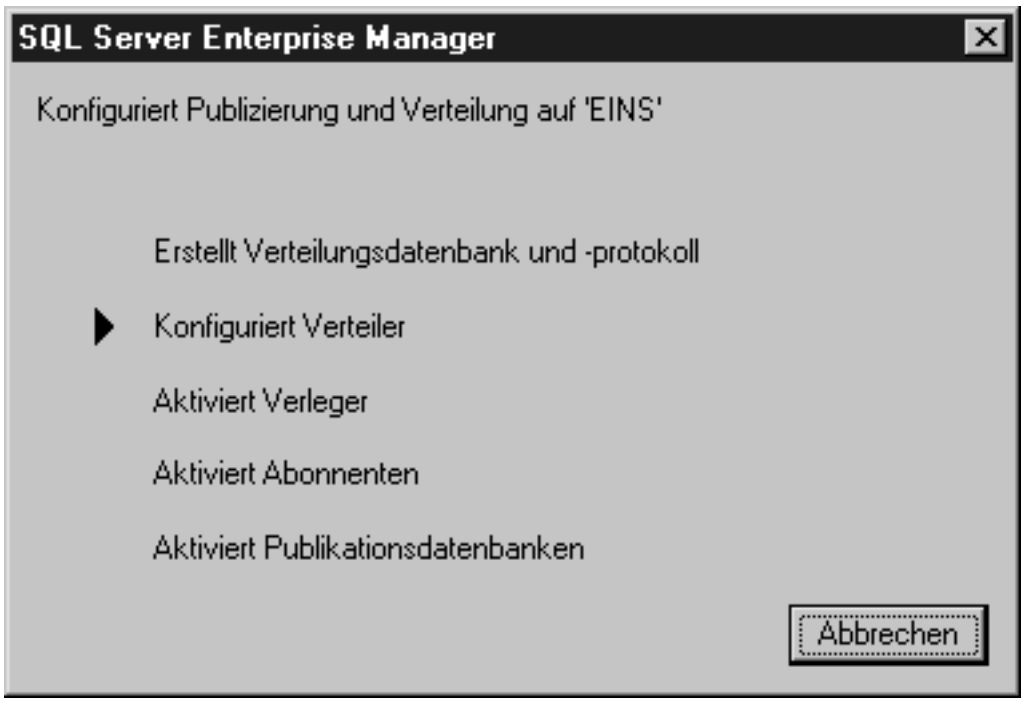


abbildung 22.13: fortschrittsmeldung von enterprise manager

schließlich weist die fertigmeldung darauf hin, daß der gewählte verteiler für den gewählten verleger konfiguriert und aktiviert wurde (siehe abbildung 22.14).

abbildung 22.14: fertigmeldung

falls sie den dienst *sql server-agent* nicht für den automatischen start eingerichtet haben, erscheint das in abbildung 22.15 gezeigte dialogfeld. den automatischen start können sie zwar auch später über das dialogfeld **dienste** der windows-nt-systemsteuerung oder durch einschalten des kontrollkästchens im sql-server-dienst-manager aktivieren, wenn sie sich aber gleich für **ja** entscheiden, haben sie schon ein potentielles problem weniger.

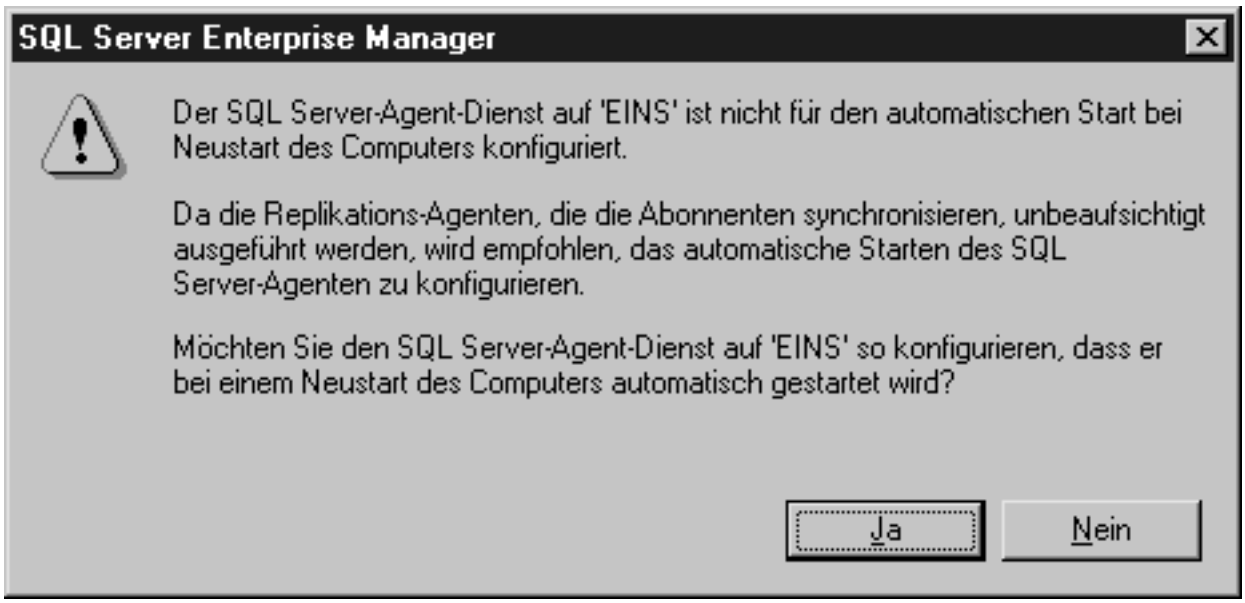


abbildung 22.15: frage, ob ein automatischer start des sql-server-agent-dienstes erfolgen soll

wenn sie die replikation erstmalig auf diesem server installieren, erscheint die in abbildung 22.16 gezeigte meldung. die konsolenstruktur des enterprise managers hat also zuwachs bekommen: mit dem replikationsmonitor läßt sich der zustand der replikation überwachen.

abbildung 22.16: meldung, daß der replikationsmonitor hinzugefügt wurde

damit ist der erste wesentliche teil bei der installation der an einer replikation beteiligten server abgeschlossen. allerdings handelt es sich bis jetzt nur um eine konfiguration, d.h., die festgelegten datenbanken und tabellen sind noch nicht reif für eine publikation - sie sind lediglich für die replikation

aktiviert worden.

die folgende übersicht faßt die bisher ausgeführten schritte zusammen. im einzelnen haben sie ...

- einen verteiler ausgewählt und die standardkonfiguration festgelegt,
- die verteilungsdatenbank mit protokoll erstellt und konfiguriert,
- verlegern den zugriff auf den konfigurierten verteilungsserver gewährt,
- publikationsdatenbanken aktiviert und die jeweilige replikationsart festgelegt,
- abonnenten aktiviert und
- terminpläne für die verteilungs- und merge-agenten vorbereitet.

diese schritte waren erforderlich, damit sie publikationen erstellen und abonnieren können. ein blick in das menü **extras** des enterprise managers zeigt, daß jetzt alle befehle im untermenü **replikation** verfügbar sind, während vorher nur die befehle aktiv waren, die sich auf die konfiguration der replikation beziehen.

22.3.2 publikationserstellungs-assistent

wie bereits erwähnt, sind die von ihnen im publikations- und verteilungskonfigurations-assistenten ausgewählten datenbanken zunächst nur für eine publikation aktiviert worden. dieser abschnitt zeigt nun, wie sie publikationen erstellen.

viele wege führen nach rom. übertragen gilt das auch für das erstellen von publikationen. zum beispiel haben sie im letzten abschnitt den publikations- und verteilungskonfigurations-assistenten über das taskpad **daten replizieren** aufgerufen. äquivalente befehle finden sie im menü **extras** des enterprise managers. die jeweiligen assistenten können sie natürlich auch über das dialogfeld **assistent auswählen** starten. dieses dialogfeld erreichen sie zum einen über die schaltfläche **assistenten auswählen** in der symbolleiste des enterprise managers, zum anderen über den befehl **assistenten** des menüs **extras**.

eine publikation erstellen sie mit dem publikationserstellungs-assistenten in folgenden schritten:

1. markieren sie in der konsolenstruktur den gewünschten server. wählen sie im menü **extras** den befehl **replikation** und aus dem aufklappenden untermenü den befehl **publikationen erstellen und verwalten**. daraufhin erscheint das in abbildung 22.17 dargestellte dialogfeld **publikationen und abonnements erstellen und verwalten**.

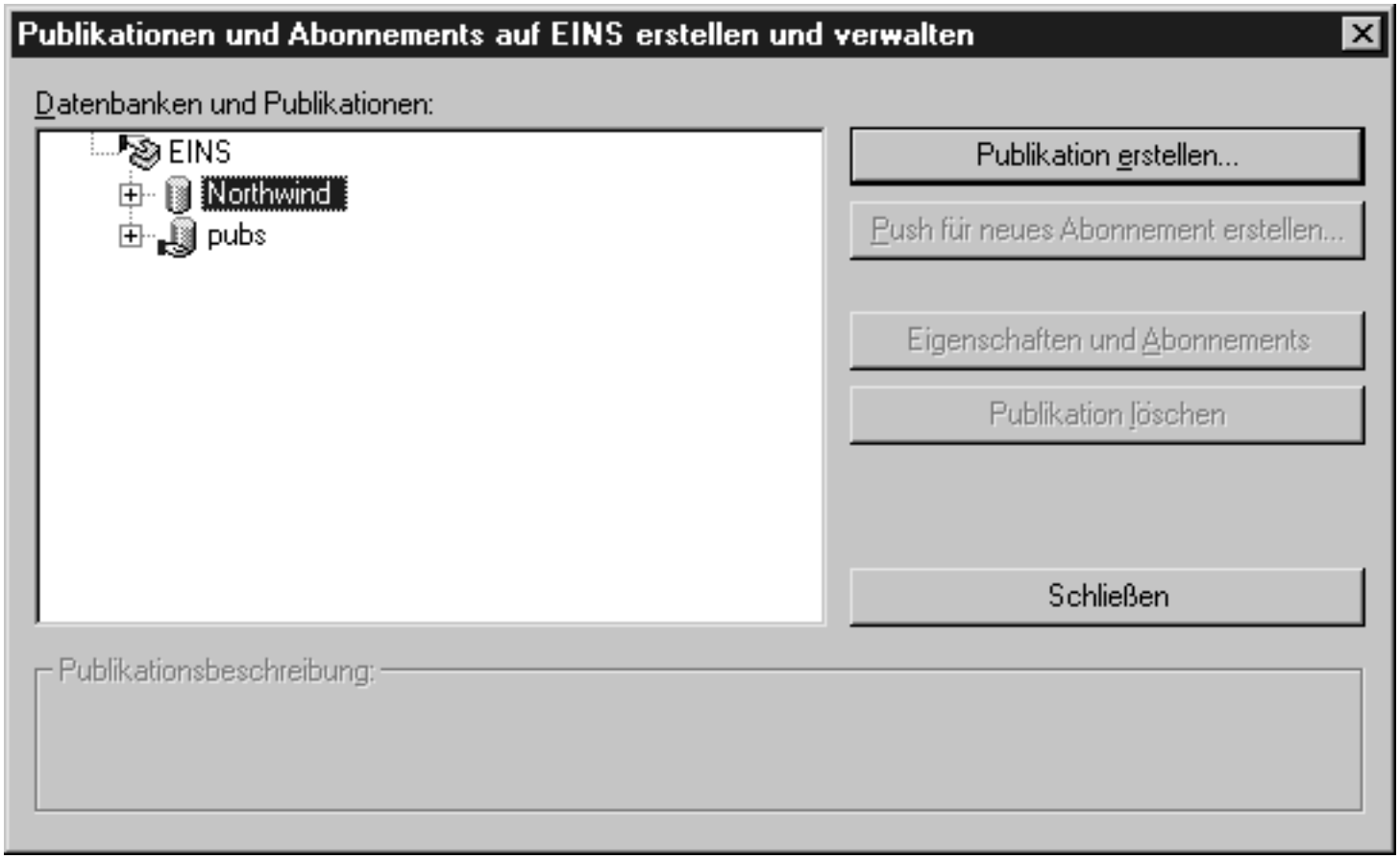


abbildung 22.17: das dialogfeld publikationen erstellen und verwalten

2. beachten sie das freigabesymbol (die geöffnete hand) am ordner für die datenbank pubs. es zeigt an, daß diese datenbank für die replikation aktiviert ist und als publikationsdatenbank verwendet werden kann. markieren sie die datenbank pubs. klikken sie dann auf die schaltfläche **publikation erstellen**. daraufhin startet der publikationserstellungs-assistent. klicken sie im startdialogfeld auf **weiter**.
3. im dialogfeld **publikationstyp auswählen** (siehe abbildung 22.18) stehen die drei optionen *snapshotreplikation*, *transaktionsreplikation* und *mergereplikation* zur wahl. zu jeder option ist eine kurze beschreibung angegeben. im letzten abschnitt wurde beim konfigurieren des verlegers und verteilers im schritt 5 die transaktionsreplikation für die datenbank pubs vorgesehen (siehe dort abbildung 22.8). wählen sie also für das beispiel die zweite option, und klicken sie auf **weiter**.

bild

abbildung 22.18: das dialogfeld sofort aktualisierbare abonnements zulassen

bild

abbildung 22.19: das dialogfeld publikationstyp auswählen

4. für snapshot- und transaktionsreplikation bietet sql server die option der sofort aktualisierbaren abonnements. wenn sie wie im beispiel eine dieser replikationsarten gewählt haben, fragt das nächste dialogfeld, ob sie sofort aktualisierbare abonnements zulassen möchten. übernehmen sie für das beispiel die voreingestellte zweite option *nein, keine sofort aktualisierbaren abonnements zulassen* (siehe abbildung 22.19). klicken sie auf **weiter**.

5. im dialogfeld **abonnemententypen angeben** (siehe abbildung 22.20) haben sie die möglichkeit, sogenannte heterogene server - d.h. server, auf denen nicht unbedingt sql server installiert sein muß - in den replikationsprozeß einzubeziehen. das beispiel geht von der voreingestellten ersten option aus. klicken sie auf **weiter**.

abbildung 22.20: das dialogfeld abonnemententypen angeben

6. im dialogfeld **artikel angeben** (siehe abbildung 22.21) legen sie die tabellen fest, die sie in die publikation einbeziehen möchten. in einer transaktionsreplikation können sie nur tabellen mit primärschlüsseln verwenden. im dialogfeld sind deshalb die kontrollkästchen der anderen tabellen durch ein schlüsselsymbol mit rotem kreuz gekennzeichnet. wählen sie für das beispiel die tabelle titles. wenn sie im rechten teil des dialogfelds das kontrollkästchen **gespeicherte prozeduren einschliessen** einschalten, erscheint ein hinweis, daß sich die verarbeitungsleistung steigern läßt (siehe abbildung 22.22). das beispiel verzichtet allerdings auf diese möglichkeit. klicken sie deshalb im hinweisfeld auf **nein**.

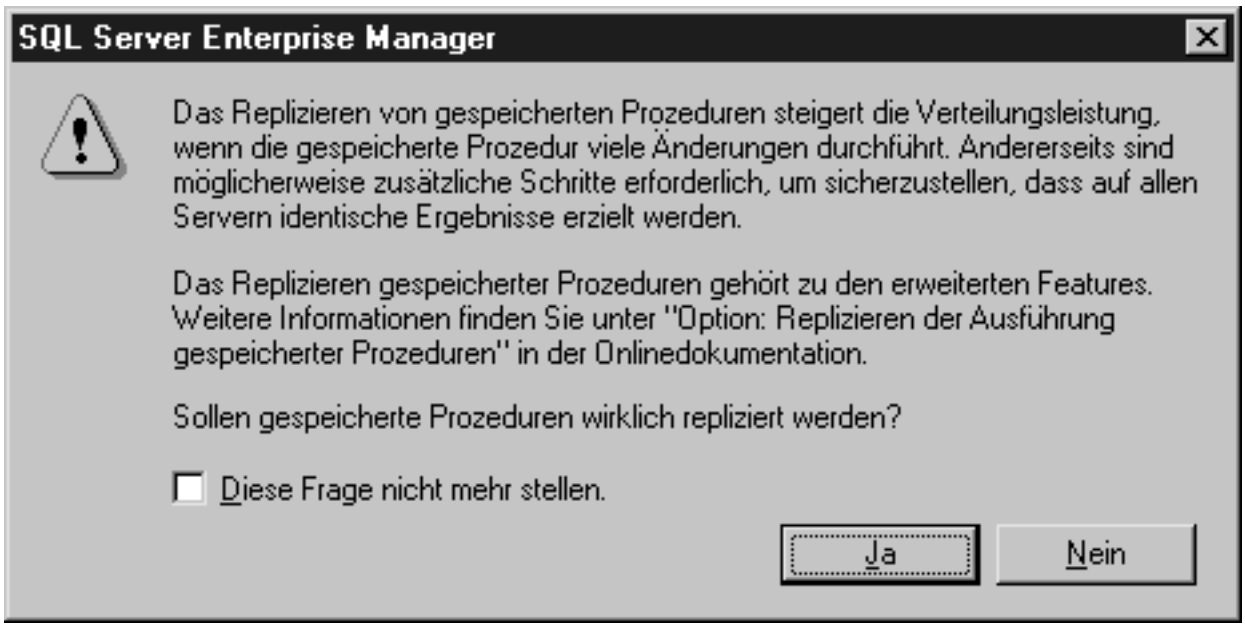


abbildung 22.21: hinweis auf verarbeitungsleistung bei verwendung gespeicherter prozeduren

bild

abbildung 22.22: das dialogfeld artikel angeben

7. wenn sie im dialogfeld **artikel angeben** auf die schaltfläche mit den drei punkten klicken, gelangen sie zum eigenschaftsdialogfeld der jeweiligen tabelle. auf der registerkarte **allgemein** (siehe abbildung 22.23) können sie den automatisch vom assistenten erzeugten artikelnamen ändern, eine beschreibung angeben und den besitzer der zieltabelle spezifizieren. die registerkarte **befehle** (siehe abbildung 22.24) erlaubt es, datenmanipulationsbefehle durch gespeicherte prozeduren zu ersetzen. schließlich finden sie auf der registerkarte **snapshot** (siehe abbildung 22.25) verschiedene optionen, die sich auf namenskonflikte, indizes, benutzerdefinierte datentypen und die referentielle integrität beziehen. zu diesen erweiterten einstellungen sei auf die online-dokumentation verwiesen. schließen sie das dialogfeld, und klicken sie auf **weiter**.

bild

abbildung 22.23: die registerkarte befehle

abbildung 22.24: die registerkarte allgemein

wählen sie auf der registerkarte **snapshot** die option *vorhandene tabelle löschen und neu erstellen*, wenn die tabelle bereits beim abonnenten vorhanden sein könnte. andernfalls kann es zu undefinierten zuständen in der tabelle kommen.

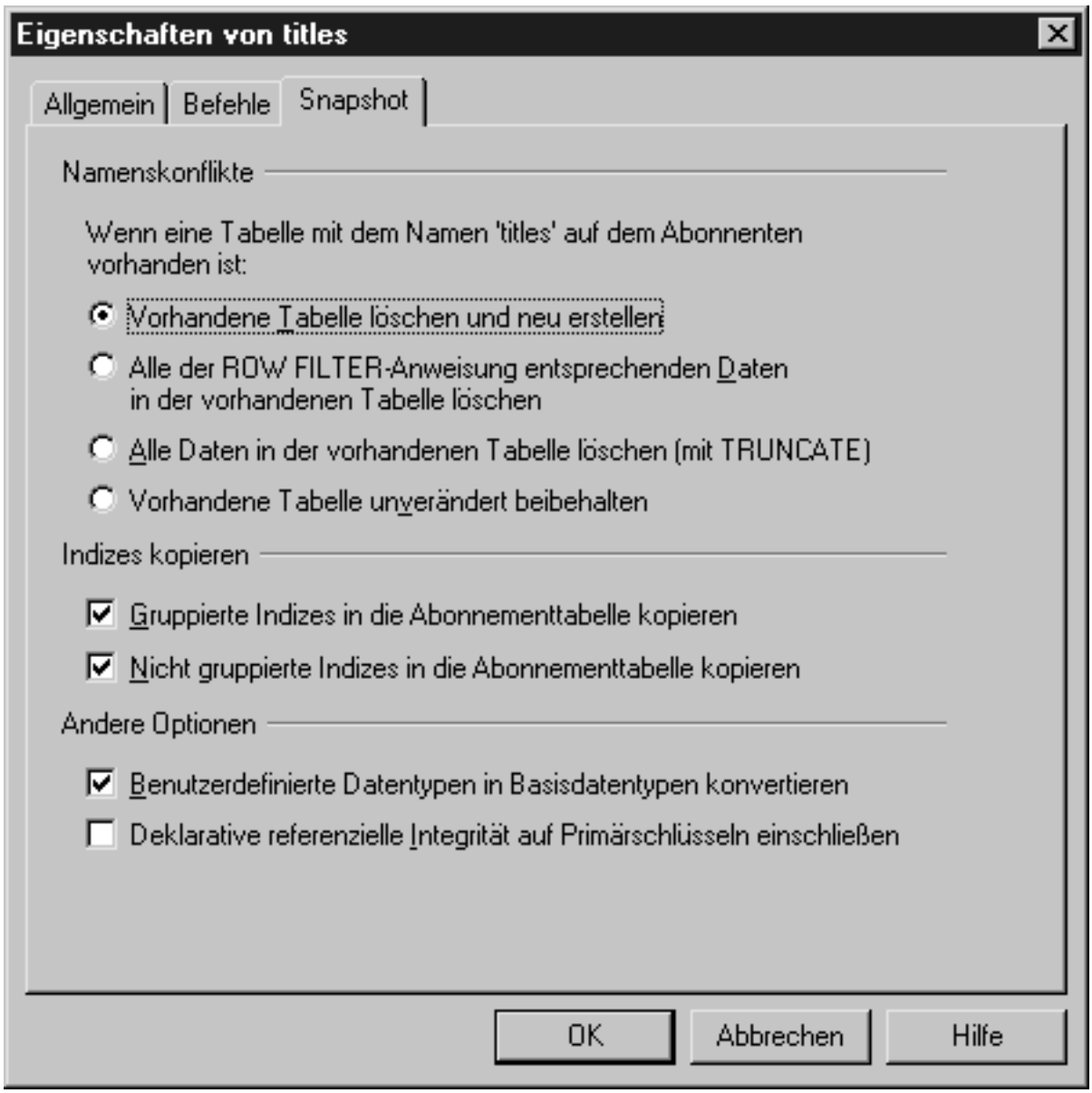


abbildung 22.25: die registerkarte snapshot

8. im nächsten dialogfeld können sie einen neuen namen für die publikation angeben und eine beschreibung eintragen. die in abbildung 22.26 gezeigte beschreibung hat der assistent automatisch generiert. klicken sie auf **weiter**.

abbildung 22.26: das dialogfeld publikationsname und beschreibung auswählen

9. wenn sie im dialogfeld **standardeigenschaften der publikation verwenden** (siehe abbildung 22.27) die zweite option wählen, gelangen sie unmittelbar zum letzten dialogfeld des assistenten. wenn sie jedoch daten filtern, das erstellen von abonnements nur für bestimmte abonnenten zulassen oder - wie im beispiel - einen terminplan für den snapshot-agenten festlegen wollen, müssen sie die erste option wählen. klicken sie auf **weiter**.

bild

abbildung 22.27: das dialogfeld standardeigenschaften der publikation verwenden

10. im beispiel geht es nur um die festlegung eines terminplans. übernehmen sie also im nächsten dialogfeld **daten filtern** die zweite option. im darauffolgenden dialogfeld **anonyme abonnenten zulassen** klicken sie ebenfalls auf **weiter**. danach gelangen sie zum dialogfeld **terminplan für snapshot-agenten festlegen** (siehe abbildung 22.28). da die replikation unverzüglich starten soll, schalten sie das kontrollkästchen **ersten snapshot sofort erstellen** ein. klikken sie auf **weiter**.

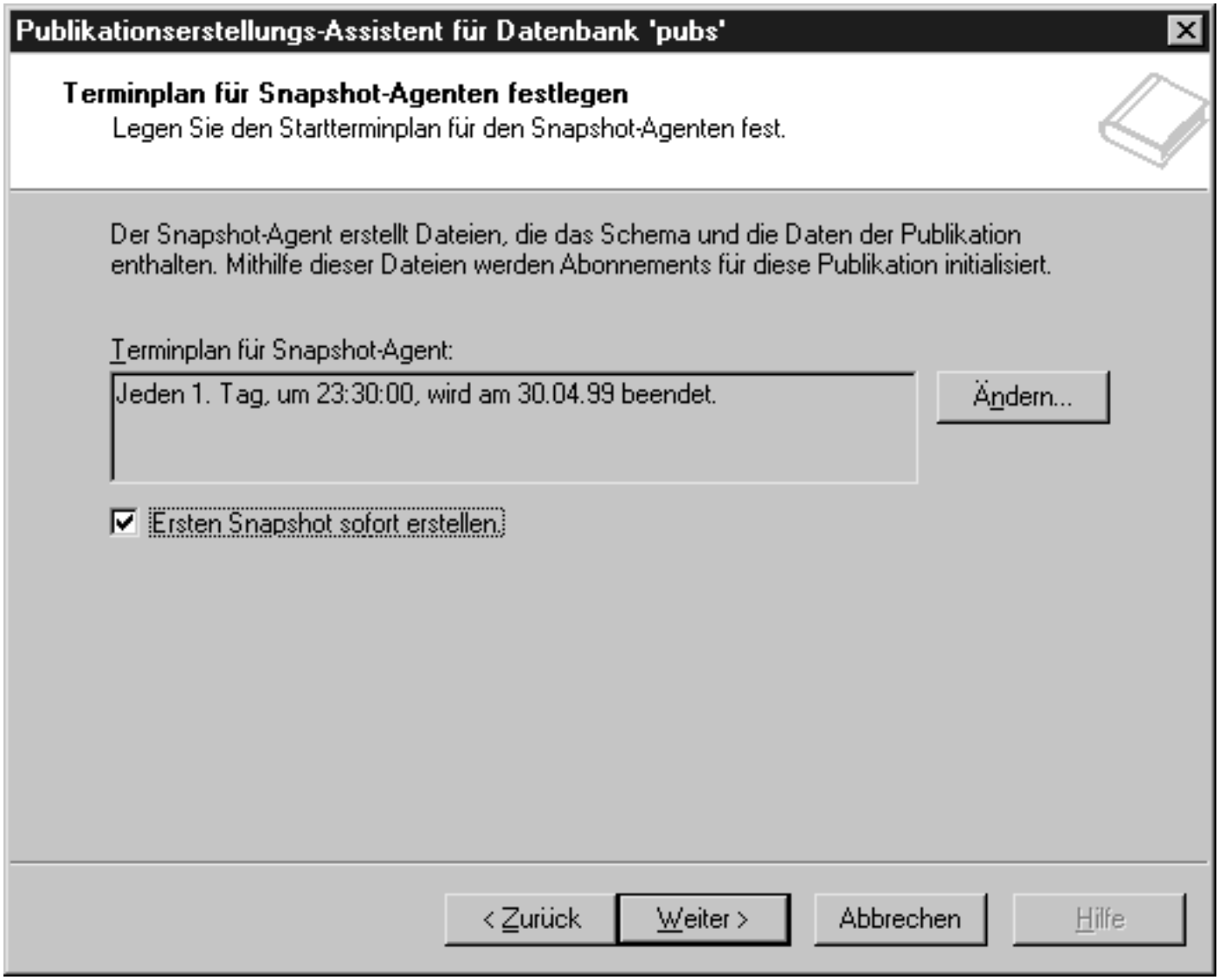


abbildung 22.28: in diesem dialogfeld legen sie den sofortigen start der replikation fest

11. das letzte dialogfeld des assistenten (siehe abbildung 22.29) zeigt eine zusammenfassung aller von ihnen gewählten einstellungen. klicken sie auf **fertigstellen**. der assistent zeigt daraufhin den fortschritt beim erstellen der publikation an und schließt das ganze mit einer erfolgsmeldung ab.

bild

abbildung 22.29: letztes dialogfeld mit einer übersicht aller einstellungen

12. klicken sie im meldungsfeld mit der fertigmeldung auf **ok**. damit gelangen sie zurück zum dialogfeld **publikationen erstellen und verwalten**, in dem die neu erstellte publikation eingetragen ist (siehe abbildung 22.30).

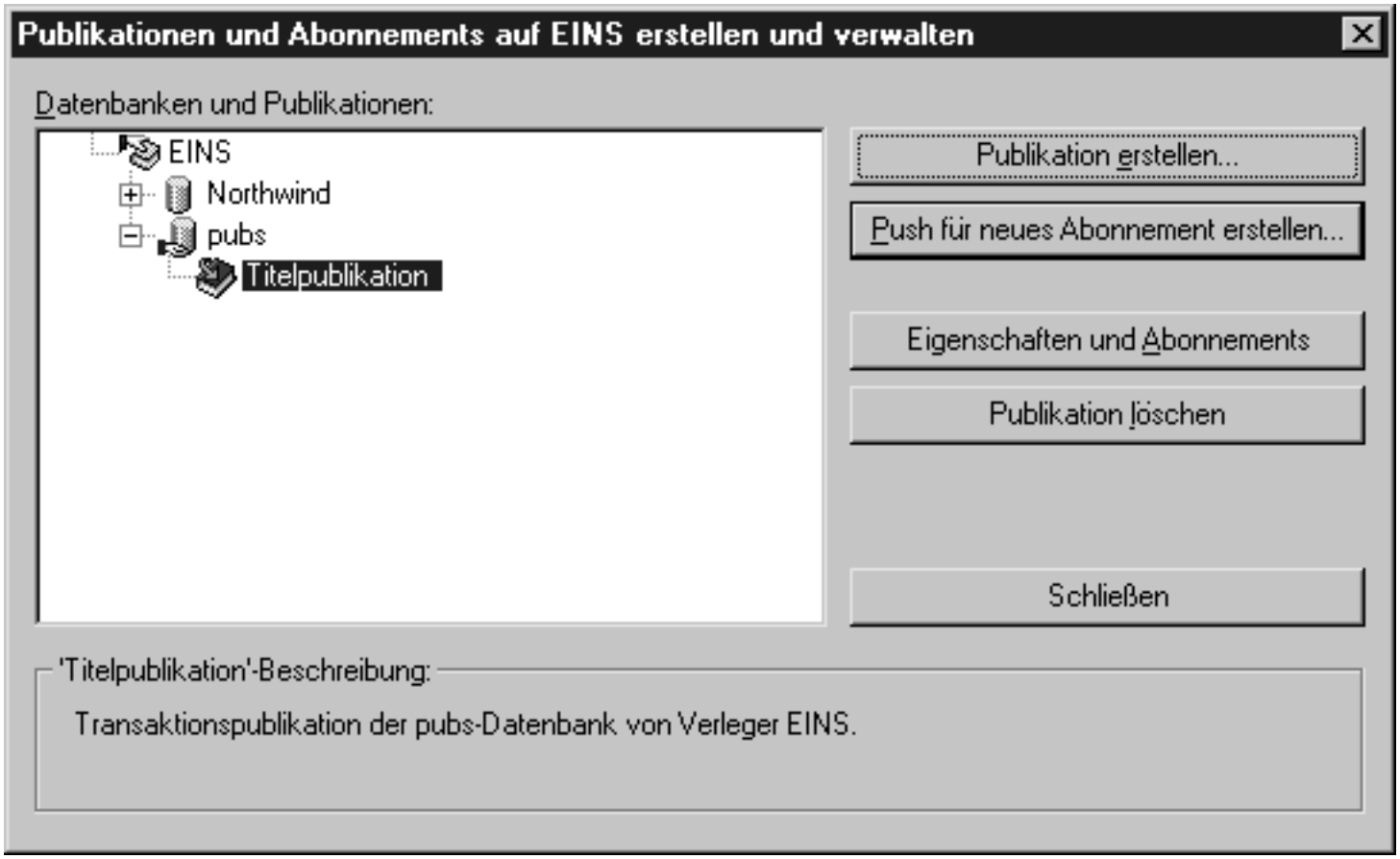


abbildung 22.30: im dialogfeld publikationen erstellen und verwalten ist die neu erstellte publikation eingetragen

wie abbildung 22.30 zeigt, können sie jetzt den push für ein neues abonnement erstellen, weitere publikationen erstellen, die eigenschaften der publikation und abonnements bearbeiten oder die publikation löschen. die nächsten abschnitte gehen auf einige punkte näher ein. klicken sie fürs erste auf **schliessen**, um das dialogfeld zu verlassen.

22.3.3 abonnements erstellen

damit die publikationen auch zu den abonnenten gelangen können, müssen sie die abonnentenserver ebenfalls konfigurieren. die folgenden abschnitte zeigen am beispiel eines pullabonnements und eines pushabonnements, wie sie dabei vorgehen müssen.

normalerweise sind die abonnentenserver vom verleger bzw. verteiler getrennt. wie sie im abschnitt zur konfiguration der replikation gesehen haben, erlaubt es sql server, daß ein und derselbe computer mehrere rollen übernimmt. die im beispiel verwendete konfiguration macht von dieser möglichkeit gebrauch. lassen sie sich also nicht irritieren, wenn als abonnent wieder der name eins auftaucht.

in jedem fall sind für den abonnenten folgende arbeiten zu erledigen:

- verleger auswählen
- publikation auswählen
- zieldatenbank festlegen
- terminplan für initialisierung und synchronisation festlegen

- anmeldungsinformationen bereitstellen

pullabonnement-assistent

wie bereits erwähnt, gehen die aktivitäten bei einem pullabonnement vom abonnenten aus. führen sie die folgenden schritte aus, um ein pullabonnement mit dem pullabonnement-assistenten einzurichten:

1. starten sie den assistenten zum beispiel über das taskpad **daten replizieren** durch klicken auf **pull für abonnement ausführen**. wählen sie im dialogfeld **pullabonnement auf** *servername* (siehe abbildung 22.31) die schaltfläche **neues pullabonnement**. damit starten sie den pullabonnement-assistenten.

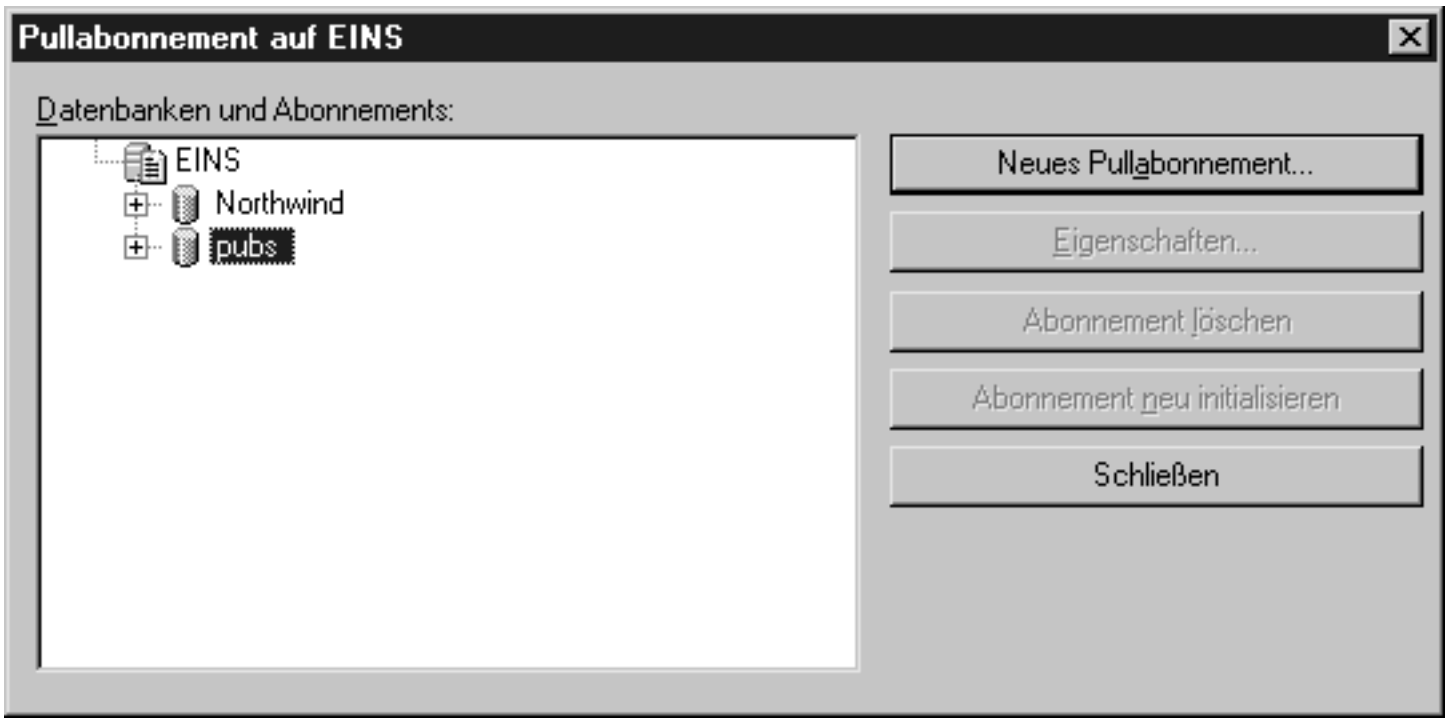


abbildung 22.31: das dialogfeld pullabonnement auf eins

2. klicken sie im startdialogfeld des assistenten auf **weiter**. im dialogfeld publikation auswählen (siehe abbildung 22.32) finden sie den server, der die publikation bereitstellt. wenn sie den serverzweig erweitern, erscheinen die aktivierten publikationen (im beispiel die titelpublikation für die datenbank pubs). klicken sie auf **weiter**.

abbildung 22.32: wählen sie die zu abonnierende publikation aus

3. geben sie im nächsten dialogfeld die anmeldeinformationen für den synchronisations-agenten ein. klicken sie auf **weiter**.
4. im dialogfeld **zieldatenbank auswählen** klicken sie auf die schaltfläche **neue datenbank**, um eine neue datenbank als ziel zu spezifizieren. legen sie im dialogfeld **datenbankeigenschaften** einen namen für die neue zieldatenbank fest, zum beispiel rptitel. die angelegte datenbank erscheint dann im dialogfeld **zieldatenbank auswählen** (siehe abbildung 22.33).

normalerweise können sie für quelle und ziel die gleichen datenbanknamen verwenden. wenn sich aber wie im beispiel beide datenbanken auf demselben server befinden, führt das unweigerlich zu problemen.

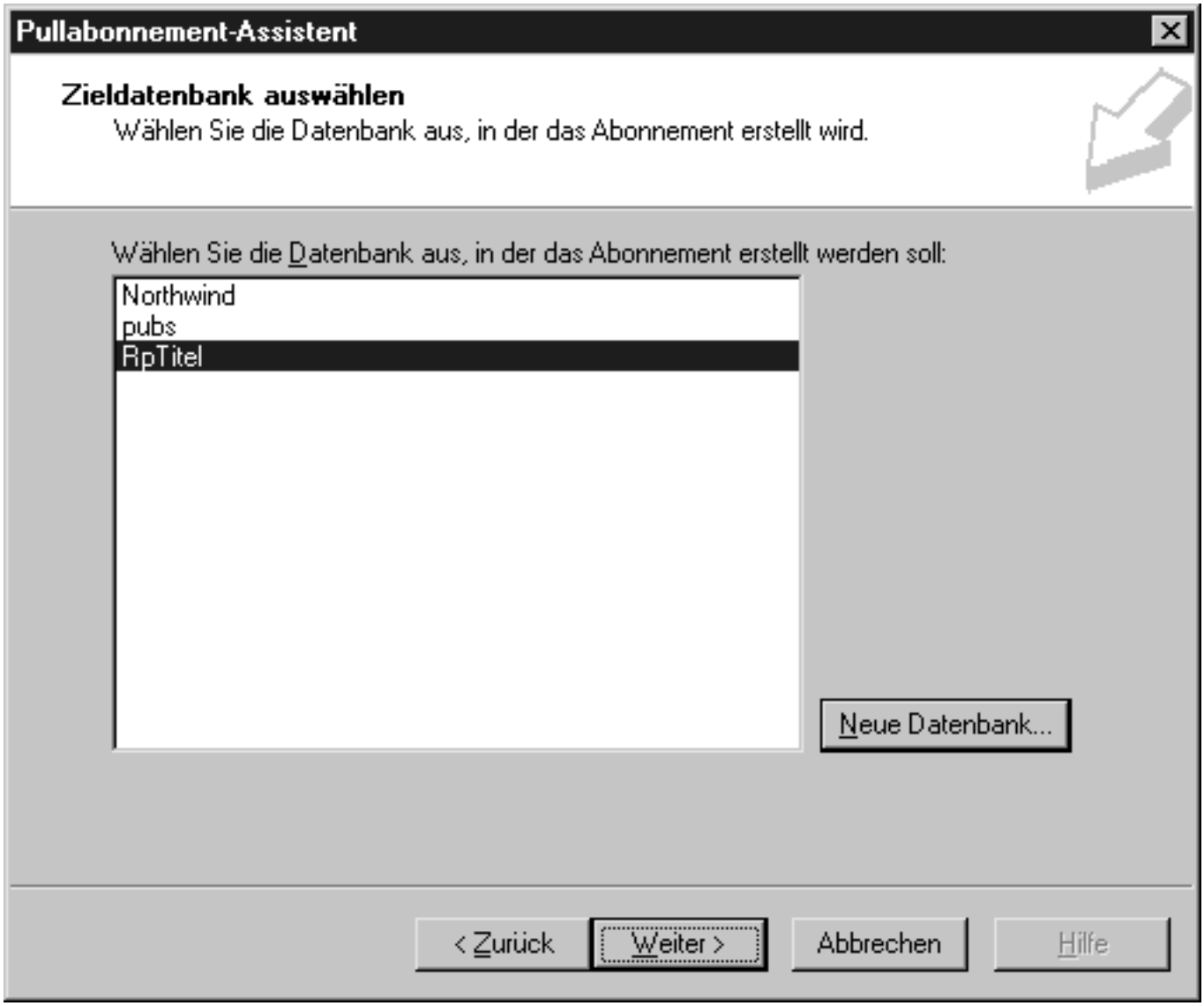


abbildung 22.33: klicken sie auf neue datenbank, um eine neue zieldatenbank zu spezifizieren

5. im dialogfeld **abonnement initialisieren** (siehe abbildung 22.34) wählen sie die erste option. klicken sie auf **weiter**.

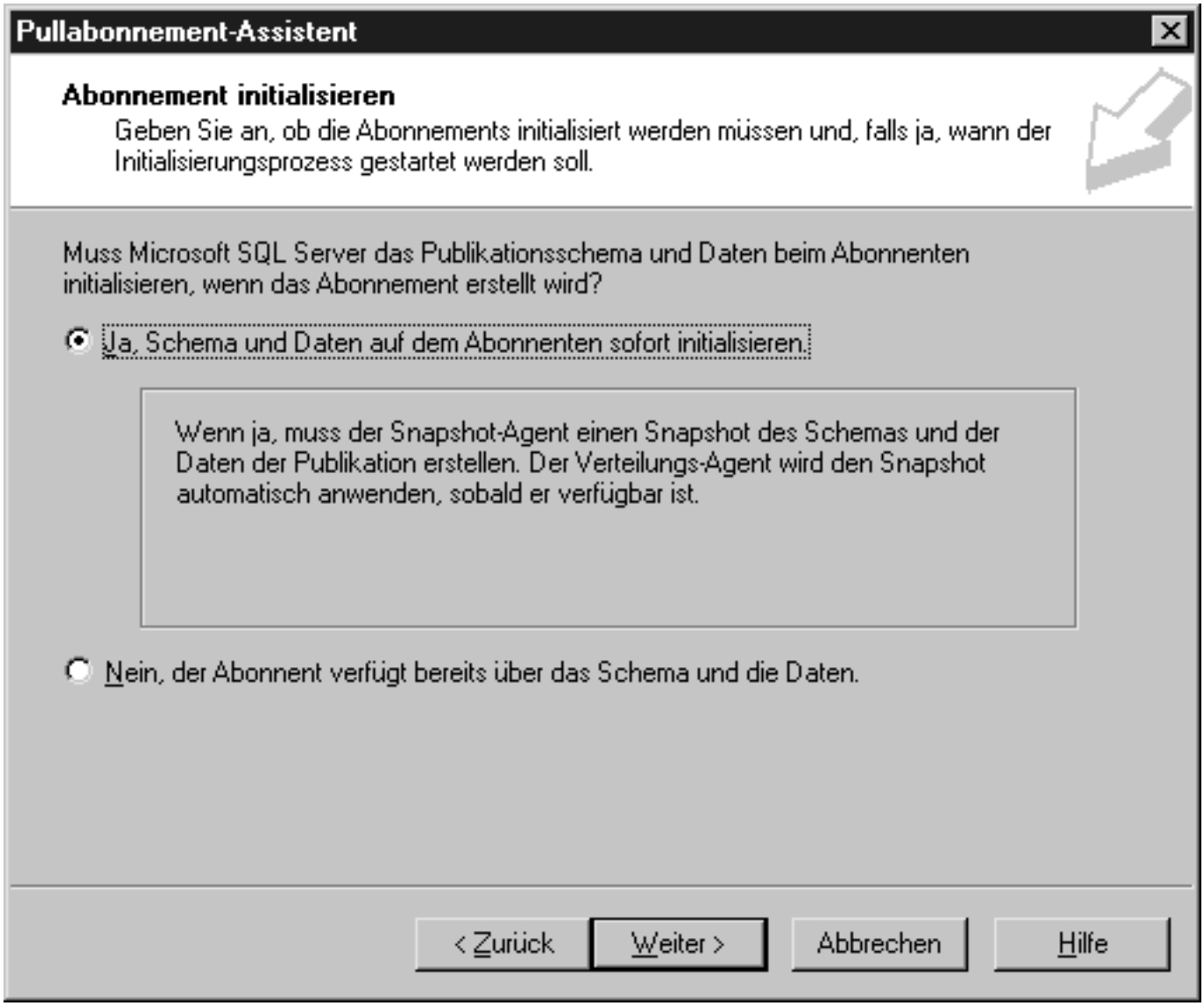


abbildung 22.34: das dialogfeld abonnement initialisieren

6. im dialogfeld **terminplan für verteilungs-agenten festlegen** (siehe abbildung 22.35) wählen sie, in welchen zeitabständen die synchronisierung erfolgen soll. übernehmen sie für das beispiel die erste option, und klicken sie auf **weiter**.

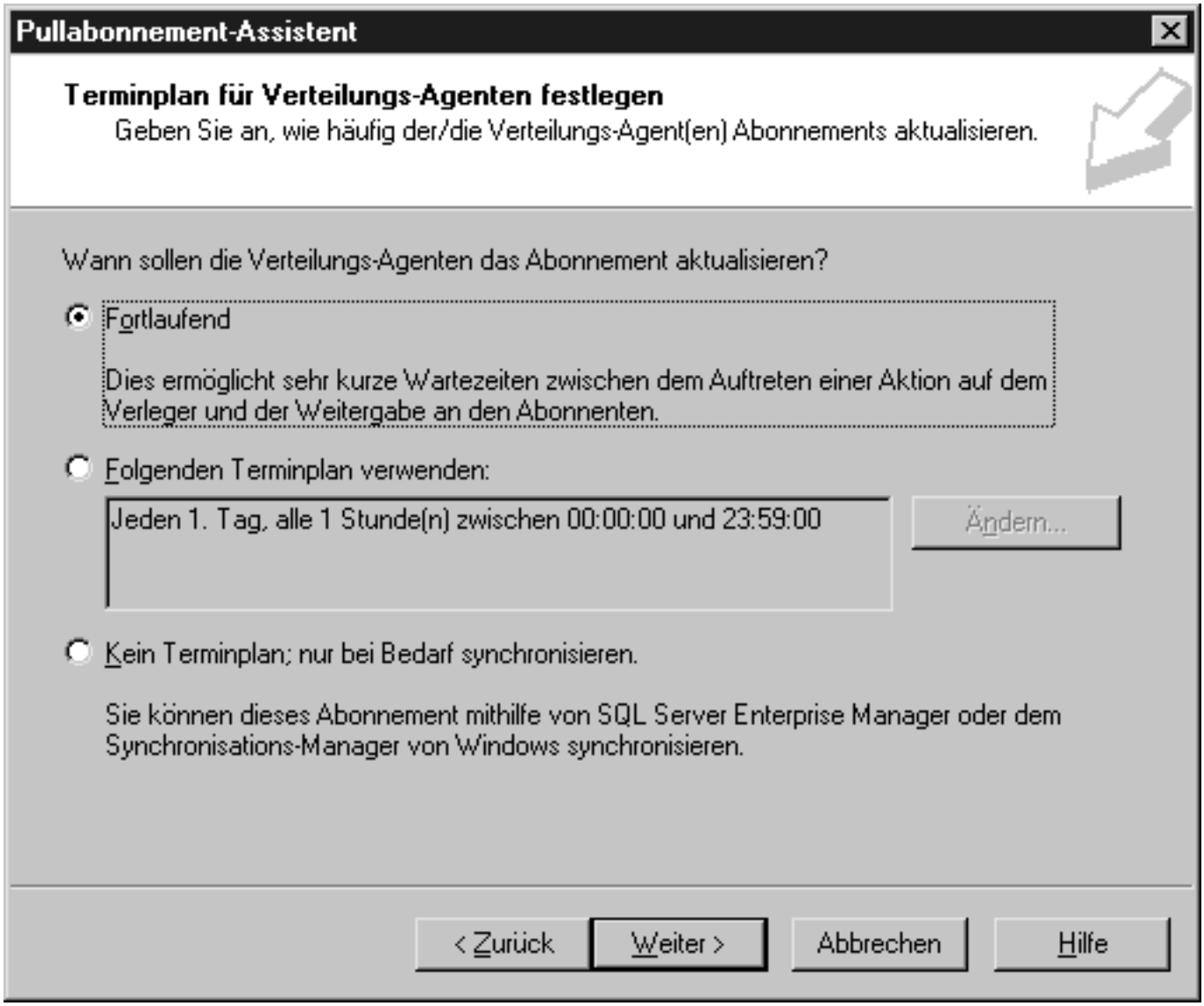


abbildung 22.35: hier legen sie die häufigkeit der synchronisierung fest

wenn sie die option *fortlaufend* wählen, bleibt der abonnent zwar immer auf dem neuesten stand, der verteilungsserver wird aber durch die laufenden abfragen in hohem maße belastet. entscheiden sie für den konkreten fall, welche option wirklich ausreichend ist. es sei noch einmal betont, daß sich die im beispiel dargestellte konfiguration nur zu testzwecken eignet. weiter unten in diesem kapitel erfahren sie, wie sich die replikation wieder von diesem server verbannen läßt.

7. damit die replikation funktioniert, muß der dienst *sql server-agent* aktiv sein. im dialogfeld **erforderliche dienste starten** (siehe abbildung 22.36) müssen sie gegebenenfalls das entsprechende kontrollkästchen einschalten. klicken sie dann auf **weiter**.

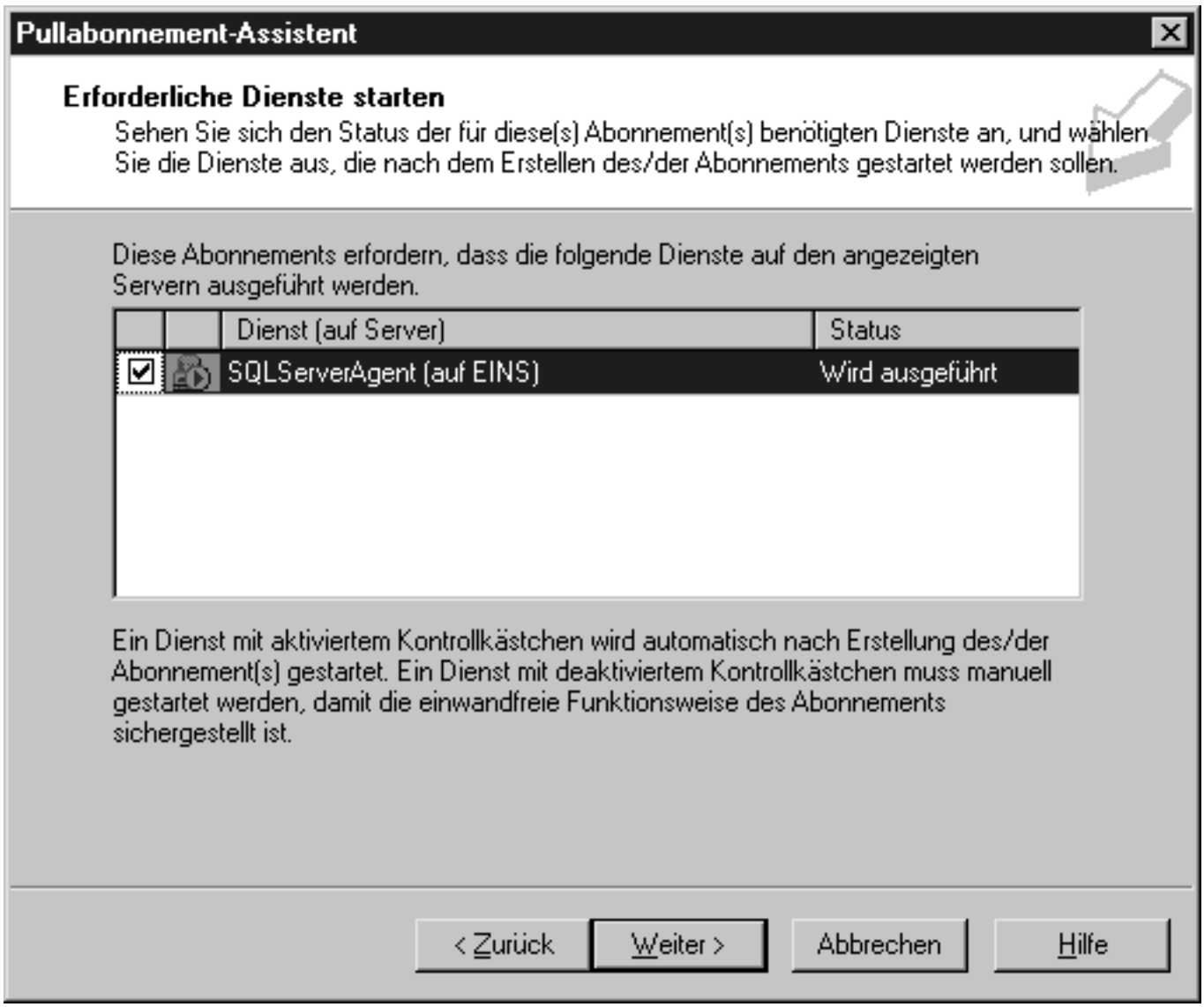


abbildung 22.36: das dialogfeld erforderliche dienste starten

8. geschafft! klicken sie im letzten dialogfeld des assistenten auf **fertigstellen**. wie gewohnt, erscheint eine fortschrittsmeldung, die mit einer erfolgsmeldung abschließt. bestätigen sie diese mit **ok**.
9. im dialogfeld **pullabonnement auf eins** ist jetzt das neue abonnement verzeichnet (siehe abbildung 22.37). bei bedarf können sie sich die eigenschaften des abonnements ansehen. klicken sie dann auf **schliessen**, um zum enterprise manager zurückzukehren.

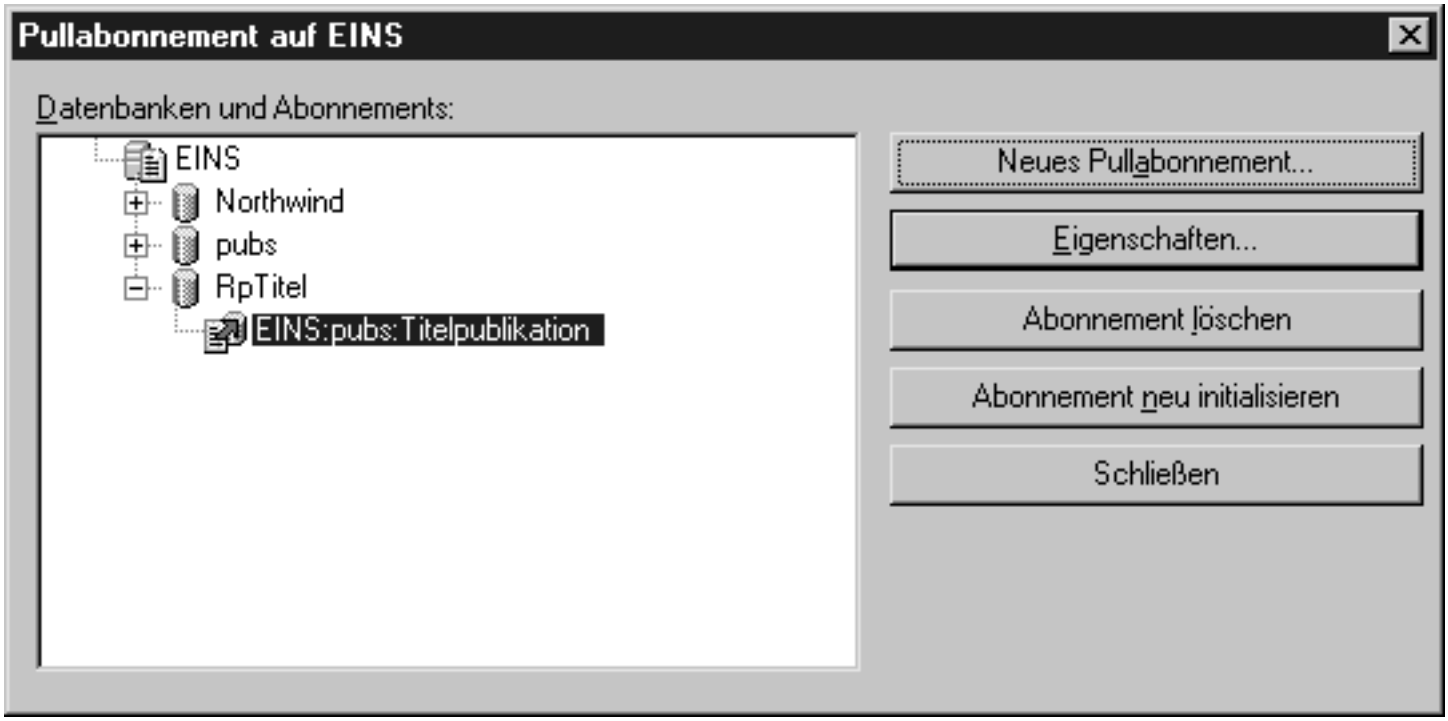


abbildung 22.37: das erstellte abonnement

pushabonnement-assistent

bei einem pushabonnement ist der verleger dafür verantwortlich, die synchronisierung einzuleiten. wenn sie ein pushabonnement erstellen wollen, durchlaufen sie mit dem pushabonnement-assistenten in etwa die gleichen schritte, wie sie sie beim pullabonnement-assistenten kennengelernt haben. dabei brauchen sie nur daran zu denken, daß jetzt alles aus der sicht des verlegers passiert.

22.3.4 manuelle synchronisation

wenn sie keine sofortige synchronisation festgelegt haben und nicht warten wollen, bis der terminplan greift, können sie die replikation auch manuell in gang setzen. erweitern sie dazu die konsolenstruktur bis zum replikationsmonitor. erweitern sie im zweig des replikationsmonitors den ordner **verleger**, und markieren sie die gewünschte publikation (siehe abbildung 22.38). im detailbereich klicken sie dann mit der rechten maustaste auf den eintrag **snapshot** und wählen im kontextmenü den befehl **starten**. in den spalten **status** und **letzte aktion** können sie ablesen, ob die manuelle synchronisation erfolgreich verlaufen ist.

bild

abbildung 22.38: manuelles starten der synchronisation für eine publikation

22.4 replikation testen

wenn sie die beispiele dieses kapitels nachvollzogen haben, finden sie in der konsolenstruktur im ordner **datenbanken** den eintrag **rptitel**. das ist die datenbank, in der das pullabonnement die artikel ablegt. markieren sie den eintrag **tabellen**, um die für die zieldatenbank angelegten tabellen im detailbereich

aufzulisten (siehe abbildung 22.39).

bild

abbildung 22.39: die tabellen in der zieldatenbank des pullabonnements

wenn die für die publikation ausgewählte tabelle titles nicht im detailbereich erscheint, hat die anfängliche synchronisation noch nicht stattgefunden. es ist allerdings auch möglich, daß sie den enterprise manager erst einmal schließen und dann erneut starten müssen - ein ähnliches vorgehen wie nach der installation eines neuen windows-programms.

zunächst einmal können sie testen, ob die anfängliche synchronisation erfolgreich verlaufen ist. führen sie dazu folgende anweisung aus:

```
use rptitel
select * from titles
```

wenn diese anweisung daten zurückgibt, hat die anfängliche synchronisation funktioniert, und sie können sich den tests der laut terminplan auszuführenden aktualisierungen widmen. führen sie dazu insert-, update- und delete-anweisungen für die betreffenden datenbanken aus, und kontrollieren sie die ergebnisse.

22.5 replikation entfernen

wie weiter oben in diesem kapitel versprochen, zeigt dieser abschnitt, wie sie die replikation wieder loswerden. sql server stellt für diesen zweck sogar einen eigenen assistenten bereit, der auf den klangvollen namen publizierungs- und verteilungsdeaktivierungs-assistent hört.

führen sie die folgenden schritte aus, um die replikation zu entfernen:

1. starten sie den assistenten über **extras / replikation / publizierung deaktivieren** oder im taskpad **daten replizieren** durch klicken auf **replikation entfernen**. daraufhin startet der publizierungs- und verteilungsdeaktivierungs-assistent (siehe abbildung 22.40). klicken sie auf **weiter**.

Bild

abbildung 22.40: startdialogfeld des publizierungs- und verteilungsdeaktivierungs-assistenten

2. im dialogfeld **publizierung deaktivieren** (siehe abbildung 22.41) wählen sie die erste option, um die publizierung auf eins gänzlich zu entfernen. klicken sie auf **weiter**.

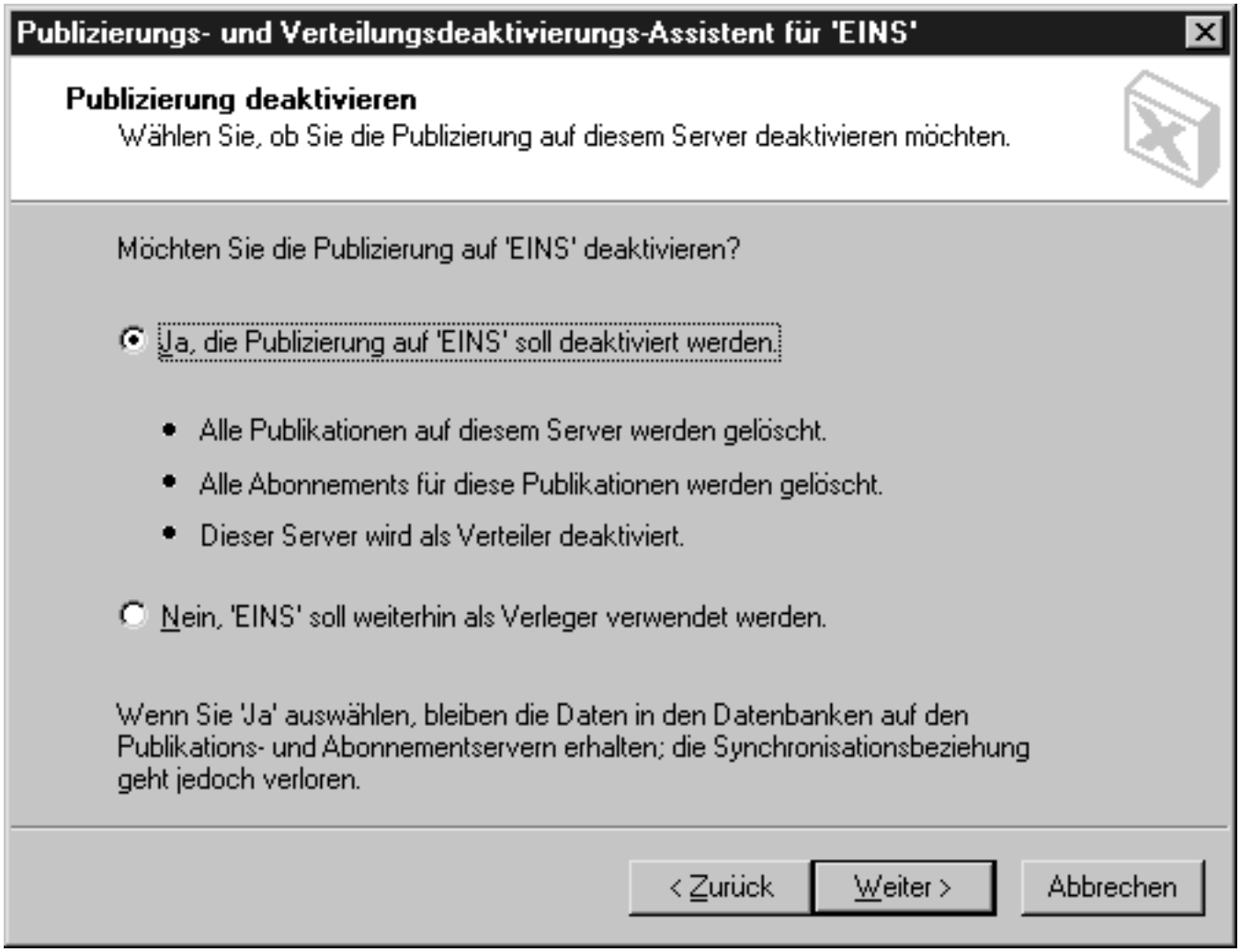


abbildung 22.41: das dialogfeld publizierung deaktivieren

3. da sie bereits publikationen erstellt haben, zeigt der assistent im dialogfeld **publikationslöschung bestätigen** (siehe abbildung 22.42) alle publikationen an, die entfernt werden. klicken sie auf **weiter**.

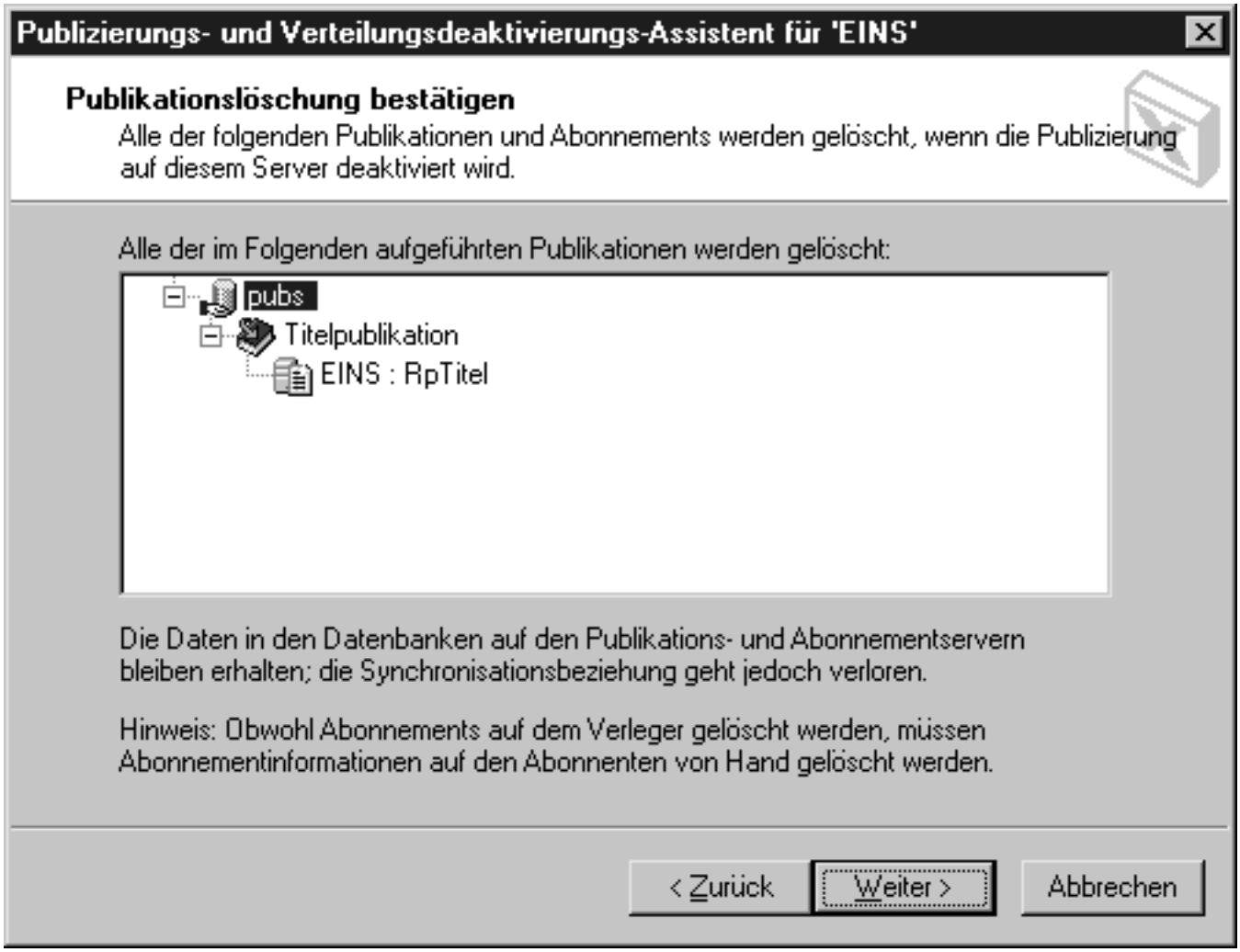


abbildung 22.42: hier sind alle publikationen aufgeführt, die der assistent löscht

4. damit sind sie beim letzten dialogfeld des assistenten angelangt (siehe abbildung 22.43). klicken sie auf **fertigstellen**, um die replikation komplett zu entfernen.

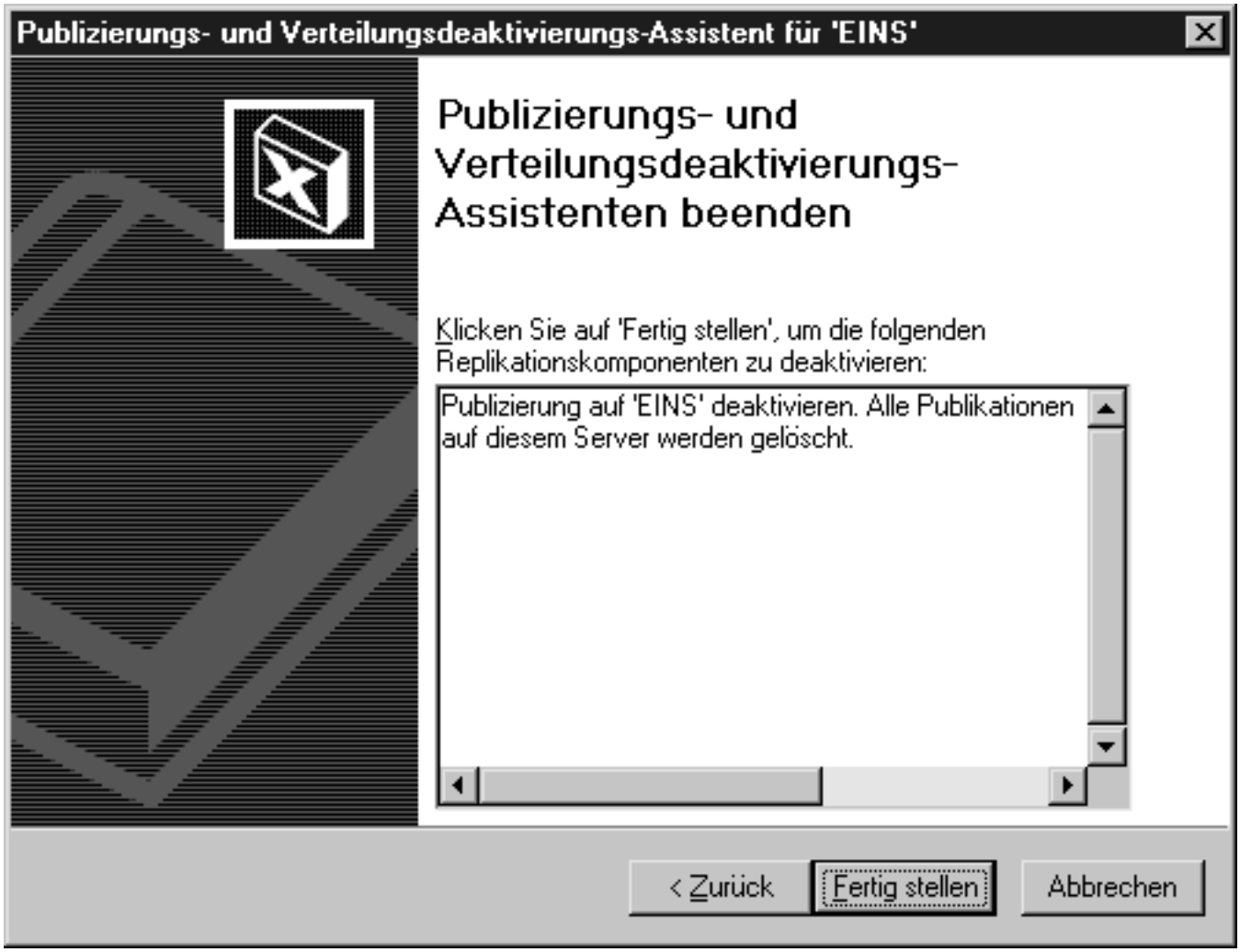


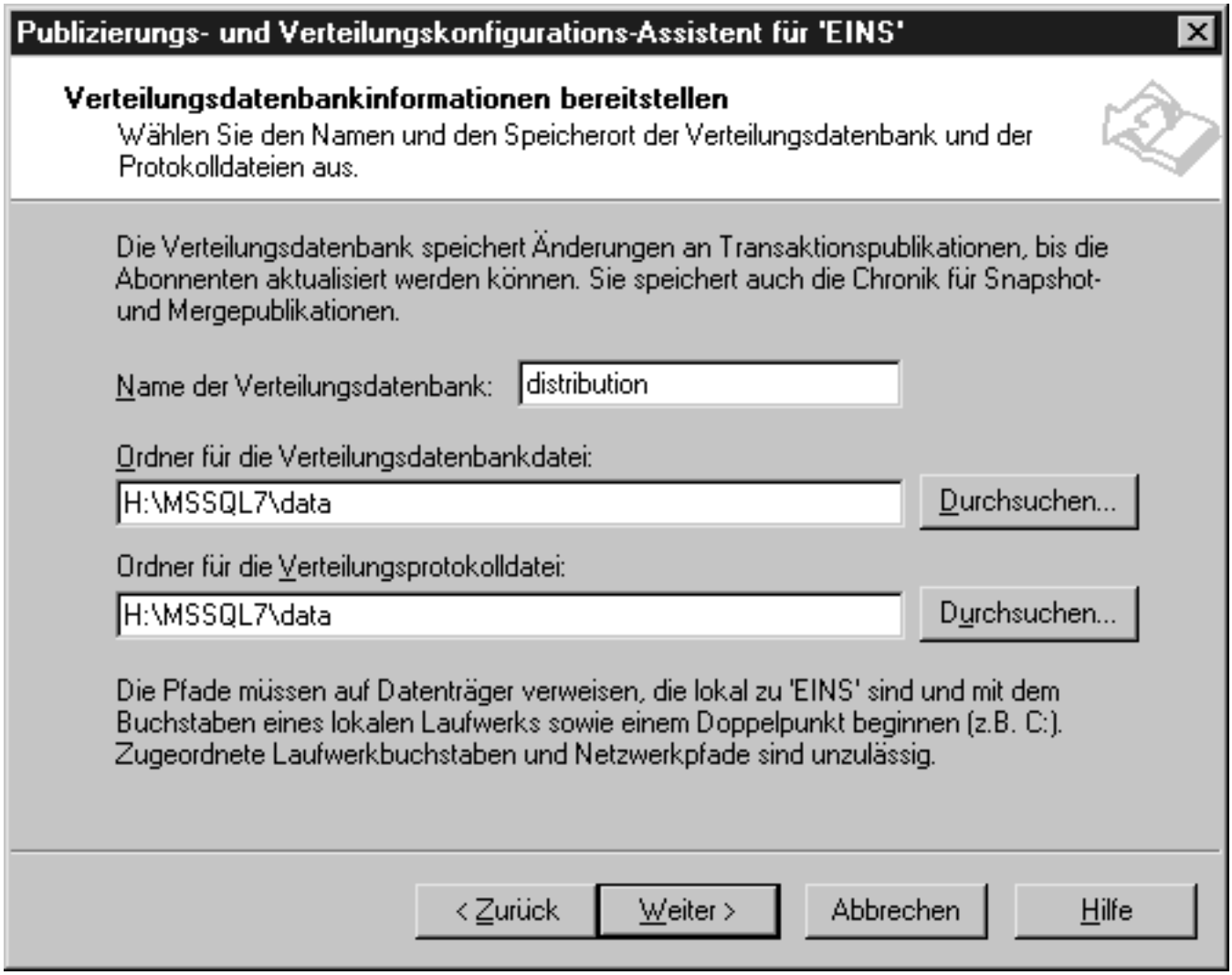
abbildung 22.43: letztes dialogfeld des assistenten

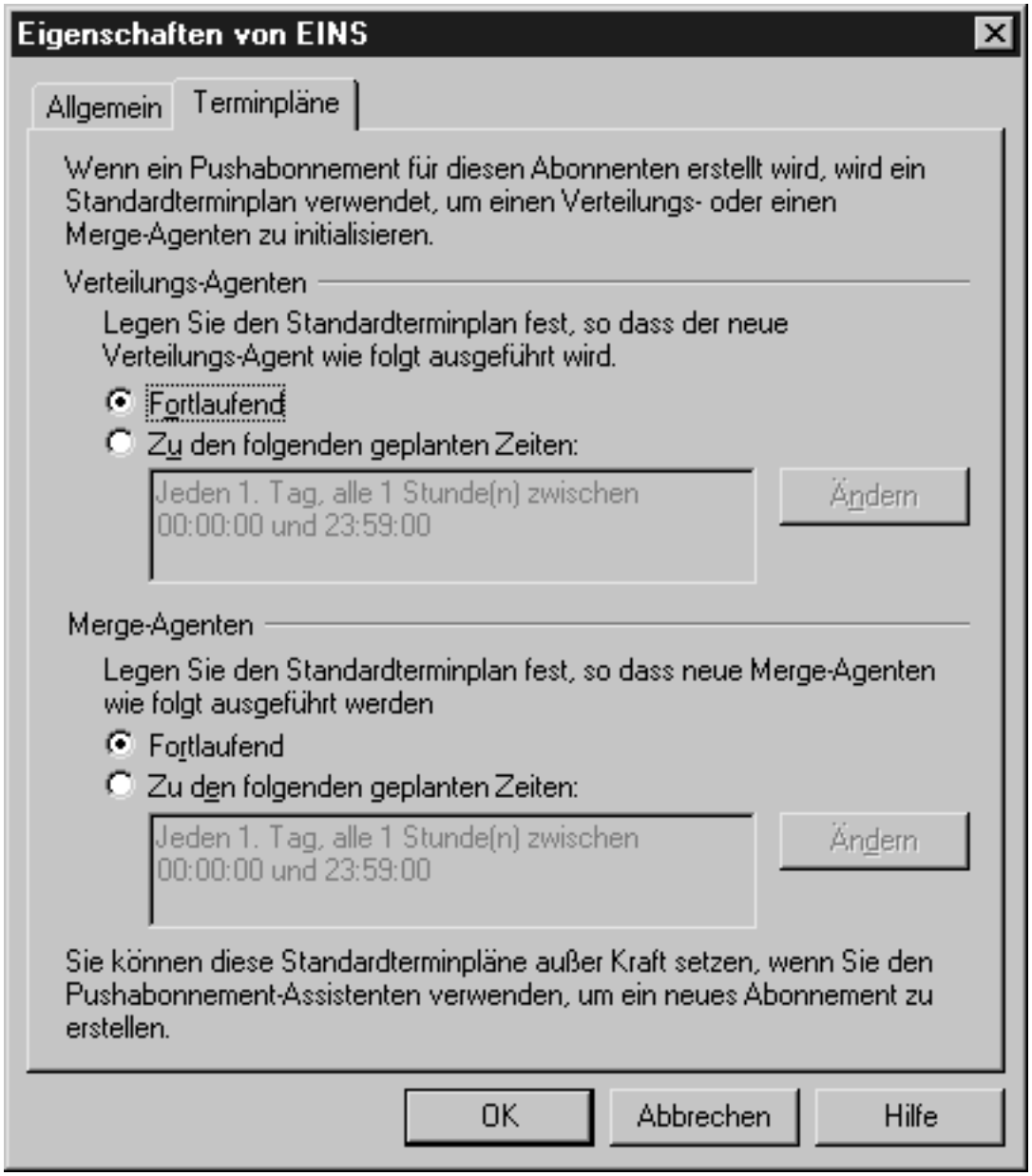
wenn eine meldung erscheint, daß sich zum beispiel die datenbank distribution nicht löschen ließ, weil sie in verwendung ist, beenden sie den dienst sql server-agent und führen den assistenten erneut aus. es ist durchaus möglich, daß ein laufender terminplan das löschen verhindert hat.

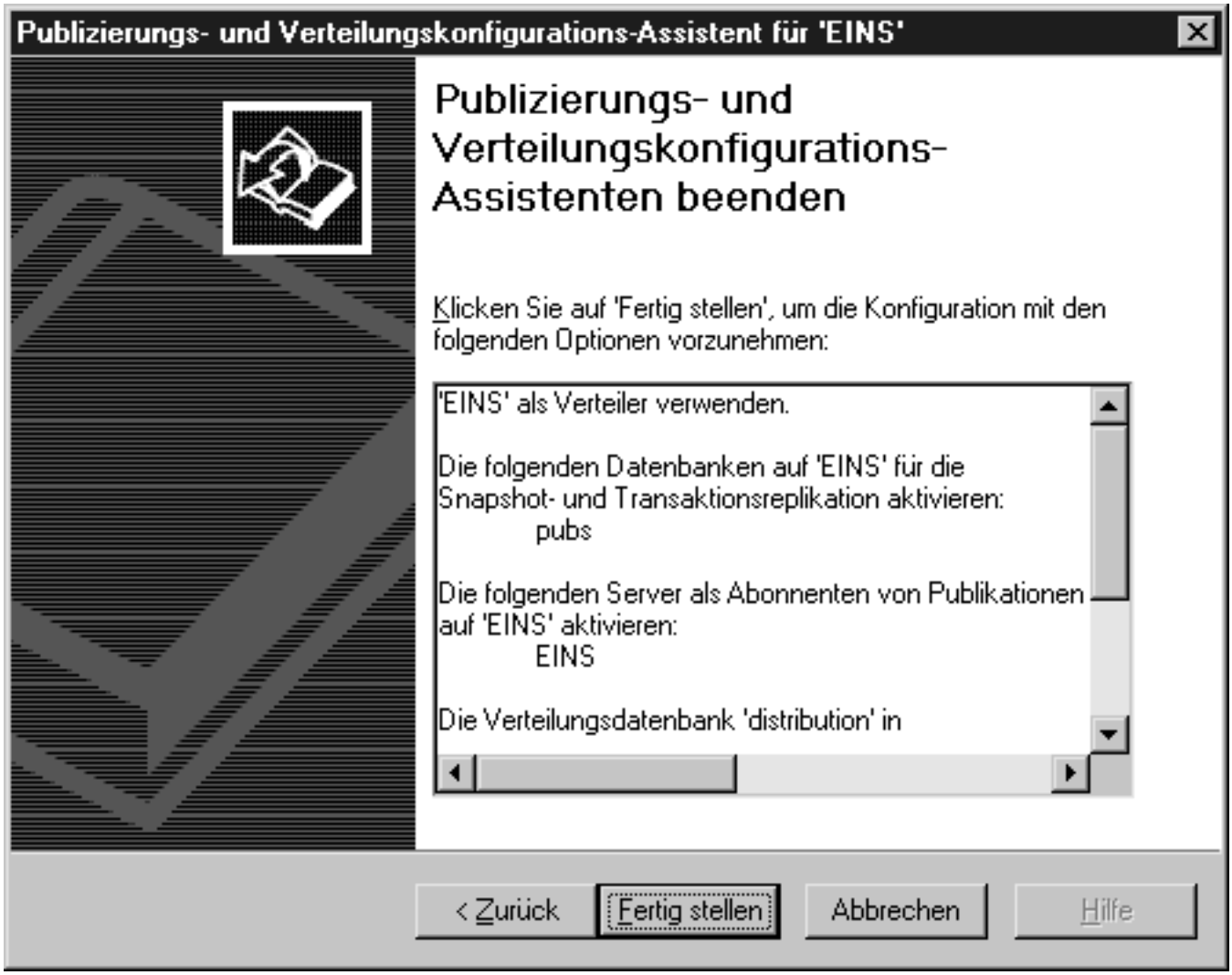
nachdem eine abschließende meldung bestätigt hat, daß die replikation vollständig entfernt wurde, befindet sich die konsolenstruktur wieder im ursprünglichen zustand, d.h., bevor sie mit der konfiguration von verlegern und verteilern begonnen haben. weiterhin sind die erwähnten freigabesymbole verschwunden. die für das pullabonnement angelegte datenbank rptitel ist allerdings weiterhin vorhanden, nur hat sie keinen bezug mehr zur quelldatenbank pubs. es handelt sich jetzt um eine ganz normale datenbank mit einer einzigen tabelle.

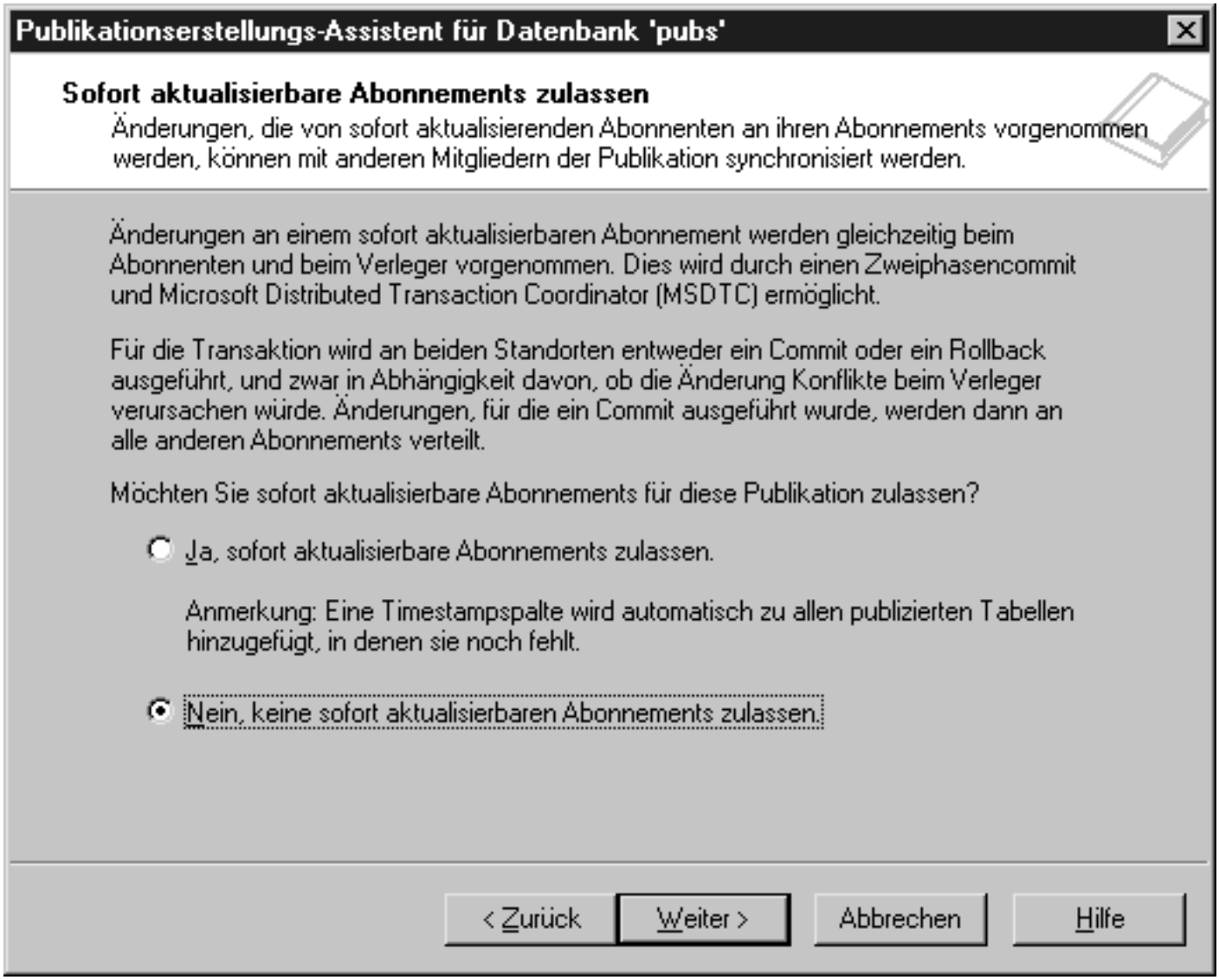

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)]
Konsole
Fenster
?
Vorgang
Ansicht
Extras
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Daten replizieren
Replikation konfigurieren
Konfigurieren Sie die Publizierung und Verteilung auf diesem Server.
entfernen
Publikationen erstellen oder verwalten
Pull für Abonnement ausführen
Push für Abonnement ausführen
Online-dokumentation
Microsoft
SQL Server
res://d:\MSSQL7\Binn\Resources\1031\sqlmmc.rll/

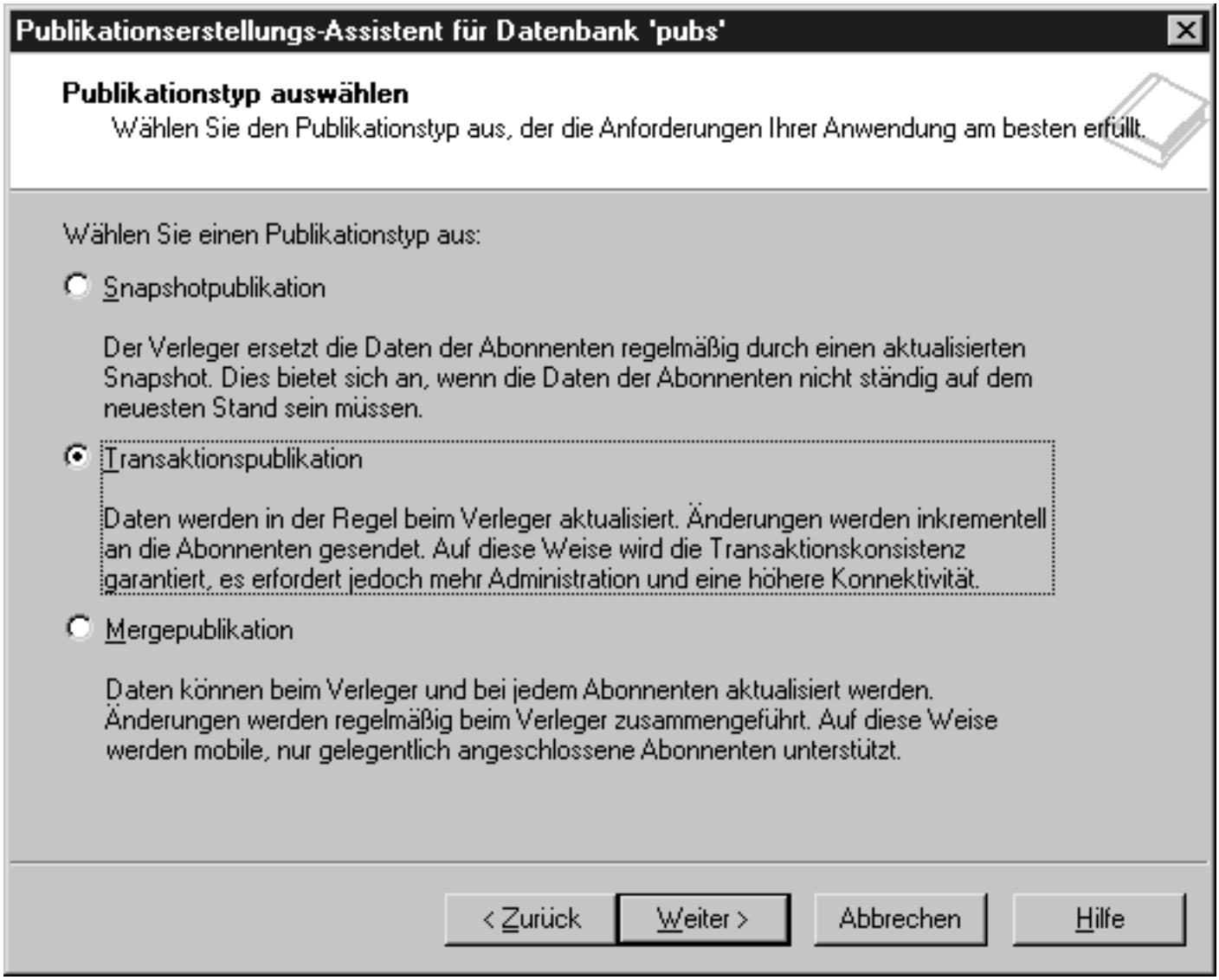

Publizierungs- und Verteilungskonfigurations-Assistent für 'EINS'
Standardkonfiguration verwenden
Sie können den Server für die Replikation unter Verwendung von Standardeinstellungen konfigurieren. Sie können die Einstellungen jedoch auch anpassen.
Möchten Sie die Publizierungs- und Verteilungseinstellungen anpassen?
Ja, ich möchte den Namen und den Speicherort der Verteilungsdatenbank festlegen, weitere Verleger aktivieren oder Publizierungseinstellungen anpassen.
Nein, folgende Standardeinstellungen verwenden:
'EINS' als Verteiler verwenden.
Die folgenden Server als Abonnenten von Publikationen auf 'EINS' aktivieren:
EINS
Die Verteilungsdatenbank 'distribution' in 'd:\MSSQL7\data' speichern.
Die folgenden Server aktivieren, um 'EINS' als ihren Verteiler zu verwenden, wenn sie später als Verleger konfiguriert werden:
EINS
< Zurück
Weiter >
Abbrechen
Hilfe

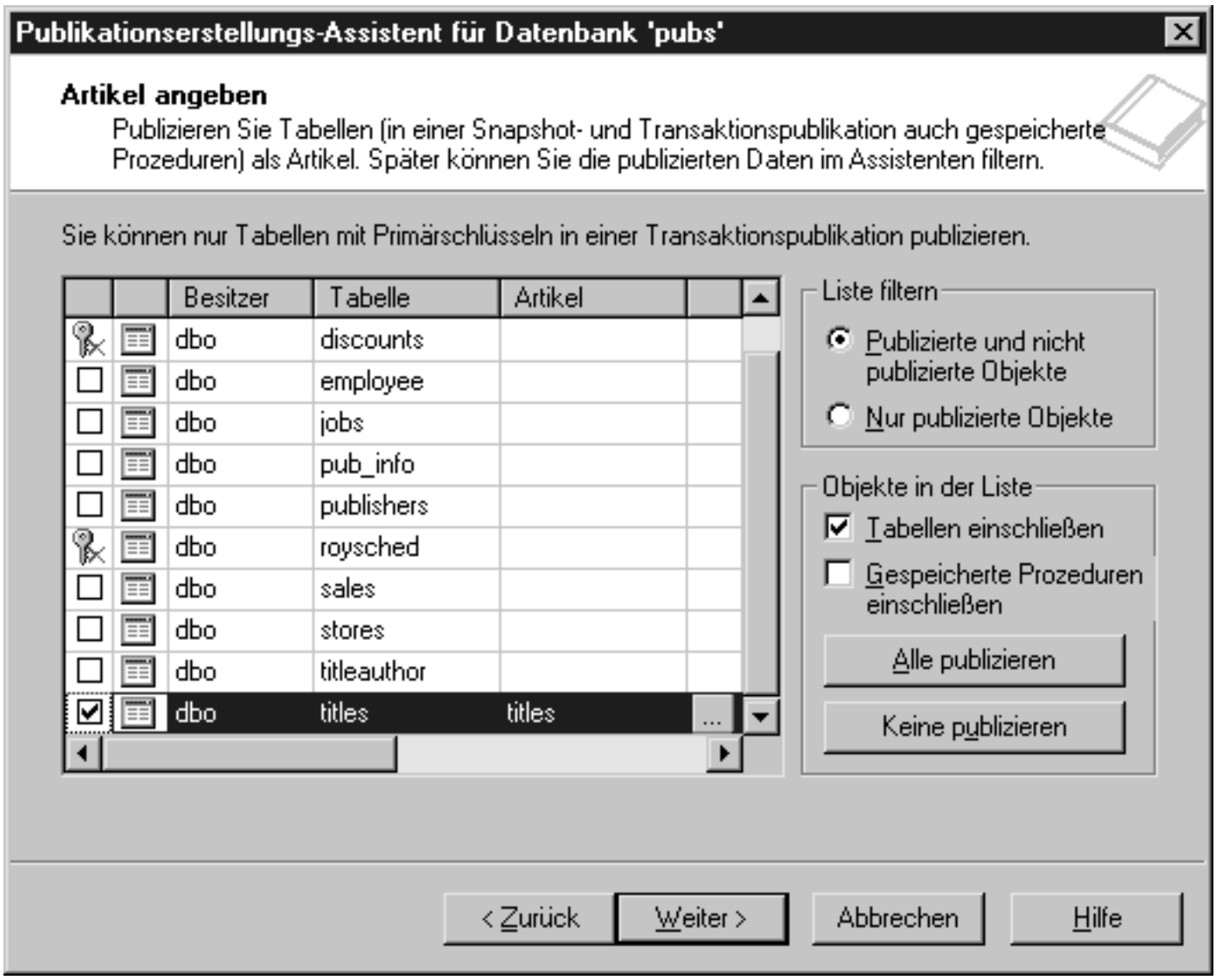

Publizierungs- und Verteilungskonfigurations-Assistent für 'EINS'
Verteilungsdatenbankinformationen bereitstellen
Wählen Sie den Namen und den Speicherort der Verteilungsdatenbank und der Protokolldateien aus.
Die Verteilungsdatenbank speichert Änderungen an Transaktionspublikationen, bis die Abonnenten aktualisiert werden können. Sie speichert auch die Chronik für Snapshot- und Mergepublikationen.
Name der Verteilungsdatenbank:
distribution
Ordner für die Verteilungsdatenbankdatei:
H:\MSSQL7\data
Durchsuchen...
Ordner für die Verteilungsprotokolldatei:
H:\MSSQL7\data
Durchsuchen...
Die Pfade müssen auf Datenträger verweisen, die lokal zu 'EINS' sind und mit dem Buchstaben eines lokalen Laufwerks sowie einem Doppelpunkt beginnen (z.B. C:). Zugeordnete Laufwerkbuchstaben und Netzwerkpfade sind unzulässig.
< Zurück
Weiter >
Abbrechen
Hilfe

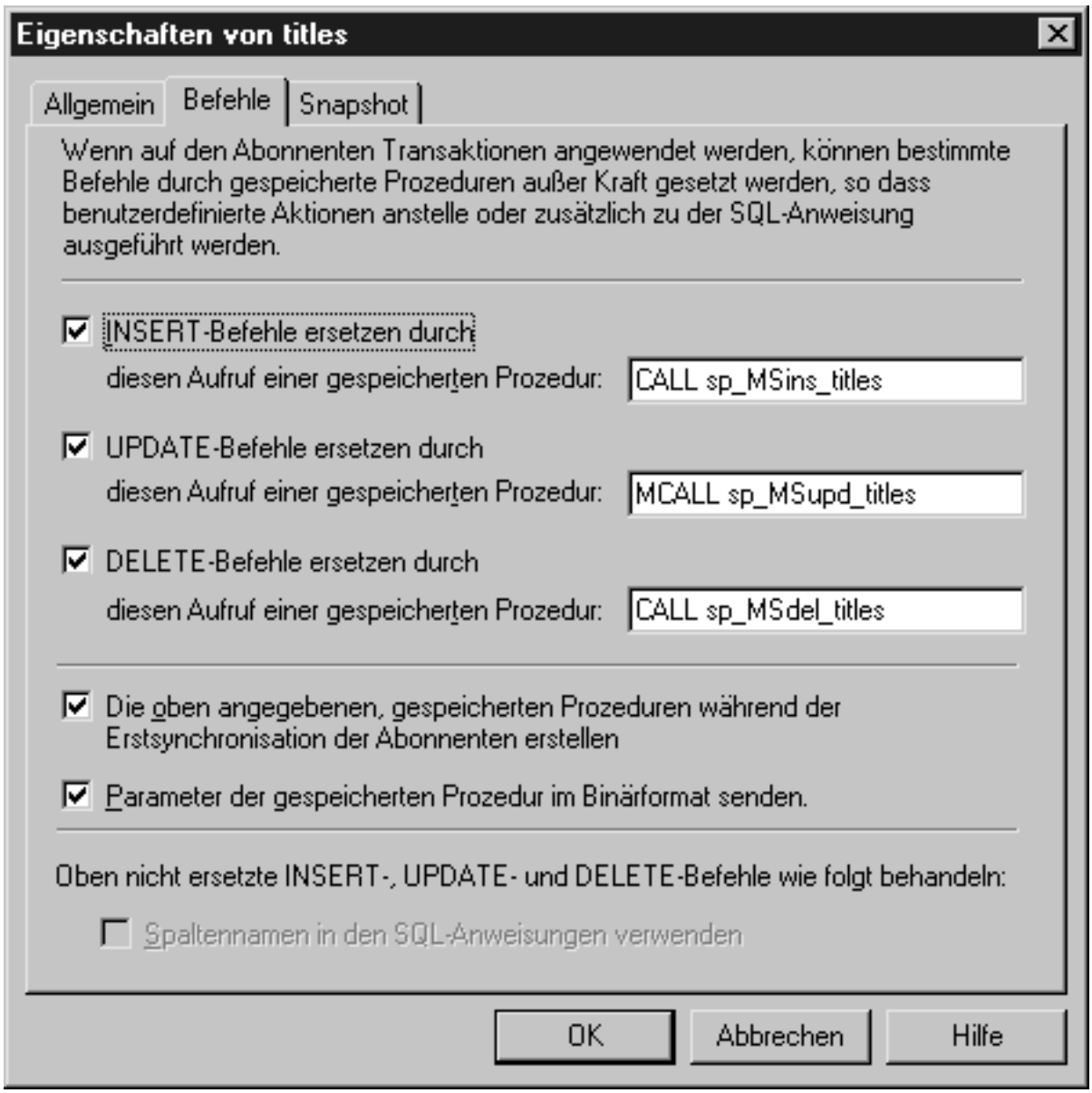

Eigenschaften von EINS
Allgemein
Terminpläne
Wenn ein Pushabonnement für diesen Abonnenten erstellt wird, wird ein Standardterminplan verwendet, um einen Verteilungs- oder einen Merge-Agenten zu initialisieren.
Verteilungs-Agenten
Legen Sie den Standardterminplan fest, so dass der neue Verteilungs-Agent wie folgt ausgeführt wird.
Fortlaufend
Zu den folgenden geplanten Zeiten:
Jeden 1. Tag, alle 1 Stunde(n) zwischen 00:00:00 und 23:59:00
Ändern
Merge-Agenten
Legen Sie den Standardterminplan fest, so dass neue Merge-Agenten wie folgt ausgeführt werden
Fortlaufend
Zu den folgenden geplanten Zeiten:
Jeden 1. Tag, alle 1 Stunde(n) zwischen 00:00:00 und 23:59:00
Ändern
Sie können diese Standardterminpläne außer Kraft setzen, wenn Sie den Pushabonnement-Assistenten verwenden, um ein neues Abonnement zu erstellen.
OK
Abbrechen
Hilfe


Publizierungs- und Verteilungskonfigurations-Assistent für 'EINS'
Publizierungs- und Verteilungskonfigurations-Assistenten beenden
Klicken Sie auf 'Fertig stellen', um die Konfiguration mit den folgenden Optionen vorzunehmen:
'EINS' als Verteiler verwenden.
Die folgenden Datenbanken auf 'EINS' für die Snapshot- und Transaktionsreplikation aktivieren:
pubs
Die folgenden Server als Abonnenten von Publikationen auf 'EINS' aktivieren:
EINS
Die Verteilungsdatenbank 'distribution' in
< Zurück
Fertig stellen
Abbrechen
Hilfe

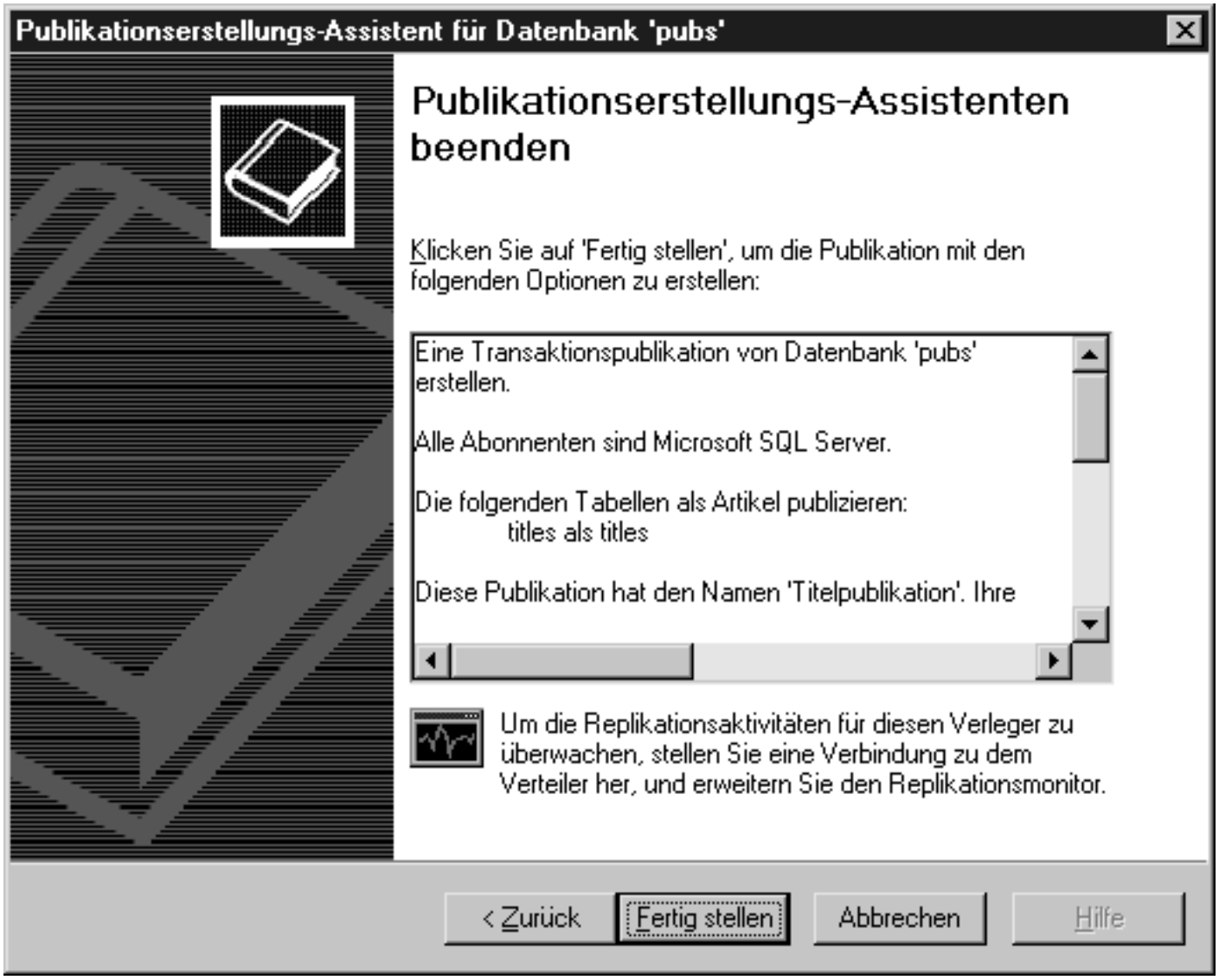

Publikationserstellungs-Assistent für Datenbank 'pubs'
Sofort aktualisierbare Abonnements zulassen
Änderungen, die von sofort aktualisierenden Abonnenten an ihren Abonnements vorgenommen werden, können mit anderen Mitgliedern der Publikation synchronisiert werden.
Änderungen an einem sofort aktualisierbaren Abonnement werden gleichzeitig beim Abonnenten und beim Verleger vorgenommen. Dies wird durch einen Zweiphasencommit und Microsoft Distributed Transaction Coordinator (MSDTC) ermöglicht.
Für die Transaktion wird an beiden Standorten entweder ein Commit oder ein Rollback ausgeführt, und zwar in Abhängigkeit davon, ob die Änderung Konflikte beim Verleger verursachen würde. Änderungen, für die ein Commit ausgeführt wurde, werden dann an alle anderen Abonnements verteilt.
Möchten Sie sofort aktualisierbare Abonnements für diese Publikation zulassen?
Ja, sofort aktualisierbare Abonnements zulassen.
Anmerkung: Eine Timestampspalte wird automatisch zu allen publizierten Tabellen hinzugefügt, in denen sie noch fehlt.
Nein, keine sofort aktualisierbaren Abonnements zulassen.
< Zurück
Weiter >
Abbrechen
Hilfe

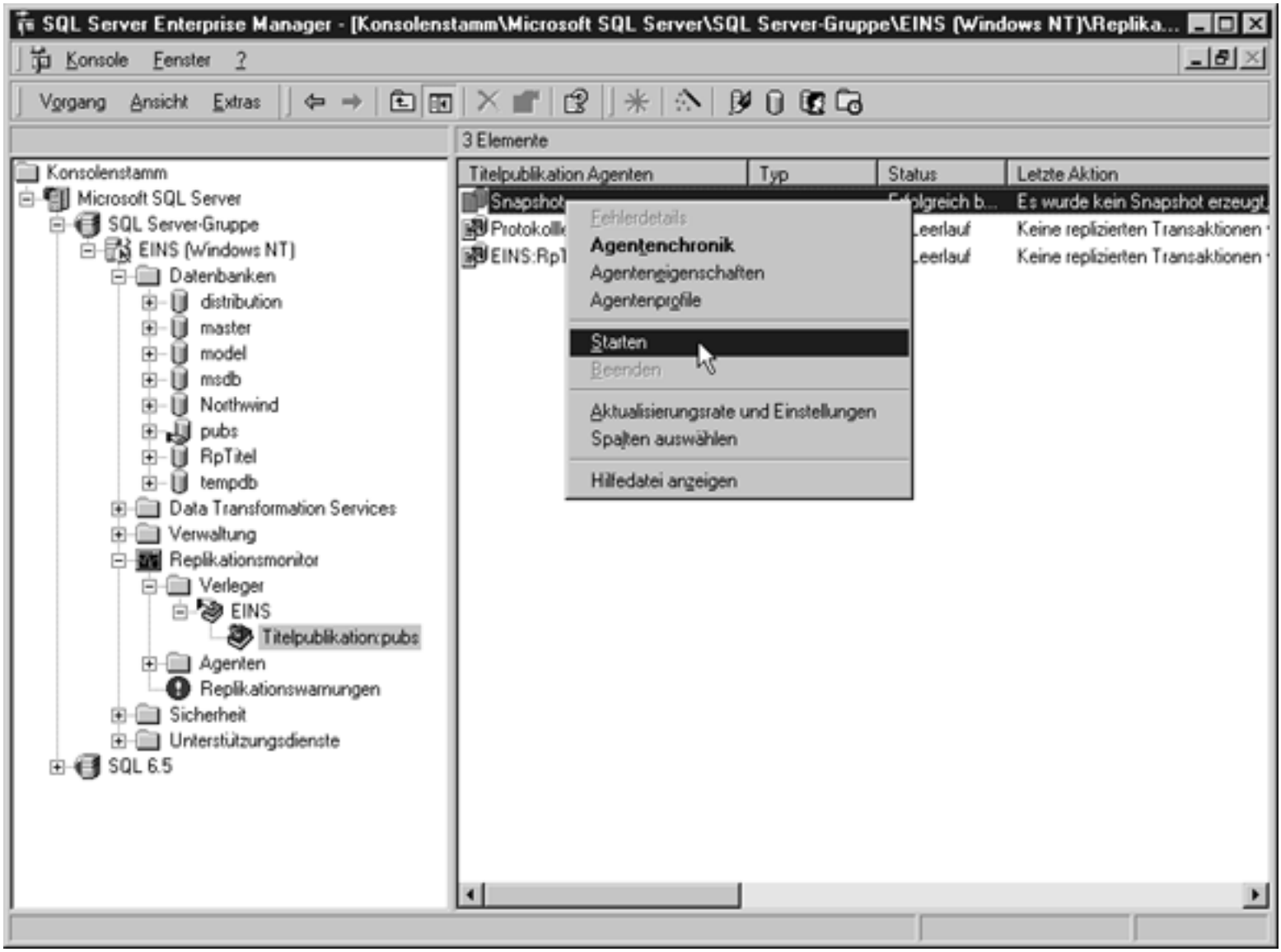

Publikationserstellungs-Assistent für Datenbank 'pubs'
Publikationstyp auswählen
Wählen Sie den Publikationstyp aus, der die Anforderungen Ihrer Anwendung am besten erfüllt.
Wählen Sie einen Publikationstyp aus:
Snapshotpublikation
Der Verleger ersetzt die Daten der Abonnenten regelmäßig durch einen aktualisierten Snapshot. Dies bietet sich an, wenn die Daten der Abonnenten nicht ständig auf dem neuesten Stand sein müssen.
Transaktionspublikation
Daten werden in der Regel beim Verleger aktualisiert. Änderungen werden inkrementell an die Abonnenten gesendet. Auf diese Weise wird die Transaktionskonsistenz garantiert, es erfordert jedoch mehr Administration und eine höhere Konnektivität.
Mergepublikation
Daten können beim Verleger und bei jedem Abonnenten aktualisiert werden. Änderungen werden regelmäßig beim Verleger zusammengeführt. Auf diese Weise werden mobile, nur gelegentlich angeschlossene Abonnenten unterstützt.
< Zurück
Weiter >
Abbrechen
Hilfe

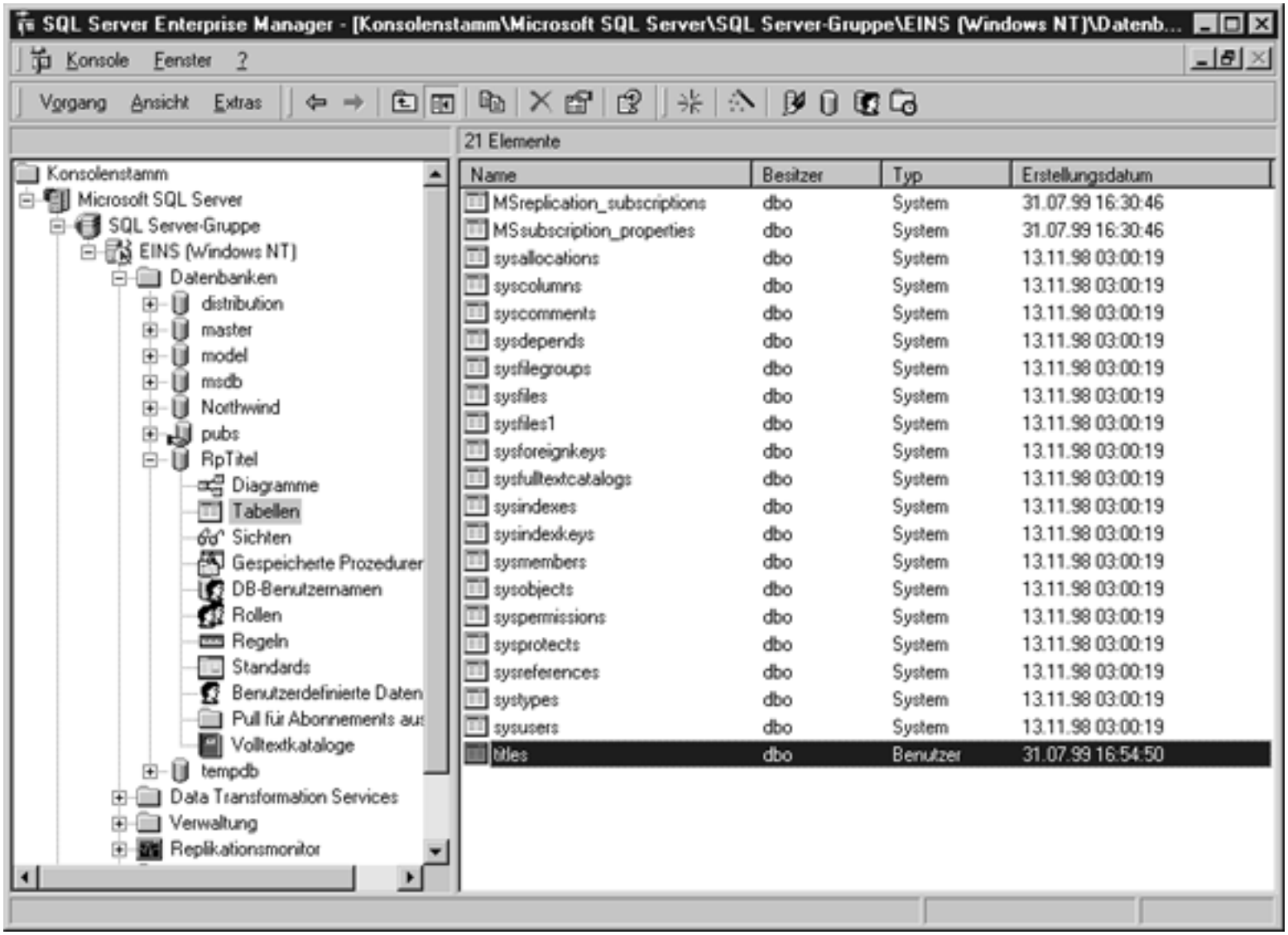

Publikationserstellungs-Assistent für Datenbank 'pubs'
Artikel angeben
Publizieren Sie Tabellen (in einer Snapshot- und Transaktionspublikation auch gespeicherte Prozeduren) als Artikel. Später können Sie die publizierten Daten im Assistenten filtern.
Sie können nur Tabellen mit Primärschlüsseln in einer Transaktionspublikation publizieren.
Besitzer
Tabelle
Artikel
dbo discounts
dbo employee
dbo jobs
dbo pub_info
dbo publishers
dbo roysched
dbo sales
dbo stores
dbo titleauthor
dbo titles titles
Liste filtern
Publizierte und nicht publizierte Objekte
Nur publizierte Objekte
Objekte in der Liste
Tabellen einschließen
Gespeicherte Prozeduren einschließen
Alle publizieren
Keine publizieren
< Zurück
Weiter >
Abbrechen
Hilfe

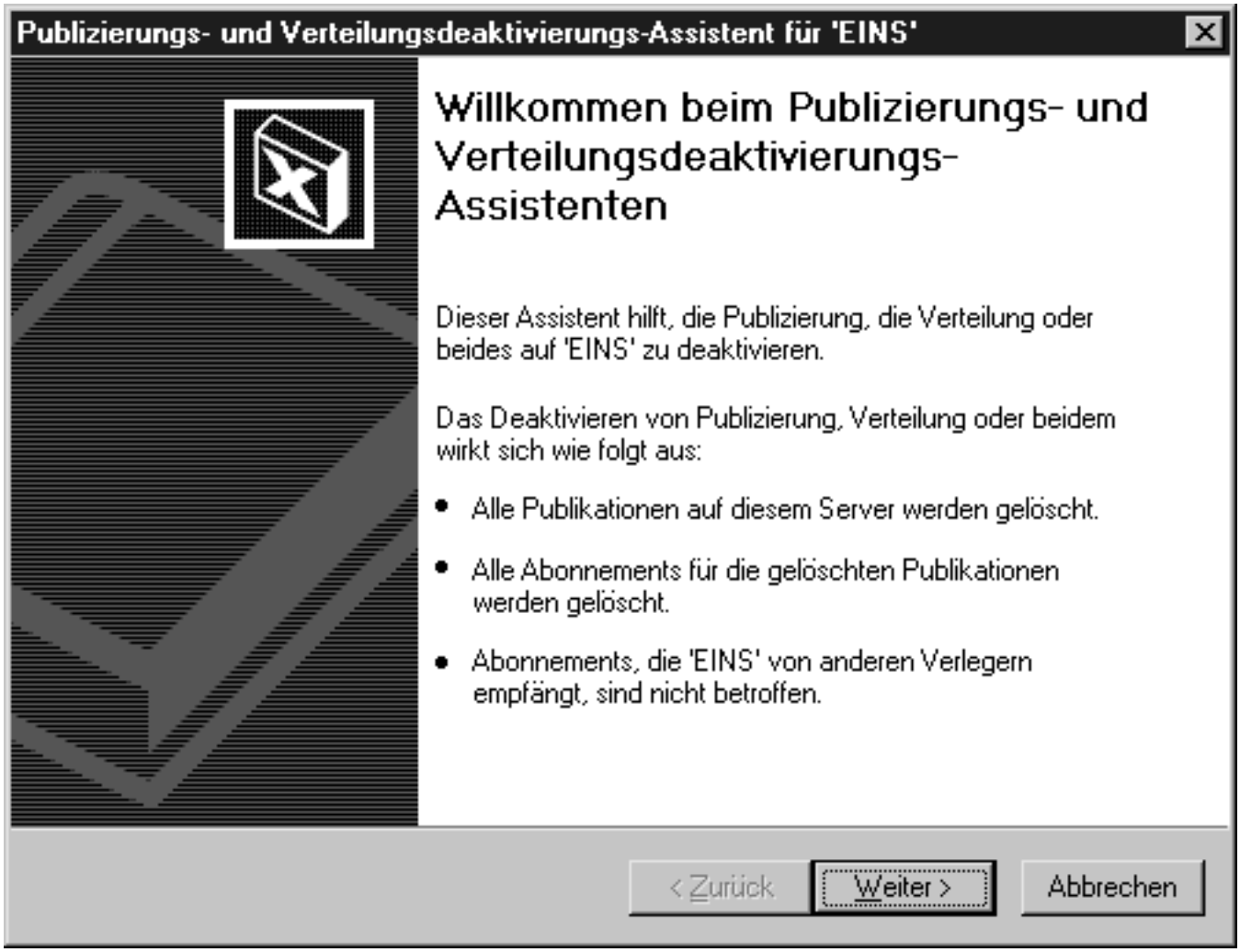

Eigenschaften von titles
Allgemein
Befehle
Snapshot
Wenn auf den Abonnenten Transaktionen angewendet werden, können bestimmte Befehle durch gespeicherte Prozeduren außer Kraft gesetzt werden, so dass benutzerdefinierte Aktionen anstelle oder zusätzlich zu der SQL-Anweisung ausgeführt werden.
INSERT-Befehle ersetzen durch
diesen Aufruf einer gespeicherten Prozedur:
CALL sp_MSins_titles
UPDATE-Befehle ersetzen durch
diesen Aufruf einer gespeicherten Prozedur:
MCALL sp_MSupd_titles
DELETE-Befehle ersetzen durch
diesen Aufruf einer gespeicherten Prozedur:
CALL sp_MSdel_titles
Die oben angegebenen, gespeicherten Prozeduren während der Erstsynchronisation der Abonnenten erstellen
Parameter der gespeicherten Prozedur im Binärformat senden.
Oben nicht ersetzte INSERT-, UPDATE- und DELETE-Befehle wie folgt behandeln:
Spaltennamen in den SQL-Anweisungen verwenden
OK
Abbrechen
Hilfe


Publikationserstellungs-Assistent für Datenbank 'pubs'
Standardeigenschaften der Publikation verwenden
Definieren Sie Datenfilter, oder passen Sie die verbleibenden Publikationseigenschaften an; andernfalls wird die Publikation wie angegeben erstellt.
Möchten Sie Datenfilter definieren oder die verbleibenden Eigenschaften dieser Publikation anpassen?
Ja, Datenfilter definieren, anonyme Abonnenten aktivieren oder andere Eigenschaften anpassen.
Nein, Publikation ohne Datenfilter und mit den folgenden Eigenschaften erstellen:
Eine Transaktionspublikation von Datenbank 'pubs' erstellen.
Alle Abonnenten sind Microsoft SQL Server.
Die folgenden Tabellen als Artikel publizieren:
titles als titles
Diese Publikation hat den Namen 'Titelpublikation'. Ihre Beschreibung lautet: 'Transaktionspublikation der pubs-Datenbank von Verleger EINS.'.
< Zurück
Weiter >
Abbrechen
Hilfe


Publikationserstellungs-Assistent für Datenbank 'pubs'
Publikationserstellungs-Assistenten beenden
Klicken Sie auf 'Fertig stellen', um die Publikation mit den folgenden Optionen zu erstellen:
Eine Transaktionspublikation von Datenbank 'pubs' erstellen.
Alle Abonnenten sind Microsoft SQL Server.
Die folgenden Tabellen als Artikel publizieren:
titles als titles
Diese Publikation hat den Namen 'Titelpublikation'. Ihre
Um die Replikationsaktivitäten für diesen Verleger zu überwachen, stellen Sie eine Verbindung zu dem Verteiler her, und erweitern Sie den Replikationsmonitor.
< Zurück
Fertig stellen
Abbrechen
Hilfe


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Replika...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
3 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
distribution
master
model
msdb
Northwind
pubs
RpTitel
tempdb
Data Transformation Services
Verwaltung
Replikationsmonitor
Verleger
EINS
Titelpublikation:pubs
Agenten
Replikationswarnungen
Sicherheit
Unterstützungsdienste
SQL 6.5
Titelpublikation Agenten
Typ
Status
Letzte Aktion
Snapshot
Protokoll
EINS:RpT
Leerlauf
Leerlauf
Keine replizierten Transaktionen
Keine replizierten Transaktionen
Fehlerdetails
Agentenchronik
Agenteneigenschaften
Agentenprofile
Starten
Beenden
Aktualisierungsrate und Einstellungen
Spalten auswählen
Hilfedatei anzeigen


SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Datenb...
Konsole Fenster ?
Vorgang Ansicht Extras
21 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
distribution
master
model
msdb
Northwind
pubs
RpTitel
Diagramme
Tabellen
Sichten
Gespeicherte Prozedurer
DB-Benutzernamen
Rollen
Regeln
Standards
Benutzerdefinierte Daten
Pull für Abonnements aus
Volltextkataloge
tempdb
Data Transformation Services
Verwaltung
Replikationsmonitor
Name
Besitzer
Typ
Erstellungsdatum
MSreplication_subscriptions dbo System 31.07.99 16:30:46
MSsubscription_properties dbo System 31.07.99 16:30:46
sysallocations dbo System 13.11.98 03:00:19
syscolumns dbo System 13.11.98 03:00:19
syscomments dbo System 13.11.98 03:00:19
sysdepends dbo System 13.11.98 03:00:19
sysfilegroups dbo System 13.11.98 03:00:19
sysfiles dbo System 13.11.98 03:00:19
sysfiles1 dbo System 13.11.98 03:00:19
sysforeignkeys dbo System 13.11.98 03:00:19
sysfulltextcatalogs dbo System 13.11.98 03:00:19
sysindexes dbo System 13.11.98 03:00:19
sysindexkeys dbo System 13.11.98 03:00:19
sysmembers dbo System 13.11.98 03:00:19
sysobjects dbo System 13.11.98 03:00:19
syspermissions dbo System 13.11.98 03:00:19
sysprotects dbo System 13.11.98 03:00:19
sysreferences dbo System 13.11.98 03:00:19
systypes dbo System 13.11.98 03:00:19
sysusers dbo System 13.11.98 03:00:19
titles dbo Benutzer 31.07.99 16:54:50


Publizierungs- und Verteilungsdeaktivierungs-Assistent für 'EINS'
Willkommen beim Publizierungs- und Verteilungsdeaktivierungs-Assistenten
Dieser Assistent hilft, die Publizierung, die Verteilung oder beides auf 'EINS' zu deaktivieren.
Das Deaktivieren von Publizierung, Verteilung oder beidem wirkt sich wie folgt aus:
Alle Publikationen auf diesem Server werden gelöscht.
Alle Abonnements für die gelöschten Publikationen werden gelöscht.
Abonnements, die 'EINS' von anderen Verlegern empfängt, sind nicht betroffen.
< Zurück
Weiter >
Abbrechen

kapitel 23 sql server-agent

23.1 es gibt viel zu tun

ein datenbankadministrator hat oftmals zeitaufwendige, aber fast immer gleichbleibende routineaufgaben zu erledigen. dazu gehören zum beispiel datenbanksicherungen oder regelmäßige aktualisierungen. diese aufgaben bieten sich für eine automatisierung an. der sql server-agent stellt dabei das werkzeug dar, mit dem sich regelmäßig wiederkehrende aktivitäten auf microsoft sql server planen oder mitteilungen an die systemadministratoren bei problemen mit dem server auslösen lassen.

23.2 komponenten des sql server-agenten

der dienst sql server-agent hat seine wurzeln in der sql executive der versionen 6.x von sql server, verfügt aber über eine wesentlich erweiterte funktionalität, die mit folgenden komponenten realisiert wird:

- operatoren
- aufträge
- warnungen

um die komponenten des sql server-agenten im enterprise manager anzuzeigen, erweitern sie die konsolenstruktur des servers bis zum ordner **verwaltung**. klicken sie auf diesen ordner, so daß die einträge im detailbereich erscheinen (siehe abbildung 23.1).

bild

abbildung 23.1: die komponenten des sql server-agenten im enterprise manager

die definitionen für aufträge, warnungen und operatoren legt sql server in der systemdatenbank msdb ab. der dienst sql server-agent fragt beim start die tabellen in der datenbank msdb ab, um eventuell anstehende aufträge oder warnungen auszulösen. sql server reicht alle ereignisse an den sql server-agenten weiter, der diese verarbeitet und daraus warnungen erzeugt oder geplante aufträge termingerecht ausführt.

wenn sie den vollen leistungsumfang des sql server-agenten nutzen wollen, sollten sie zunächst einen oder mehrere operatoren festlegen, die benachrichtigungen über den status von aufträgen erhalten sollen.

23.3 operatoren

bei programmiersprachen und der programmierung von datenbanken im speziellen denkt man beim begriff *operatoren* unwillkürlich an die symbole oder schlüsselwörter, die ausdrücke verknüpfen oder spezielle aktionen bewirken. im sinne des sql server-agenten handelt es sich aber um personen, die

warnungen und benachrichtigungen erhalten - sozusagen den bereitschaftsdienst für sql server. bei problemen mit sql server sind aktionen zur fehlerbehebung einzuleiten. dazu können operatoren, die sich anhand ihres netzwerkkontos oder ihrer e-mail-adresse identifizieren lassen, gerufen werden. das kann per e-mail, pager oder net send erfolgen.

23.3.1 operatoren einrichten

im enterprise manager richten sie einen operator in folgenden schritten ein:

1. erweitern sie die konsolenstruktur bis zum zweig **sql server-agent**, und markieren sie den eintrag **operatoren**. klicken sie mit der rechten maustaste auf diesen eintrag, und wählen sie aus dem kontextmenü den befehl **neuer operator**.
2. im dialogfeld **neue operatoreigenschaften** (siehe abbildung 23.2) geben sie den namen des operators ein und stellen - falls vorhanden - die adressen für e-mail, pager und den befehl net send bereit.

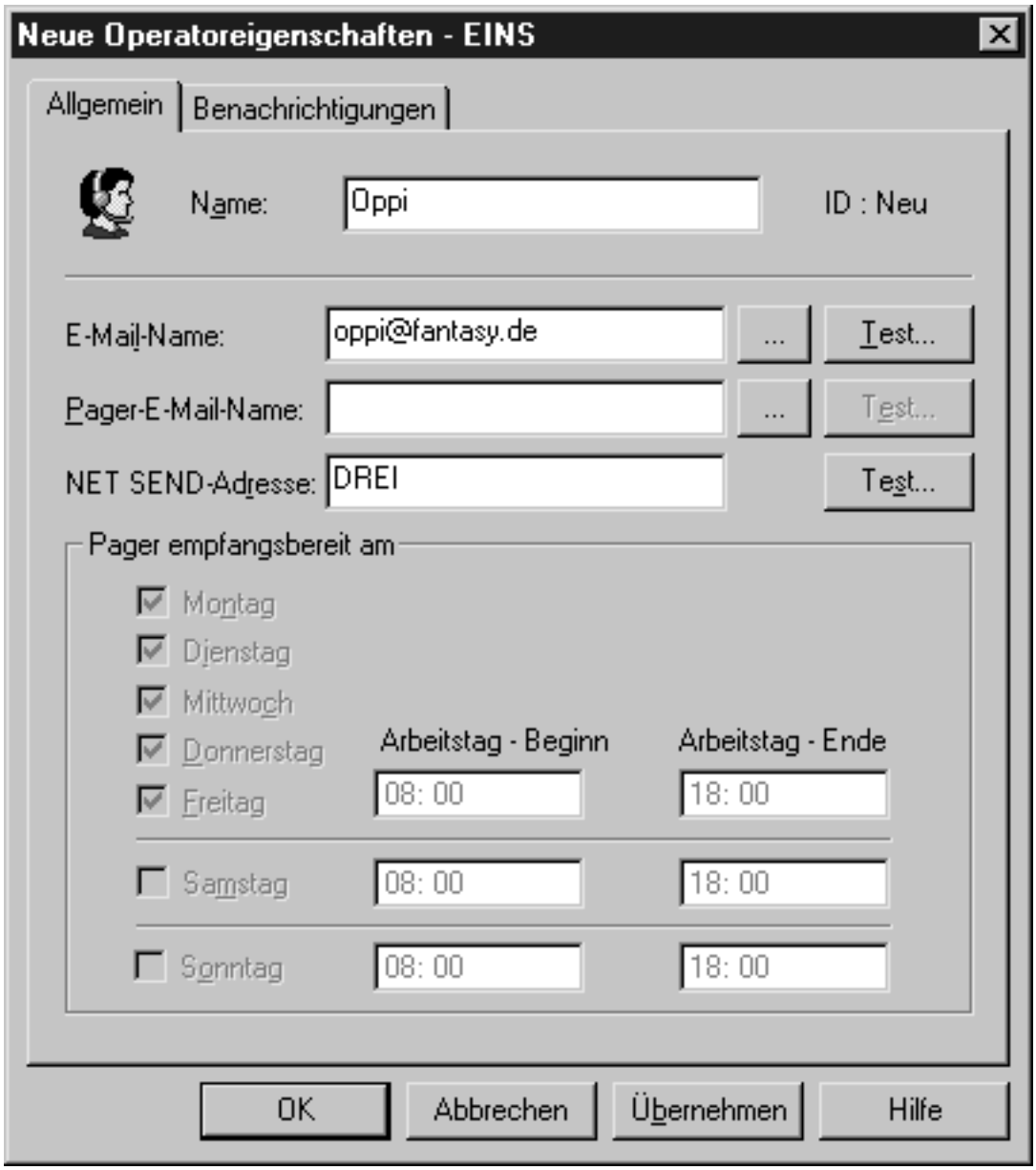


abbildung 23.2: hier geben sie den namen und die empfangsadressen des operators an

die benachrichtigung des operators über net send muß nicht unbedingt an einen unter windows 95/98 oder windows nt laufenden computer erfolgen. die installation von sql server auf dem zielcomputer ist ebenfalls nicht erforderlich. im beispiel handelt es sich bei computer drei um einen 386er (ja, die gibt es noch), der unter windows for workgroups 3.11 läuft.

wenn sie im feld **net send-adresse** einen computernamen eingeben und auf die schaltfläche **test** klicken, erscheint die in abbildung 23.3 gezeigte meldung.

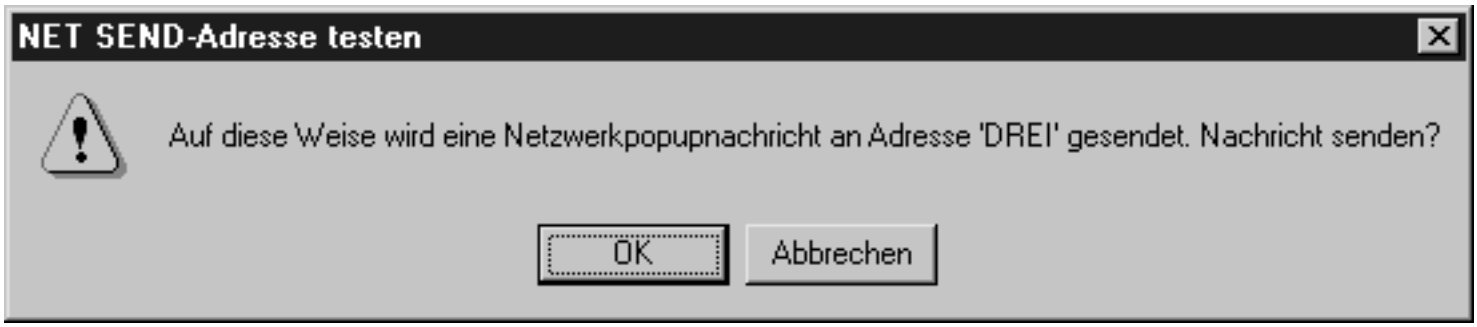


abbildung 23.3: hinweis auf das senden einer testnachricht

klicken sie auf **ok**, um die testnachricht abzuschicken. nach kurzer zeit erhalten sie auf dem zielcomputer die folgende popup-nachricht:

```
test der operatorbenachrichtigung über
netzwerkpopupnachricht - bitte ignorieren
```

damit ist der test der benachrichtigung via net send erfolgreich abgeschlossen.

3. auf der registerkarte **benachrichtigungen** (siehe abbildung 23.4) legen sie fest, bei welchen ereignissen und auf welchem weg der operator zu benachrichtigen ist. im unteren teil dieses dialogfelds ist angegeben, wann die letzten benachrichtigungen des operators erfolgt sind.

bild

abbildung 23.4: hier legen sie die art der warnungen fest, über die der operator zu informieren ist

4. klicken sie auf **übernehmen** und dann auf **ok**, um den operator in sql server einzubinden.

wenn sie in der konsolenstruktur auf den eintrag **operatoren** klicken, ist der eben angelegte operator im detailbereich zu sehen.

23.4 aufträge

unter aufträgen versteht man operationen, die zu festgelegten zeitpunkten automatisch ablaufen. praktisch sind das transact-sql-anweisungen, die sql server entsprechend dem vom sql server-agenten überwachten terminplan ausführt.

23.4.1 aufträge anzeigen und bearbeiten

in kapitel 20 haben sie mit dem datenbankwartungsplanungs-assistenten den wartungsplan für eine datenbanksicherung erstellt. der assistent hat aus dem von ihnen festgelegten wartungsplan verschiedene aufträge gemacht. um diese aufträge anzuzeigen, erweitern sie die konsolenstruktur bis zum ordner **verwaltung**, erweitern den ordner **verwaltung** und dann den eintrag **sql server-agent**.

einen auftrag können sie anzeigen und bearbeiten, indem sie mit der rechten maustaste im detailbereich des enterprise managers auf den jeweiligen auftrag klicken und **eigenschaften** aus dem kontextmenü wählen. abbildung 23.5 zeigt als beispiel das eigenschaftsdialogfeld für den in kapitel 20 erstellten sicherungsauftrag.

bild

abbildung 23.5: eigenschaftsdialogfeld für einen sicherungsauftrag

auf der registerkarte **allgemein** legen sie im feld **name** den namen des auftrags fest. das kontrollkästchen **aktiviert** ist bereits eingeschaltet, da ein auftrag beim erstellen automatisch aktiviert wird. wenn ein auftrag deaktiviert ist, führt ihn der sql server-agent nicht aus. allerdings können sie einen deaktivierten auftrag auch manuell starten. die optionen *ziel: lokaler server* bzw. *ziel: mehrere server* legen fest, wo der auftrag auszuführen ist. die zweite option ist nur auf einem registrierten masterserver verfügbar.

interessanter wird es auf der registerkarte **schritte** (siehe abbildung 23.6). zunächst einmal ist festzustellen, daß sich ein auftrag aus mehreren schritten zusammensetzen kann (auch wenn es im beispiel momentan nur ein schritt ist).

bild

abbildung 23.6: die registerkarte schritte

mit den schaltflächen im unteren teil des dialogfelds können sie die reihenfolge der schritte ändern, neue schritte erstellen oder einfügen, vorhandene schritte bearbeiten und schritte löschen.

im drop-down-listenfeld **schritt starten** legen sie fest, mit welchem schritt der auftrag zu beginnen ist.

wenn sie sehen wollen, wie ein schritt überhaupt definiert ist, markieren sie einen schritt und klicken auf **bearbeiten**. daraufhin wird das dialogfeld **schritt bearbeiten** wie in abbildung 23.7 geöffnet.

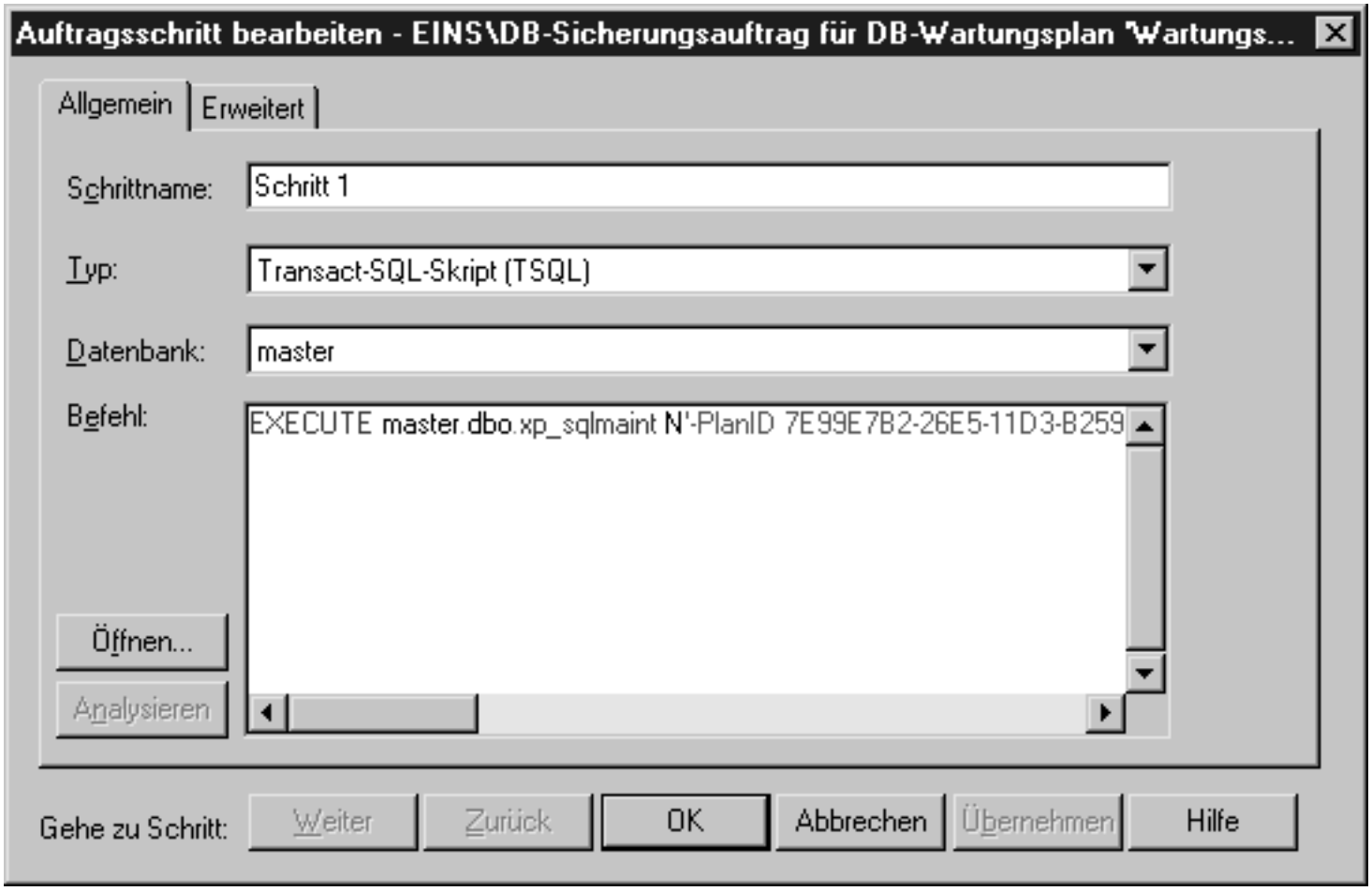


abbildung 23.7: das dialogfeld schritt bearbeiten

ein blick in das feld **befehl** zeigt, daß der datenbankwartungsplanungs-assistent einen execute-befehl für die erweiterte gespeicherte prozedur xp_sqlmaint erzeugt hat. diese prozedur ruft das dienstprogramm sqlmaint mit den angegebenen parametern auf. die kryptische zahl nach dem parameter planid ist ein vom datenbankwartungsplanungs-assistenten definierter global eindeutiger bezeichner (guid). eine erläuterung der parameter finden sie bei der beschreibung des dienstprogramms sqlmaint in kapitel 20.

nehmen wir als beispiel an, daß sie im datenbankwartungsplan noch keine konsistenzprüfung für die datenbank master vorgesehen haben und diesen schritt in den auftrag einfügen möchten. da diese prüfung vor ausführung des wartungsplans stattfinden soll, wählen sie auf der registerkarte **schritte** die schaltfläche **einfügen**. daraufhin wird das dialogfeld **neuer auftragsschritt** geöffnet (siehe abbildung 23.8).

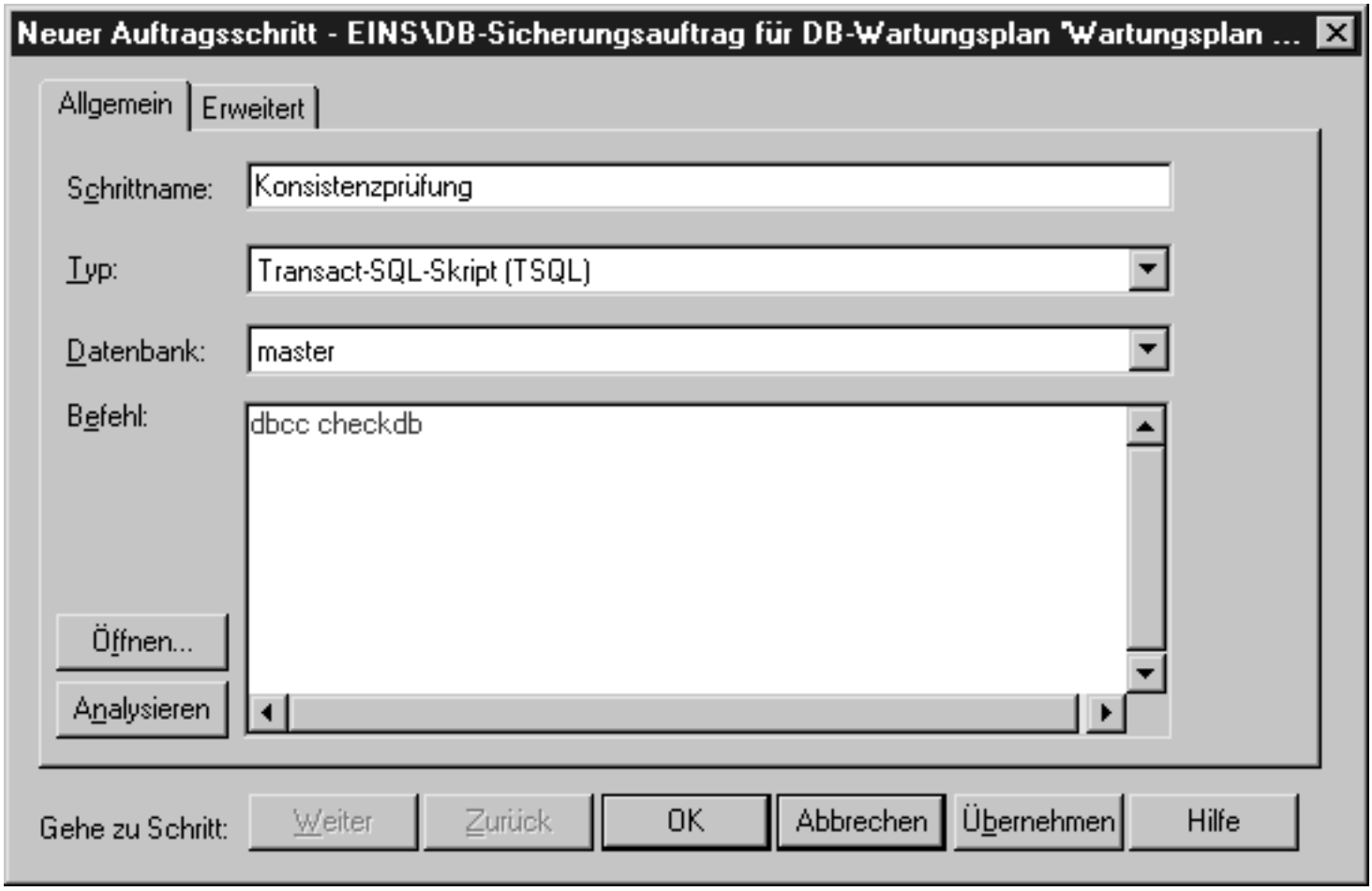


abbildung 23.8: in diesem dialogfeld spezifizieren sie einen neuen auftragsschritt

geben sie im bearbeitungsfeld **schrittname** zum beispiel konsistenzprüfung ein. als typ übernehmen sie die voreinstellung *transact-sql-skript*. im drop-down-listenfeld **datenbank** wählen sie die datenbank aus, auf die sich der auftrag beziehen soll, im beispiel die datenbank master. die für den auftragsschritt auszuführenden befehle geben sie in das bearbeitungsfeld **befehl** ein. für dieses einfache beispiel soll der befehl dbcc checkdb zur integritätsprüfung der datenbank genügen. vor allem bei umfangreicheren befehlsfolgen empfiehlt es sich, die syntax zu überprüfen. klicken sie dazu auf die schaltfläche **analysieren**. bei einer fehlermeldung können sie die befehlsfolge korrigieren und erneut testen.

da dieser auftrag aus mehreren schritten besteht, legen sie auf der registerkarte **erweitert** (siehe abbildung 23.9) fest, welcher schritt als nächstes auszuführen ist, wenn der aktuelle schritt erfolgreich bzw. mit fehlern ausgeführt wurde.

bild

abbildung 23.9: nächsten schritt nach erfolg oder mißerfolg des aktuellen schrittes festlegen

klicken sie auf **übernehmen** und dann auf **ok**, um den neuen auftragsschritt in den auftrag zu übernehmen. der eingefügte schritt hat den bisherigen schritt und damit auch den startpunkt (angezeigt durch eine grüne flagge) nach unten verschoben. der schritt befindet sich zwar an der richtigen position, der start soll aber mit der konsistenzprüfung beginnen. klicken sie auf den drop-down-pfeil im feld **schritt starten**, und markieren sie den eintrag *(1) konsistenzprüfung*. damit der auftrag verständlicher wird, können sie gleich noch den zweiten schritt umbenennen. dazu markieren sie den schritt, klikken auf

bearbeiten und tragen einfach einen neuen namen - im beispiel wartungsplan - ein. das dialogfeld **sicherungsauftrag** hat nun das in abbildung 23.10 gezeigte aussehen.

bild

abbildung 23.10: der bearbeitete auftrag

auf der registerkarte **terminpläne** (siehe abbildung 23.11) können sie die ausführungszeiten des auftrags bearbeiten, einen neuen terminplan festlegen oder den terminplan löschen. über die schaltfläche **neue warnung** gelangen sie zum gleichnamigen dialogfeld, in dem sie warnungen für bestimmte ereignisse definieren können. darauf geht der entsprechende abschnitt weiter unten in diesem kapitel ein.

bild

abbildung 23.11: die registerkarte terminpläne

auf der registerkarte **benachrichtigungen** (siehe abbildung 23.12) legen sie fest, wer und bei welchem auftragsstatus eine benachrichtigung auf welchem weg erhalten soll.

bild

abbildung 23.12: registerkarte benachrichtigungen

klicken sie abschließend auf **übernehmen** und dann auf **ok**, um die änderungen in den bearbeiteten auftrag einfließen zu lassen.

23.4.2 neuen auftrag erstellen

um einen neuen auftrag zu erstellen, klicken sie in der konsolenstruktur des enterprise managers mit der rechten maustaste auf **aufträge** und wählen aus dem kontextmenü den befehl **neuer auftrag**. im dialogfeld **neuer auftrag** (siehe abbildung 23.13) geben sie einen namen für den neuen auftrag an und wählen eine dazu passende kategorie aus (falls in der liste vorhanden). für den besitzer des auftrags können sie die standardeinstellung übernehmen, da jeder benutzer von sql server einen auftrag erstellen kann.

bild

abbildung 23.13: erstellen eines neuen auftrags

nachdem sie diese angaben bereitgestellt haben, läuft die auftragserstellung nach den gleichen schritten weiter, wie sie der letzte abschnitt »aufträge anzeigen und bearbeiten« erläutert hat.

23.4.3 auftragserstellungs-assistent

wenn sie einen auftrag lieber mit der unterstützung eines assistenten erstellen, öffnen sie das dialogfeld **assistent auswählen**, erweitern den zweig **verwaltung** und wählen den eintrag **auftragserstellungs-assistent**. führen sie dann folgende schritte aus:

1. klicken sie im startdialogfeld auf **weiter**. im nächsten dialogfeld des assistenten (siehe abbildung 23.14) wählen sie die art des auszuführenden befehls. übernehmen sie für das beispiel die

voreinstellung *transact-sql-befehl*.

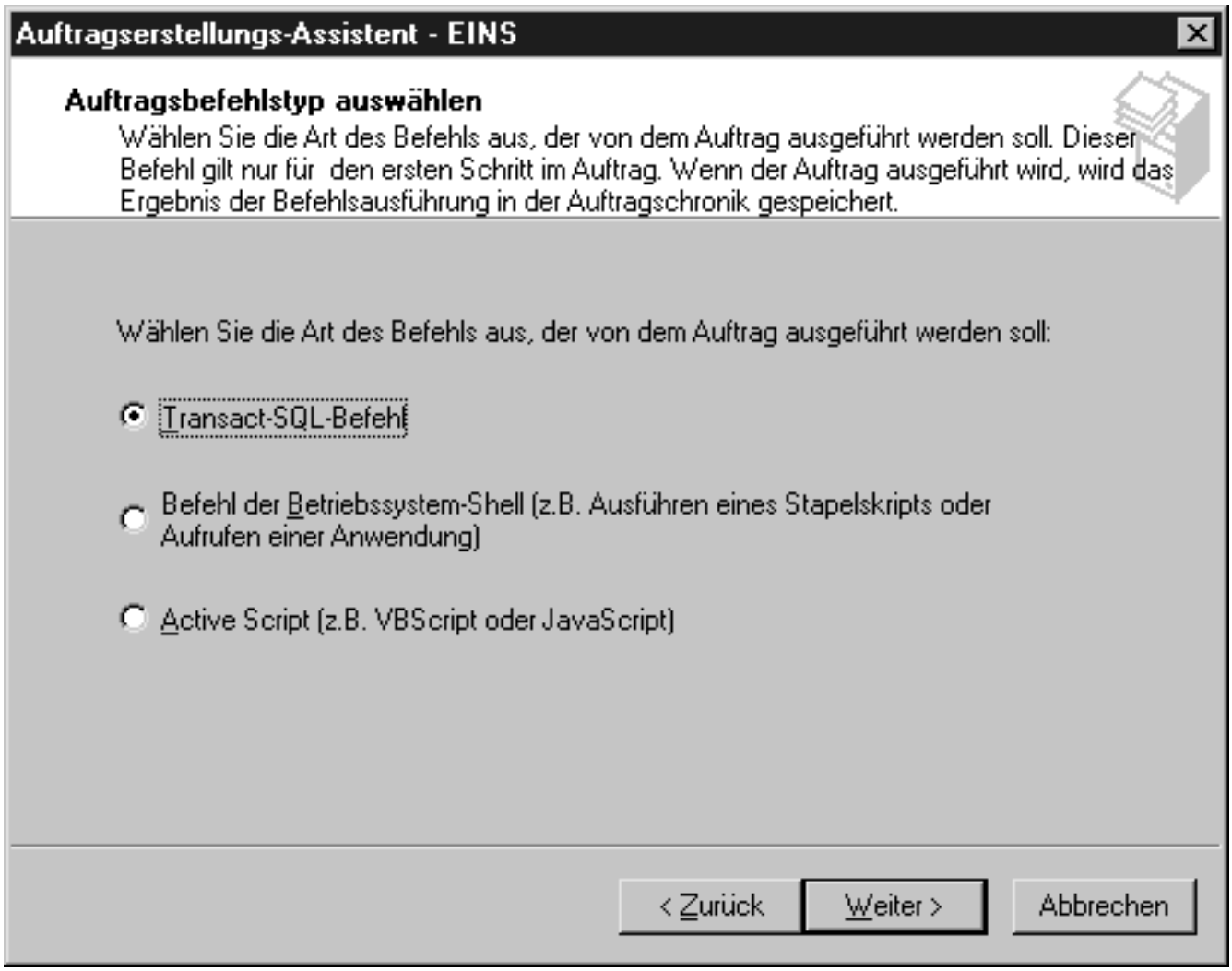


abbildung 23.14: auswahl des befehlstyps

2. im dialogfeld **transact-sql-anweisung eingeben** (siehe abbildung 23.15) wählen sie zunächst die datenbank aus, auf die sich der auftrag beziehen soll, und geben dann die gewünschten transact-sql-anweisungen ein, im beispiel die anweisung dbcc checkdb. klicken sie auf die schaltfläche **analysieren**, um sich von der ordnungsgemäßen syntax der anweisungen zu überzeugen. klicken sie dann auf **weiter**.

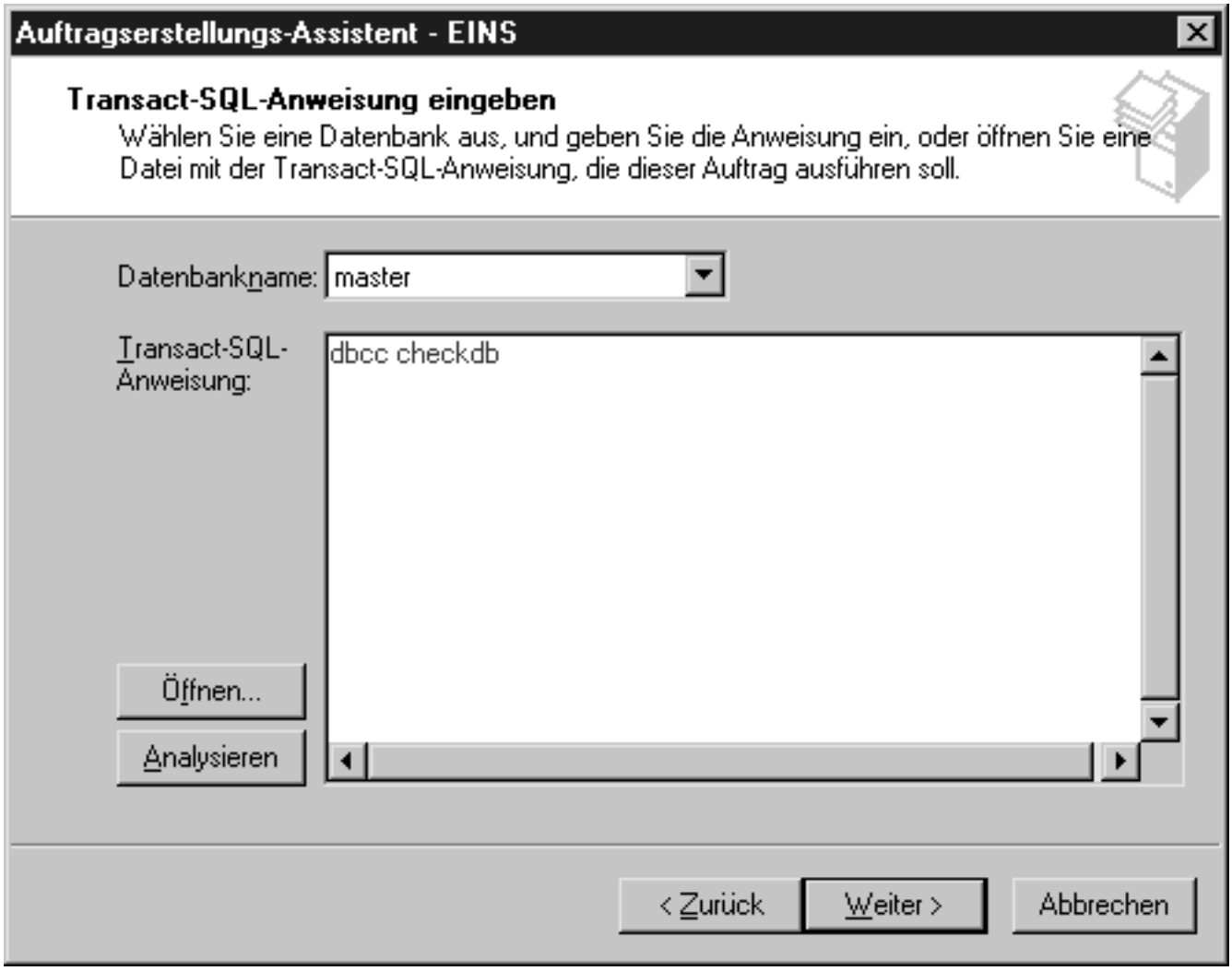


abbildung 23.15: wählen sie die datenbank aus, und geben sie die transact-sql-anweisung(en) ein

3. im dialogfeld **auftragsterminplan angeben** (siehe abbildung 23.16) spezifizieren sie die zeitpunkte, zu denen der auftrag ausgeführt werden soll. klicken sie auf **ändern**, wenn sie den vorgegebenen terminplan an ihre vorstellungen anpassen möchten. klicken sie dann auf **weiter**.

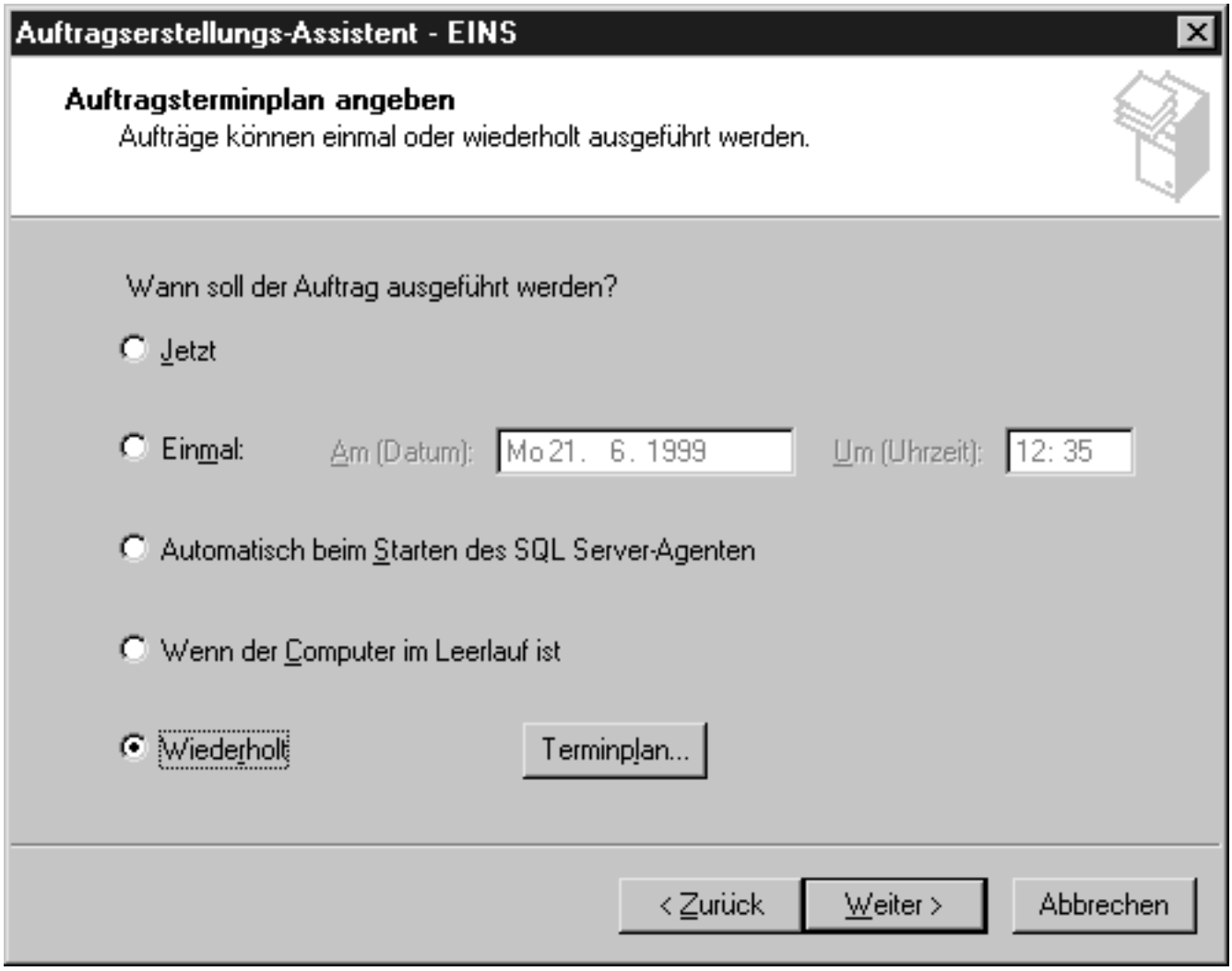


abbildung 23.16: hier legen sie den terminplan für den auftrag fest

4. im nächsten dialogfeld spezifizieren sie bei bedarf die operatoren, die sie über den status des auftrags benachrichtigen wollen. nach klicken auf **weiter** gelangen sie zum letzen dialogfeld des assistenten. hier können sie den vorgegebenen namen des auftrags überschreiben. klicken sie auf **fertigstellen**, um den auftrag erstellen zu lassen.

23.5 warnungen

mit der komponente *warnungen* spezifizieren sie ereignisse innerhalb von sql server, die der sql server-agent überwacht. wenn eines dieser ereignisse eintritt und zu einer warnung führt, wird der dafür vorgesehene operator benachrichtigt.

23.5.1 warnungen konfigurieren

im enterprise manager legen sie die warnungen in folgenden schritten fest:

1. erweitern sie die konsolenstruktur bis zum ordner **sql server-agent**, und markieren sie diesen ordner. klicken sie mit der rechten maustaste auf den eintrag **warnungen**. wählen sie aus dem kontextmenü den befehl **neue warnung**.
2. im dialogfeld **neue warnung** (siehe abbildung 23.17) geben sie zunächst einen namen für die

warnung ein. im kombinationsfeld **typ** können sie zwischen *sql server-ereigniswarnung* und *sql server-leistungsstatuswarnung* wählen. übernehmen sie für das beispiel die voreingestellte erste option, um interne ereignisse von sql server zu überwachen, zum beispiel schweregrade oder bestimmte fehlermeldungen.

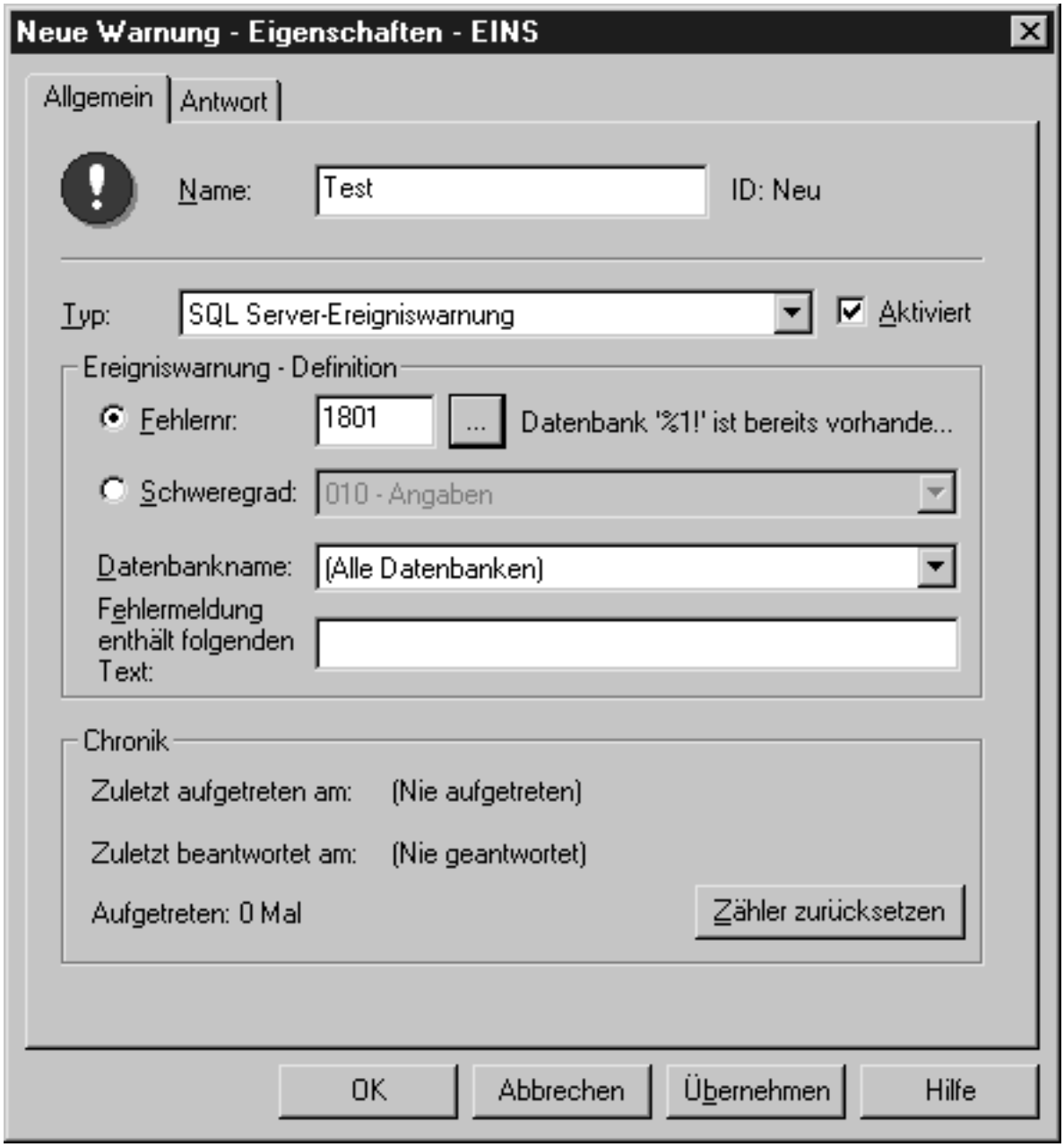


abbildung 23.17: das dialogfeld neue warnung

3. im abschnitt **ereigniswarnung - definition** können sie entweder eine fehlernummer oder einen schweregrad spezifizieren. wählen sie für das beispiel die erste option, und klicken sie auf die schaltfläche mit den drei punkten. damit gelangen sie zum dialogfeld **sql server-meldungen verwalten** (siehe abbildung 23.18). im nachfolgenden testbeispiel soll der operator eine benachrichtigung erhalten, wenn ein benutzer versucht, eine bereits vorhandene datenbank anzulegen. im bearbeitungsfeld **meldungstext enthält** können sie ein stichwort eingeben und meldungen mit diesem stichwort auflisten lassen. geben sie im beispiel das stichwort vorhanden ein, und klicken sie auf die schaltfläche **suchen**.

bild

abbildung 23.18: in diesem dialogfeld können sie nach vordefinierten meldungen suchen

4. die ergebnisse erscheinen auf der registerkarte **meldungen** (siehe abbildung 23.19). die gesuchte meldung hat die fehlernummer 1801. markieren sie die betreffende zeile, und klicken sie auf ok. die fehlernummer erscheint daraufhin auf der registerkarte **allgemein** des dialogfelds **neue warnung**.

bild

abbildung 23.19: die gesuchten meldungen erscheinen auf der registerkarte meldungen

die fehlernummer läßt sich auch über eine anweisung ermitteln, die den gewünschten fehler liefert. das setzt natürlich voraus, daß durch diese operation kein schaden an der datenbank entstehen kann.

5. im feld **datenbankname** können sie eine bestimmte datenbank oder wie im beispiel alle datenbanken auswählen.
6. gehen sie auf die registerkarte **antwort** (siehe abbildung 23.20), um den operator und die art und weise seiner benachrichtigung festzulegen. wenn sie das kontrollkästchen **auftrag ausführen** einschalten, können sie aus der dann verfügbaren liste einen auftrag auswählen, der auf die warnung hin ausgeführt werden soll. im beispiel soll lediglich der operator per net send eine benachrichtigung erhalten.

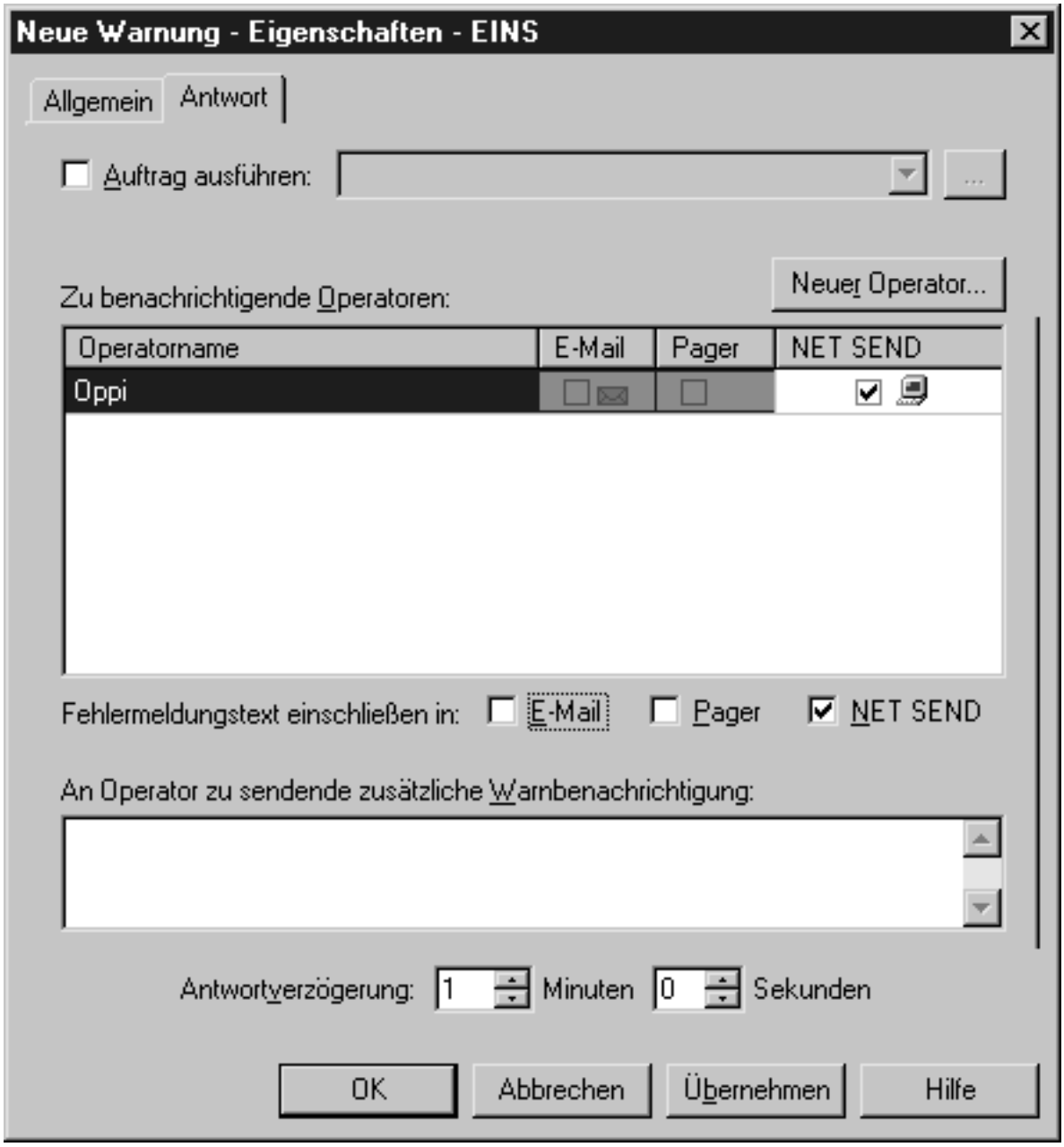


abbildung 23.20: hier legen sie fest, wer die benachrichtigung erhalten soll

7. klicken sie auf **übernehmen**. bestätigen sie den hinweis, daß der fehler 1801 standardmäßig keine warnung auslöst, mit **ja**.

wenn sie diese änderung des meldungsverhaltens nicht wünschen, können sie das beispiel auch mit anderen fehlermeldungen nachvollziehen. beispielsweise bietet sich der fehler 208 an, der bei einem nicht vorhandenen objekt ausgelöst wird.

klicken sie auf **ok**, um die neue warnung in sql server zu übernehmen.

23.5.2 warnungserstellungs-assistent

auch für das erstellen von warnungen hält sql server unterstützung bereit. den warnungserstellungs-assistenten starten sie wie gewohnt über das dialogfeld **assistent auswählen** (zweig *verwaltung*). führen sie dann die folgenden schritte aus:

1. klicken sie im startdialogfeld auf **weiter**. im nächsten dialogfeld (siehe abbildung 23.21)

definieren sie die warnung - wie im letzten beispiel durch die fehlernummer 1801. klicken sie auf **weiter**.

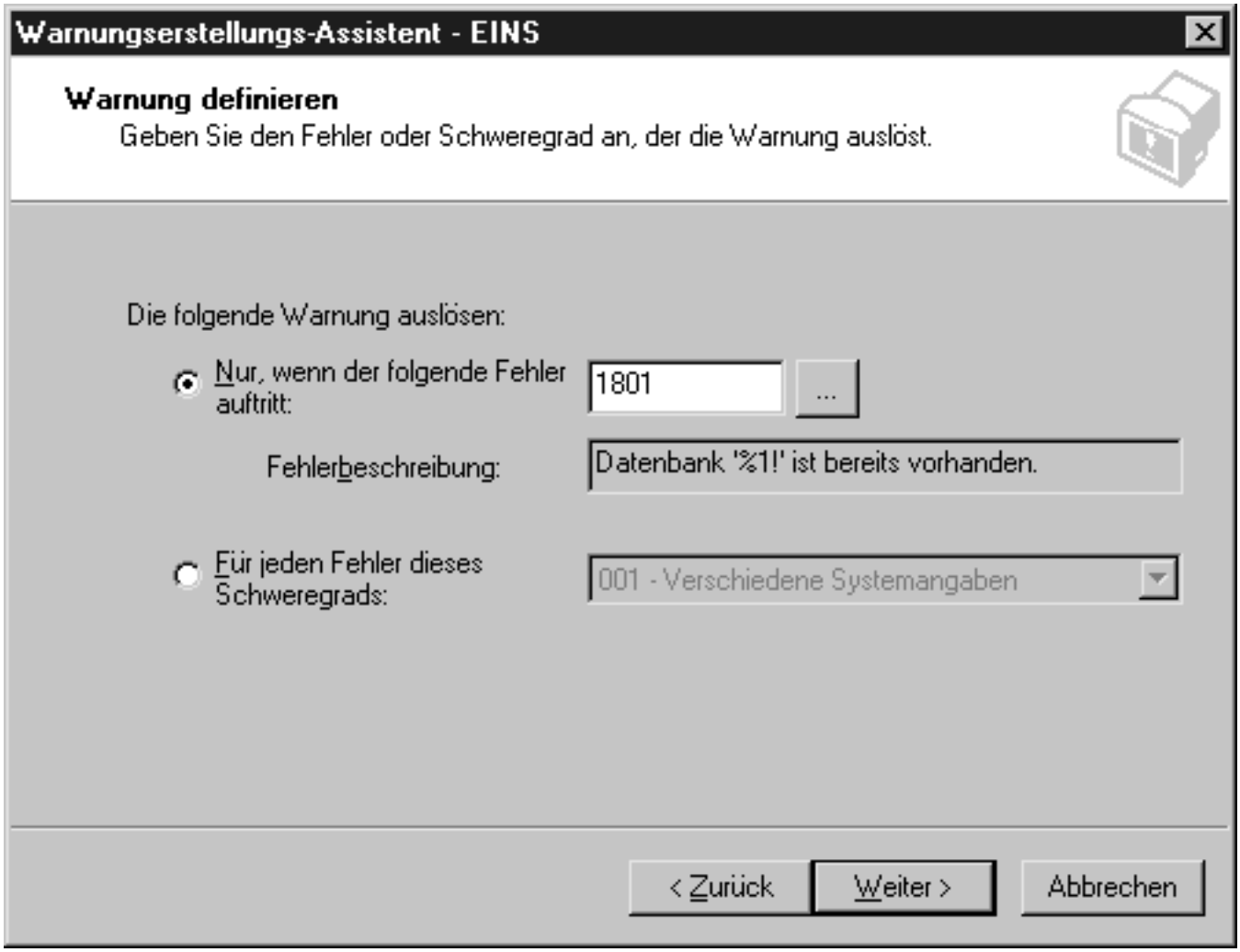


abbildung 23.21: das dialogfeld warnung definieren

2. im nächsten dialogfeld (siehe abbildung 23.22) wählen sie eine bestimmte datenbank oder alle datenbanken aus. klicken sie auf **weiter**.

abbildung 23.22: das dialogfeld datenbank oder fehlerschlüsselwörter angeben

3. das dialogfeld **warnungsantwort** (abbildung 23.23) entspricht der registerkarte **antwort** im dialogfeld **neue warnung**. hier ist wieder das kontrollkästchen in der spalte **net send** markiert, damit der operator am computer drei eine nachricht erhält. klicken sie auf **weiter**.

abbildung 23.23: das dialogfeld warnungsantwort definieren

4. im dialogfeld **warnbenachrichtigung definieren** (siehe abbildung 23.24) können sie dem operator einen zusätzlichen hinweis liefern. klicken sie auf **weiter**.

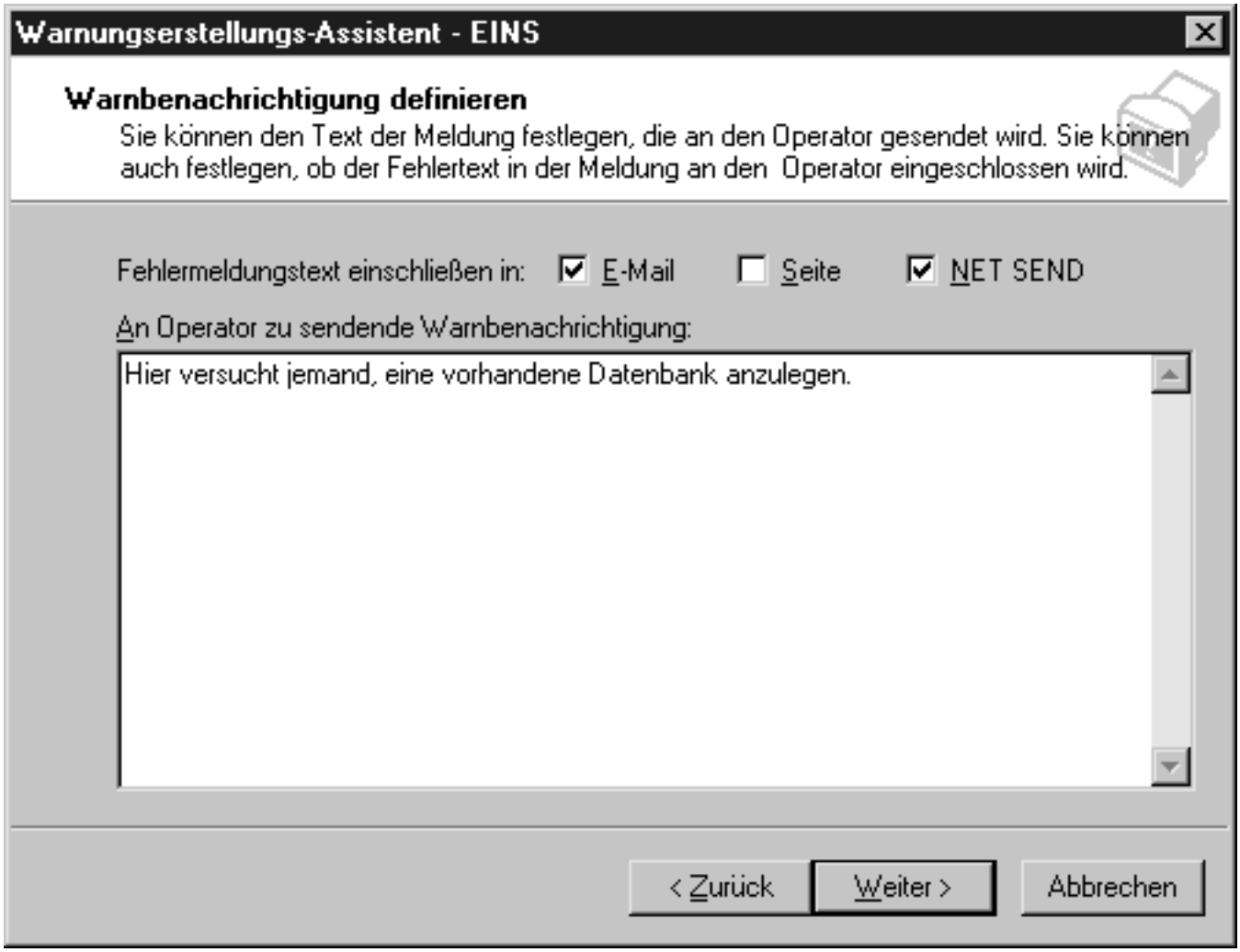


abbildung 23.24: hier können sie eine zusätzliche nachricht an den operator verfassen

5. schon sind sie beim letzten dialogfeld des assistenten angelangt (siehe abbildung 23.25). im feld **warnungsname** können sie jetzt einen neuen titel für die warnung festlegen. klicken sie auf **fertigstellen**, um die warnung in sql server zu übernehmen.

Bild

abbildung 23.25: letztes dialogfeld des warnungserstellungs-assistenten

23.5.3 warnungen testen

probieren sie jetzt aus, ob die im letzten abschnitt erstellte warnung eine benachrichtigung des operators bewirkt. führen sie die folgende anweisung aus:

```
create database pubs
```

auf dem bildschirm erscheint zunächst die meldung:

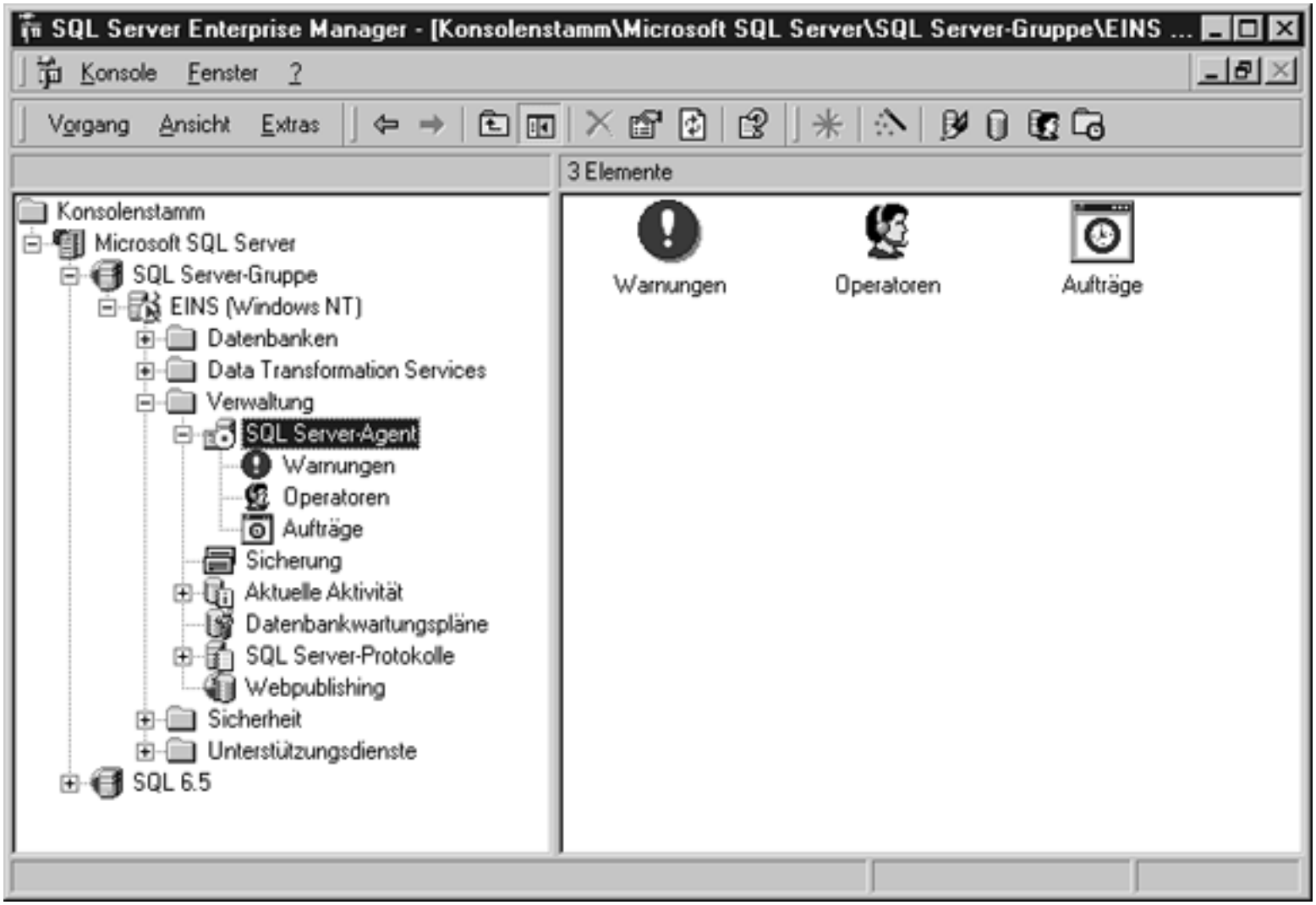
```
server: nachr.-nr. 1801, schweregrad 16, status 3, zeile 1
```

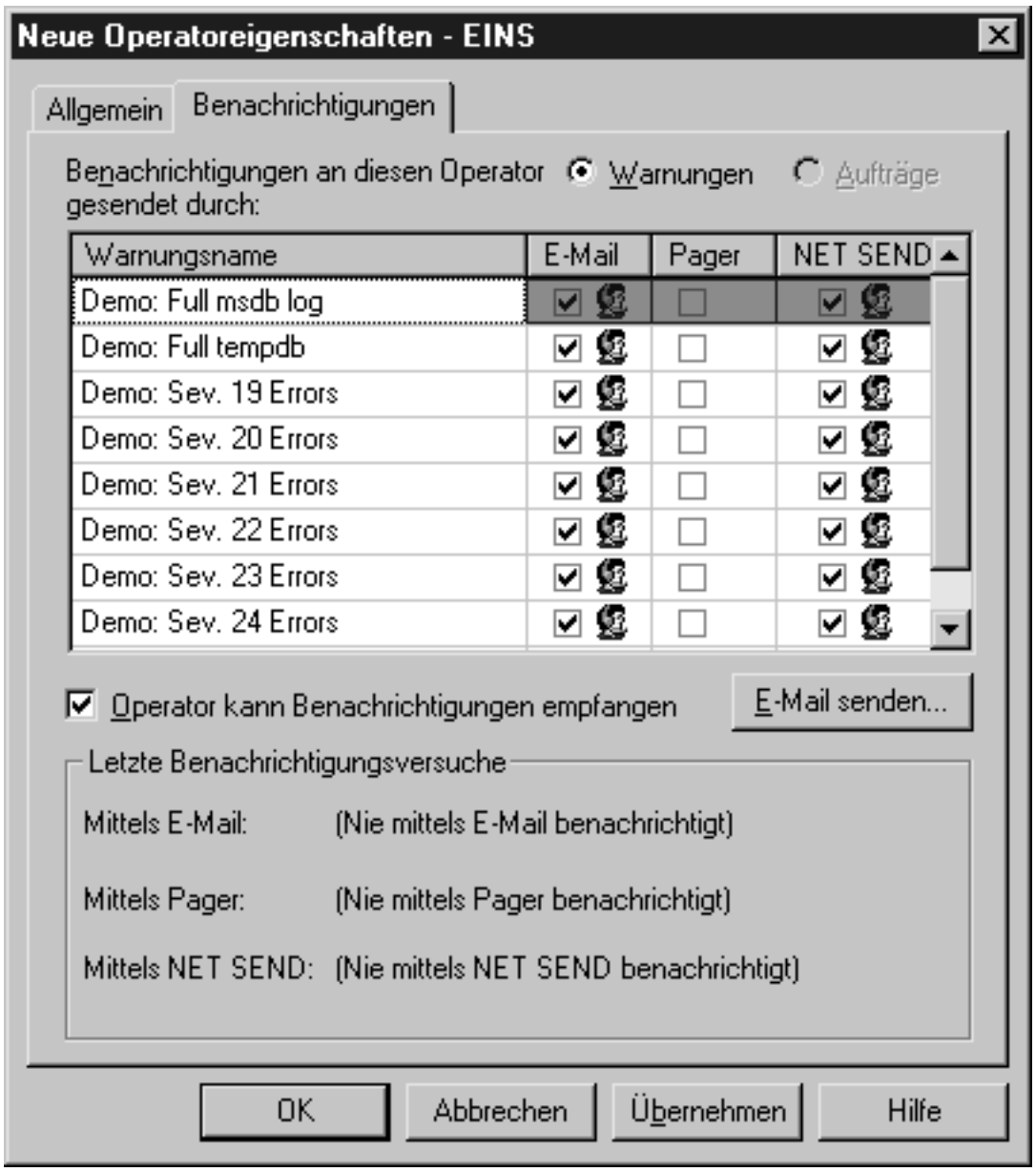
```
datenbank 'pubs' ist bereits vorhanden.
```

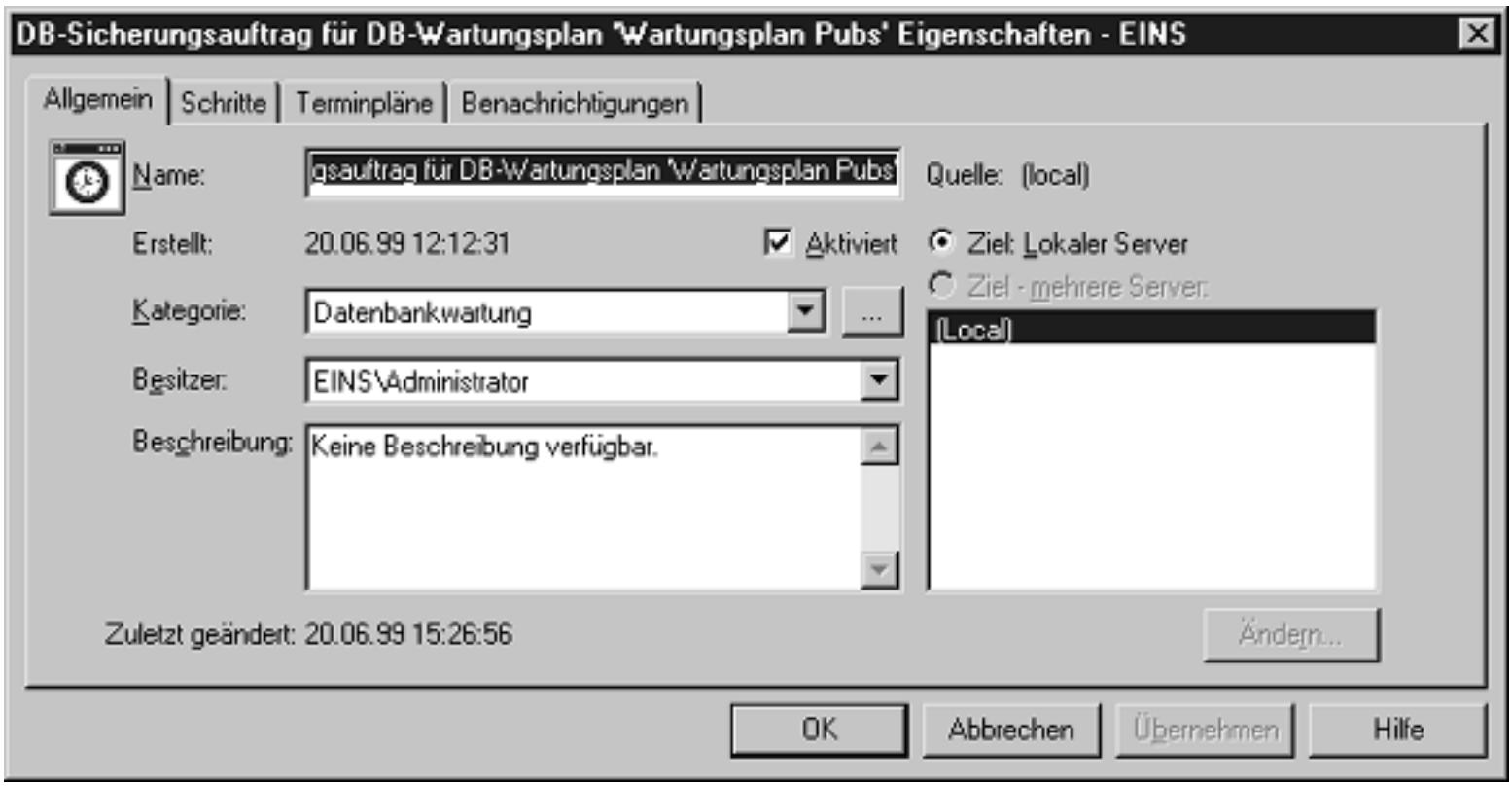
und im winpopup-fenster des computers drei ist folgende nachricht zu sehen:

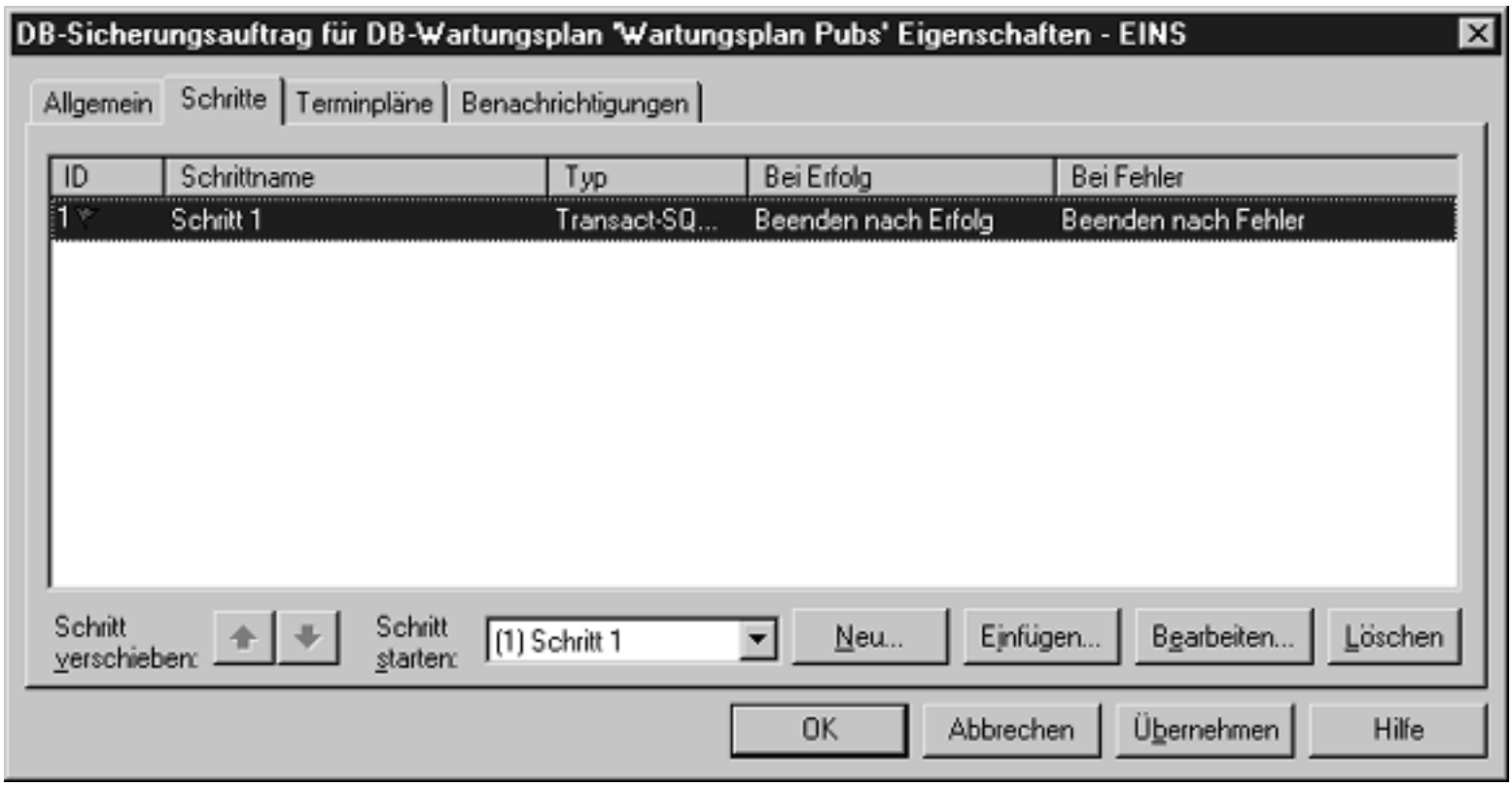
```
sql server-warnungssystem: 'test' aufgetreten auf \\eins

datum/zeit:      21.06.99/15:32:42

beschreibung:    fehler: 1801, schweregrad: 16, status: 3
datenbank 'pubs' ist bereits vorhanden.

kommentar:       (keine)

auftragsausführung:    (keine)
```

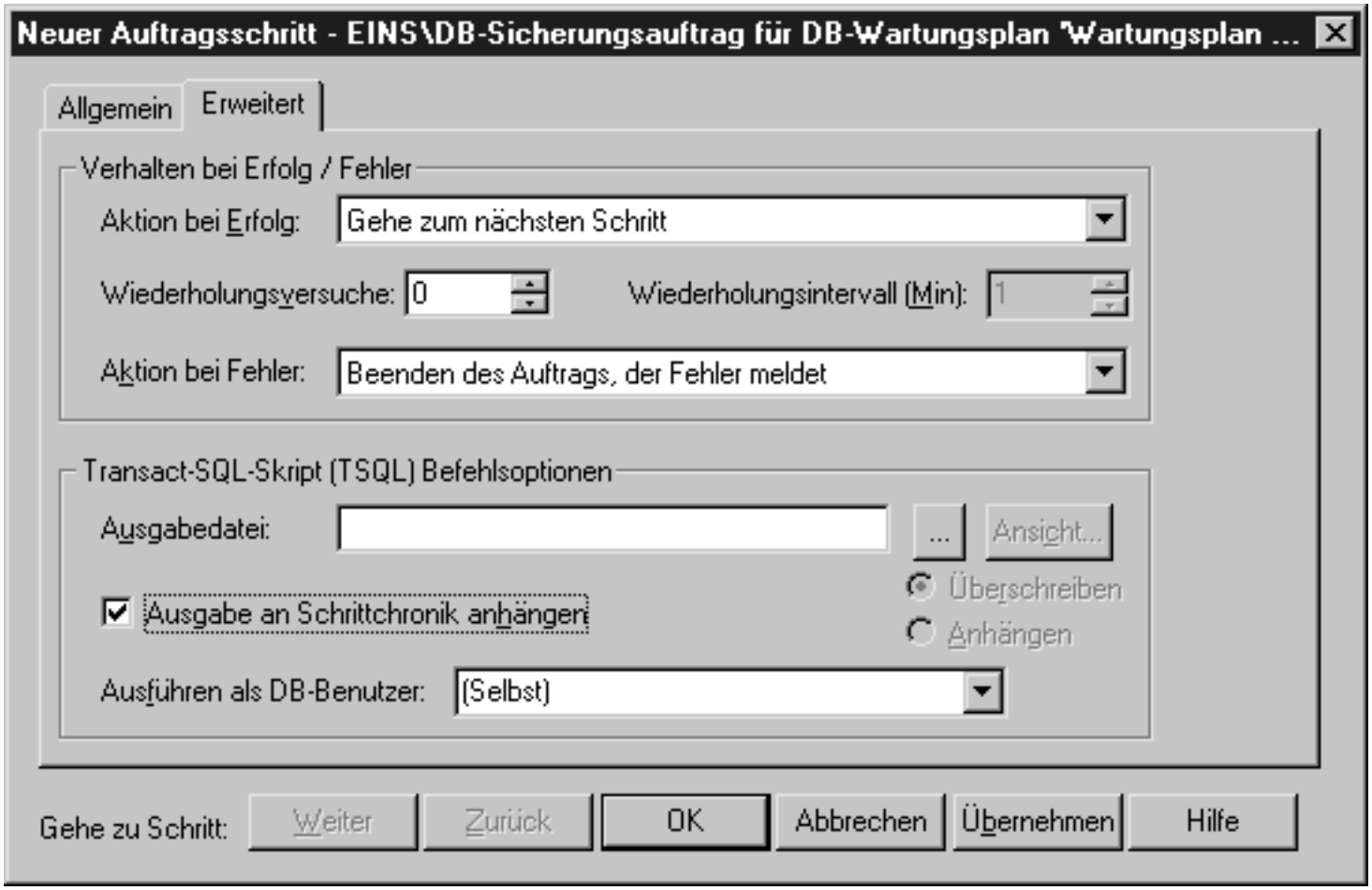
da sql server die warnungen in das windows-nt-anwendungsprotokoll schreibt und der sql server-agent das anwendungsprotokoll liest, um die einträge mit den definierten warnungen zu vergleichen und gegebenenfalls eine warnung auszulösen, stehen die beschriebenen funktionen für warnungen nicht unter windows 95/98, sondern nur unter windows nt zur verfügung.

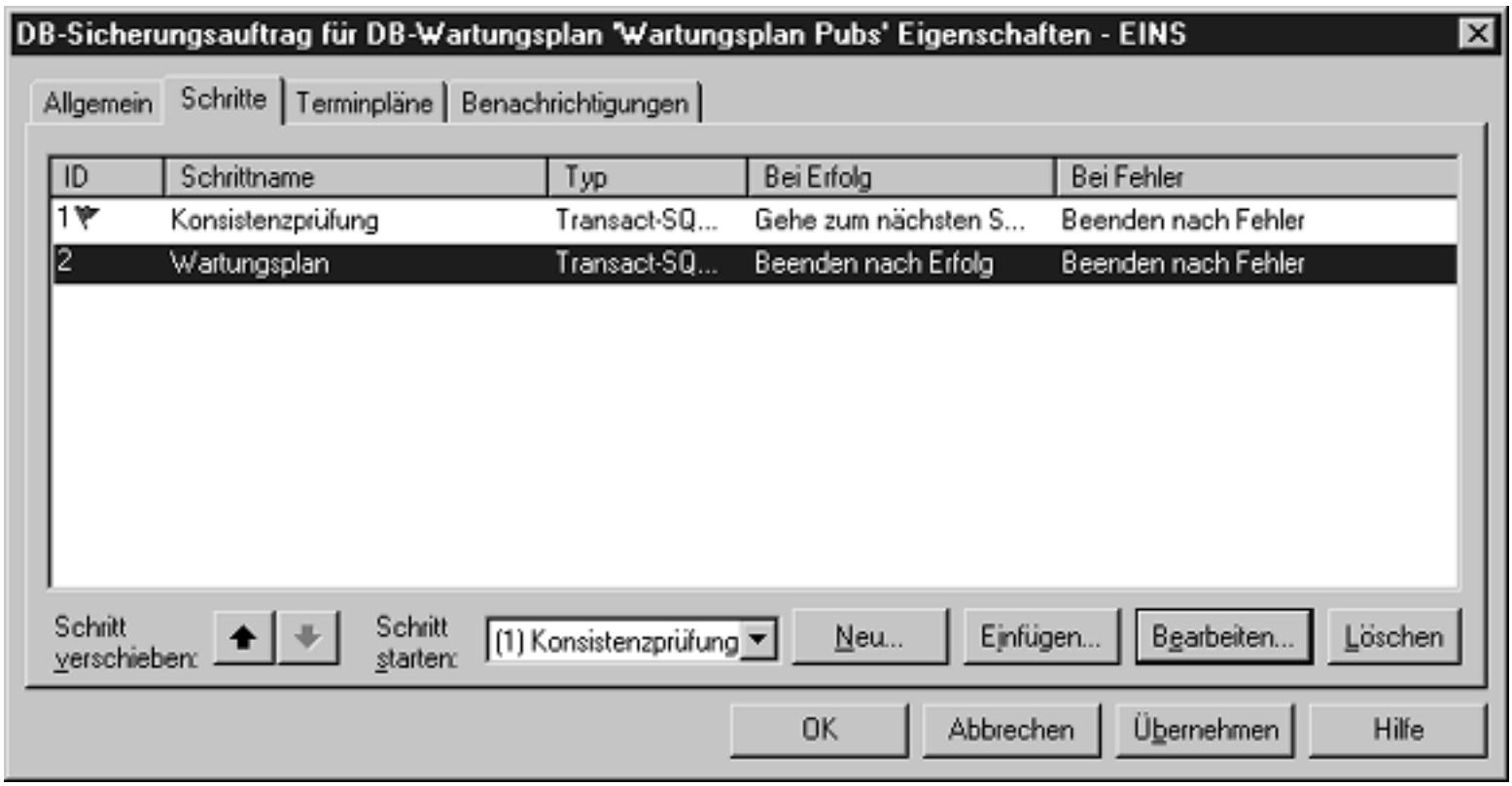

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS ...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
3 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation Services
Verwaltung
SQL Server-Agent
Warnungen
Operatoren
Aufträge
Sicherung
Aktuelle Aktivität
Datenbankwartungspläne
SQL Server-Protokolle
Webpublishing
Sicherheit
Unterstützungsdienste
SQL 6.5
Warnungen
Operatoren
Aufträge

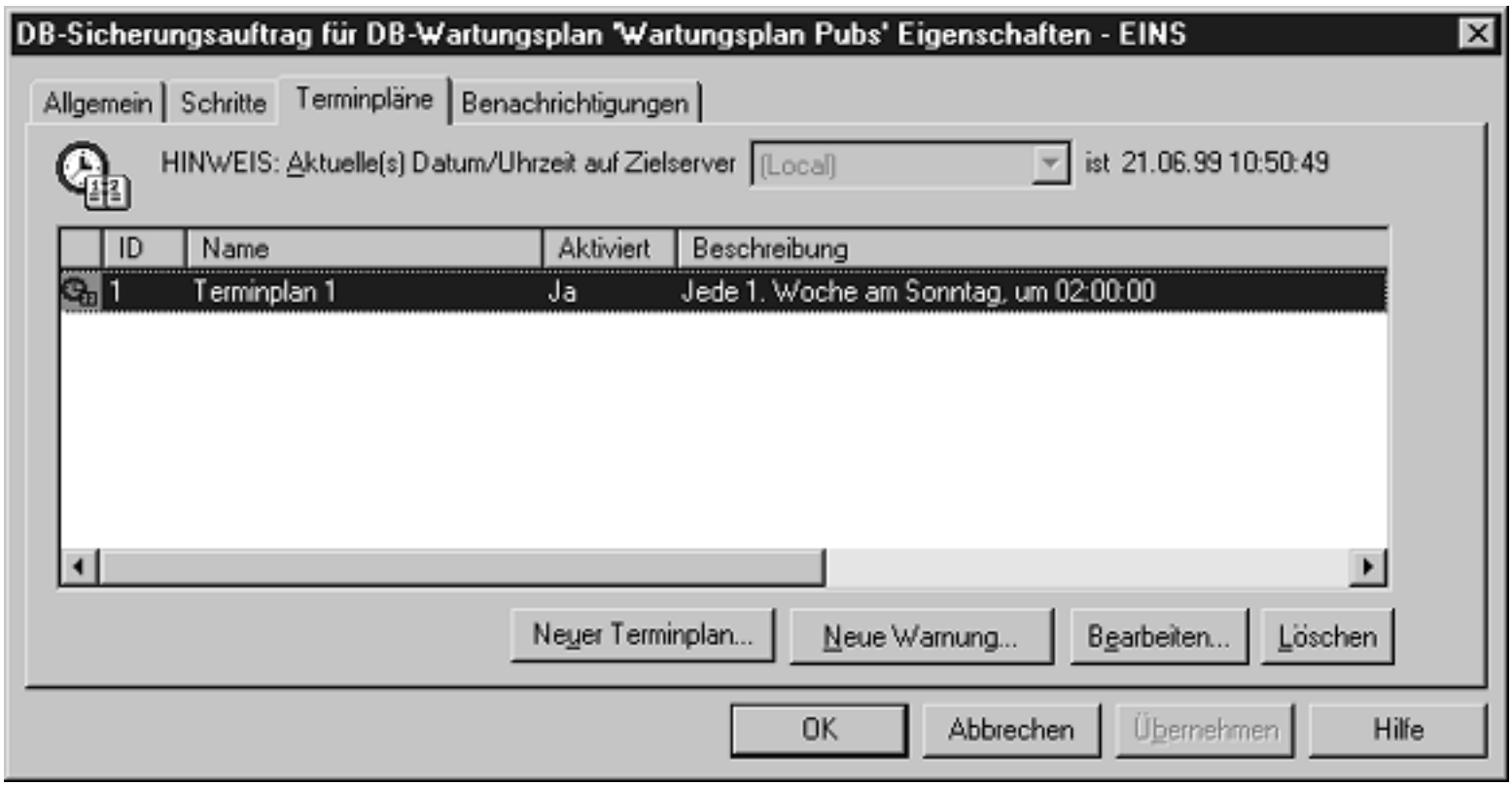

Neue Operatoreigenschaften - EINS
Allgemein
Benachrichtigungen
Benachrichtigungen an diesen Operator gesendet durch:
Warnungen
Aufträge
Warnungsname
E-Mail
Pager
NET SEND
Demo: Full msdb log
Demo: Full tempdb
Demo: Sev. 19 Errors
Demo: Sev. 20 Errors
Demo: Sev. 21 Errors
Demo: Sev. 22 Errors
Demo: Sev. 23 Errors
Demo: Sev. 24 Errors
Operator kann Benachrichtigungen empfangen
E-Mail senden...
Letzte Benachrichtigungsversuche
Mittels E-Mail: (Nie mittels E-Mail benachrichtigt)
Mittels Pager: (Nie mittels Pager benachrichtigt)
Mittels NET SEND: (Nie mittels NET SEND benachrichtigt)
OK
Abbrechen
Übernehmen
Hilfe

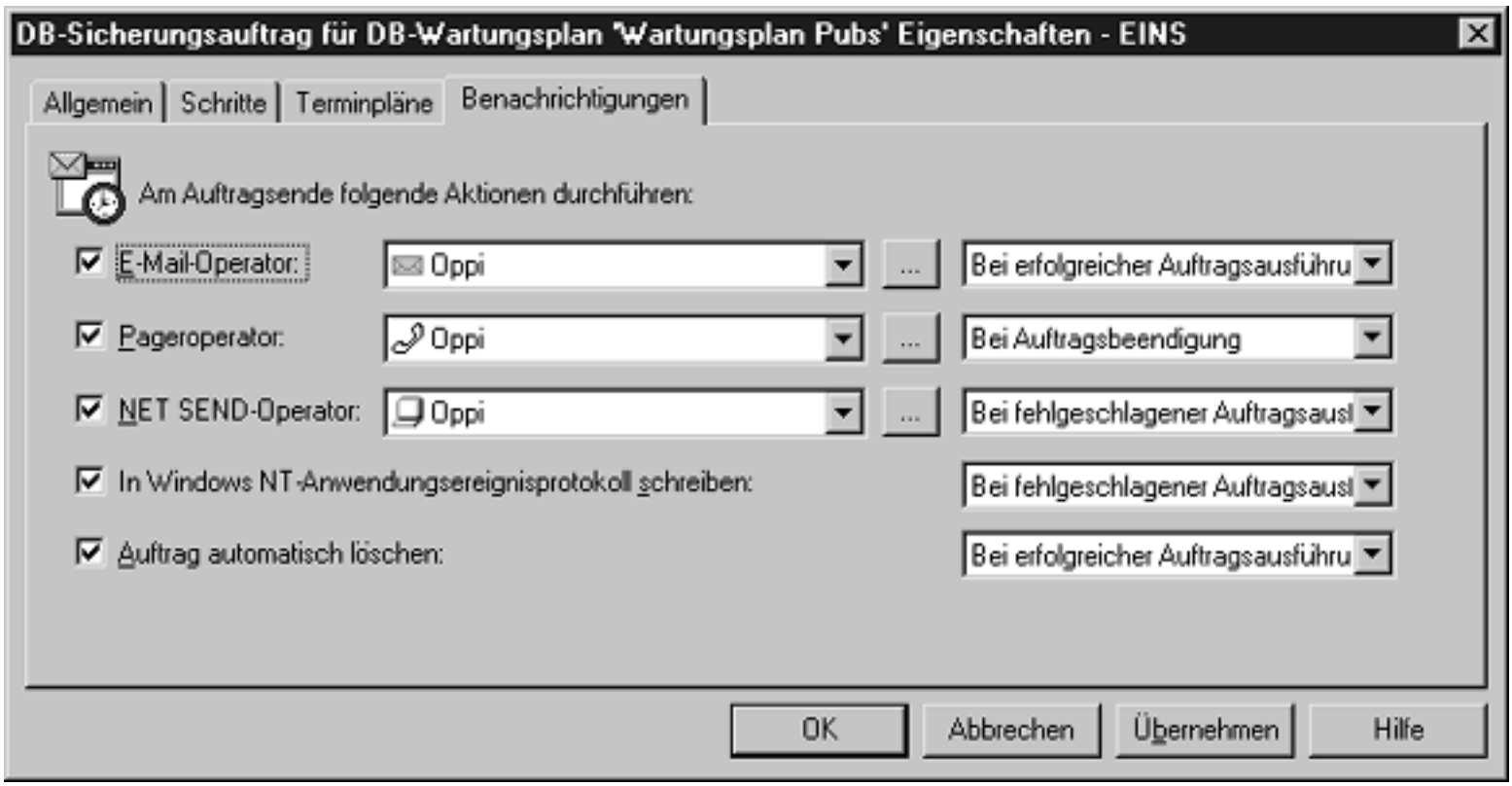

DB-Sicherungsauftrag für DB-Wartungsplan 'Wartungsplan Pubs' Eigenschaften - EINS
Allgemein
Schritte
Terminpläne
Benachrichtigungen
Name:
gsauftrag für DB-Wartungsplan 'Wartungsplan Pubs'
Quelle: (local)
Erstellt:
20.06.99 12:12:31
Aktiviert
Ziel: Lokaler Server
Ziel - mehrere Server:
Kategorie:
Datenbankwartung
...
(Local)
Besitzer:
EINS\Administrator
Beschreibung:
Keine Beschreibung verfügbar.
Zuletzt geändert: 20.06.99 15:26:56
Ändern...
OK
Abbrechen
Übernehmen
Hilfe

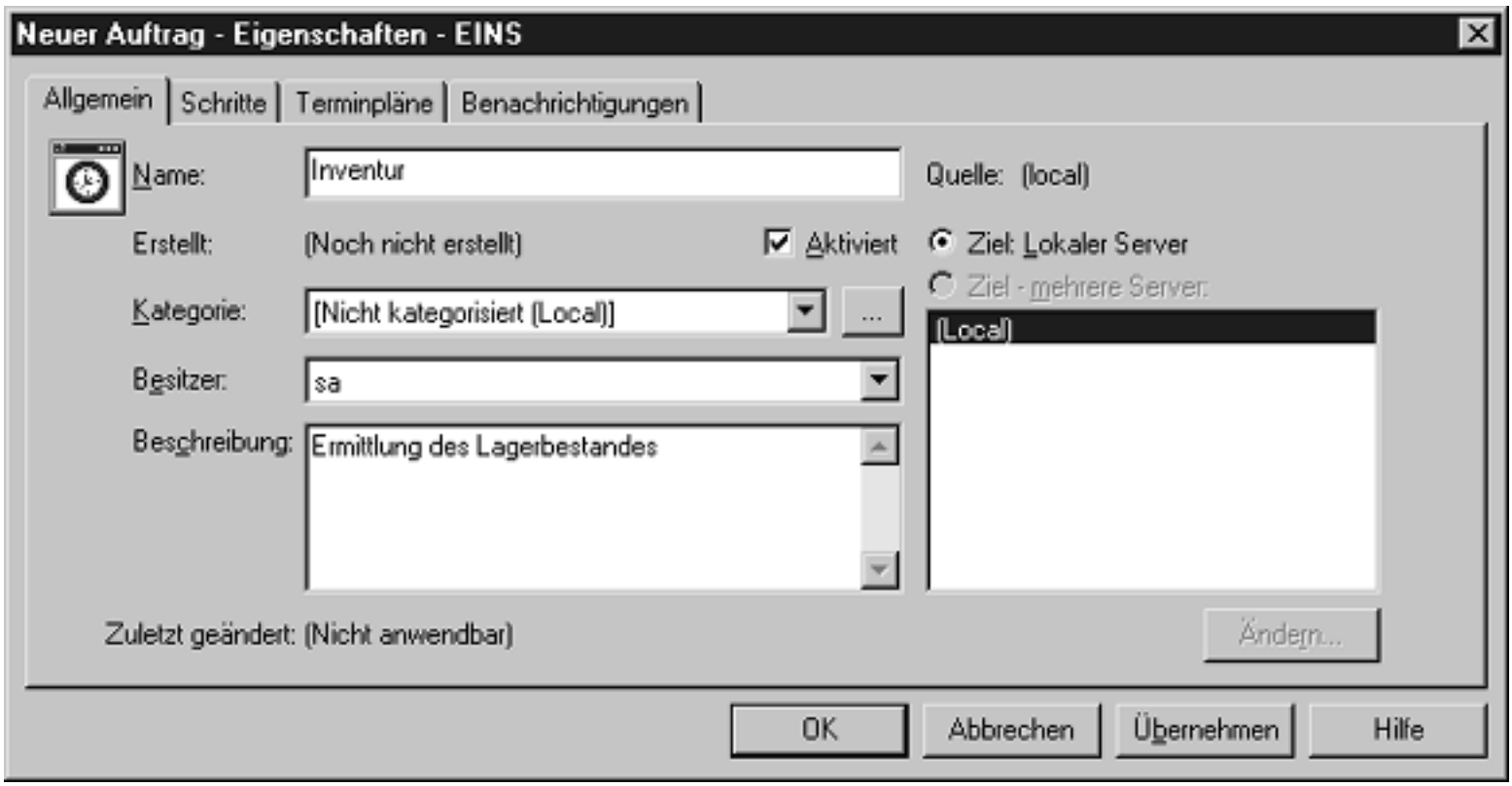

DB-Sicherungsauftrag für DB-Wartungsplan 'Wartungsplan Pubs' Eigenschaften - EINS
Allgemein
Schritte
Terminpläne
Benachrichtigungen
ID
Schrittname
Typ
Bei Erfolg
Bei Fehler
1
Schritt 1
Transact-SQ...
Beenden nach Erfolg
Beenden nach Fehler
Schritt verschieben:
Schritt starten:
(1) Schritt 1
Neu...
Einfügen...
Bearbeiten...
Löschen
OK
Abbrechen
Übernehmen
Hilfe

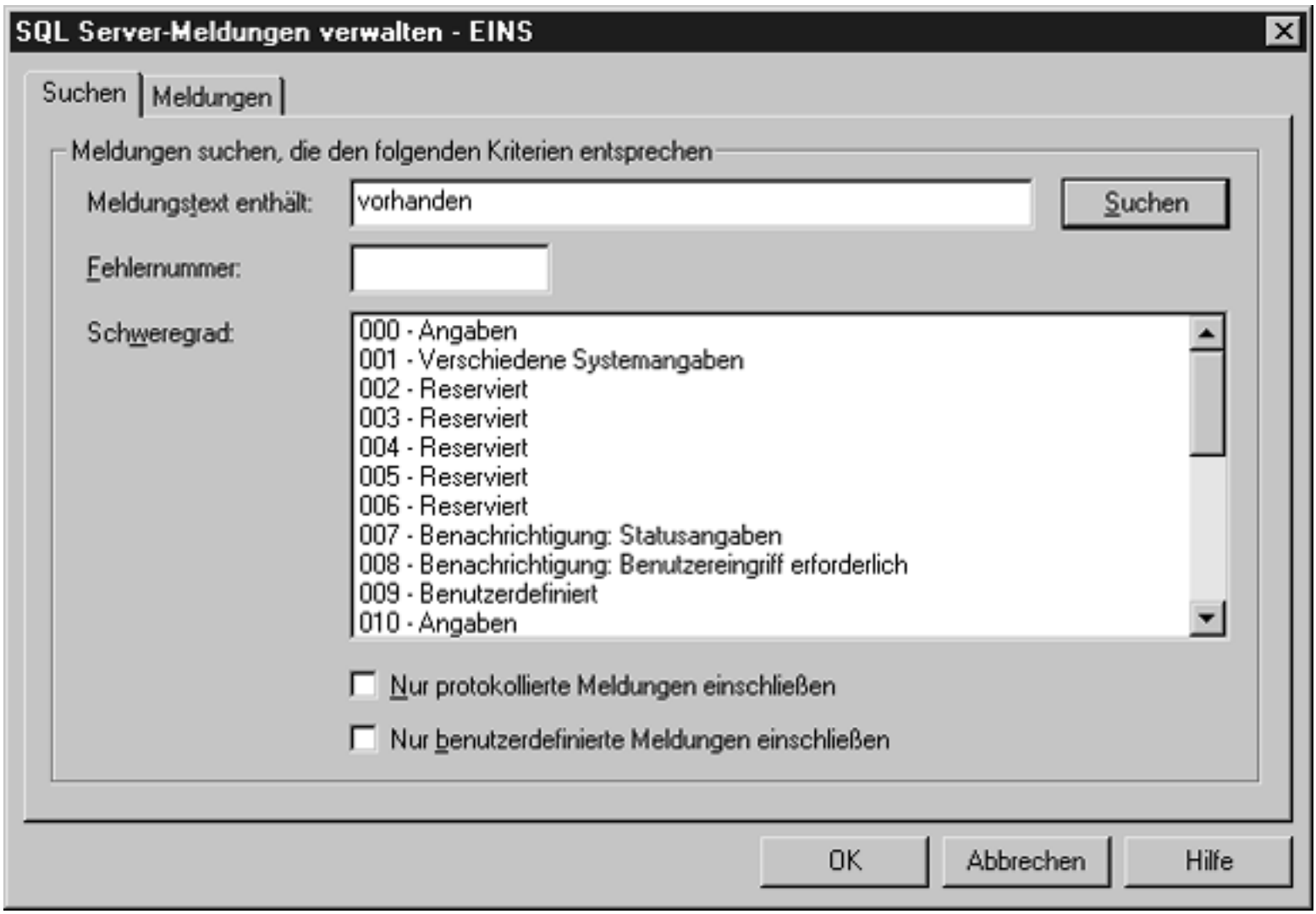

Neuer Auftragsschritt - EINS\DB-Sicherungsauftrag für DB-Wartungsplan 'Wartungsplan ...
Allgemein
Erweitert
Verhalten bei Erfolg / Fehler
Aktion bei Erfolg:
Gehe zum nächsten Schritt
Wiederholungsversuche:
0
Wiederholungsintervall (Min):
1
Aktion bei Fehler:
Beenden des Auftrags, der Fehler meldet
Transact-SQL-Skript (TSQL) Befehlsoptionen
Ausgabedatei:
...
Ansicht...
Ausgabe an Schrittchronik anhängen
Überschreiben
Anhängen
Ausführen als DB-Benutzer:
(Selbst)
Gehe zu Schritt:
Weiter
Zurück
OK
Abbrechen
Übernehmen
Hilfe

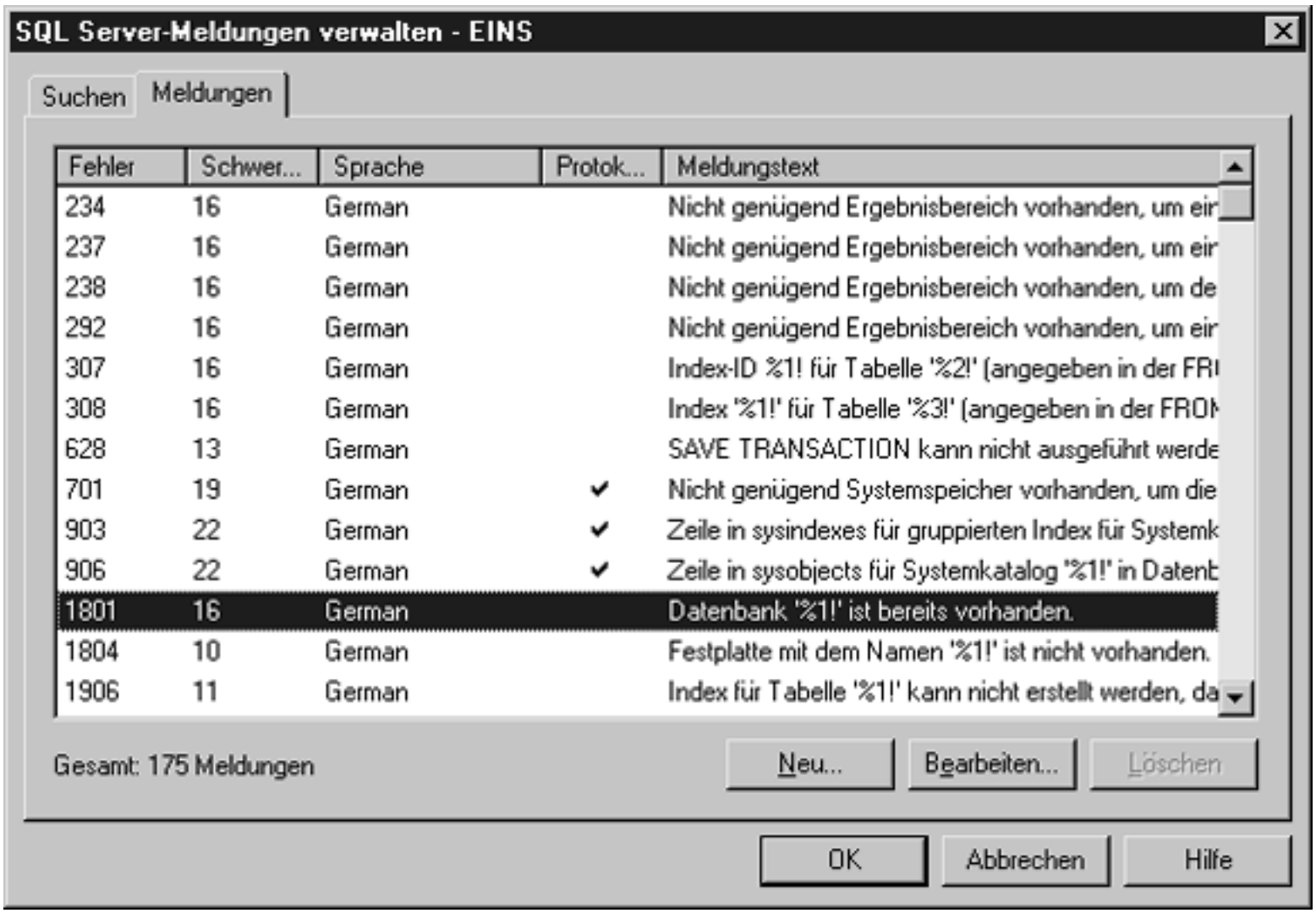

DB-Sicherungsauftrag für DB-Wartungsplan 'Wartungsplan Pubs' Eigenschaften - EINS
Allgemein
Schritte
Terminpläne
Benachrichtigungen
ID
Schrittname
Typ
Bei Erfolg
Bei Fehler
1
Konsistenzprüfung
Transact-SQ...
Gehe zum nächsten S...
Beenden nach Fehler
2
Wartungsplan
Transact-SQ...
Beenden nach Erfolg
Beenden nach Fehler
Schritt verschieben:
Schritt starten:
(1) Konsistenzprüfung
Neu...
Einfügen...
Bearbeiten...
Löschen
OK
Abbrechen
Übernehmen
Hilfe

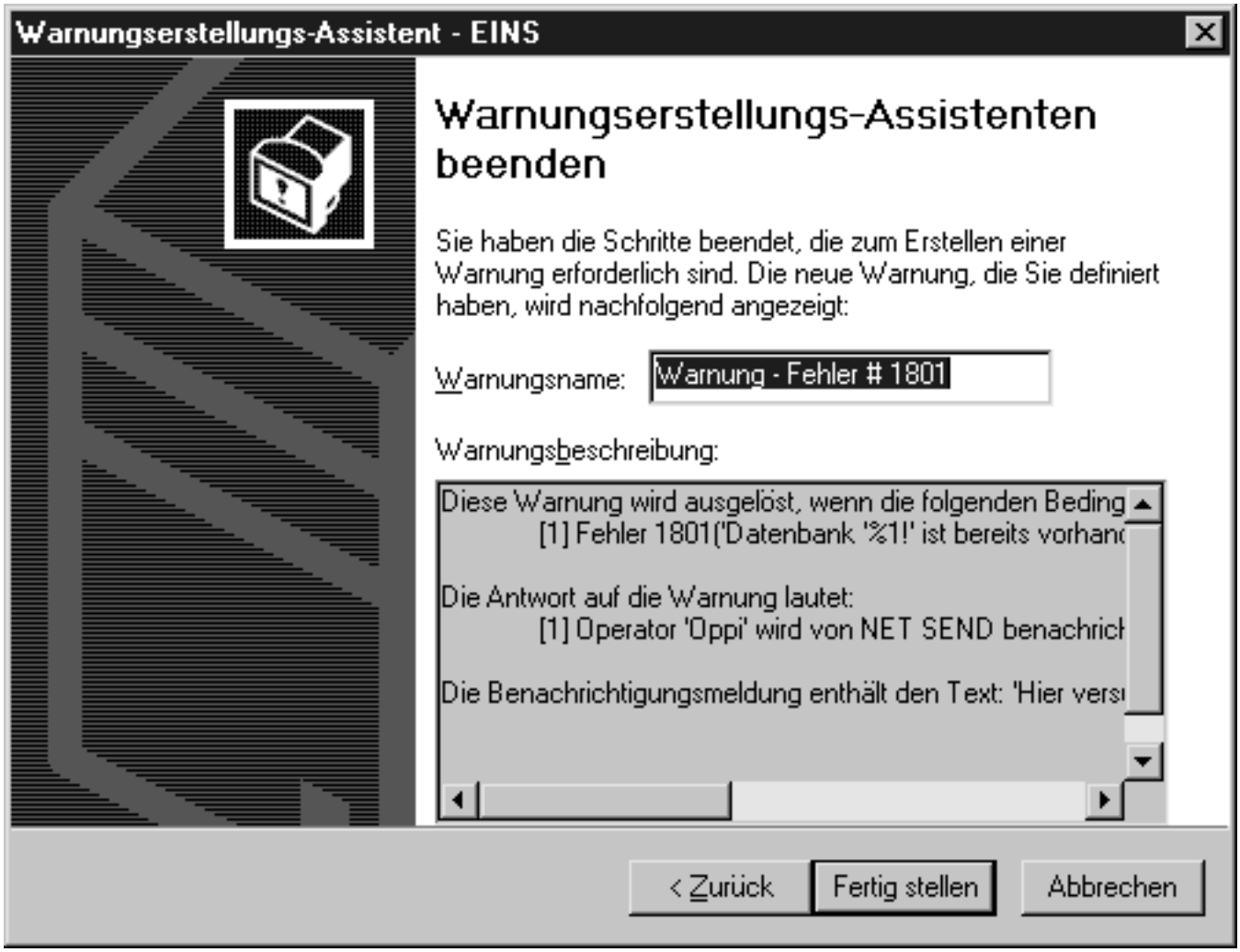

DB-Sicherungsauftrag für DB-Wartungsplan 'Wartungsplan Pubs' Eigenschaften - EINS
Allgemein
Schritte
Terminpläne
Benachrichtigungen
HINWEIS: Aktuelle(s) Datum/Uhrzeit auf Zielserver
(Local)
ist 21.06.99 10:50:49
ID
Name
Aktiviert
Beschreibung
1
Terminplan 1
Ja
Jede 1. Woche am Sonntag, um 02:00:00
Neuer Terminplan...
Neue Warnung...
Bearbeiten...
Löschen
OK
Abbrechen
Übernehmen
Hilfe


DB-Sicherungsauftrag für DB-Wartungsplan 'Wartungsplan Pubs' Eigenschaften - EINS
Allgemein
Schritte
Terminpläne
Benachrichtigungen
Am Auftragsende folgende Aktionen durchführen:
E-Mail-Operator:
Oppi
...
Bei erfolgreicher Auftragsausführu
Pageroperator:
Oppi
...
Bei Auftragsbeendigung
NET SEND-Operator:
Oppi
...
Bei fehlgeschlagener Auftragsausl
In Windows NT-Anwendungsereignisprotokoll schreiben:
Bei fehlgeschlagener Auftragsausl
Auftrag automatisch löschen:
Bei erfolgreicher Auftragsausführu
OK
Abbrechen
Übernehmen
Hilfe


Neuer Auftrag - Eigenschaften - EINS
Allgemein
Schritte
Terminpläne
Benachrichtigungen
Name:
Inventur
Quelle: (local)
Erstellt:
(Noch nicht erstellt)
Aktiviert
Ziel: Lokaler Server
Ziel - mehrere Server:
Kategorie:
[Nicht kategorisiert (Local)]
...
(Local)
Besitzer:
sa
Beschreibung:
Ermittlung des Lagerbestandes
Zuletzt geändert: (Nicht anwendbar)
Ändern...
OK
Abbrechen
Übernehmen
Hilfe


SQL Server-Meldungen verwalten - EINS
Suchen
Meldungen
Meldungen suchen, die den folgenden Kriterien entsprechen
Meldungstext enthält:
vorhanden
Suchen
Fehlernummer:
Schweregrad:
000 - Angaben
001 - Verschiedene Systemangaben
002 - Reserviert
003 - Reserviert
004 - Reserviert
005 - Reserviert
006 - Reserviert
007 - Benachrichtigung: Statusangaben
008 - Benachrichtigung: Benutzereingriff erforderlich
009 - Benutzerdefiniert
010 - Angaben
Nur protokollierte Meldungen einschließen
Nur benutzerdefinierte Meldungen einschließen
OK
Abbrechen
Hilfe


SQL Server-Meldungen verwalten - EINS
Suchen
Meldungen
Fehler
Schwer...
Sprache
Protok...
Meldungstext
234 16 German Nicht genügend Ergebnisbereich vorhanden, um ein
237 16 German Nicht genügend Ergebnisbereich vorhanden, um ein
238 16 German Nicht genügend Ergebnisbereich vorhanden, um de
292 16 German Nicht genügend Ergebnisbereich vorhanden, um ein
307 16 German Index-ID %1! für Tabelle '%2!' (angegeben in der FR
308 16 German Index '%1!' für Tabelle '%3!' (angegeben in der FRO
628 13 German SAVE TRANSACTION kann nicht ausgeführt werde
701 19 German ✔ Nicht genügend Systemspeicher vorhanden, um die
903 22 German ✔ Zeile in sysindexes für gruppierten Index für Systemk
906 22 German ✔ Zeile in sysobjects für Systemkatalog '%1!' in Datenb
1801 16 German Datenbank '%1!' ist bereits vorhanden.
1804 10 German Festplatte mit dem Namen '%1!' ist nicht vorhanden.
1906 11 German Index für Tabelle '%1!' kann nicht erstellt werden, da
Gesamt: 175 Meldungen
Neu...
Bearbeiten...
Löschen
OK
Abbrechen
Hilfe


Warnungserstellungs-Assistent - EINS
Warnungserstellungs-Assistenten beenden
Sie haben die Schritte beendet, die zum Erstellen einer Warnung erforderlich sind. Die neue Warnung, die Sie definiert haben, wird nachfolgend angezeigt:
Warnungsname:
Warnung - Fehler # 1801
Warnungsbeschreibung:
Diese Warnung wird ausgelöst, wenn die folgenden Beding
[1] Fehler 1801('Datenbank '%1!' ist bereits vorhan
Die Antwort auf die Warnung lautet:
[1] Operator 'Oppi' wird von NET SEND benachrich
Die Benachrichtigungsmeldung enthält den Text: 'Hier vers
< Zurück
Fertig stellen
Abbrechen

kapitel 24 überwachen und optimieren

24.1 geschwindigkeit ist keine hexerei

schnelligkeit ist die devise bei nahezu allen programmierern und benutzern. datenbank-managementsysteme sind hiervon nicht ausgeschlossen. gerade bei umfangreichen datenbeständen kommt es darauf an, wie lange der benutzer auf die ergebnisse einer abfrage warten muß. dabei lassen sich bereits beim entwurf einer datenbank vorkehrungen treffen, die schnelle abfragen ermöglichen. indizes sind zum beispiel ein mittel dazu. das sql-server-system bietet allerdings noch weitere möglichkeiten, effiziente operationen zu gestalten. zunächst muß man aber wissen, welche elemente überhaupt einen einfluß auf die leistung ausüben. zuerst sollte man also die server-leistung überwachen und danach gezielt an die optimierung - oder fehlerbeseitigung - gehen.

24.2 werkzeuge zur überwachung

sql server ist vorrangig für das betriebssystem windows nt ausgelegt. damit stehen zum einen der windows-nt-systemmonitor und der vom sql-server-system installierte sql server profiler zur überwachung der systemleistung zur verfügung. der windows-nt-systemmonitor ist nicht speziell auf sql server ausgerichtet. mit diesem werkzeug überwacht man die ressourcen bei server-prozessen. der sql server profiler dient dagegen der überwachung sogenannter modulereignisse. dazu gehören zum beispiel das herstellen einer verbindung, die ausführung einer transact-sql-anweisung, der start oder das ende einer gespeicherten prozedur, die ausführung eines transact-sql-stapels oder ähnliche ereignisse.

dieses kapitel stellt einige ausgewählte grafische werkzeuge vor:

- enterprise manager - fenster **aktuelle aktivität**
- sql server query analyzer
- sql server profiler
- indexoptimierungs-assistent

24.3 enterprise manager - fenster aktuelle aktivität

das fenster **aktuelle aktivität** des enterprise managers bietet einen einstieg in die überwachung von sql server. erweitern sie die konsolenstruktur bis zum ordner **verwaltung**. in diesem zweig befindet sich der ordner **aktuelle aktivität**. erweitern sie auch diesen ordner. die konsolenstruktur hat dann etwa das in abbildung 24.1 gezeigte aussehen.

bild

abbildung 24.1: das fenster aktuelle aktivität im enterprise manager

bei markiertem eintrag **prozessinfo** zeigt der detailbereich die laufenden prozesse - d.h. die anmeldungen an sql server. wenn sie auf einen eintrag mit der rechten maustaste klicken, zeigt das kontextmenü die befehle **nachricht senden**, **prozess abbrechen**, **eigenschaften** und **hilfedatei anzeigen**.

bei wahl von **eigenschaften** wird das dialogfeld **prozessdetails** geöffnet (siehe abbildung 24.2), das den letzten ausgeführten transact-sql-stapel zeigt.

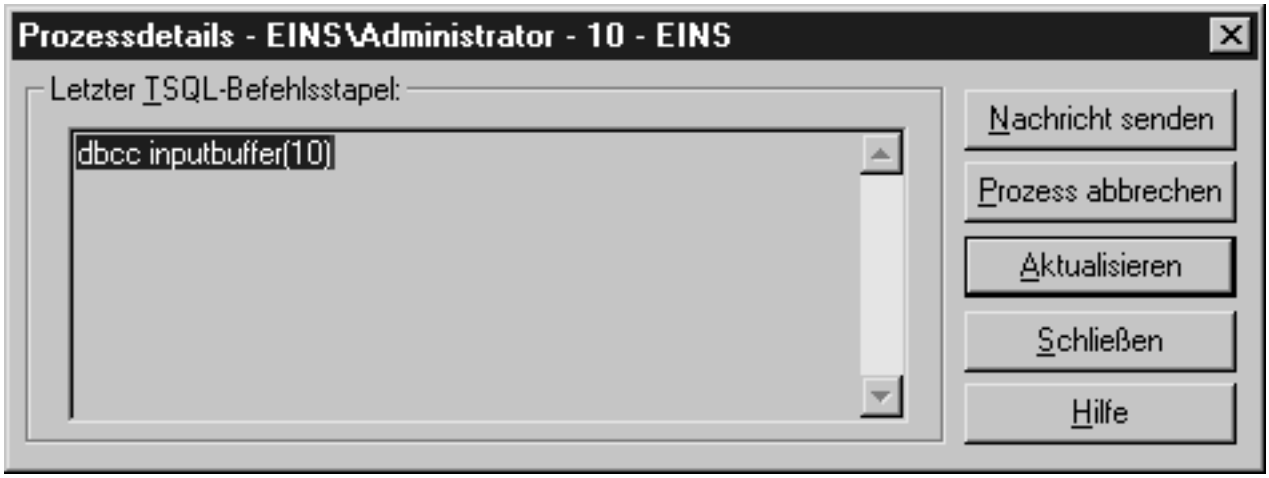


abbildung 24.2: das dialogfeld prozessdetails

hier stehen ebenfalls die befehle **nachricht senden** und **prozess abbrechen** zur verfügung. wenn sie zum beispiel einen prozeß abbrechen möchten, können sie zuvor dem jeweiligen benutzer eine nachricht über das netzwerk senden. klicken sie dazu auf die schaltfläche **nachricht senden**. im dialogfeld **nachricht senden** (siehe abbildung 24.3) geben sie eine meldung ein und spezifizieren als ziel entweder den benutzernamen oder den servernamen. klicken sie auf **senden**, um die nachricht über den nachrichtendienst des betriebssystems abzuschicken.

abbildung 24.3: senden sie dem benutzer eine nachricht, bevor sie den prozeß beenden

die beiden anderen einträge im ordner **aktuelle aktivität** wurden in kapitel 18 im zusammenhang mit

transaktionen besprochen.

24.4 sql server query analyzer

bisher haben sie den sql server query analyzer nur eingesetzt, um transact-sql-anweisungen auszuführen. wie der name vermuten läßt, kann man mit diesem programm auch abfragen analysieren.

starten sie sql server query analyzer, wählen sie aus dem menü **abfrage** den befehl **ausführungsplan anzeigen**, und führen sie als beispiel die in abbildung 24.4 dargestellte anweisung aus.

bild

abbildung 24.4: ausführungsplan im sql server query analyzer anzeigen

die interpretation der zum teil etwas kryptischen darstellung soll an dieser stelle nicht weiter vertieft werden. in der online-dokumentation finden sie unter dem stichwort **ausführungsplan** eine ausführliche erläuterung der verschiedenen operatorsymbole.

wenn sie mit der rechten maustaste klicken, finden sie im kontextmenü unter anderem die befehle **indizes verwalten**, **statistiken verwalten** und **fehlende statistiken erstellen**.

24.5 sql server profiler

mit dem grafischen werkzeug sql server profiler kann der systemadministrator sogenannte modulereignisse von sql server überwachen. es eignet sich nicht nur für die leistungsoptimierung, sondern auch für die fehlersuche - oder umgekehrt, je nach betrachtungsweise. sql server profiler ist praktisch das alter ego für sql server nach dem motto »big brother is watching you«.

wenn sie engpässe oder fehler aufspüren wollen, müssen sie zunächst einmal folgende fragen klären:

- welche ereignisse sind zu überwachen?
- welche ereignisdaten sind aufzuzeichnen?

bei der auswahl der ereignisdaten können sie filter anwenden, um die informationsflut einzudämmen. die aufgezeichneten ereignisdaten können sie später analysieren oder erneut wiedergeben.

um eine ablaufverfolgung zu erstellen, die sie später wiederverwenden können, rufen sie den sql server profiler über **start / programme / microsoft sql server 7.0 / profiler** auf.

24.5.1 neue ablaufverfolgung

wählen sie im menü **datei** von sql server profiler den befehl **neu / ablaufverfolgung**. im dialogfeld **ablaufverfolgungseigenschaften** (siehe abbildung 24.5) legen sie zunächst einen namen für die ablaufverfolgung fest.

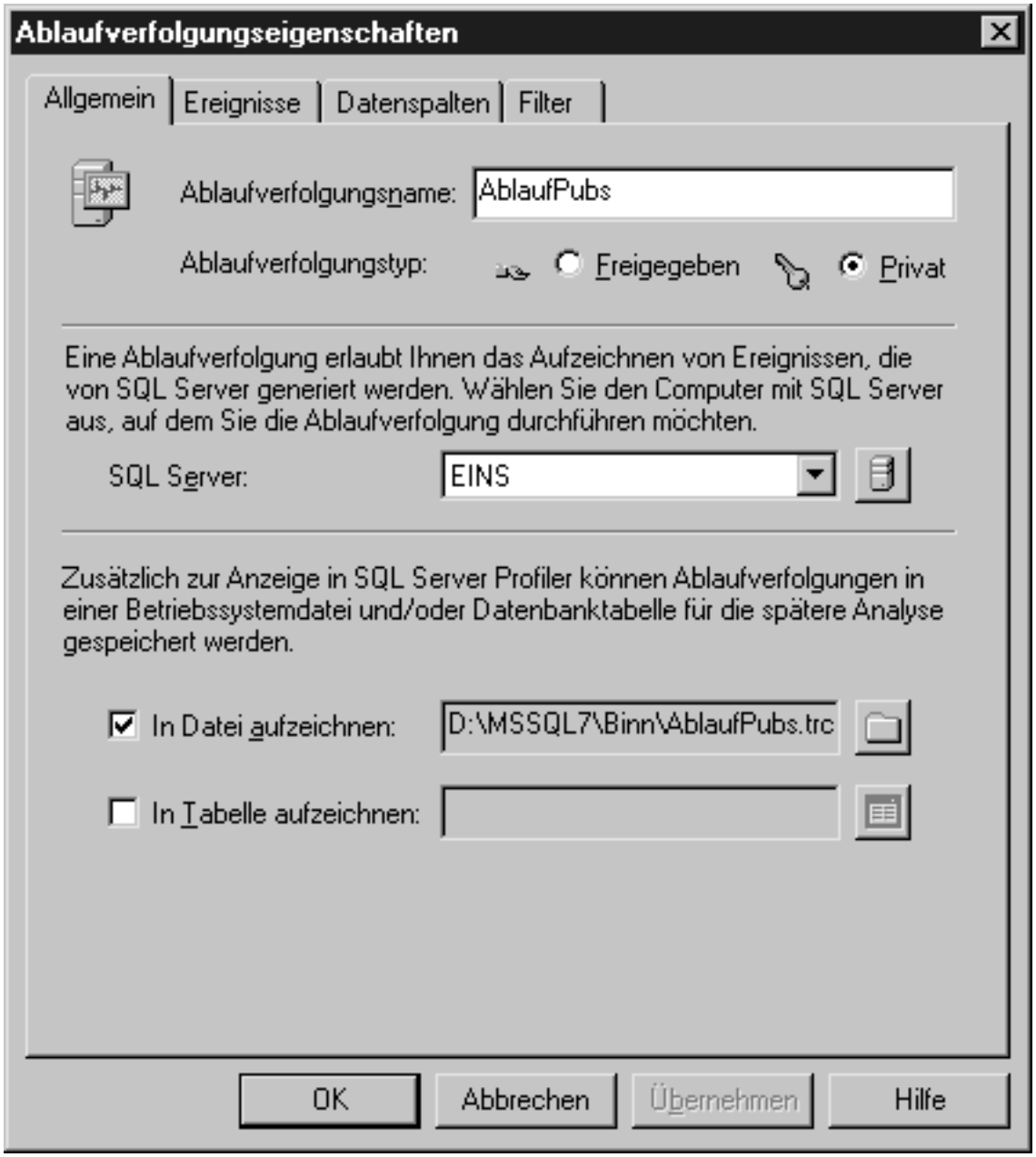


abbildung 24.5: die registerkarte allgemein im dialogfeld ablaufverfolgungseigenschaften

über die schaltfläche rechts neben dem drop-down-listenfeld **sql server** gelangen sie zum dialogfeld **quellserver** (siehe abbildung 24.6). hier lassen sich verschiedene eigenschaften festlegen, die vor allem dann wichtig sind, wenn während der ablaufverfolgung eine große zahl von ereignissen anfällt.

wenn sie die ablaufverfolgung später auswerten wollen, können sie die ergebnisse in einer datei oder datenbanktabelle speichern. im beispiel wurde die datei ablaufpubs.trc gewählt. schalten sie das kontrollkästchen **in datei aufzeichnen** ein, und klicken sie dann auf die schaltfläche mit dem ordnersymbol. damit gelangen sie zum dialogfeld **datei speichern unter**, in dem sie einen namen und gegebenenfalls einen speicherort für die verfolgungsdatei festlegen können.

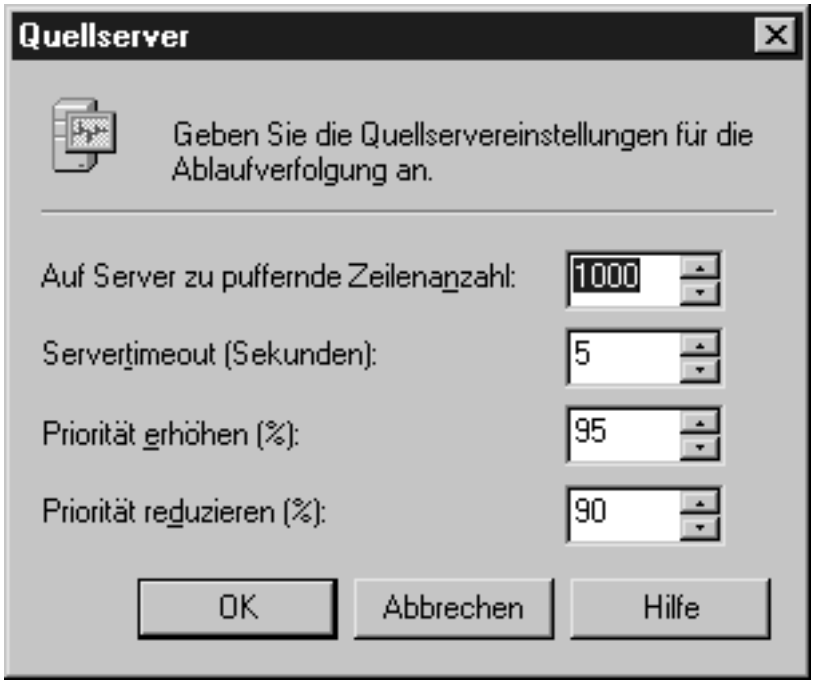


abbildung 24.6: das dialogfeld quellserver

auf der registerkarte **ereignisse** (siehe abbildung 24.7) legen sie fest, welche ereignisse sql server profiler überwachen und aufzeichnen soll.

bild

abbildung 24.7: die registerkarte ereignisse

auf der registerkarte **datenspalten** (siehe abbildung 24.8) wählen sie die aufzuzeichnenden datenspalten aus.

bild

abbildung 24.8: die registerkarte datenspalten

auf der registerkarte **filter** (siehe abbildung 24.9) können sie die arten der zu überwachenden ereignisse einschränken.

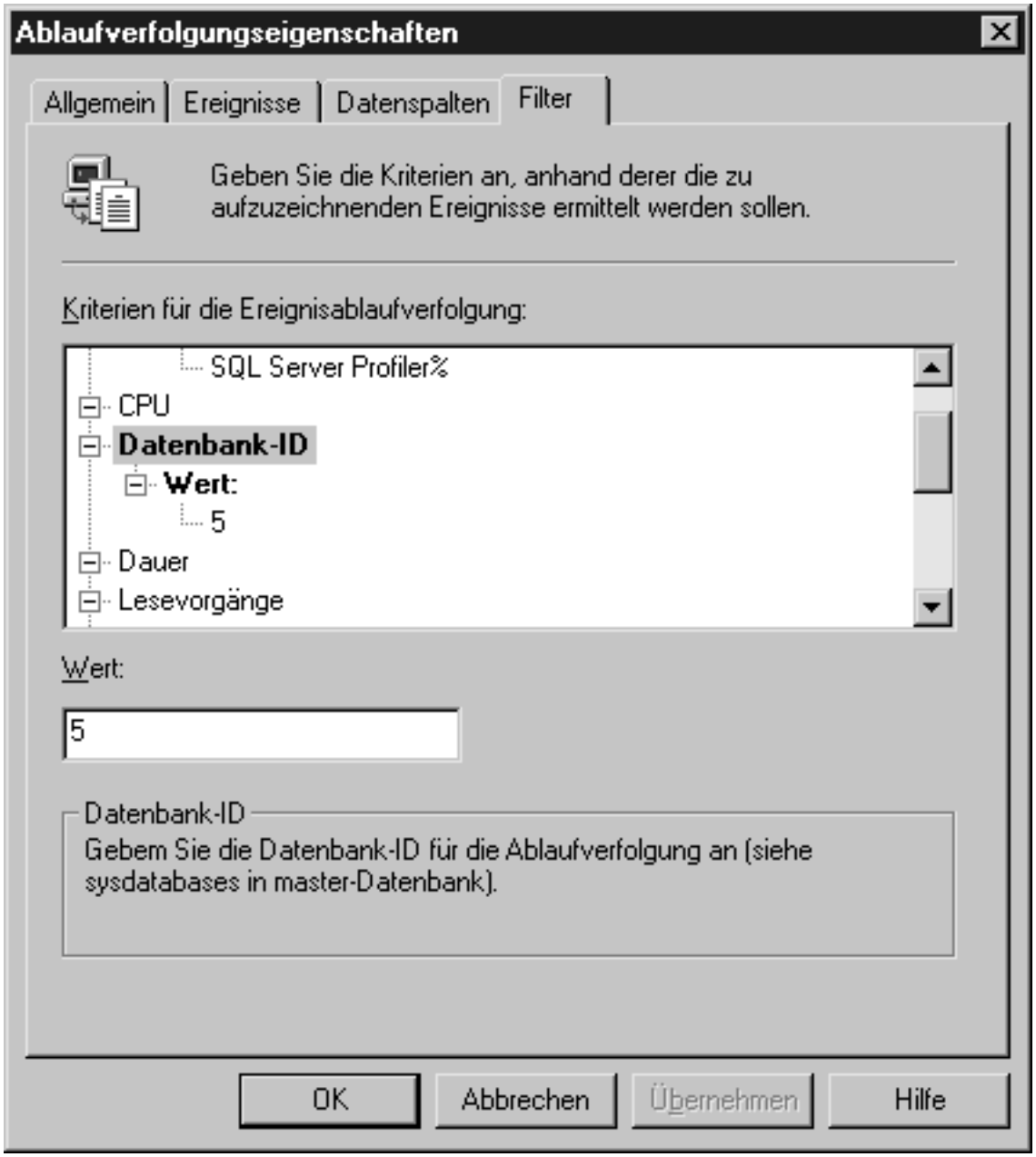


abbildung 24.9: die registerkarte filter

im beispiel soll nur die datenbank pubs überwacht werden. dazu brauchen sie die datenbank-id. diese rufen sie zum beispiel mit der folgenden transact-sql-anweisung aus der systemtabelle sysdatabases der datenbank master ab:

```
use master
select dbid from sysdatabases
where name='pubs'
```

das ergebnis lautet:

```
dbid
```

```
------
5
```

die datenbank-id 5 übernehmen sie in das feld **wert**.

sobald sie das dialogfeld **ablaufverfolgungseigenschaften** mit **ok** schließen, startet sql server profiler die ablaufverfolgung. darauf weist der eintrag »tracestart« in der spalte **ereignisklasse** hin. wenn sie nun transact-sql-anweisungen für die datenbank pubs ausführen, zeichnet sql server profiler die entsprechenden ereignisse auf und gibt sie im fenster wieder. abbildung 24.10 zeigt den zustand nach ausführen der anweisungen

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

bild

abbildung 24.10: eine laufende ablaufverfolgung

klicken sie auf die schaltfläche **diese ablaufverfolgung beenden**, um die aktuelle aufzeichnung zu stoppen. die ergebnisse finden sie in der datei ablaufpubs.trc, die sie auf der registerkarte **allgemein** des dialogfelds **ablaufverfolgungseigenschaften** festgelegt haben.

24.5.2 ablaufverfolgung anzeigen

wenn sie sich die aufgezeichneten ergebnisse ansehen wollen, wählen sie **datei / öffnen / ablaufverfolgungsdatei** und dann die datei ablaufpubs.trc aus dem dialogfeld **öffnen**. die datei wird im sql server profiler angezeigt (siehe abbildung 24.11).

bild

abbildung 24.11: die aufgezeichneten ereignisse in der datei ablaufpubs.trc

24.5.3 ablaufverfolgungserstellungs-assistent

in der symbolleiste des sql server profilers befindet sich ganz rechts eine schaltfläche, die den namen **ablaufverfolgungserstellungs-assistent** trägt und den gleichnamigen assistenten startet (siehe abbildung 24.12).

bild

abbildung 24.12: startdialogfeld des ablaufverfolgungserstellungs-assistenten

im zweiten dialogfeld **problem identifizieren** (siehe abbildung 24.13) bietet der assistent im drop-down-listenfeld **problem** folgende optionen:

- abfragen mit der schlechtesten leistung suchen
- ablauf der transact-sql-aktivitäten auf anwendungsbasis verfolgen
- ablauf der transact-sql-aktivitäten auf benutzerbasis verfolgen
- leistungsprofil einer gespeicherten prozedur erstellen
- scans von größeren tabellen identifizieren
- ursache für einen deadlock suchen

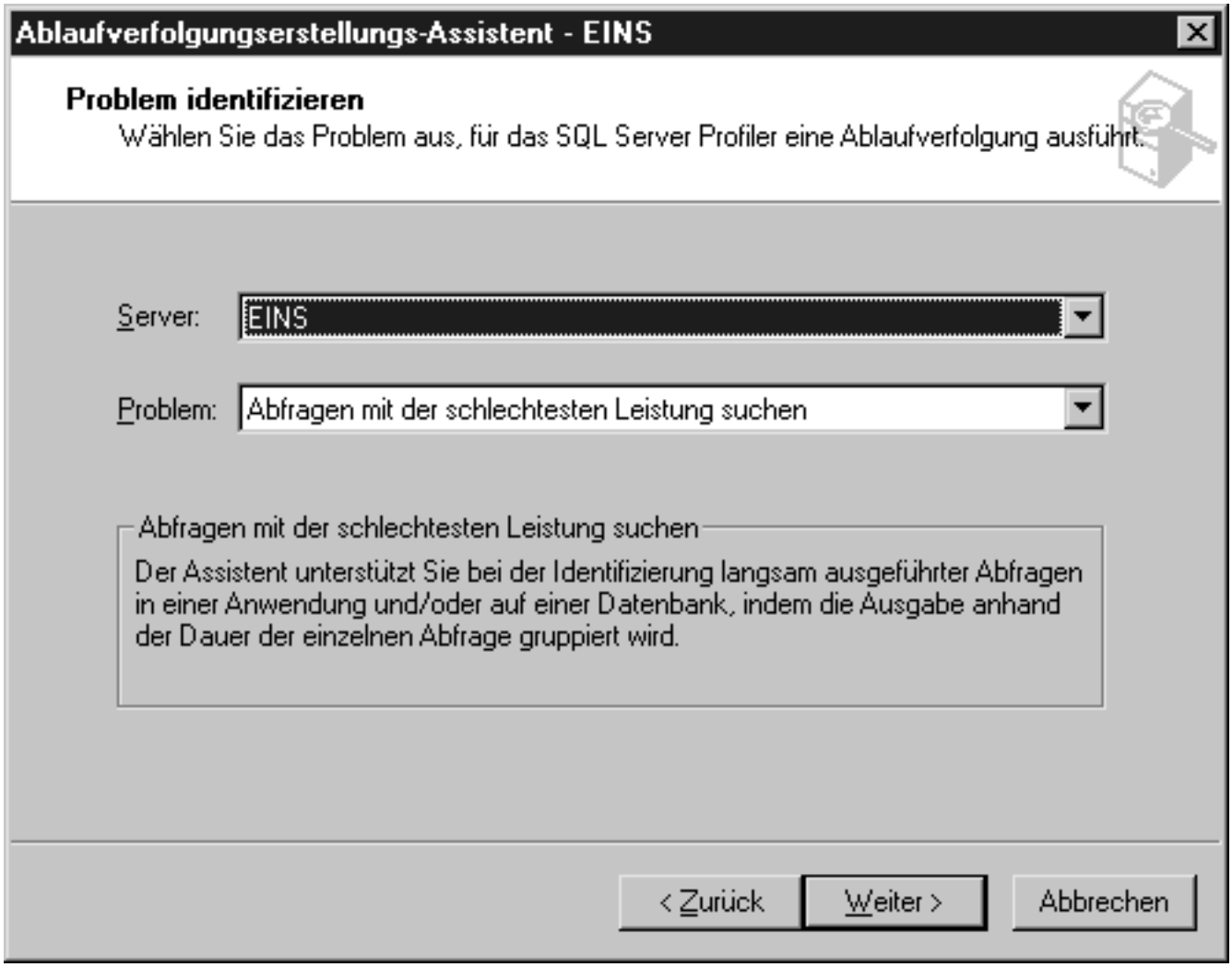


abbildung 24.13: das dialogfeld problem identifizieren

je nach ausgewähltem problem führt sie der assistent zu speziellen dialogfeldern, die jeweils den titel **ablaufverfolgungsfilter angeben** tragen.

bei der option *abfragen mit der schlechtesten leistung suchen*, kommen sie zum dialogfeld nach abbildung 24.14. hier wählen sie zunächst die datenbank(en) aus, für die sie abfragen überwachen wollen. und da es um besonders langwierige abfragen geht, geben sie im feld **mindestdauer** ein zeitlimit vor. sql server profiler zeichnet dann alle abfragen auf, die das gewählte limit überschreiten. wenn die ablaufverfolgung zu viele abfragen zutage fördert, erhöhen sie das zeitlimit, bei zu wenigen treffern setzen sie es entsprechend herunter. mit **weiter** gelangen sie zum zweiten dialogfeld **ablaufverfolgungsfilter angeben** (siehe abbildung 24.15).

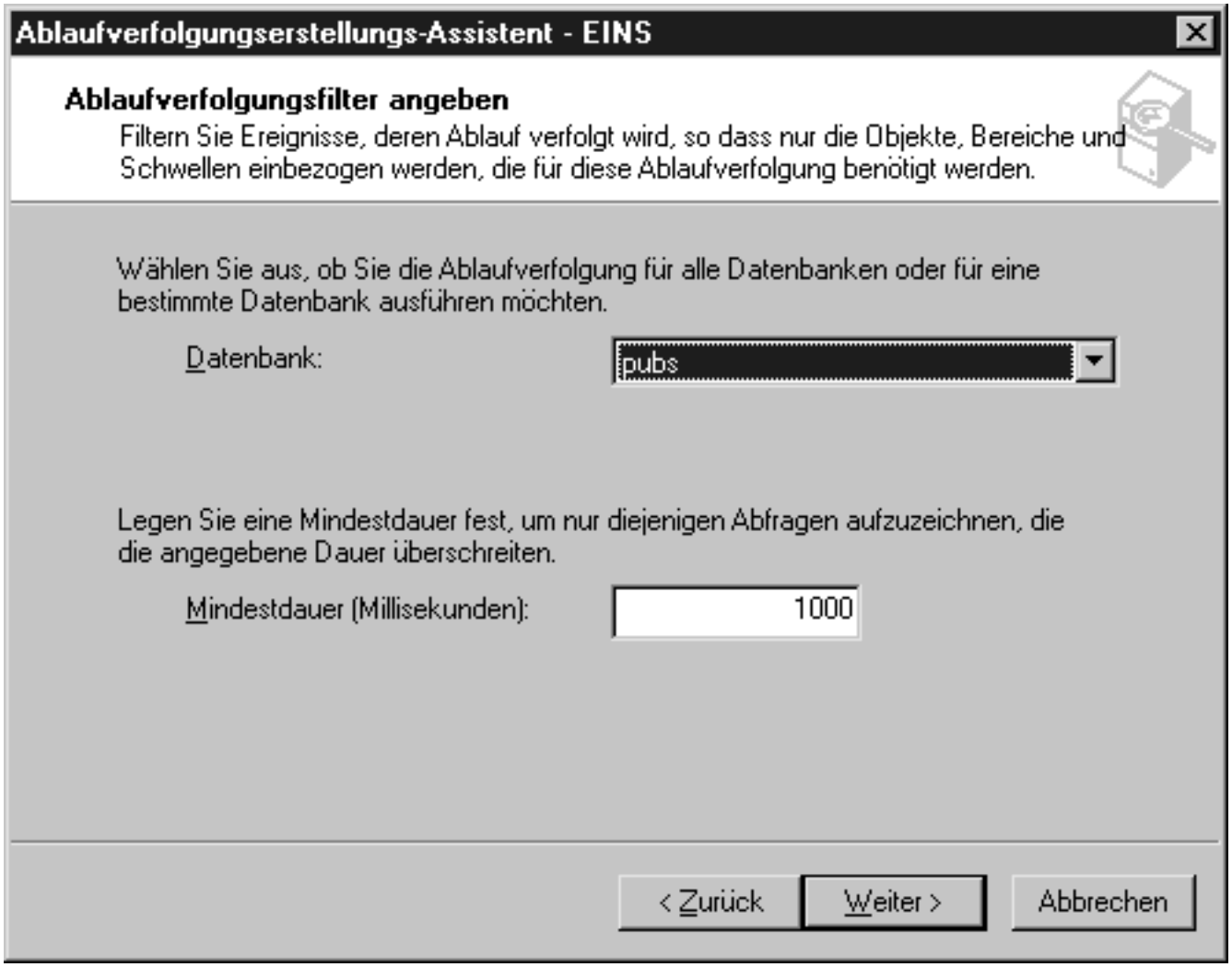


abbildung 24.14: das dialogfeld ablaufverfolgungsfilter angeben

nach wahl der option *ablauf der transact-sql-aktivitäten auf anwendungsbasis verfolgen* kommen sie direkt zum dialogfeld **ablaufverfolgungsfilter angeben**, in dem sie anwendungen, die ereignisse generieren, ein-/ausschließen können. durch klicken auf **weiter** kommen sie zum letzten dialogfeld des assistenten (siehe abbildung 24.19).

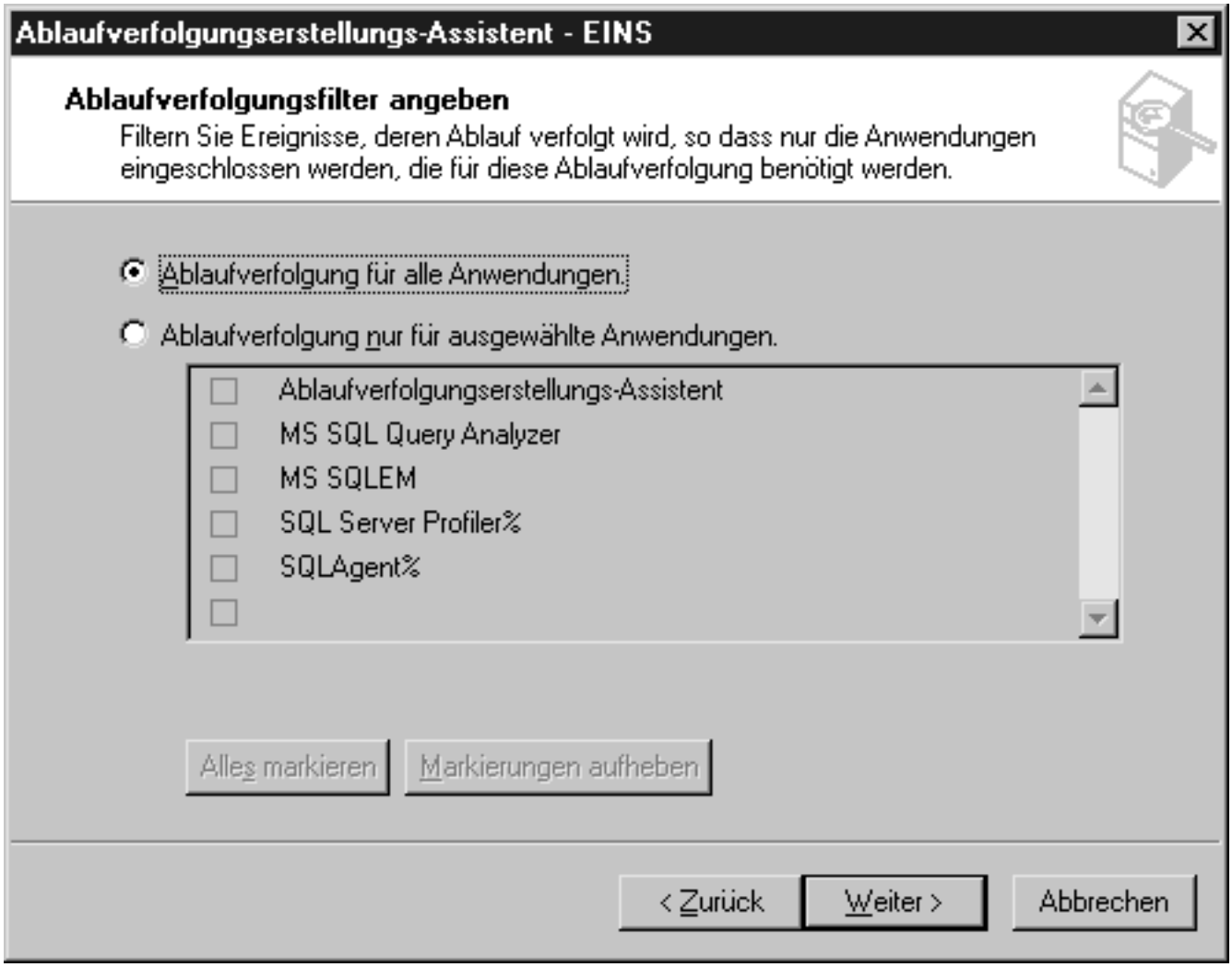


abbildung 24.15: anwendungen ein-/ausschließen

nach wahl der option *ablauf der transact-sql-aktivitäten auf benutzerbasis verfolgen* kommen sie zum dialogfeld **ablaufverfolgungsfilter angeben**, in dem sie die ablaufverfolgung auf bestimmte benutzer einschränken können (siehe abbildung 24.16). durch klicken auf **weiter** gelangen sie zum letzten dialogfeld des assistenten (siehe abbildung 24.19).

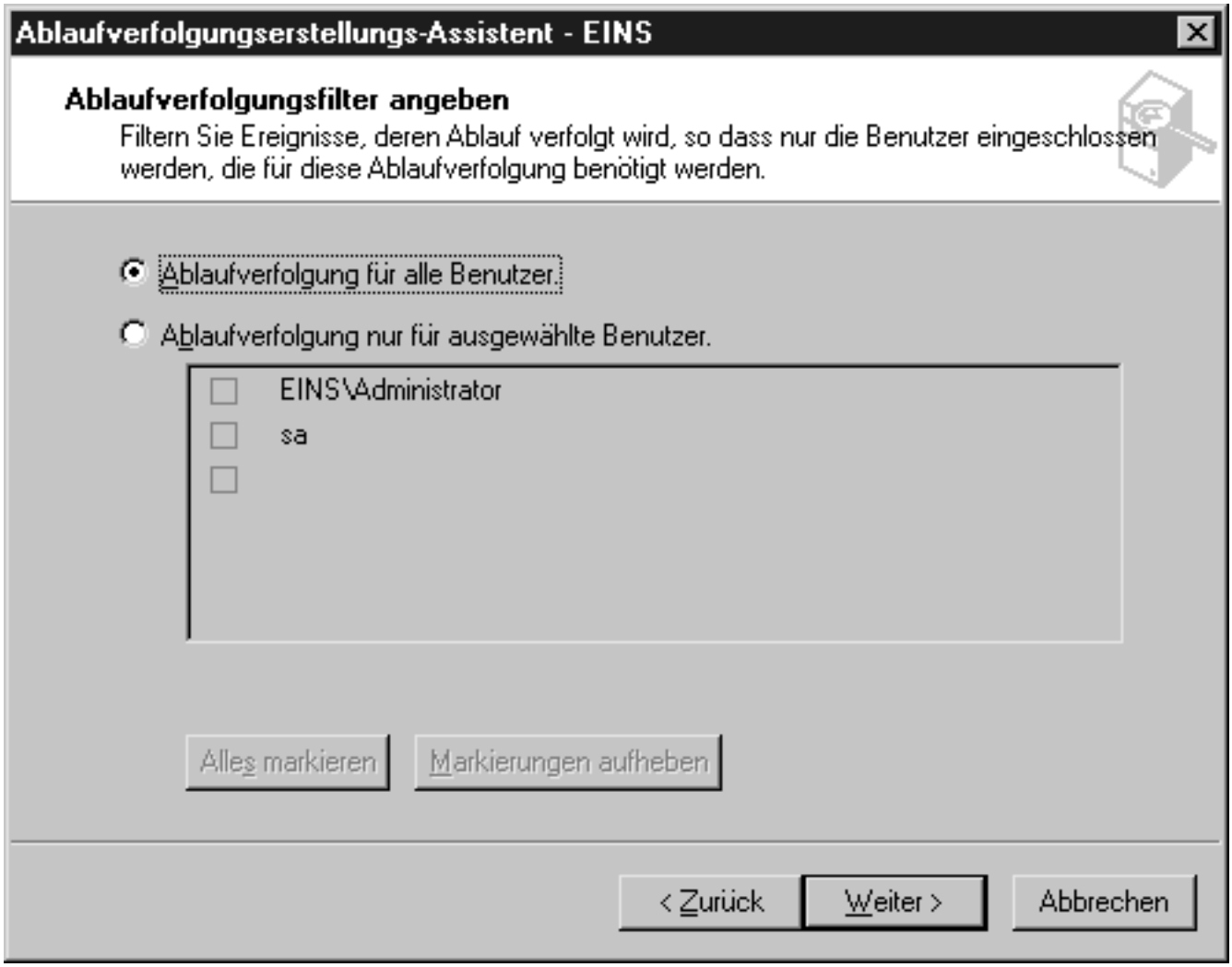


abbildung 24.16: benutzer ein-/ausschließen

im dialogfeld **ablaufverfolgungsfilter angegeben** für die option *leistungsprofil einer gespeicherten prozedur erstellen* (siehe abbildung 24.17) wählen sie die datenbank und im drop-down-listenfeld **gespeicherte prozedur** entweder alle oder eine bestimmte gespeicherte prozedur dieser datenbank aus. durch klicken auf **weiter** gelangen sie zum letzten dialogfeld des assistenten (siehe abbildung 24.19).

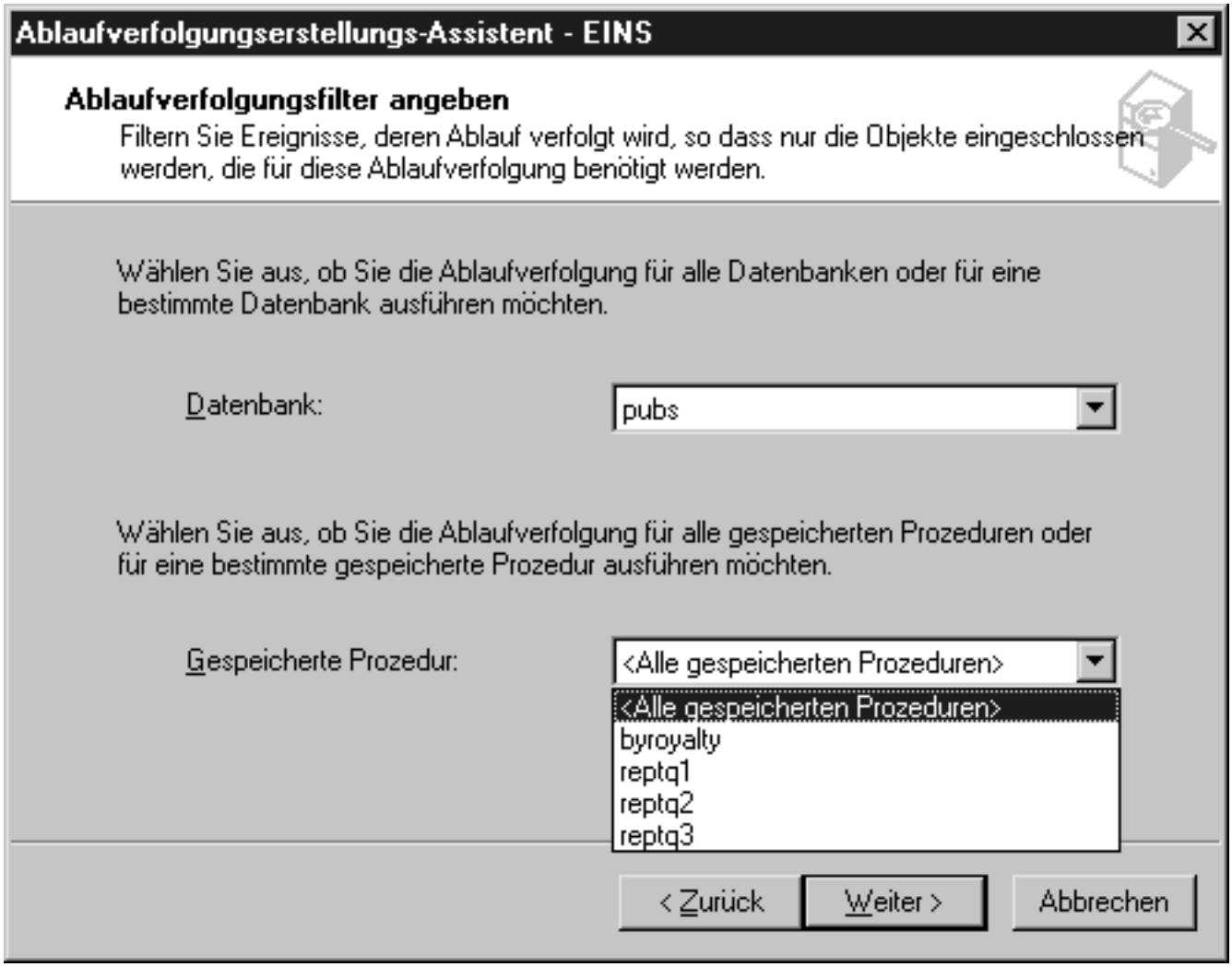


abbildung 24.17: ablaufverfolgungsfilter für gespeicherte prozeduren

nach wahl der option *scans von größeren tabellen identifizieren* kommen sie zum dialogfeld **ablaufverfolgungsparameter für tabellenscan identifizieren** (siehe abbildung 24.18). hier wählen sie die datenbank(en) und tabelle(n) aus, die in die ablaufverfolgung einzubeziehen sind. durch klicken auf **weiter** gelangen sie zum letzten dialogfeld des assistenten (siehe abbildung 24.19).

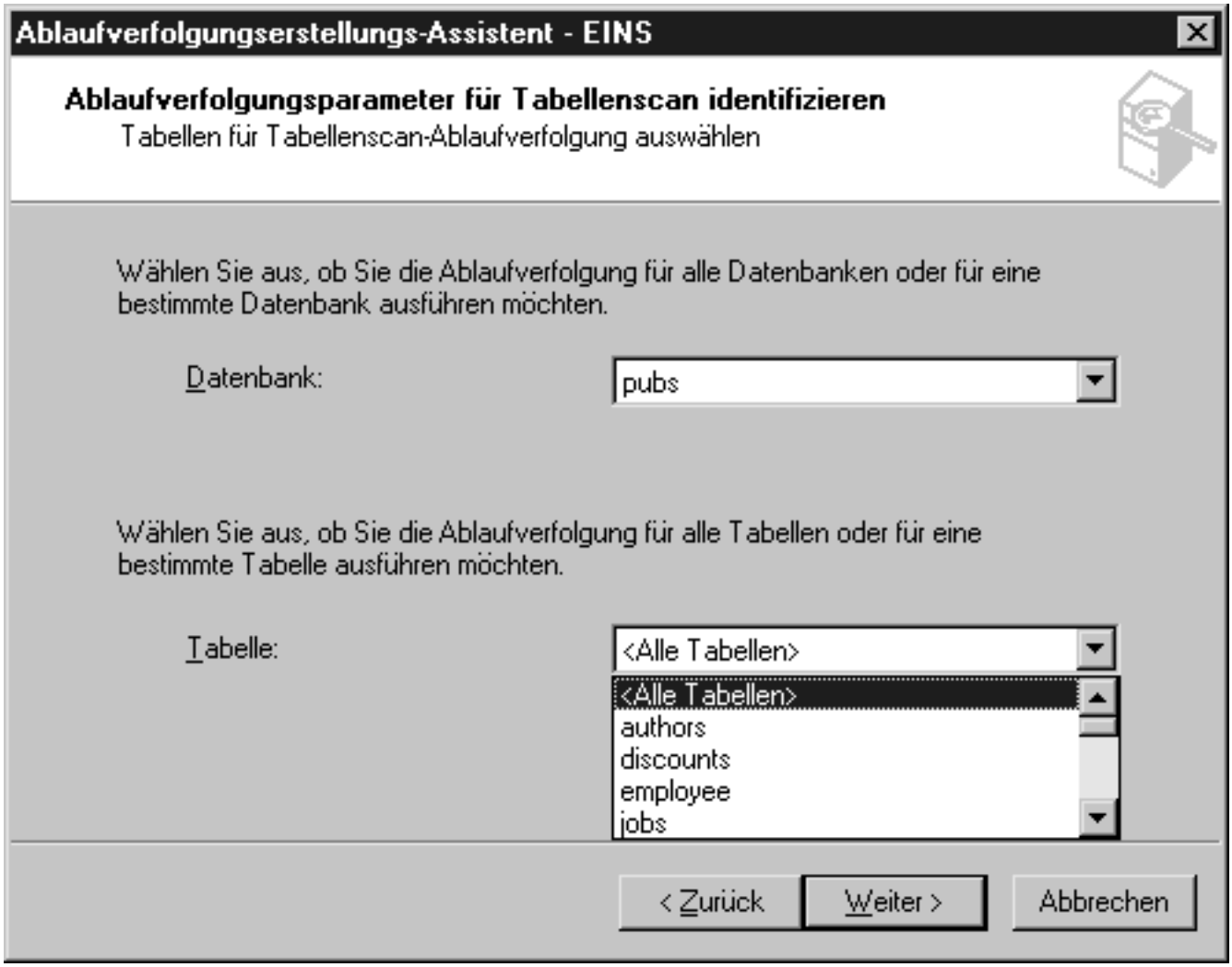


abbildung 24.18: ablaufverfolgungsparameter für tabellenscan identifizieren

das letzte dialogfeld des assistenten, das abbildung 24.19 für die option *abfragen mit der schlechtesten leistung suchen* zeigt und das für die anderen optionen ähnlich aussieht, faßt alle gewählten eigenschaften für die ablaufverfolgung zusammen. klicken sie auf **fertigstellen**, um die definition der ablaufverfolgung erstellen zu lassen und die ablaufverfolgung zu starten.

bild

abbildung 24.19: letztes dialogfeld des ablaufverfolgungserstellungs-assistenten

die gestartete ablaufverfolgung wird im fenster von sql server profiler angezeigt, wie es bereits abbildung 24.10 im abschnitt »neue ablaufverfolgung« dargestellt hat.

24.6 indexoptimierungs-assistent

der indexoptimierungs-assistent liefert auf der basis einer arbeitsauslastung eine empfehlung, welche indizes oder statistiken sie für eine sql-server-datenbank auswählen oder erstellen sollten.

wie sie eine arbeitsauslastung erstellen, haben sie bei der behandlung des sql server profilers gelernt. wenn sie eine neue arbeitsauslastung erstellen, zeichnen sie die standardereignisse und datenspalten auf und speichern die ergebnisse in einer datei.

den indexoptimierungs-assistenten können sie sowohl vom enterprise manager als auch vom sql server profiler starten. im profiler wählen sie dazu aus dem menü **extras** den befehl **indexoptimierungs-assistent**. führen sie dann folgende schritte aus:

1. klicken sie im startdialogfeld (siehe abbildung 24.20) auf **weiter**.

abbildung 24.20: startdialogfeld des indexoptimierungs-assistenten

2. im nächsten dialogfeld (siehe abbildung 24.21) wählen sie den server und die datenbank, für die der index zu optimieren ist. schalten sie das kontrollkästchen **ausführliche analyse durchführen** ein. klicken sie auf **weiter**.

bild

abbildung 24.21: server und datenbank auswählen

3. im ersten dialogfeld **arbeitsauslastung angeben** (siehe abbildung 24.22) entscheiden sie, ob sie eine vorhandene arbeitsauslastung verwenden oder eine neue erstellen wollen. wählen sie für das beispiel die erste option. die arbeitsauslastung haben sie bereits mit dem profiler erstellt. klicken sie auf **weiter**.

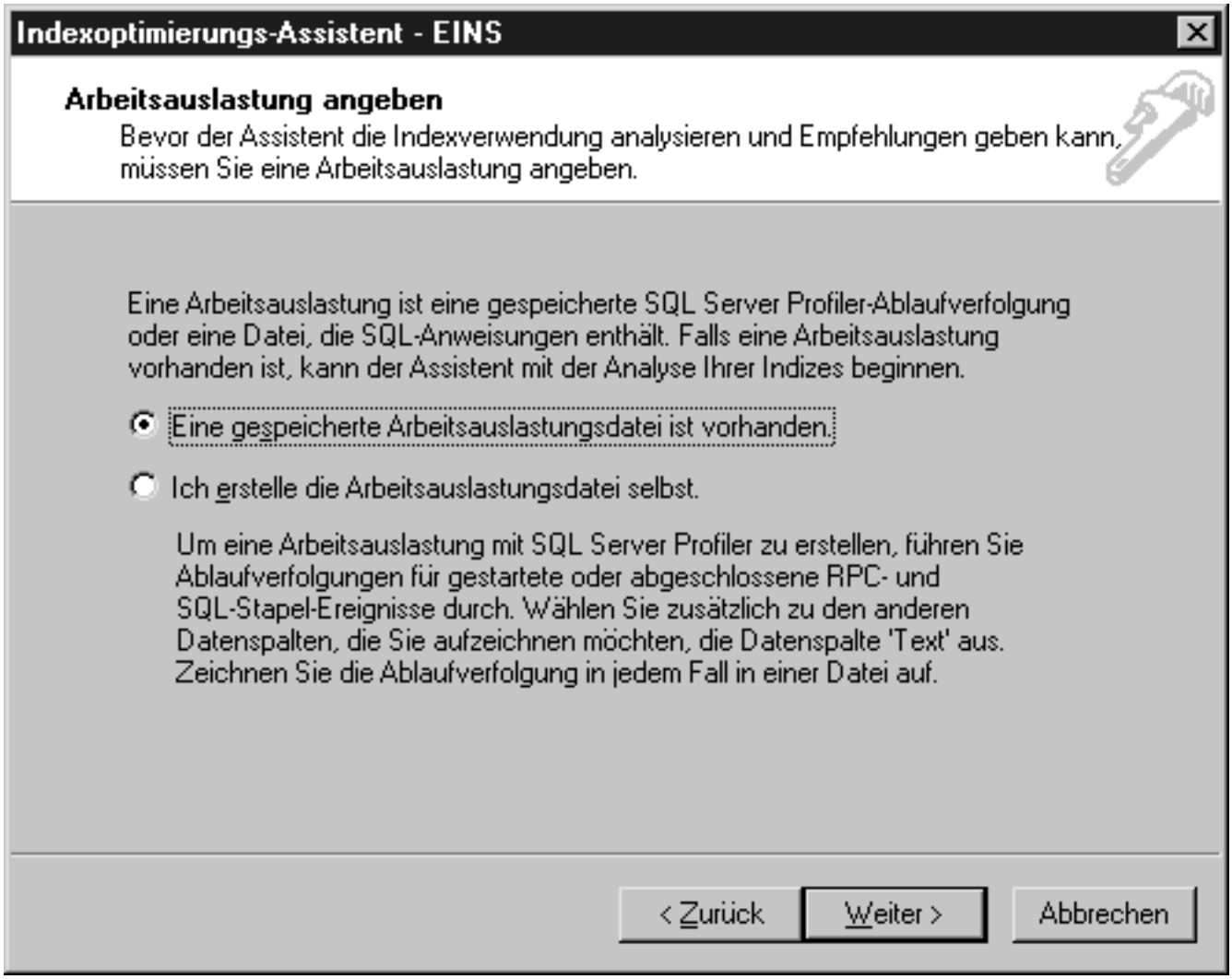


abbildung 24.22: arbeitsauslastung angeben (1. teil)

4. im zweiten teil dieses dialogfelds (siehe abbildung 24.23) spezifizieren sie die vorhandene arbeitsauslastung ablaufpubs.trc. klicken sie auf **weiter**.

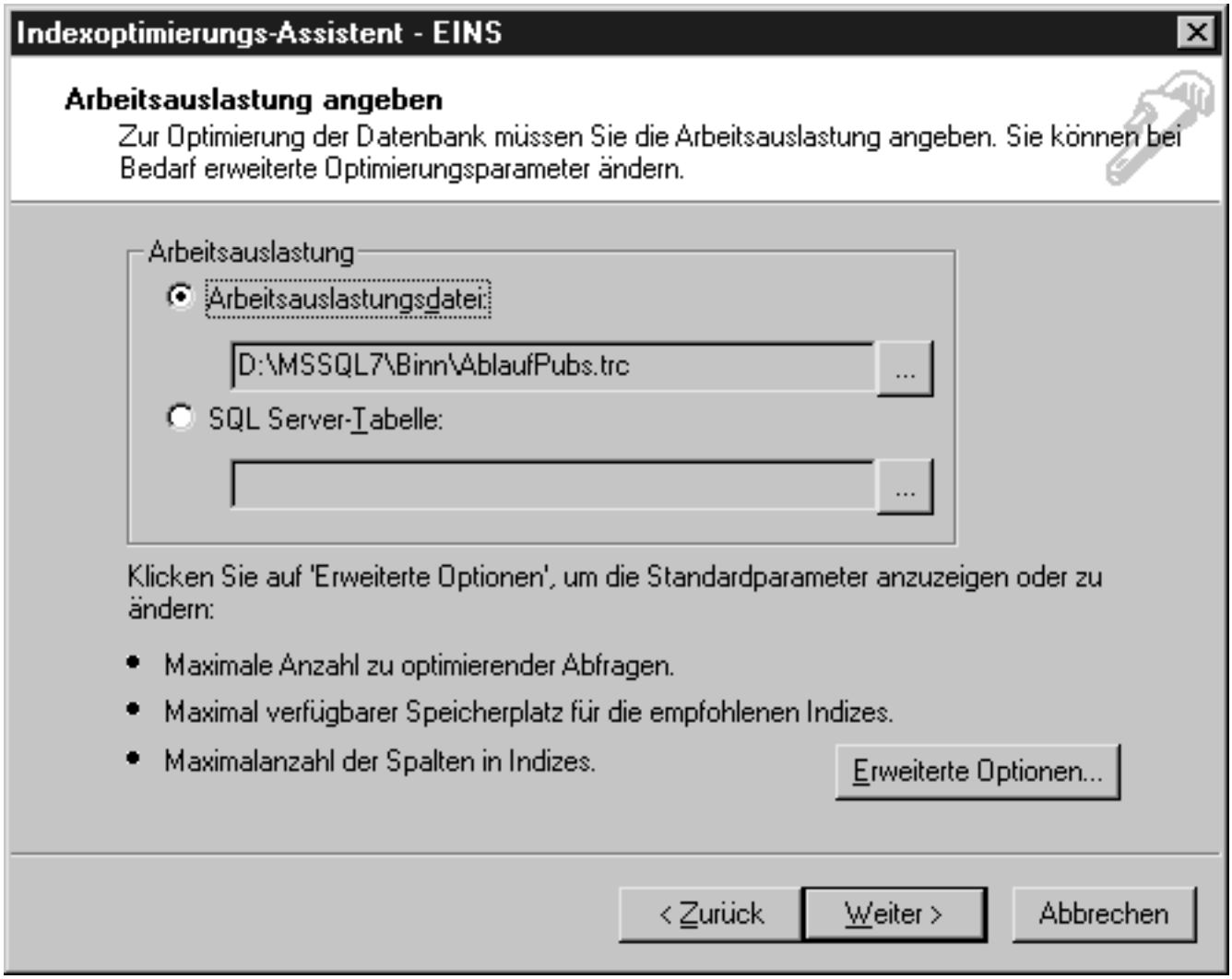


abbildung 24.23: arbeitsauslastung angeben (2. teil)

5. klicken sie auf die schaltfläche **erweiterte optionen**, um die standardparameter der indexoptimierung anzuzeigen und gegebenenfalls zu ändern. es erscheint das in abbildung 24.24 gezeigte dialogfeld. nehmen sie bei bedarf änderungen vor, und schließen sie dann das dialogfeld.

bild

abbildung 24.24: zu optimierende tabellen auswählen

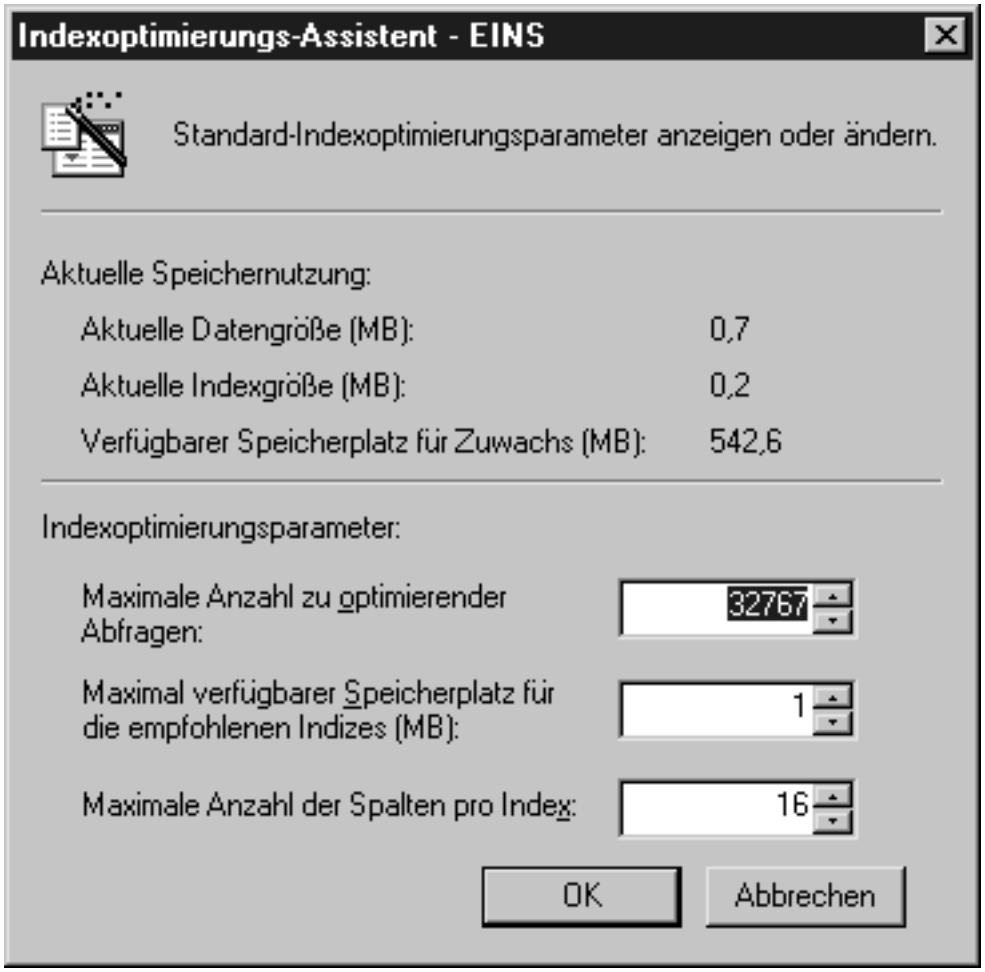


abbildung 24.25: erweiterte eigenschaften

6. im nächsten schritt des assistenten (siehe abbildung 24.25) wählen sie die tabellen aus, die sie in die optimierung einbeziehen wollen. in der abbildung sind alle tabellen ausgewählt. über die schaltflächen unter den beiden listenfeldern können sie bestimmte tabellen entfernen oder wieder hinzufügen. klikken sie dann auf **weiter**.
7. nachdem sie auf **weiter** geklickt haben, erscheint eine fortschrittsmeldung wie in abbildung 24.26.

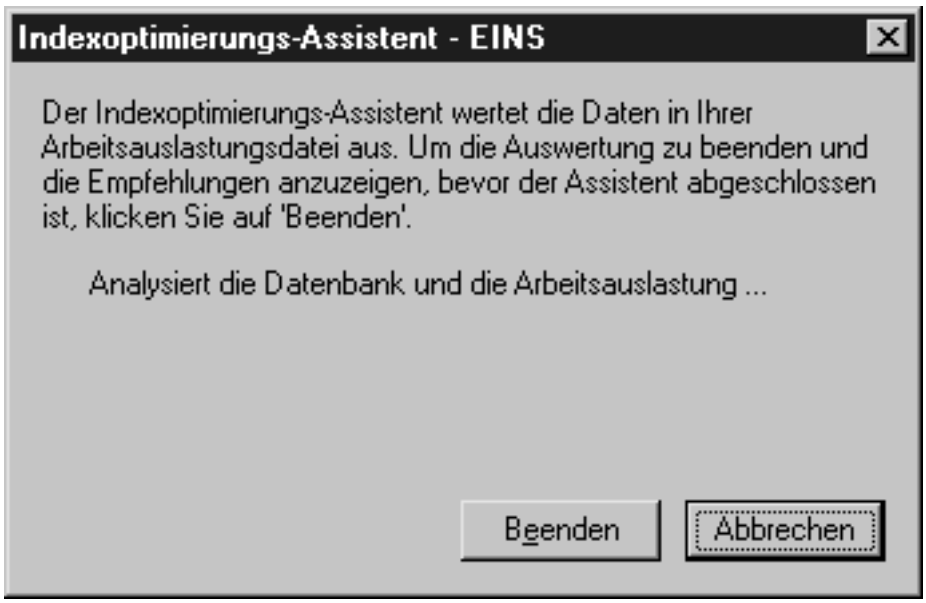


abbildung 24.26: fortschrittsmeldung der analyse

8. wenn der assistent die analyse beendet hat, erhalten sie indexempfehlungen (siehe abbildung 24.27).

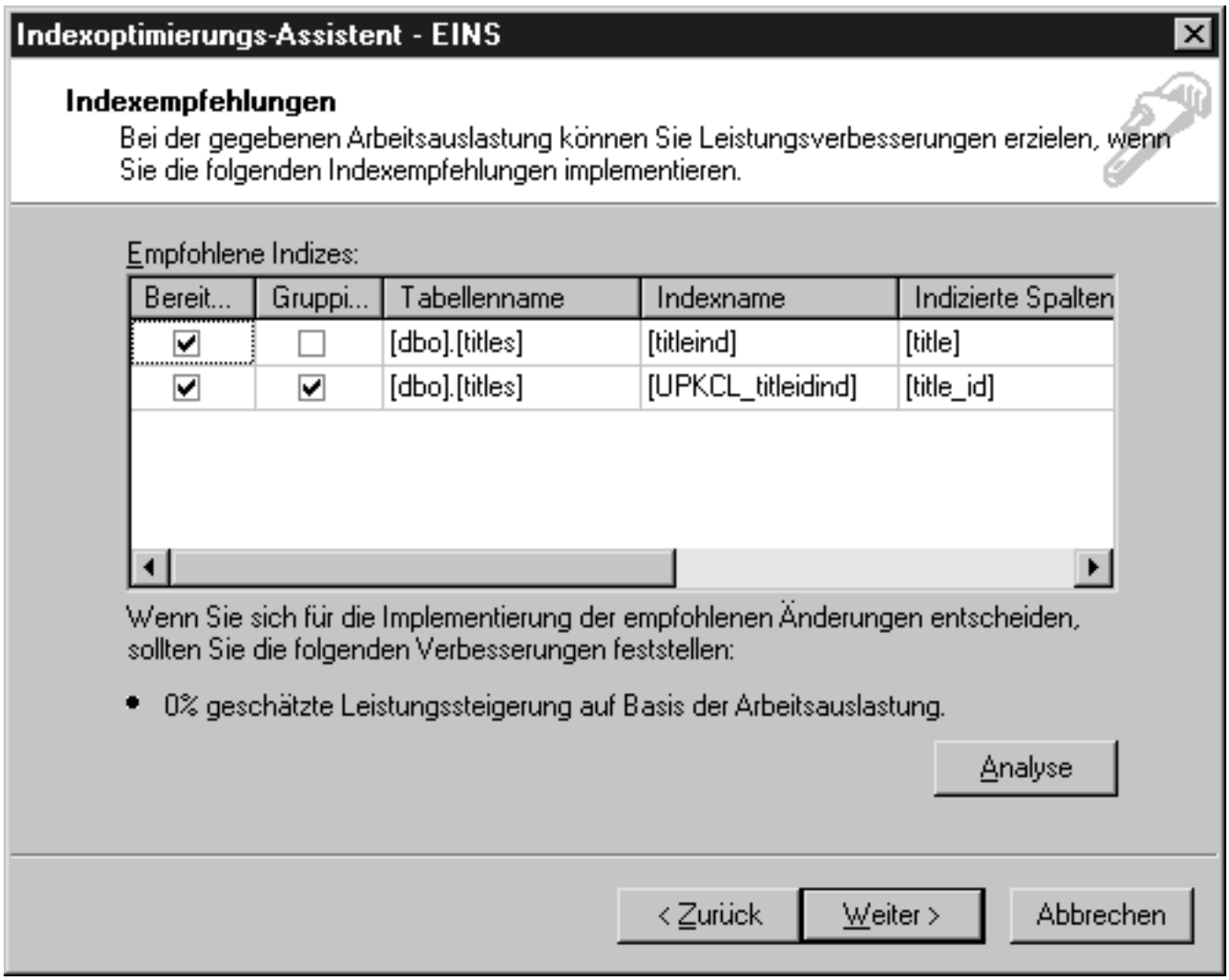


abbildung 24.27: indexempfehlungen

9. wenn sie auf die schaltfläche **analyse** klicken, erscheint das in abbildung 24.28 dargestellte dialogfeld. wählen sie im drop-down-listenfeld **berichte** einen bericht aus, und klicken sie auf **speichern**.

bild

abbildung 24.28: analyse in bericht speichern

10. nachdem sie den bericht gespeichert haben, klicken sie auf **schliessen**, um zum dialogfeld **indexempfehlungen** zurückzukehren. klicken sie auf **weiter**. damit kommen sie zum dialogfeld **termin für auftrag zur indexaktualisierung planen** (siehe abbildung 24.29). wenn sie das kontrollkästchen **skriptdatei speichern** einschalten, wird das dialogfeld **datei speichern unter** geöffnet, in welchem sie einen namen für das skript eingeben können, im beispiel indexempf. klicken sie dann auf **weiter**.

bild

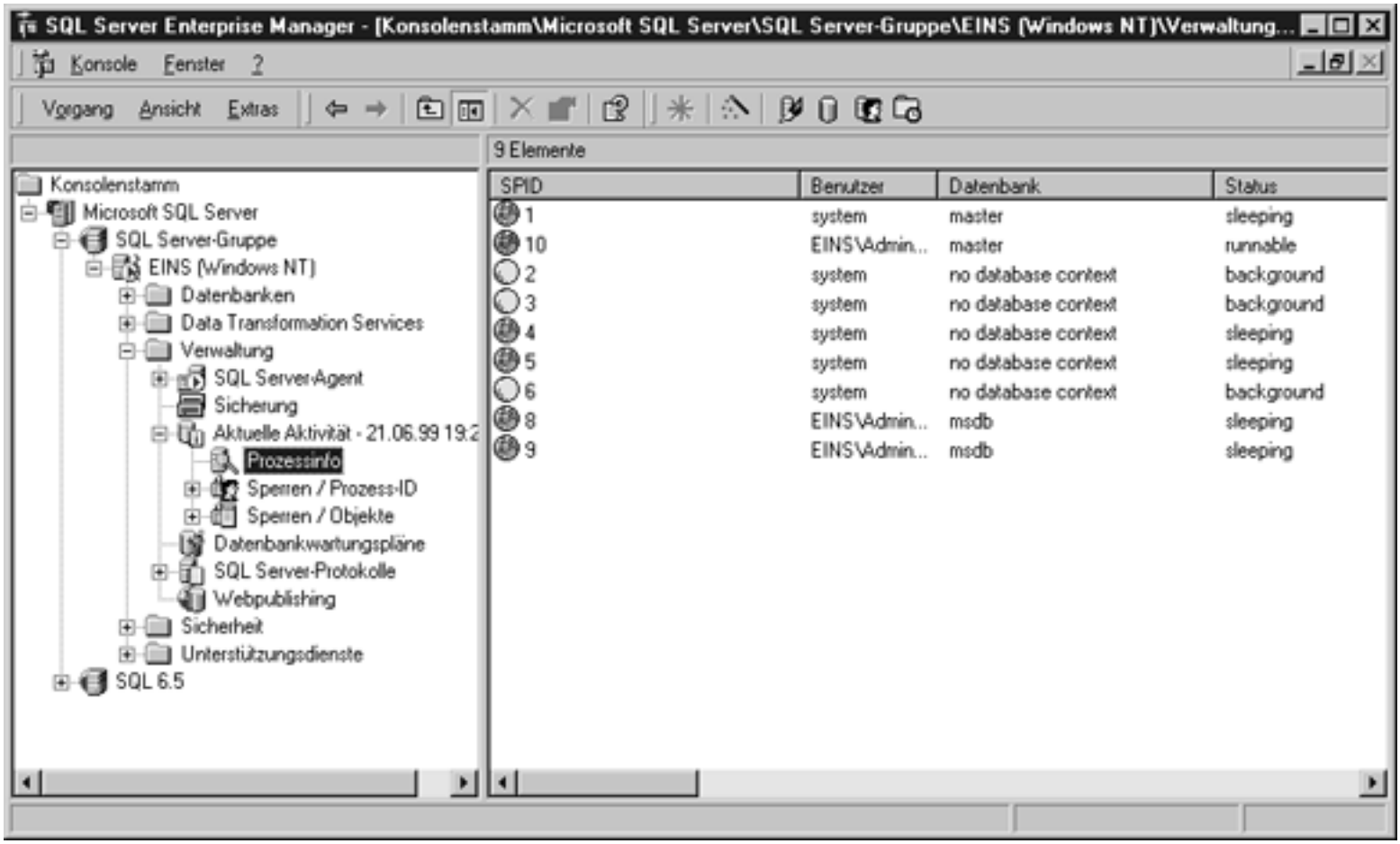
abbildung 24.29: termin für indexaktualisierung planen

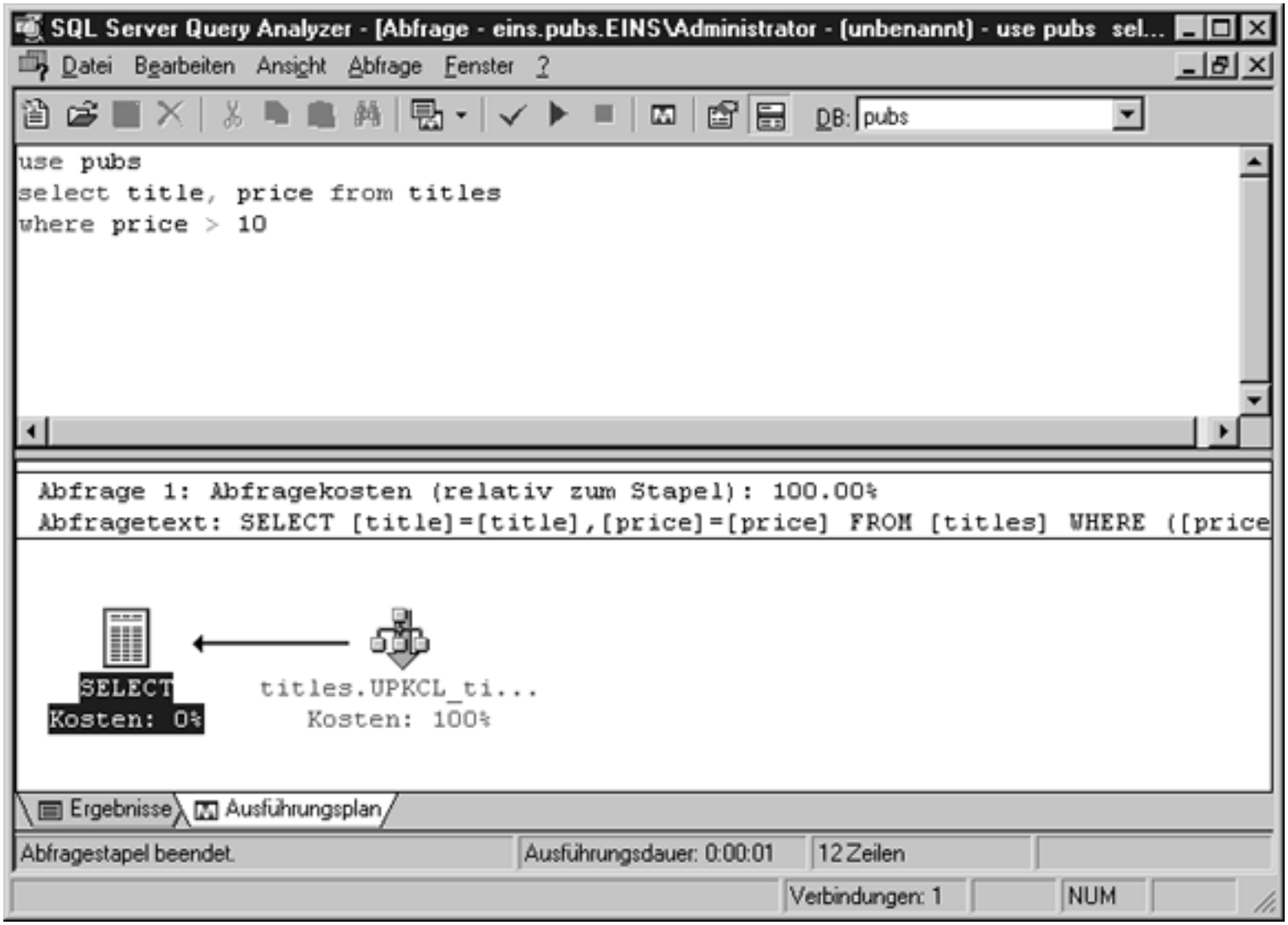
11. das letzte dialogfeld des assistenten erläutert, was der assistent unternimmt bzw. was sie unternehmen können - je nach den von ihnen getroffenen einstellungen (siehe abbildung 24.30).

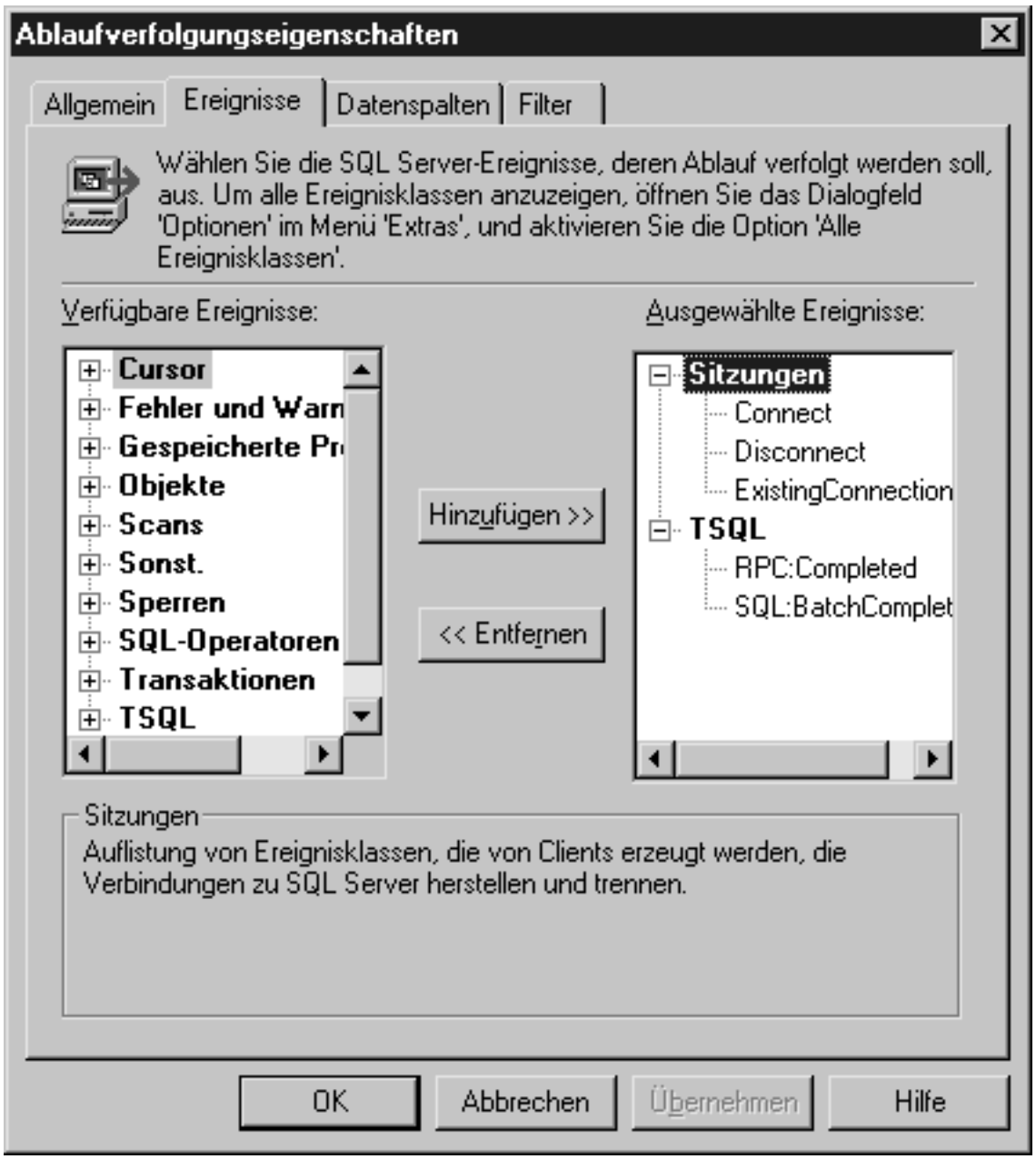
Bild

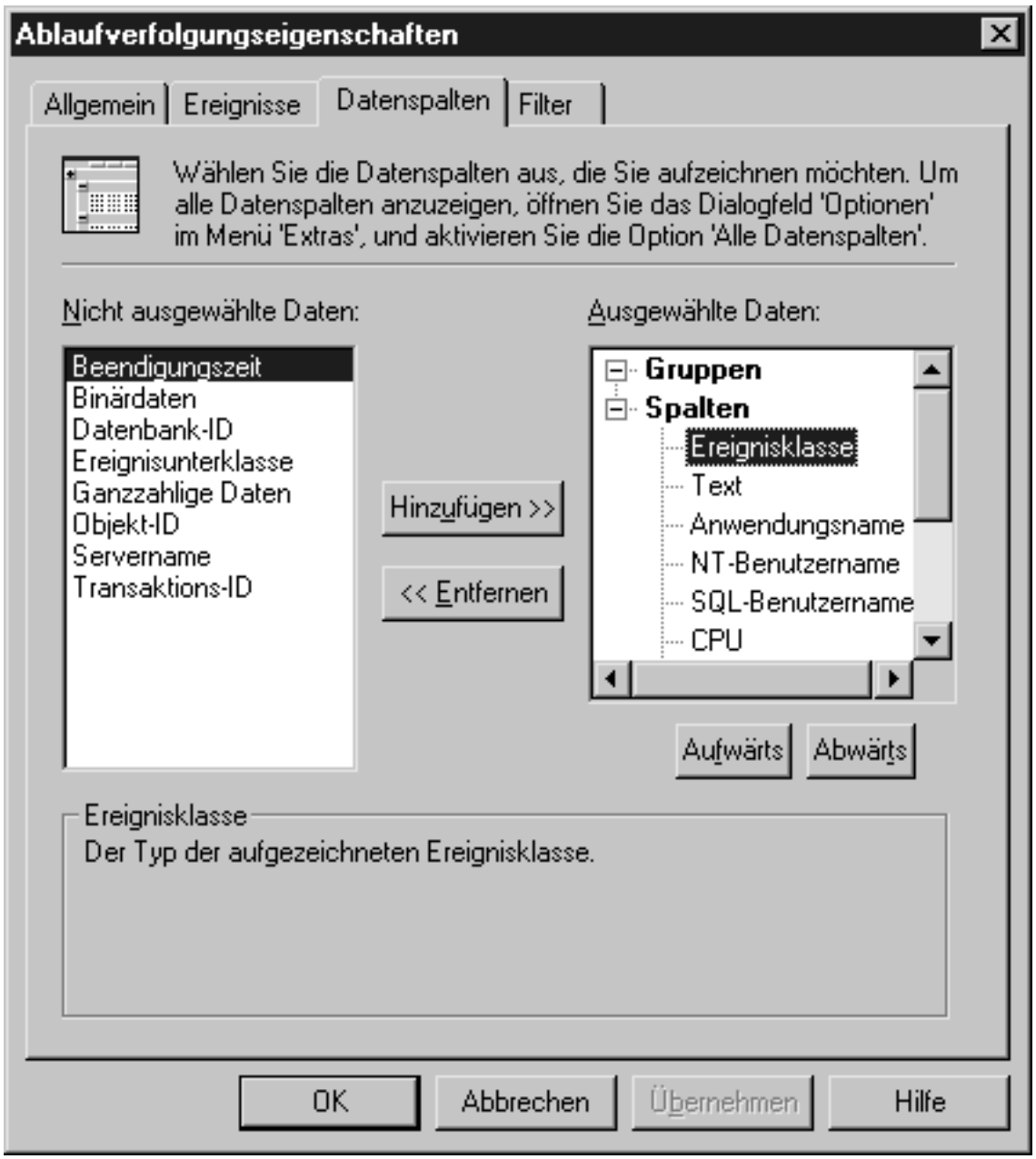
abbildung 24.30: letztes dialogfeld des indexoptimierungs-assistenten

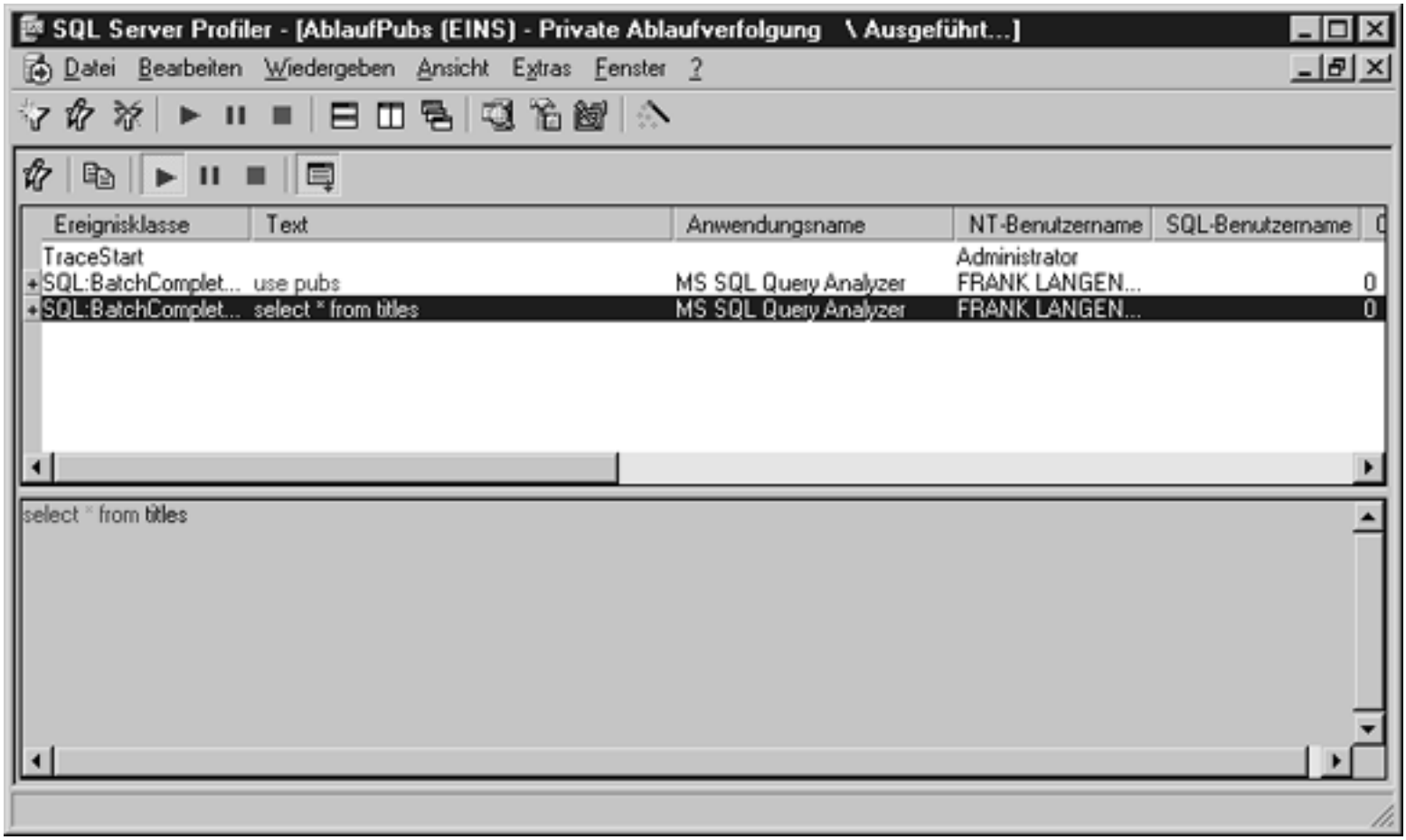
12. klicken sie auf **fertigstellen**. nach kurzer zeit erscheint die meldung »der indexoptimierungs-assistent wurde erfolgreich beendet.«

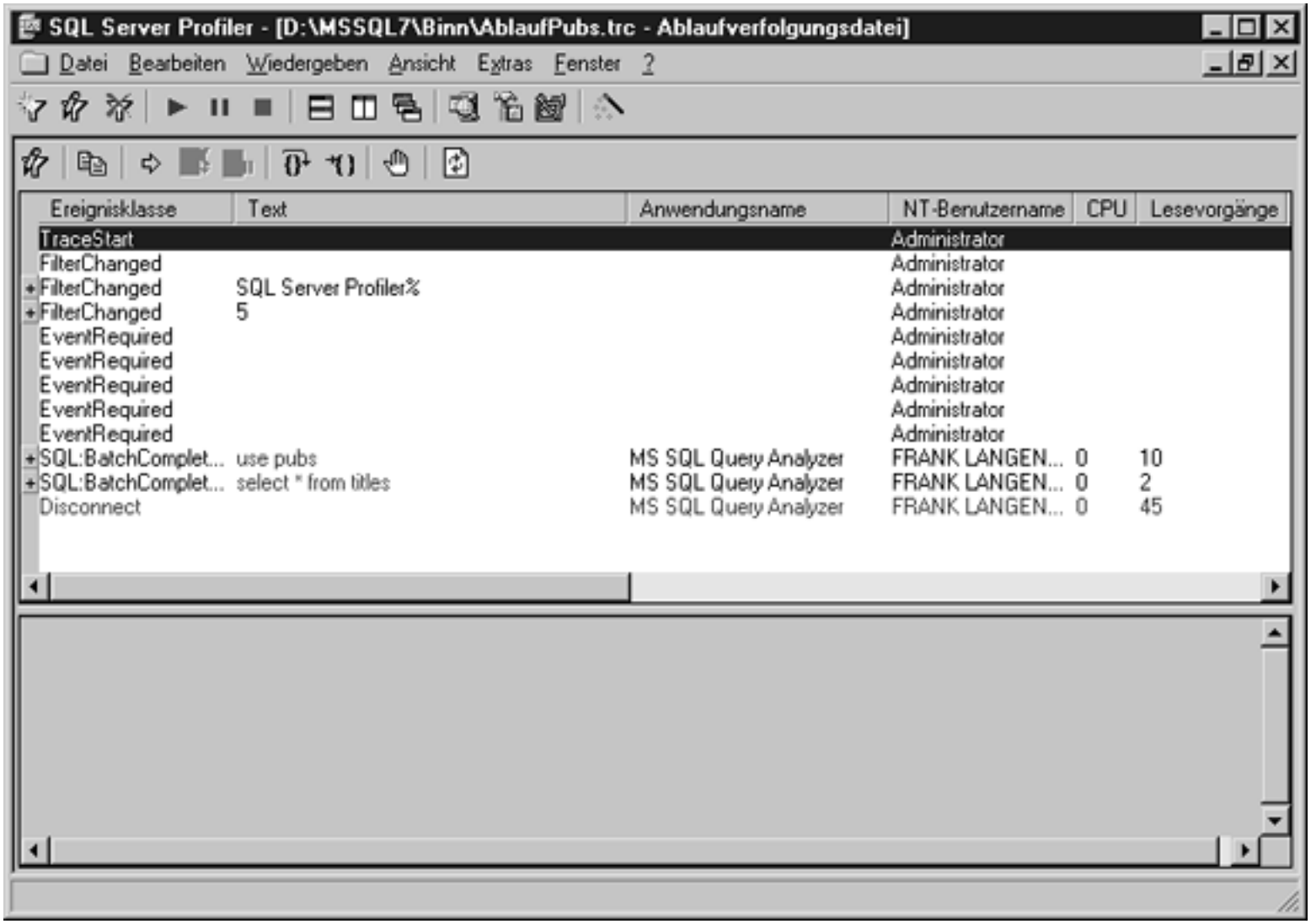

SQL Server Enterprise Manager - [Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Verwaltung...
Konsole
Fenster
?
Vorgang
Ansicht
Extras
9 Elemente
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation Services
Verwaltung
SQL Server-Agent
Sicherung
Aktuelle Aktivität - 21.06.99 19:2
Prozessinfo
Sperren / Prozess-ID
Sperren / Objekte
Datenbankwartungspläne
SQL Server-Protokolle
Webpublishing
Sicherheit
Unterstützungsdienste
SQL 6.5
SPID
Benutzer
Datenbank
Status
1 system master sleeping
10 EINS\Admin... master runnable
2 system no database context background
3 system no database context background
4 system no database context sleeping
5 system no database context sleeping
6 system no database context background
8 EINS\Admin... msdb sleeping
9 EINS\Admin... msdb sleeping

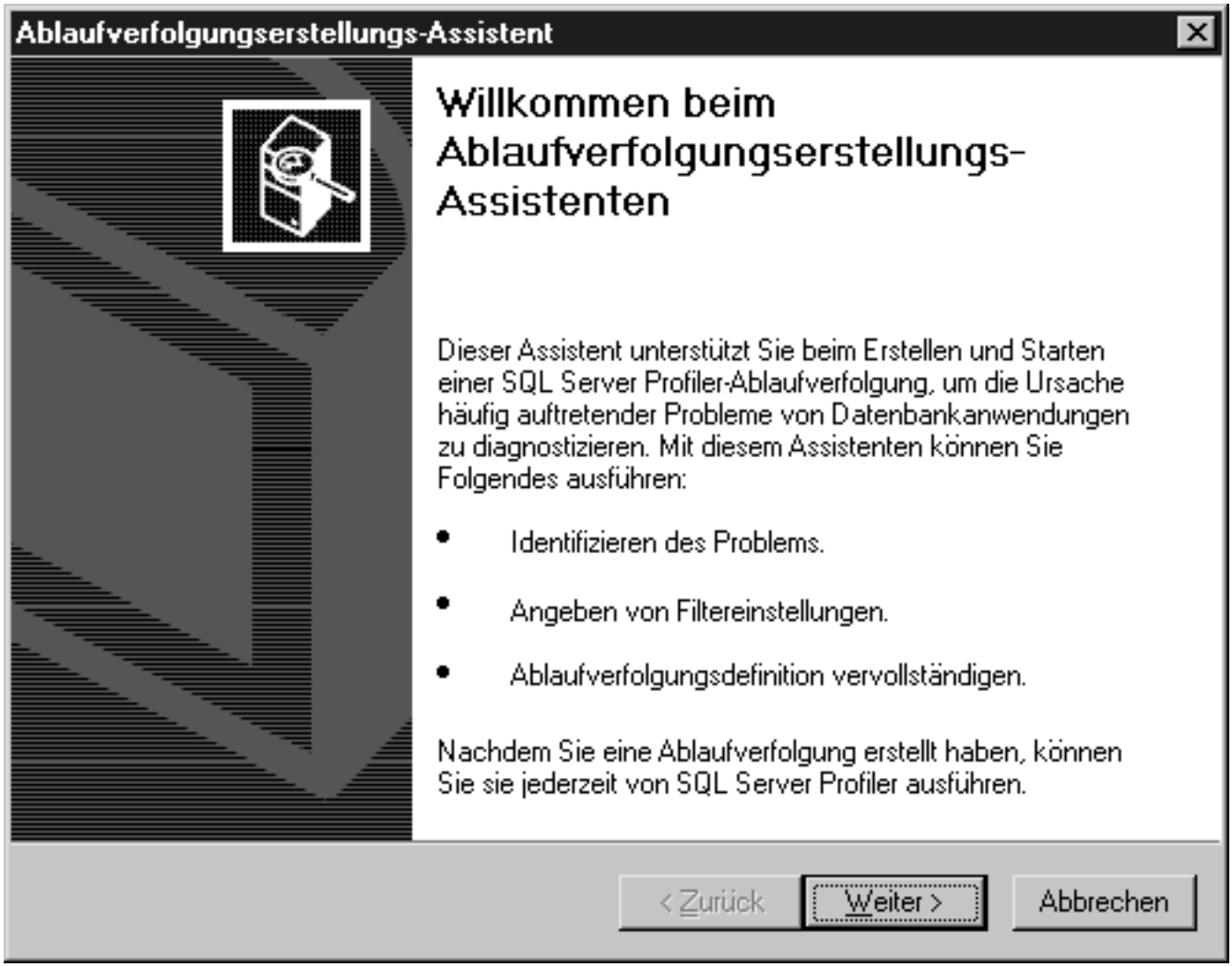

SQL Server Query Analyzer - [Abfrage - eins.pubs.EINS\Administrator - (unbenannt) - use pubs sel...
Datei Bearbeiten Ansicht Abfrage Fenster ?
DB: pubs
use pubs
select title, price from titles
where price > 10
Abfrage 1: Abfragekosten (relativ zum Stapel): 100.00%
Abfragetext: SELECT [title]=[title],[price]=[price] FROM [titles] WHERE ([price
SELECT
Kosten: 0%
titles.UPKCL_ti...
Kosten: 100%
Ergebnisse
Ausführungsplan
Abfragestapel beendet.
Ausführungsdauer: 0:00:01
12 Zeilen
Verbindungen: 1
NUM


Ablaufverfolgungseigenschaften
Allgemein
Ereignisse
Datenspalten
Filter
Wählen Sie die SQL Server-Ereignisse, deren Ablauf verfolgt werden soll, aus. Um alle Ereignisklassen anzuzeigen, öffnen Sie das Dialogfeld 'Optionen' im Menü 'Extras', und aktivieren Sie die Option 'Alle Ereignisklassen'.
Verfügbare Ereignisse:
Cursor
Fehler und Warn
Gespeicherte Pr
Objekte
Scans
Sonst.
Sperren
SQL-Operatoren
Transaktionen
TSQL
Hinzufügen >>
<< Entfernen
Ausgewählte Ereignisse:
Sitzungen
Connect
Disconnect
ExistingConnection
TSQL
RPC:Completed
SQL:BatchComplet
Sitzungen
Auflistung von Ereignisklassen, die von Clients erzeugt werden, die Verbindungen zu SQL Server herstellen und trennen.
OK
Abbrechen
Übernehmen
Hilfe

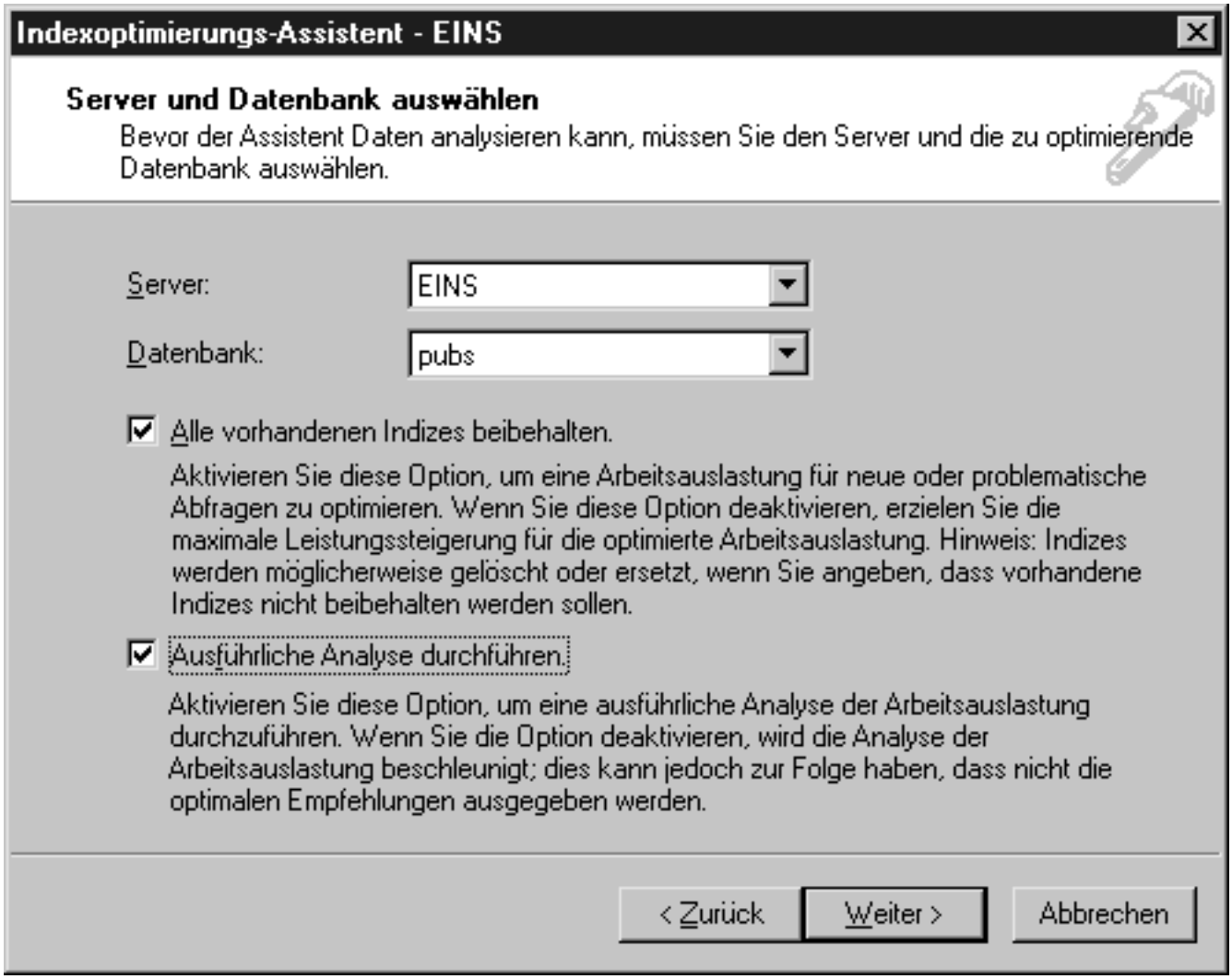

Ablaufverfolgungseigenschaften
Allgemein
Ereignisse
Datenspalten
Filter
Wählen Sie die Datenspalten aus, die Sie aufzeichnen möchten. Um alle Datenspalten anzuzeigen, öffnen Sie das Dialogfeld 'Optionen' im Menü 'Extras', und aktivieren Sie die Option 'Alle Datenspalten'.
Nicht ausgewählte Daten:
Beendigungszeit
Binärdaten
Datenbank-ID
Ereignisunterklasse
Ganzzahlige Daten
Objekt-ID
Servername
Transaktions-ID
Hinzufügen >>
<< Entfernen
Ausgewählte Daten:
Gruppen
Spalten
Ereignisklasse
Text
Anwendungsname
NT-Benutzername
SQL-Benutzername
CPU
Aufwärts
Abwärts
Ereignisklasse
Der Typ der aufgezeichneten Ereignisklasse.
OK
Abbrechen
Übernehmen
Hilfe

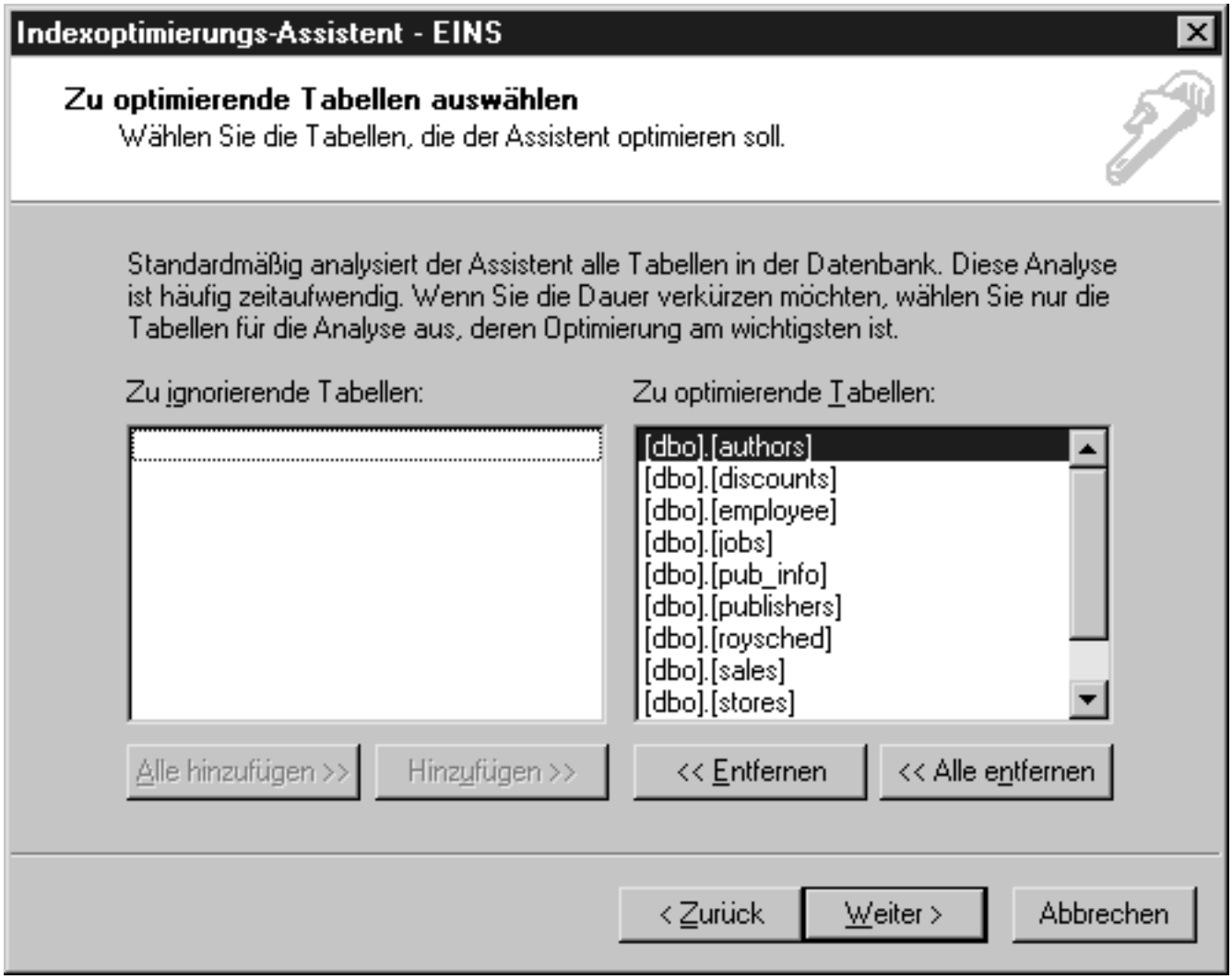

SQL Server Profiler - [AblaufPubs (EINS) - Private Ablaufverfolgung \ Ausgeführt...]
Datei Bearbeiten Wiedergeben Ansicht Extras Fenster ?
Ereignisklasse
Text
Anwendungsname
NT-Benutzername
SQL-Benutzername
TraceStart
Administrator
SQL:BatchComplet... use pubs
MS SQL Query Analyzer
FRANK LANGEN...
0
SQL:BatchComplet... select * from titles
MS SQL Query Analyzer
FRANK LANGEN...
0
select * from titles

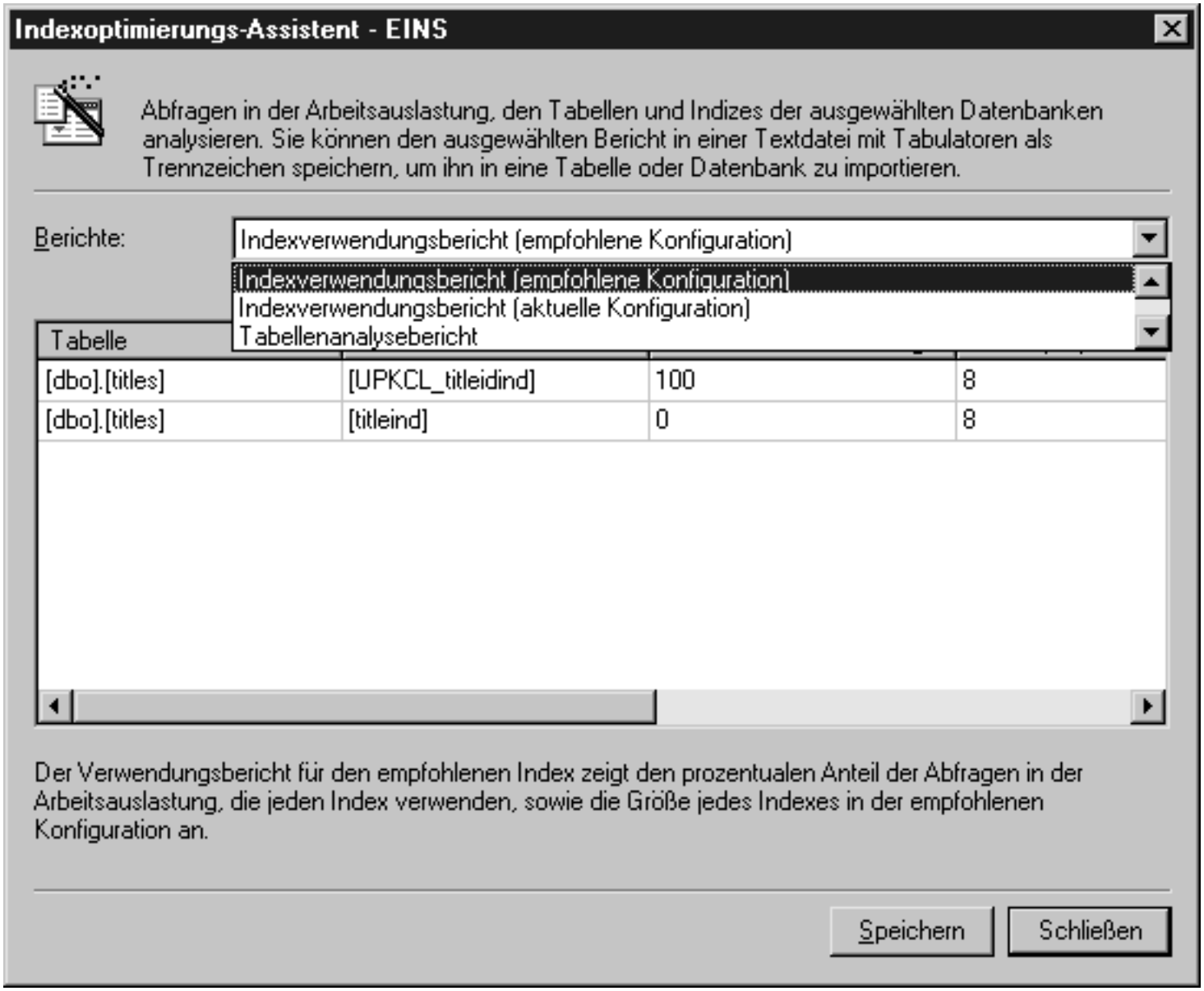

SQL Server Profiler - [D:\MSSQL7\Binn\AblaufPubs.trc - Ablaufverfolgungsdatei]
Datei Bearbeiten Wiedergeben Ansicht Extras Fenster ?
Ereignisklasse Text Anwendungsname NT-Benutzername CPU Lesevorgänge
TraceStart Administrator
FilterChanged Administrator
FilterChanged SQL Server Profiler% Administrator
FilterChanged 5 Administrator
EventRequired Administrator
EventRequired Administrator
EventRequired Administrator
EventRequired Administrator
EventRequired Administrator
SQL:BatchComplet... use pubs MS SQL Query Analyzer FRANK LANGEN... 0 10
SQL:BatchComplet... select * from titles MS SQL Query Analyzer FRANK LANGEN... 0 2
Disconnect MS SQL Query Analyzer FRANK LANGEN... 0 45

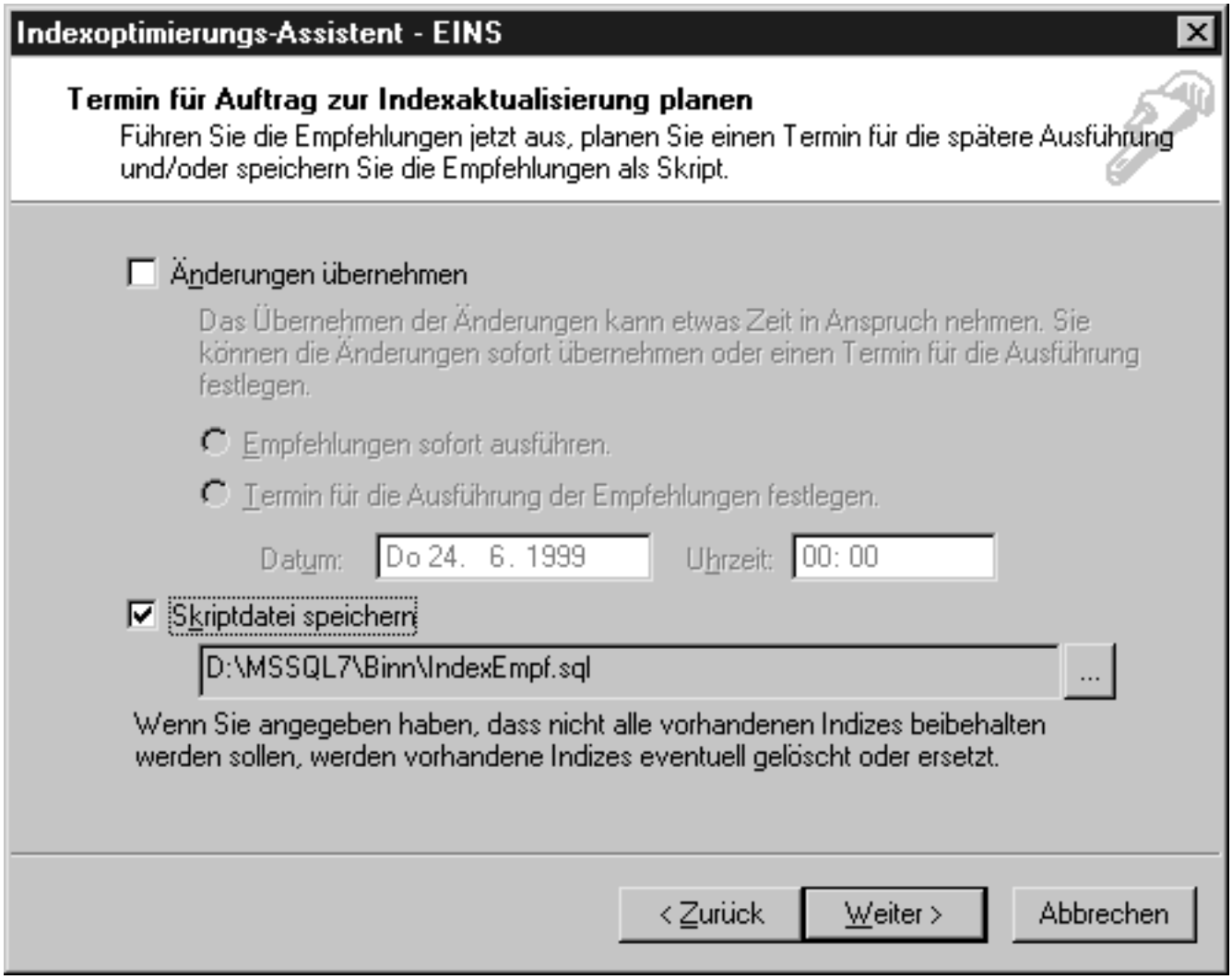

Ablaufverfolgungserstellungs-Assistent
Willkommen beim Ablaufverfolgungserstellungs-Assistenten
Dieser Assistent unterstützt Sie beim Erstellen und Starten einer SQL Server Profiler-Ablaufverfolgung, um die Ursache häufig auftretender Probleme von Datenbankanwendungen zu diagnostizieren. Mit diesem Assistenten können Sie Folgendes ausführen:
Identifizieren des Problems.
Angeben von Filtereinstellungen.
Ablaufverfolgungsdefinition vervollständigen.
Nachdem Sie eine Ablaufverfolgung erstellt haben, können Sie sie jederzeit von SQL Server Profiler ausführen.
< Zurück
Weiter >
Abbrechen


Ablaufverfolgungserstellungs-Assistent - EINS
Ablaufverfolgungserstellungs-Assistenten beenden
Sie haben die Schritte abgeschlossen, die zum Erstellen einer Ablaufverfolgung erforderlich sind. Der Assistent erstellt eine Ablaufverfolgung mit der folgenden Definition.
Ablaufverfolgungsname:
agen mit der schlechtesten Leistung
Ablaufverfolgungseigenschaften:
Ereignisse:
RPC:Completed
SQL:BatchCompleted
Datenspalten:
Gruppen:
Dauer
Spalten:
Ereignisklasse
Text
< Zurück
Fertig stellen
Abbrechen


Indexoptimierungs-Assistent - EINS
Server und Datenbank auswählen
Bevor der Assistent Daten analysieren kann, müssen Sie den Server und die zu optimierende Datenbank auswählen.
Server:
EINS
Datenbank:
pubs
Alle vorhandenen Indizes beibehalten.
Aktivieren Sie diese Option, um eine Arbeitsauslastung für neue oder problematische Abfragen zu optimieren. Wenn Sie diese Option deaktivieren, erzielen Sie die maximale Leistungssteigerung für die optimierte Arbeitsauslastung. Hinweis: Indizes werden möglicherweise gelöscht oder ersetzt, wenn Sie angeben, dass vorhandene Indizes nicht beibehalten werden sollen.
Ausführliche Analyse durchführen.
Aktivieren Sie diese Option, um eine ausführliche Analyse der Arbeitsauslastung durchzuführen. Wenn Sie die Option deaktivieren, wird die Analyse der Arbeitsauslastung beschleunigt; dies kann jedoch zur Folge haben, dass nicht die optimalen Empfehlungen ausgegeben werden.
< Zurück
Weiter >
Abbrechen


Indexoptimierungs-Assistent - EINS
Zu optimierende Tabellen auswählen
Wählen Sie die Tabellen, die der Assistent optimieren soll.
Standardmäßig analysiert der Assistent alle Tabellen in der Datenbank. Diese Analyse ist häufig zeitaufwendig. Wenn Sie die Dauer verkürzen möchten, wählen Sie nur die Tabellen für die Analyse aus, deren Optimierung am wichtigsten ist.
Zu ignorierende Tabellen:
Zu optimierende Tabellen:
[dbo].[authors]
[dbo].[discounts]
[dbo].[employee]
[dbo].[jobs]
[dbo].[pub_info]
[dbo].[publishers]
[dbo].[roysched]
[dbo].[sales]
[dbo].[stores]
Alle hinzufügen >>
Hinzufügen >>
<< Entfernen
<< Alle entfernen
< Zurück
Weiter >
Abbrechen


Indexoptimierungs-Assistent - EINS
Abfragen in der Arbeitsauslastung, den Tabellen und Indizes der ausgewählten Datenbanken analysieren. Sie können den ausgewählten Bericht in einer Textdatei mit Tabulatoren als Trennzeichen speichern, um ihn in eine Tabelle oder Datenbank zu importieren.
Berichte:
Indexverwendungsbericht (empfohlene Konfiguration)
Indexverwendungsbericht (empfohlene Konfiguration)
Indexverwendungsbericht (aktuelle Konfiguration)
Tabellenanalysebericht
Tabelle
[dbo].[titles] [UPKCL_titleidind] 100 8
[dbo].[titles] [titleind] 0 8
Der Verwendungsbericht für den empfohlenen Index zeigt den prozentualen Anteil der Abfragen in der Arbeitsauslastung, die jeden Index verwenden, sowie die Größe jedes Indexes in der empfohlenen Konfiguration an.
Speichern
Schließen


Indexoptimierungs-Assistent - EINS
Termin für Auftrag zur Indexaktualisierung planen
Führen Sie die Empfehlungen jetzt aus, planen Sie einen Termin für die spätere Ausführung und/oder speichern Sie die Empfehlungen als Skript.
Änderungen übernehmen
Das Übernehmen der Änderungen kann etwas Zeit in Anspruch nehmen. Sie können die Änderungen sofort übernehmen oder einen Termin für die Ausführung festlegen.
Empfehlungen sofort ausführen.
Termin für die Ausführung der Empfehlungen festlegen.
Datum:
Do 24. 6. 1999
Uhrzeit:
00: 00
Skriptdatei speichern
D:\MSSQL7\Binn\IndexEmpf.sql
...
Wenn Sie angegeben haben, dass nicht alle vorhandenen Indizes beibehalten werden sollen, werden vorhandene Indizes eventuell gelöscht oder ersetzt.
< Zurück
Weiter >
Abbrechen


Indexoptimierungs-Assistent - EINS
Indexoptimierungs-Assistenten beenden
Sie haben die Schritte beendet, die zum Optimieren des Indexes erforderlich sind. Wählen Sie 'Fertig stellen', um die gewählten Optionen anzuwenden.
Wenn Sie sich für die direkte Ausführung entscheiden, wird die Datenbank auf die empfohlene Konfiguration aktualisiert.
Wenn Sie einen Termin für einen Auftrag planen möchten, können Sie den Auftrag in SQL Server Enterprise Manager anzeigen.
Wenn Sie sich für das Speichern des Skripts in einer Datei entscheiden, können Sie das Skript mit einem Texteditor anzeigen oder bearbeiten.
Sie sollten eine Sicherung Ihrer Datenbank durchführen, bevor Sie die empfohlene Konfiguration durchführen.
< Zurück
Fertig stellen
Abbrechen

kapitel 25 fehler

25.1 der teufel steckt im detail

wer in irgendeiner form mit pcs zu tun hat, lebt von anbeginn an mit fehlern. es ist schon mehrfach nachgewiesen worden, daß es beim heutigen stand der software-technologie praktisch nicht möglich ist, vollkommen fehlerfreie programme zu schreiben. kapitel 7 ist im rahmen der sicherung und wiederherstellung bereits auf fehlerursachen eingegangen. während es dort vor allem darum ging, mögliche datenverluste zu vermeiden, beschäftigt sich dieses kapitel in erster linie mit logischen fehlern, die auf der anweisungsebene liegen.

oftmals sind es nur kleinere dinge, die man übersehen hat und die zu unverständlichen fehlermeldungen führen. im buch finden sie einige hinweise, die sich auf die schreibweise von befehlen - wie etwa raiseerror statt raiserror - beziehen. ein falsch geschriebener befehl zieht zwar in der regel noch keinen datenverlust nach sich, dennoch ist es zumindest ärgerlich, wenn auf diese weise eine ganze transaktion zum scheitern verurteilt ist.

bevor man fehler behandeln kann, muß man natürlich die ursache des fehlers kennen. einen ersten anhaltspunkt bieten die fehlermeldungen von sql server. darüber hinaus stellt sql server verschiedene werkzeuge bereit, mit denen man fehlern auf die spur kommen kann.

25.2 fehlermeldungen

gerade wenn sie ad-hoc-abfragen im sql server query analyzer ausführen, kann es leicht passieren, daß sie auf ein nicht vorhandenes datenbankobjekt zugreifen. probieren sie als beispiel die folgende anweisung aus:

```
use pubs
select * from titel
```

als ergebnis erhalten sie die fehlermeldung:

```
server: nachr.-nr. 208, schweregrad 16, status 1, zeile 1
ungültiger objektname 'titel'.
```

ein blick in das datenbankdiagramm der datenbank pubs zeigt, daß die tabelle nicht titel, sondern titles heißt.

insbesondere während der entwicklung einer datenbank oder einer datenbankanwendung empfiehlt es sich, ein aktuelles datenbankdiagramm griffbereit zu haben. wie kapitel 6 gezeigt hat, lassen sich datenbankdiagramme in sql server schnell und problemlos erstellen.

in vielen situationen läßt sich allerdings nicht so schnell auf die fehlerursache schließen. deshalb ist es hilfreich, wenn sie die fehlermeldungen interpretieren und bei bedarf zusätzliche informationen einholen können.

wie das obige beispiel zeigt, besteht eine fehlermeldung von sql server aus folgenden elementen:

- fehlernummer (nachr.-nr.)
- schweregrad
- statuscode
- zeilennummer
- beschreibung

fehlernummer

jeder fehlermeldung ist in sql server eine eindeutige fehlernummer zugeordnet. die von sql server definierten oder reservierten fehlernummern umfassen den bereich 0 bis 50000. für benutzerdefinierte fehlermeldungen sind fehlernummern größer als 50000 bis maximal 2147483647 (231 - 1) vorgesehen. benutzerdefinierte fehlermeldungen lassen sich mit der anweisung raiserror zurückgeben.

schweregrad

sql server kategorisiert alle fehler nach dem schweregrad. damit haben sie bereits einen anhaltspunkt zur herkunft des fehlers. festgelegt sind die ebenen 0 bis 25 entsprechend tabelle 25.1.

schweregrad	beschreibung
0 bis 10	informationsmeldungen, die ihre ursache in den eingegebenen informationen haben und keine eigentlichen fehler darstellen
11 bis 16	probleme, die der benutzer selbst beheben kann
17	unzureichende ressourcen, beispielsweise benutzerverbindungen oder sperren. die probleme kann der systemadministrator und in bestimmten fällen auch der datenbankbesitzer beheben.
18	mittelschwerer interner fehler (software-problem).
19	ressourcenfehler. zeigt an, daß ein nicht konfigurierbarer interner grenzwert überschritten wurde.
20	fataler fehler im aktuellen prozeß.
21	fataler fehler in datenbankprozessen.
22	tabelle oder index beschädigt.
23	datenbankintegrität zweifelhaft.

24	hardware-fehler.
25	systemfehler.

tabelle 25.1: klassifizierung von fehlermeldungen nach dem schweregrad

statuscode

der statuscode - eine ganze zahl zwischen 1 und 127 - ist nur für die supporttechniker von microsoft von bedeutung und weist auf eine bestimmte stelle im code von sql server hin.

zeilennummer

die zeilennummer bezeichnet die position der anweisung, die den fehler verursacht hat, in bezug auf den stapel oder die gespeicherte prozedur.

beschreibung

da fehlernummern kaum aussagekräftig sind, wird eine fehlermeldung von einer sogenannten fehlermeldungszeichenfolge - einer unicode-zeichenfolge - begleitet.

sql server speichert die fehlernummern, beschreibungen und schweregrade in der systemtabelle sysmessages. kapitel 19 geht näher auf diese systemtabelle ein und erläutert auch, wie sie benutzerdefinierte fehlermeldungen erstellen und ausgeben.

25.2.1 suchen in der online-dokumentation

wenn ihnen die von sql server ausgegebene beschreibung des fehlers nicht genügt und sie weitere hinweise benötigen, suchen sie einfach nach dem fehlercode in der online-dokumentation. gehen sie auf die registerkarte **suchen**, geben sie die fehlernummer ein, und klicken sie auf **themen auflisten**. wählen sie dann das thema in der ergebnisliste aus. abbildung 25.1 zeigt die beschreibung für den fehler 208 des eingangs dargestellten beispiels.

Bild

abbildung 25.1: suchen des fehlers in der online-dokumentation

25.3 fehlerprotokolle

sql server zeichnet systemnachrichten und fehlermeldungen in einem fehlerprotokoll auf. dabei handelt es sich um eine textdatei, die sie mit einem normalen editor öffnen können. damit die aktuellen meldungen bei einem neustart von sql server nicht gleich wieder verschwinden, verwaltet sql server sogar sieben dateien: das aktuelle protokoll wird in die datei errorlog geschrieben, während die jeweils älteren fehlerprotokolle in den dateien errorlog.1 bis errorlog.6 zu finden sind. das standardverzeichnis für diese fehlerprotokolle ist \mssql7\log.

das folgende beispiel zeigt ein fehlerprotokoll für den server eins:

```
1999-06-22 09:44:53.63 kernel   microsoft sql server  7.00 -
 7.00.623 (intel x86)
    nov 27 1998 22:20:07
    copyright (c) 1988-1998 microsoft corporation
    standard edition on windows nt 4.0 (build 1381: service
 pack 4)

1999-06-22 09:44:54.17 kernel   copyright (c) 1988-1997
 microsoft corporation.
1999-06-22 09:44:54.18 kernel   alle rechte vorbehalten.
1999-06-22 09:44:54.18 kernel   protokolliert sql server-
meldungen in datei 'd:\mssql7\log\errorlog'.
1999-06-22 09:44:54.63 kernel   initconfig: anzahl der
 benutzerverbindungen wurde auf 32767 begrenzt.
1999-06-22 09:44:54.63 kernel   sql server startet mit
 prioritätsklasse 'normal' (1 cpu vorgefunden).
1999-06-22 09:44:55.28 kernel   user mode scheduler configured
 for thread processing
1999-06-22 09:44:57.65 server   directory size: 2559
1999-06-22 09:44:57.97 spid1    using dynamic lock allocation.
 [2500] lock blocks, [5000] lock owner blocks
1999-06-22 09:44:57.97 kernel   initialisiert distributed
 transaction coordinator.
1999-06-22 09:45:01.51 spid1    failed to obtain
 transactiondispenserinterface: xact_e_tmnotavailable
1999-06-22 09:45:01.72 spid1    startet datenbank 'master'.
1999-06-22 09:45:01.72 spid1    opening file
 d:\mssql7\data\master.mdf.
1999-06-22 09:45:01.98 spid1    opening file
 d:\mssql7\data\mastlog.ldf.
1999-06-22 09:45:02.95 spid1    lädt unicode-sortierreihenfolge
 von sql server .
1999-06-22 09:45:03.14 spid1    lädt nicht-unicode-
sortierreihenfolge und -zeichensatz von sql server .
1999-06-22 09:45:04.42 spid1    rollforward für 6 transaktionen
 in datenbank 'master' (1) ausgeführt.
1999-06-22 09:45:04.48 spid1    rollback für 0 transaktionen in
 datenbank 'master' (1) ausgeführt.
1999-06-22 09:45:05.07 spid1    startet datenbank 'model'.
1999-06-22 09:45:05.07 spid1    opening file
 d:\mssql7\data\model.mdf.
1999-06-22 09:45:05.55 spid1    opening file
 d:\mssql7\data\modellog.ldf.
1999-06-22 09:45:06.59 spid1    löscht tempdb-datenbank.
1999-06-22 09:45:09.08 spid1    creating file
```

```
 d:\mssql7\data\tempdb.mdf.
1999-06-22 09:45:09.91 spid1    closing file
 d:\mssql7\data\tempdb.mdf.
1999-06-22 09:45:10.00 spid1    creating file
 d:\mssql7\data\templog.ldf.
1999-06-22 09:45:10.36 spid1    closing file
 d:\mssql7\data\templog.ldf.
1999-06-22 09:45:10.56 spid1    opening file
 d:\mssql7\data\tempdb.mdf.
1999-06-22 09:45:10.58 spid1    opening file
 d:\mssql7\data\templog.ldf.
1999-06-22 09:45:11.91 spid1    closing file
 d:\mssql7\data\tempdb.mdf.
1999-06-22 09:45:11.98 spid1    closing file
 d:\mssql7\data\templog.ldf.
1999-06-22 09:45:12.00 spid1    startet datenbank 'tempdb'.
1999-06-22 09:45:12.00 spid1    opening file
 d:\mssql7\data\tempdb.mdf.
1999-06-22 09:45:12.20 spid1    opening file
 d:\mssql7\data\templog.ldf.
1999-06-22 09:45:13.52 spid1    servername ist 'eins'.
1999-06-22 09:45:13.58 kernel   verwendet 'sqlevn70.dll',
 version '7.00.623'.
1999-06-22 09:45:13.68 kernel   verwendet 'opends60.dll',
 version '7.00.00.0623'.
1999-06-22 09:45:14.07 ods      verwendet 'ssnmpn70.dll',
 version '7.0.623', zur überwachung von '\\.\pipe\sql\query'.
1999-06-22 09:45:14.12 ods      verwendet 'ssmsso70.dll',
 version '7.0.623', zur überwachung von '1433'.
1999-06-22 09:45:14.31 ods      verwendet 'ssmsrp70.dll',
 version '7.0.623', zur überwachung von 'eins'.
1999-06-22 09:45:23.58 spid1    ods does not start in 10
 seconds
1999-06-22 09:45:24.08 spid6    startet datenbank 'msdb'.
1999-06-22 09:45:24.08 spid6    opening file
 d:\mssql7\data\msdbdata.mdf.
1999-06-22 09:45:24.08 spid7    startet datenbank 'pubs'.
1999-06-22 09:45:24.08 spid7    opening file
 d:\mssql7\data\pubs.mdf.
1999-06-22 09:45:24.09 spid8    startet datenbank 'northwind'.
1999-06-22 09:45:24.09 spid8    opening file
 d:\mssql7\data\northwnd.mdf.
1999-06-22 09:45:24.09 spid9    startet datenbank 'rptitel'.
1999-06-22 09:45:24.09 spid9    opening file
 d:\mssql7\data\rptitel_daten.mdf.
1999-06-22 09:45:24.60 spid6    opening file
```

```
 d:\mssql7\data\msdblog.ldf.
1999-06-22 09:45:24.60 spid7    opening file
 d:\mssql7\data\pubs_log.ldf.
1999-06-22 09:45:24.72 spid9    opening file
 d:\mssql7\data\rptitel_protokoll.ldf.
1999-06-22 09:45:24.72 spid8    opening file
 d:\mssql7\data\northwnd.ldf.
1999-06-22 09:45:26.66 spid7    startet datenbank 'lotto'.
1999-06-22 09:45:26.66 spid7    opening file
 d:\mssql7\data\lotto.mdf.
1999-06-22 09:45:27.29 spid7    opening file
 d:\mssql7\data\lotto_log.ldf.
1999-06-22 09:45:29.57 spid1    wiederherstellung
 abgeschlossen.
1999-06-22 09:45:30.11 spid1    die unicode-sortierreihenfolge
 von sql server ist:
1999-06-22 09:45:30.11 spid1            'english' (id = 1033).
1999-06-22 09:45:30.11 spid1            vergleichsart = 196609.
1999-06-22 09:45:30.11 spid1    die nicht-unicode-
sortierreihenfolge von sql server ist:
1999-06-22 09:45:30.11 spid1            'nocase_iso' (id = 52).
1999-06-22 09:45:30.12 spid1    der nicht-unicode-zeichensatz
 von sql server ist:
1999-06-22 09:45:30.12 spid1            'iso_1' (id = 1).
1999-06-22 14:06:25.21 spid8    'xpsql70.dll'-version
 '1998.11.13' wird verwendet, um die erweiterte gespeicherte
 prozedur 'xp_msver' auszuführen.
```

wie sie dem protokoll entnehmen können, zeichnet sql server die startsequenz, verschiedene systemeinstellungen und auch laufende aktionen auf. falls möglich, sollten sie sich die zeit nehmen und ein fehlerprotokoll (das im beispiel wiedergegebene oder ein protokoll ihres servers) genauer unter die lupe nehmen.

auch wenn sie sich im normalfall nicht weiter um das fehlerprotokoll kümmern müssen, empfiehlt es sich, das protokoll einer erfolgreich verlaufenen sql-server-sitzung auszudrucken und zu archivieren, damit sie bei problemen eine vergleichsmöglichkeit zur hand haben.

25.4 sqldiag

das dienstprogramm sqldiag erstellt einen umfangreichen bericht, der diagnoseinformationen sowohl zu sql server als auch zum servercomputer enthält. die syntax von sqldiag lautet:

```
sqldiag [[-uanmeldename] [-pkennwort] | [-e]]
```

```
[-o ausgabedateiname]
```

bei *anmeldename* und *kennwort* ist die groß-/kleinschreibung zu beachten. nach der installation von sql server gilt der anmeldename sa und das standardkennwort null (d.h. keine kennworteingabe erforderlich). wenn sie den schalter -e angeben, verwendet sql server eine vertraute verbindung, die nur bei windows nt möglich ist.

mit dem parameter -o können sie eine ausgabedatei spezifizieren. ist dieser parameter nicht angegeben, verwendet sql server die standarddateien sqldiag.trc und sqldiag.txt.

die ausgabedatei enthält unter anderem den inhalt der sieben fehlerprotokolle, angaben zur systemumgebung, versionsinformationen zu dll-dateien, konfigurationseinstellungen und die ausgaben verschiedener gespeicherter systemprozeduren.

das dienstprogramm starten sie zum beispiel mit der folgenden anweisung von der eingabeaufforderung:

```
sqldiag -e -odiagnose.txt
```

die anmeldung erfolgt hier mit windows-nt-authentifizierung. die ergebnisse schreibt sqldiag in die datei diagnose.txt.

25.5 dbcc-anweisungen

die transact-sql-anweisungen zur datenbankkonsistenzprüfung (dbcc - database consistency checker) setzt man normalerweise für die wartung von datenbanken und datenbankobjekten ein. diese anweisungen lassen sich folgenden kategorien zuordnen:

- verwaltungsanweisungen
- überprüfungsanweisungen
- statusanweisungen
- verschiedene anweisungen

im anhang c finden sie eine übersicht zu diesen anweisungen zusammen mit einer kurzbeschreibung.

im zusammenhang mit der fehlersuche sind drei anweisungen der beiden letzten gruppen von interesse:

- dbcc traceon
- dbcc traceoff
- dbcc tracestatus

mit diesen anweisungen lassen sich sogenannte ablaufverfolgungsflags setzen, zurücksetzen und testen.

25.5.1 ablaufverfolgungsflags

mit ablaufverfolgungsflags können sie vorübergehend bestimmte server-eigenschaften einstellen oder verhaltensweisen beeinflussen. beispielsweise verwendet man ablaufverfolgungsflags bei der diagnose der systemleistung oder der fehlersuche in gespeicherten prozeduren.

mit der anweisung

```
dbcc traceon (flags)
```

aktivieren sie das jeweilige ablaufverfolgungsflag. in einer anweisung können sie auch mehrere, durch kommas getrennte flags angeben.

die anweisung

```
dbcc traceoff (flags)
```

schaltet ein oder mehrere (durch kommas getrennte) ablaufverfolgungsflag(s) aus.

den zustand bestimmter flags rufen sie mit

```
dbcc tracestatus(flags)
```

ab. wenn sie wissen wollen, welche flags überhaupt aktiviert sind, geben sie diese anweisung mit dem parameter -1 ein. das folgende beispiel aktiviert zunächst die flags 106, 325, 506, 1704 und 2701, zeigt alle aktiven flags an, schaltet mit der zweiten anweisung die flags 325 und 506 aus und fragt schließlich mit der letzten anweisung ab, welche flags noch aktiv sind:

```
dbcc traceon(106,325,506,1704,2701)
dbcc tracestatus(-1)
dbcc traceoff(325,506)
dbcc tracestatus(-1)
```

das ergebnis lautet:

```
dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.
traceflag   status
----------- -----------
```

```
106         1
325         1
506         1
1704        1
2701        1

(5 row(s) affected)

dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.

dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.

traceflag   status
----------- -----------
106         1
1704        1
2701        1

(3 row(s) affected)

dbcc-ausführung abgeschlossen. falls dbcc fehlermeldungen
ausgegeben hat, wenden sie sich an den systemadministrator.
```

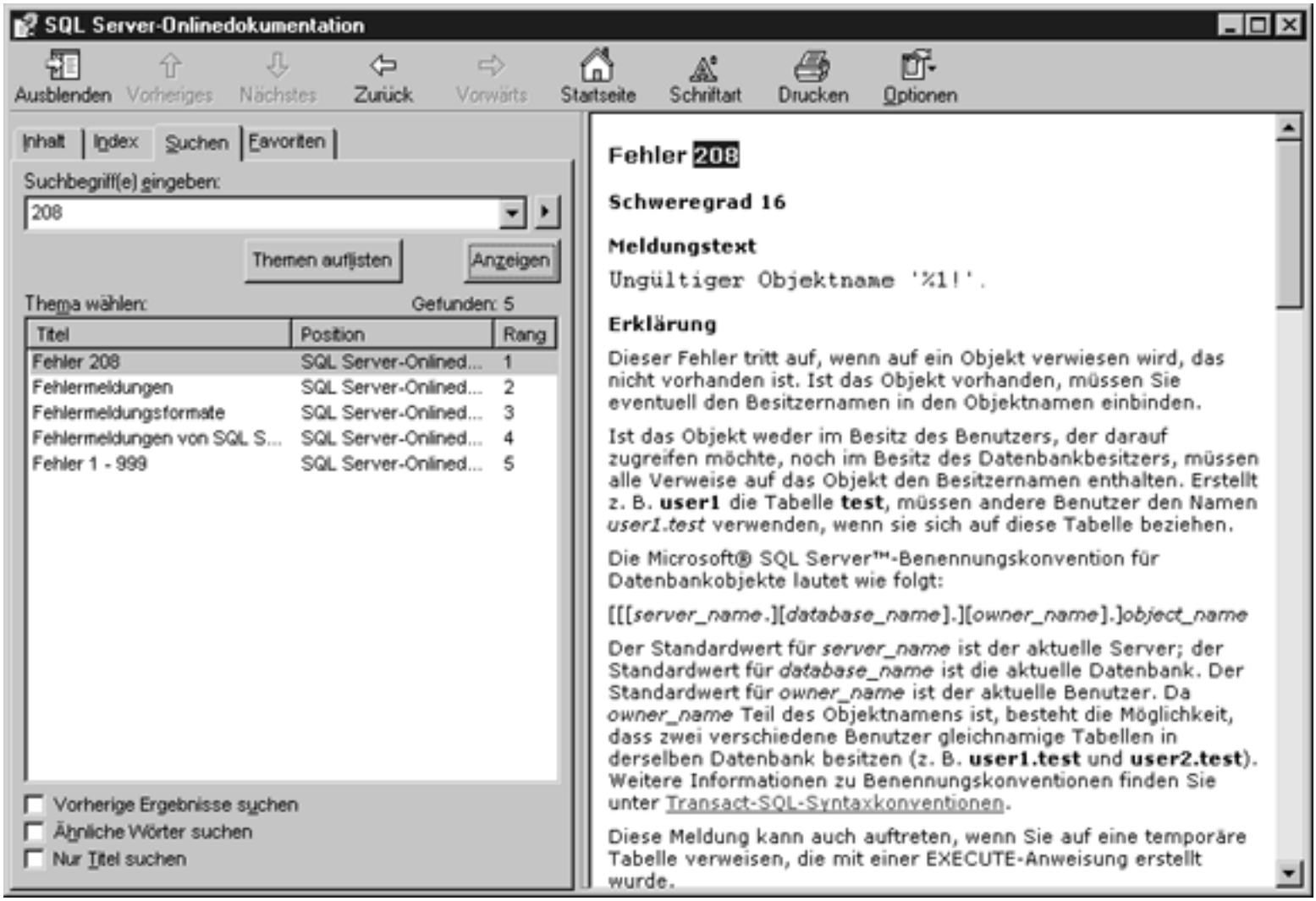
der zustand eines ablaufverfolgungsflags ist entweder 1 (aktiviert) oder 0 (deaktiviert).

ablaufverfolgungsflags können sie auch beim start von sql server von der eingabeaufforderung aktivieren, wie es folgendes beispiel zeigt:

```
sqlservr /dd:\mssql7\data\master.dat /t8783
```

der schalter /d gibt den pfad zur datenbankdatei master an, der schalter /t legt das flag 8783 zur ablaufverfolgung fest (siehe dazu kapitel 2).

eine vollständige liste der ablaufverfolgungsflags finden sie in der online-dokumentation.


SQL Server-Onlinedokumentation
Ausblenden
Vorheriges
Nächstes
Zurück
Vorwärts
Startseite
Schriftart
Drucken
Optionen
Inhalt
Index
Suchen
Favoriten
Suchbegriff(e) eingeben:
208
Themen auflisten
Anzeigen
Thema wählen:
Gefunden: 5
Titel
Position
Rang
Fehler 208
SQL Server-Onlined...
1
Fehlermeldungen
SQL Server-Onlined...
2
Fehlermeldungsformate
SQL Server-Onlined...
3
Fehlermeldungen von SQL S...
SQL Server-Onlined...
4
Fehler 1 - 999
SQL Server-Onlined...
5
Vorherige Ergebnisse suchen
Ähnliche Wörter suchen
Nur Titel suchen
Fehler 208
Schweregrad 16
Meldungstext
Ungültiger Objektname '%1!'.
Erklärung
Dieser Fehler tritt auf, wenn auf ein Objekt verwiesen wird, das nicht vorhanden ist. Ist das Objekt vorhanden, müssen Sie eventuell den Besitzernamen in den Objektnamen einbinden.
Ist das Objekt weder im Besitz des Benutzers, der darauf zugreifen möchte, noch im Besitz des Datenbankbesitzers, müssen alle Verweise auf das Objekt den Besitzernamen enthalten. Erstellt z. B. user1 die Tabelle test, müssen andere Benutzer den Namen user1.test verwenden, wenn sie sich auf diese Tabelle beziehen.
Die Microsoft® SQL Server™-Benennungskonvention für Datenbankobjekte lautet wie folgt:
[[[server_name.][database_name].][owner_name.]object_name
Der Standardwert für server_name ist der aktuelle Server; der Standardwert für database_name ist die aktuelle Datenbank. Der Standardwert für owner_name ist der aktuelle Benutzer. Da owner_name Teil des Objektnamens ist, besteht die Möglichkeit, dass zwei verschiedene Benutzer gleichnamige Tabellen in derselben Datenbank besitzen (z. B. user1.test und user2.test). Weitere Informationen zu Benennungskonventionen finden Sie unter Transact-SQL-Syntaxkonventionen.
Diese Meldung kann auch auftreten, wenn Sie auf eine temporäre Tabelle verweisen, die mit einer EXECUTE-Anweisung erstellt wurde.

kapitel 26 data warehousing und olap

26.1 ansichtssache

bisher hat dieses buch gezeigt, wie man daten mit einem relationalen datenbank-managementsystem speichert und abruft. insbesondere größere unternehmen benötigen ein instrument, um die operativen daten zu analysieren und für eine entscheidungsfindung heranzuziehen. oftmals sind diese daten dezentral und in unterschiedlichen formaten gespeichert. damit sich die daten analysieren lassen, muß man sie aus den verschiedenen quellen extrahieren, aufbereiten, bereinigen und zusammenfassen. diese konsolidierung erlaubt eine neue betrachtungsweise der daten und bietet einen besseren überblick über das unternehmen.

vergleichbar mit einem warenhaus, das ein breites sortiment unterschiedlichster artikel an einem zentralen standort anbietet, fassen die elektronischen pendants große datenmengen aus allen unternehmensbereichen zusammen und stellen sie in einer speziell zugeschnittenen datenbank - einem data warehouse - bereit. während man aber in einem richtigen warenhaus die ladenhüter aussortiert, kommt es in einem data warehouse gerade darauf an, historische informationen zu sammeln, um daraus trends ableiten und entscheidungen für die zukunft treffen zu können.

26.2 begriffsbestimmung

nehmen wir an, daß sie mittlerweile sql server perfekt beherrschen, ihre datenbanken im griff haben und alles zur zufriedenheit läuft. nun ist die unternehmensleitung durch mehrere veröffentlichungen in einschlägigen fachzeitschriften darauf aufmerksam geworden, daß entscheidungssysteme für die wettbewerbsfähigkeit unabdingbar sind. folgerichtig bekommen sie die aufgabe, ein data warehouse aufzubauen, das die grundlage für derartige systeme bildet.

in diesem zusammenhang soll zunächst geklärt werden, was entscheidungssysteme sind, worin der unterschied zwischen operativer und analytischer datenverarbeitung liegt und was ein data warehouse ist.

26.2.1 entscheidungssysteme

systeme zur entscheidungsunterstützung (dss - decision support systems) liefern den benutzern informationen, auf deren basis sich situationen analysieren und entscheidungen treffen lassen. die grundlage für entscheidungssysteme bilden data warehouses, die umfangreiche daten zusammentragen und für eine analyse bereitstellen. entscheidungssysteme sind anwendungen, die daten aus einem data warehouse abrufen, um sie in vordefinierten berichten oder analysen zu präsentieren.

26.2.2 oltp - online-transaktionsverarbeitung

systeme der operativen datenverarbeitung sind dadurch gekennzeichnet, daß viele benutzer gleichzeitig auf die datenbanken zugreifen, um daten zu ändern, während es nur wenige benutzer gibt, die ausschließlich daten zu berichtszwecken abrufen. die inhalte der datenbanken unterliegen einem ständigen wandel - im mittelpunkt steht die verarbeitung von transaktionen.

für diese form der datenverarbeitung sind relationale datenbank-managementsysteme geradezu prädestiniert. in einer oltp-datenbank werden unmengen von detaildatensätzen erfaßt. um doppelte daten zu vermeiden, macht sich die normalisierung der tabellen erforderlich. das führt allerdings zwangsläufig zu einer komplexen struktur mit vielen tabellen, die untereinander in beziehung stehen. somit ist diese organisationsform für entscheidungssysteme nicht sonderlich geeignet. oltp-datenbanken dienen vor allem der abwicklung des täglichen geschäftsbetriebs. beispiele dafür sind auftragserfassung, lagerverwaltung oder platzreservierung.

26.2.3 olap - analytische online-verarbeitung

systeme der analytischen datenverarbeitung fassen informationen zusammen und bringen die daten in eine organisationsform, die für analytische abfragen optimiert ist. es finden keine änderungen der daten im sinne einer transaktionsverarbeitung statt. der benutzer kann die daten nur lesen, hat aber trotzdem die möglichkeit, die datenbasis entsprechend seinen anforderungen an die datenanalyse zu wählen. im vordergrund steht der schnelle zugriff auf die daten, der mit multidimensionalen strukturen realisiert wird.

olap-systeme gehören zur kategorie der entscheidungssysteme und stützen sich ebenfalls auf die in einem data warehouse gespeicherten daten. häufig verwendet man die begriffe olap und data warehousing gleichberechtigt nebeneinander. genaugenommen ist olap aber ein verfahren, während ein data warehouse die zugrundeliegenden daten bereitstellt.

26.2.4 data warehouses und datamarts

data warehouses

ein data warehouse läßt sich als eine art sammelsystem für unternehmensdaten charakterisieren. es handelt sich um datenbanken, die große datenmengen aus allen unternehmensbereichen und für einen länger zurückliegenden zeitraum speichern. in der regel stammen die daten aus heterogenen quellen, die sich sowohl in der struktur als auch im format unterscheiden.

es liegt auf der hand, daß man derartige daten nur mit großem aufwand für eine entscheidungsfindung heranziehen könnte. es kommt darauf an, einen gemeinsamen nenner für die datenstrukturen zu finden und gleichzeitig die daten so aufzubereiten, daß sie für eine schnelle analyse geeignet sind. der benutzer eines data warehouse ist nicht an einzeldaten interessiert, die den blick aufs ganze versperren. er möchte trends erkennen und entscheidungen darauf stützen. außerdem sollen sich die daten aus unterschiedlichen perspektiven betrachten lassen. eine schnelle analyse setzt voraus, daß die daten etwa nach thematischen, chronologischen oder regionalen kriterien zusammengefaßt werden.

beispielsweise ist es in einem kaufhaus für eine saisonale analyse des kaufverhaltens der kunden

unerheblich, wie hoch die einnahmen an einer kasse zu einer bestimmten tageszeit sind. dagegen wären diese daten von bedeutung, wenn die leitung des kaufhauses eine veränderung der öffnungszeiten plant.

datamarts

während ein data warehouse die daten aus allen unternehmensbereichen vereinigt, kann man sich datamarts als selbständige und spezialisierte abteilungen eines data warehouse vorstellen. die daten können sowohl vom - übergeordneten - data warehouse als auch direkt von den quellen, aus denen auch das data warehouse die daten bezieht, stammen.

ein datamart läßt sich einfacher und kostengünstiger als ein data warehouse implementieren. im allgemeinen sind die datenmengen geringer, was sich positiv auf die antwortzeiten auswirkt. der kreis der benutzer ist aufgrund der spezialisierung ebenfalls kleiner.

konzeptionelle aspekte

bei der planung eines data warehouse geht man gewöhnlich von den vorhandenen datenbeständen aus und prüft, wie sie sich für einen spätere nutzung sammeln, aufbereiten und speichern lassen. mit einem data warehouse verfolgt man also eine strategie, die auf eine universelle nutzung ausgerichtet ist.

datamarts orientieren sich dagegen in erster linie an den wünschen der benutzer. deshalb faßt man hier auch gezielt daten zusammen, die für den vorgesehenen benutzerkreis von interesse sind. die quelldaten für einen datamart können aus einem bereits aufgebauten data warehouse oder direkt aus den datenbanken der einzelnen geschäftsbereiche kommen.

der entwurf eines data warehouse kann sich »von unten nach oben« oder »von oben nach unten« vollziehen. beide methoden haben vor- und nachteile. bei der ersten methode beginnt man mit dem aufbau von datamarts für einzelne geschäftsbereiche und verknüpft die datamarts später zu einem data warehouse, das die daten für das gesamte unternehmen bereitstellt. da sich das profil der jeweiligen datamarts aus den anforderungen der benutzer ergibt und die datenbestände überschaubar sind, lassen sich relativ schnell und kostengünstig brauchbare ergebnisse erzielen. probleme können allerdings dann auftreten, wenn die entwicklung der datamarts ohne zentrale koordination in verschiedene richtungen läuft, so daß der weg für eine unternehmensweite zusammenfassung der daten versperrt ist.

bei der von-oben-nach-unten-methode faßt man die daten aus den heterogenen quellen der einzelnen geschäftsbereiche zusammen und erstellt ein data warehouse für das gesamte unternehmen. auf diese weise läßt sich ein einheitliches konzept durchsetzen, und man kann sämtliche schnittstellen zu den später aufzubauenden datamarts definieren. damit hat man die potentiellen probleme unterschiedlicher datenstrukturen bereits im vorfeld beseitigt. dieses verfahren bedingt allerdings eine entsprechend längere vorbereitungsphase und einen höheren aufwand.

die entscheidung für ein konzept wird nicht zuletzt von den kosten geprägt. dennoch sollte man immer bedenken, daß sich intensive grundlagenarbeit auf lange sicht auszahlt.

26.3 komponenten

die umgebung eines entscheidungssystems besteht aus folgenden komponenten:

- heterogene quellen
- datentransformation
- data warehouse
- metadaten
- frontend-anwendungen und abfragewerkzeuge

26.3.1 heterogene quellen

die für eine datenanalyse erforderlichen basisdaten sind oftmals auf verschiedene geschäftsbereiche oder filialen verteilt und liegen in unterschiedlichen formaten vor. die daten können in datenbanken von oltp-systemen oder sogar als einfache textdateien gespeichert sein.

26.3.2 datentransformation

die rohdaten aus den heterogenen quellen eignen sich nur in den wenigsten fällen für eine schnelle analyse. um die daten in einem data warehouse zweckmäßig bereitstellen zu können, muß man sie aus den quellen extrahieren, modifizieren, filtern oder konsolidieren und in das data warehouse laden.

gleichartige daten sind bei der übernahme in das data warehouse in ein einheitliches format umzuwandeln. die datenlieferanten arbeiten oftmals als autonome systeme, die von vornherein nicht für eine zusammenarbeit ausgelegt sein müssen. selbst wenn die datensätze auf vergleichbaren systemen eine ähnliche struktur aufweisen, können felder mit derselben bedeutung in unterschiedlichen formaten gespeichert sein. zum beispiel kann man das geschlecht eines mitarbeiters auf folgende weise darstellen:

- durch die wörter »frau« und »herr«
- durch die großbuchstaben w und m
- durch die kleinbuchstaben w und m
- durch die zifferncodes 1 und 2 (1 für männlich, 2 für weiblich)
- durch ein einzelnes bit (gesetzt entspricht männlich, zurückgesetzt entspricht weiblich - oder umgekehrt, wenn ihnen das lieber ist)

für die zieltabelle im data warehouse gibt man ein bestimmtes format vor, das für alle derartigen daten gilt, beispielsweise w und m. bei der datentransformation wird dann für jede datenquelle festgelegt, wie die konkreten daten in das zielformat umzuwandeln sind.

die konsistente darstellung der informationen ist voraussetzung, um die daten konsolidieren, d.h. aus mehreren geschäftsbereichen zusammenfassen, zu können.

26.3.3 metadaten

metadaten kann man auch als »informationen über daten« bezeichnen. die metadaten in einem data warehouse beschreiben sowohl die eigenschaften der datenquellen (name des quellservers, datentypen der originaldaten) als auch die geschäftsregeln oder transformationen, die beim überführen der daten in das data warehouse anzuwenden sind.

26.3.4 frontend-anwendungen

für das abrufen der daten aus einem data warehouse oder einem datamart sind entsprechend zugeschnittene anwendungen erforderlich, die mit den multidimensionalen speicherformen umgehen können.

26.4 ein data warehouse entwerfen und erstellen

die speicherung der daten in einem data warehouse erfordert neue konzepte, um die riesigen datenmengen zu bewältigen. typische oltp-systeme arbeiten mit einem normalisierten datenbankaufbau, um redundante daten zu vermeiden und die speicherung zu optimieren. bei einem data warehouse kommt es dagegen darauf an, dem benutzer eine möglichst einfache darstellungsform der daten zu liefern und schnelle abfragen zu garantieren. dabei nimmt man ein gewisses maß an redundanz in kauf.

als logisches schema für die speicherung von daten in einem data warehouse verwendet man das dimensionale modell in den formen sternschema und schneeflockenschema.

26.4.1 sternschema

das verbreitetste schema für die gestaltung und anordnung der tabellen in einer data warehouse-datenbank ist das *sternschema*. im mittelpunkt steht eine faktentabelle, die mit mehreren dimensionstabellen über schlüssel verknüpft ist. dabei trennt man die geschäftsvorgänge in logische ereignisse (fakten) und einen satz von korrespondierenden dimensionen (beschreibungen der fakten) auf.

26.4.2 schneeflockenschema

eine abwandlung des sternschemas ist das *schneeflockenschema*. in diesem schema weisen die tabellen einen höheren grad der normalisierung als beim sternschema auf. beispielsweise kann eine dimensionstabelle mit einer weiteren verknüpft sein.

26.4.3 faktentabellen

die zentralen elemente eines data warehouse sind die faktentabellen mit numerischen werten (measures) und schlüsseln, die die fakten mit einer dimensionstabelle verknüpfen. die daten in einer faktentabelle beschreiben ein bestimmtes ereignis in einem unternehmen, beispielsweise bestellungen, lieferungen oder anfragen an den service.

in einer faktentabelle werden vor allem chronologische daten in einer zusammengefaßten form gespeichert. deshalb handelt es sich in der regel um statische daten. diese daten können bereits zum teil

zusammengefaßte zahlen darstellen.

26.4.4 dimensionstabellen

die daten in einer dimensionstabelle beschreiben daten in einer faktentabelle. dimensionstabellen sind geschäftseinheiten. die datensätze in den dimensionstabellen sind in der regel nicht statisch und können aktualisiert werden. wenn zum beispiel die faktentabelle über die schlüsselspalte kundennummer mit der dimensionstabelle kunde verknüpft ist, bleiben die daten in der faktentabelle mit den umsätzen unverändert, während sich aber die adresse des kunden ändern kann. der entsprechende kundendatensatz in der dimensionstabelle ist deshalb zu aktualisieren.

26.4.5 multidimensionale daten - cube

die tabellen in einer relationalen datenbank bestehen aus zeilen und spalten. es handelt sich also um eine darstellung in der ebene, die lediglich eine zweidimensionale sicht auf die daten erlaubt.

erweitert man diese darstellung um eine dritte dimension, entsteht der in olap-systemen verwendete *cube* (würfel). in einer multidimensionalen datenbank besteht ein cube aus einer gruppe von dimensionen und measures.

eine multidimensionale datenbank verfügt gewöhnlich über mehr als drei dimensionen. allerdings sind derartige gebilde unserem vorstellungsvermögen nicht zugänglich und existieren nur als mathematisches modell. deshalb beläßt man es der einfachheit halber bei der bezeichnung *würfel*.

die mit den olap services (siehe weiter unten in diesem kapitel) erstellten multidimensionalen datenbanken basieren alle auf zugrundeliegenden quelldaten, die in anderen relationalen speicherobjekten enthalten sind. wenn man einen cube verarbeitet, müssen die daten in einem solchen format vorliegen, daß sie sich durch die olap services abrufen und als ergebnismenge an eine client-anwendung zurückgeben lassen. olap services unterstützt die speichermodi molap, rolap und holap.

- molap: multidimensionale olap speichert die fakten und aggregationen in einer proprietären multidimensionalen struktur. dieser speichermodus bedeutet eine vollständige abkehr vom relationalen modell, bringt aber eine höhere geschwindigkeit der abfragen.
- rolap: relationale olap speichert die fakten und aggregationen in relationalen datenbanken. die daten verbleiben also an ihrem angestammten platz und werden nicht verschoben.
- holap: hybride olap ist eine mischform aus molap und rolap. dabei werden die detaildaten eines cube in relationalen datenbanken und die aggregationen in multidimensionalen strukturen gespeichert.

26.4.6 measures

die in einem cube gespeicherten numerischen daten bezeichnet man als *measures* (d.h. maßzahlen). es handelt sich dabei in der regel um daten, die aus mehreren spalten der zugrundeliegenden tabellen zusammengefaßt - aggregiert - werden. die aggregation dient dem ziel, die daten möglichst schnell in aufbereiteter form dem benutzer präsentieren zu können.

26.4.7 dimensionen

die kategorien, über die man die daten in einem cube analysiert und zusammenfaßt, bezeichnet man als dimensionen.

26.5 sql server und data warehousing

zur version 7.0 von sql server gehört das entscheidungssystem olap server (in den beta-versionen als plato bezeichnet). die olap-dienste von sql server unterstützen multidimensionale speicherung und navigation durch die datenbestände. damit lassen sich schnelle und komplexe analysen aus den vorhandenen unternehmensdaten durchführen.

26.5.1 datentransformationsdienste (dts)

die neu mit sql server 7.0 eingeführten datentransformationsdienste erlauben das importieren und exportieren von daten in einer großen auswahl von formaten, sofern diese daten über ole db oder odbc zugänglich sind. darüber hinaus unterstützen die datentransformationsdienste das überprüfen und transformieren der übertragenen daten.

die bei einer transformation auszuführenden aktionen werden in paketen gespeichert. das kann in einer betriebssystemdatei, als gemeinsam genutzte einheit im repository (siehe nächster abschnitt) oder in der datenbank msdb auf dem server geschehen. pakete sind in sich geschlossene beschreibungen der arbeitsschritte, die als bestandteil einer transformation auszuführen sind. die pakete lassen sich vom dts-assistenten, von der befehlszeile oder von einem skript ausführen. wenn man die pakete im repository oder in msdb ablegt, kann man darauf mit dem enterprise manager über den ordner **data transformation services** zugreifen.

damit die pakete nur autorisierten personen zugänglich sind, lassen sich berechtigungen auch nach paketen getrennt zuweisen.

26.5.2 repository

das repository agiert als systemkatalog eines data warehouse. es stellt die infrastruktur für gemeinsam genutzte metadaten bereit und wird standardmäßig in der datenbank msdb gespeichert.

26.6 die schritte zum data warehouse

um die daten in ein data warehouse zu laden, muß man sie zunächst aus den datenquellen extrahieren und geeignet transformieren. sql server bietet hier mit den dts-assistenten und dem dts-designer hervorragende grafische werkzeuge, mit denen sie die genannten schritte problemlos durchführen können.

26.6.1 daten importieren und transformieren

in kapitel 9 haben sie bereits ein beispiel für das importieren einer textdatei kennengelernt. die übernahme von daten in ein data warehouse läuft prinzipiell nach dem gleichen schema ab. dieser abschnitt zeigt, wie sie daten mit dem dts-assistenten aus einer access-tabelle importieren, die daten transformieren und in einer zieldatenbank ablegen. dabei legen sie ein dts-paket an, das sich wiederholt ausführen läßt, um das data warehouse aus derselben datenbank mit neuen daten zu versorgen.

zum lieferumfang der olap-services gehört die beispieldatenbank foodmart. dabei handelt es sich um eine access-datenbank, aus der wir hier die tabelle customer importieren. um die beispieldatenbank unverändert zu lassen, sollten sie sie kopieren und unter einem anderen namen speichern. die originalinstallation der olap services legt diese datenbank im ordner c:\programme\olap services\samples\ ab. für das beispiel in diesem kapitel wurde die datei foodmart.mdb kopiert und unter fm.mdb gespeichert.

den dts-assistenten können sie aus dem enterprise manager heraus oder über **start / programme / microsoft sql server 7.0 / daten importieren und exportieren** starten. klicken sie im startdialogfeld auf **weiter**. führen sie dann folgende schritte aus:

1. im dialogfeld **datenquelle wählen** (siehe abbildung 26.1) spezifizieren sie die herkunft der daten. öffnen sie das drop-down-listenfeld **quelle**, und wählen sie den eintrag *microsoft access*. im feld **dateiname** klicken sie auf die schaltfläche mit den drei punkten, navigieren zum ordner c:\programme\olap services\samples und wählen die datei fm.mdb aus. klicken sie auf **weiter**.

bild

abbildung 26.1: das dialogfeld datenquelle wählen

2. im dialogfeld **ziel wählen** (siehe abbildung 26.2) übernehmen sie *microsoft ole db-provider für sql server*, wählen den zielserver und den anmeldemodus aus und klicken dann auf den pfeil im feld **datenbank**. markieren sie den eintrag <neu>. daraufhin erscheint das dialogfeld **datenbank erstellen**. geben sie hier einen namen für die neue datenbank ein (im beispiel fm), und schließen sie das dialogfeld mit **ok**. klicken sie auf **weiter**.

bild

abbildung 26.2: das dialogfeld ziel wählen

3. im nächsten dialogfeld wählen sie die erste option *tabelle(n) aus quelldatenbank kopieren* und klicken auf **weiter**. im dialogfeld **quelltabellen auswählen** (siehe abbildung 26.3) markieren sie für das beispiel die tabelle customer.

bild

abbildung 26.3: das dialogfeld quelltabellen auswählen

4. klicken sie in der spalte **transformieren** auf die schaltfläche mit den drei punkten. daraufhin gelangen sie zum dialogfeld **spaltenzuordnungen und transformationen.** auf der registerkarte **spaltenzuordnungen** (siehe abbildung 26.4) können sie die spalten der quelltabelle auf vorhandene spalten der zieltabelle abbilden.

bild

abbildung 26.4: die registerkarte spaltenzuordnungen

5. gehen sie jetzt auf die registerkarte **transformationen** (siehe abbildung 26.5). hier ist die erste option *quellspalten direkt in die zielspalten kopieren* voreingestellt. wenn sich eine umwandlung der spalten erforderlich macht, wählen sie die zweite option. daraufhin wird das darunter befindliche bearbeitungsfeld aktiviert. für die unveränderte übernahme der spalten ist bereits ein standardskript vorgegeben.

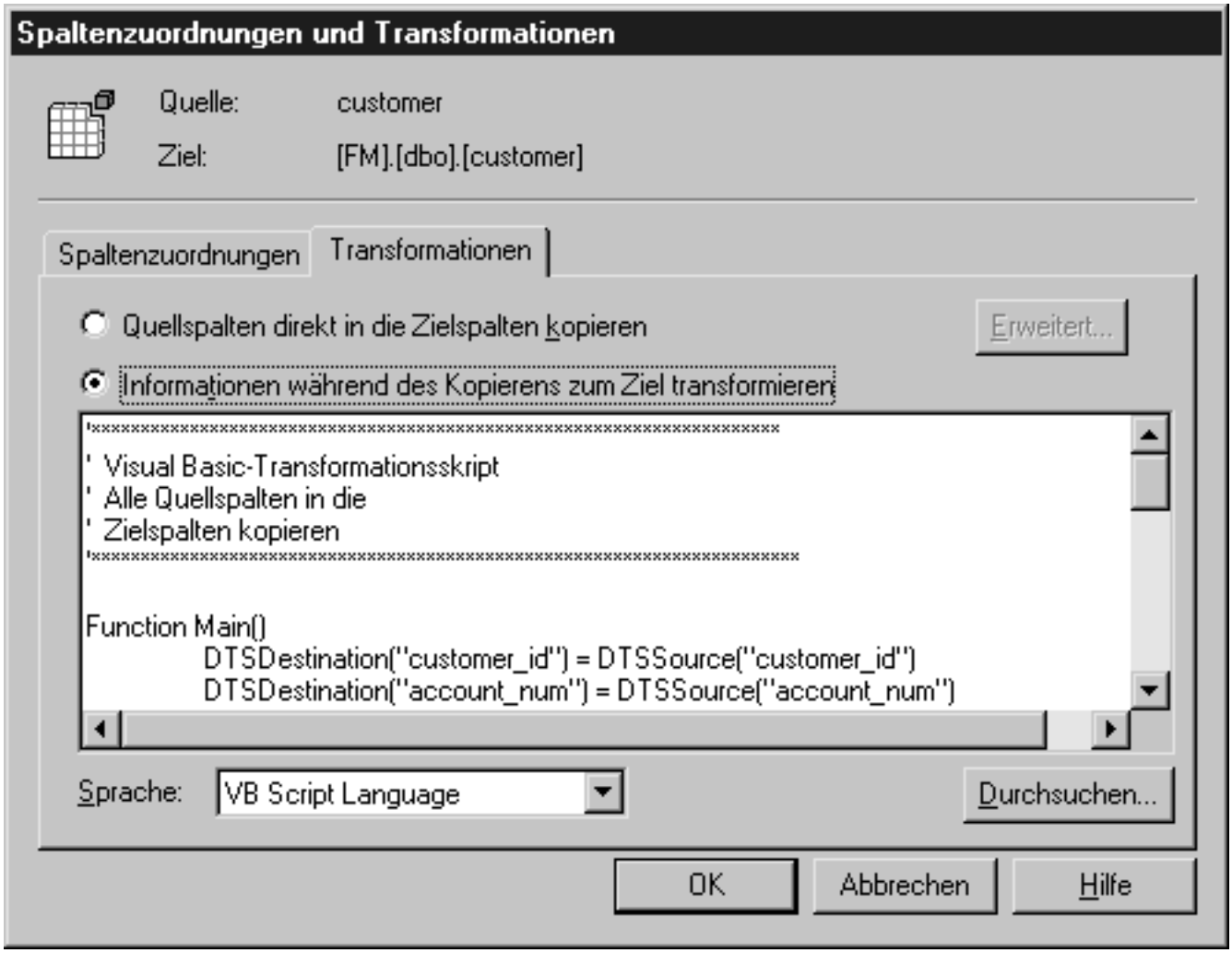


abbildung 26.5: die registerkarte transformationen

das in abbildung 26.5 in einem ausschnitt sichtbare transformationsskript hat insgesamt folgendes aussehen:

```
'**********************************************************************
'  visual basic-transformationsskript
'  alle quellspalten in die
'  zielspalten kopieren
'**********************************************************************

function main()
    dtsdestination("customer_id") = dtssource("customer_id")
```

```
    dtsdestination("account_num") = dtssource("account_num")
    dtsdestination("lname") = dtssource("lname")
    dtsdestination("fname") = dtssource("fname")
    dtsdestination("mi") = dtssource("mi")
    dtsdestination("address1") = dtssource("address1")
    dtsdestination("address2") = dtssource("address2")
    dtsdestination("address3") = dtssource("address3")
    dtsdestination("address4") = dtssource("address4")
    dtsdestination("city") = dtssource("city")
    dtsdestination("state_province") =
            dtssource("state_province")
    dtsdestination("postal_code") = dtssource("postal_code")
    dtsdestination("country") = dtssource("country")
    dtsdestination("customer_region_id") =
            dtssource("customer_region_id")
    dtsdestination("phone1") = dtssource("phone1")
    dtsdestination("phone2") = dtssource("phone2")
    dtsdestination("birthdate") = dtssource("birthdate")
    dtsdestination("marital_status") =
            dtssource("marital_status")
    dtsdestination("yearly_income") =
            dtssource("yearly_income")
    dtsdestination("gender") = dtssource("gender")
    dtsdestination("total_children") =
            dtssource("total_children")
    dtsdestination("num_children_at_home") =
            dtssource("num_children_at_home")
    dtsdestination("education") = dtssource("education")
    dtsdestination("date_accnt_opened") =
            dtssource("date_accnt_opened")
    main = dtstransformstat_ok
end function
```

um ein vorhandenes transformationsskript zu laden, klicken sie auf **durchsuchen** und wählen das gewünschte skript aus. klicken sie auf **ok**, um zum dialogfeld **spaltenzuordnungen und transformationen** zurückzukehren. klicken sie dann auf **weiter**.

6. im dialogfeld **paket speichern** (siehe abbildung 26.6) können sie das vom dts-assistenten erzeugte paket sofort ausführen, für die replikation erstellen oder speichern. schalten sie für das beispiel das kontrollkästchen **sofort ausführen** aus und das kontrollkästchen **dts-paket speichern** ein, um das paket später im dts-designer bearbeiten zu können. klicken sie auf **weiter**.

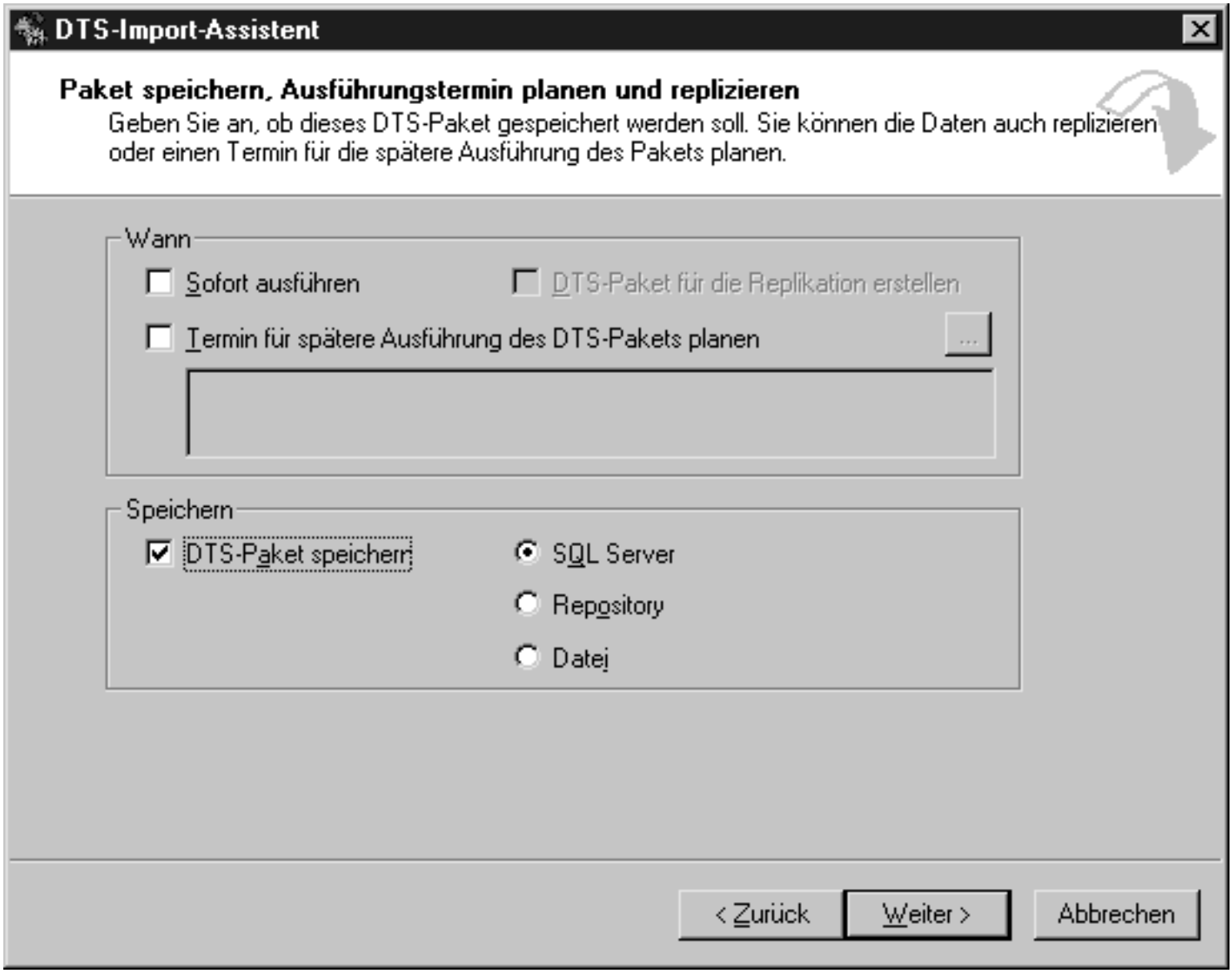


abbildung 26.6: hier legen sie fest, ob sie das paket für eine spätere verwendung speichern wollen

7. im dialogfeld **dts-paket speichern** (siehe abbildung 26.7) geben sie einen aussagekräftigen namen für das paket - im beispiel fm_customer - und optional eine beschreibung ein. stellen sie gegebenenfalls die anmeldeoptionen ein, und klicken sie dann auf **weiter**.

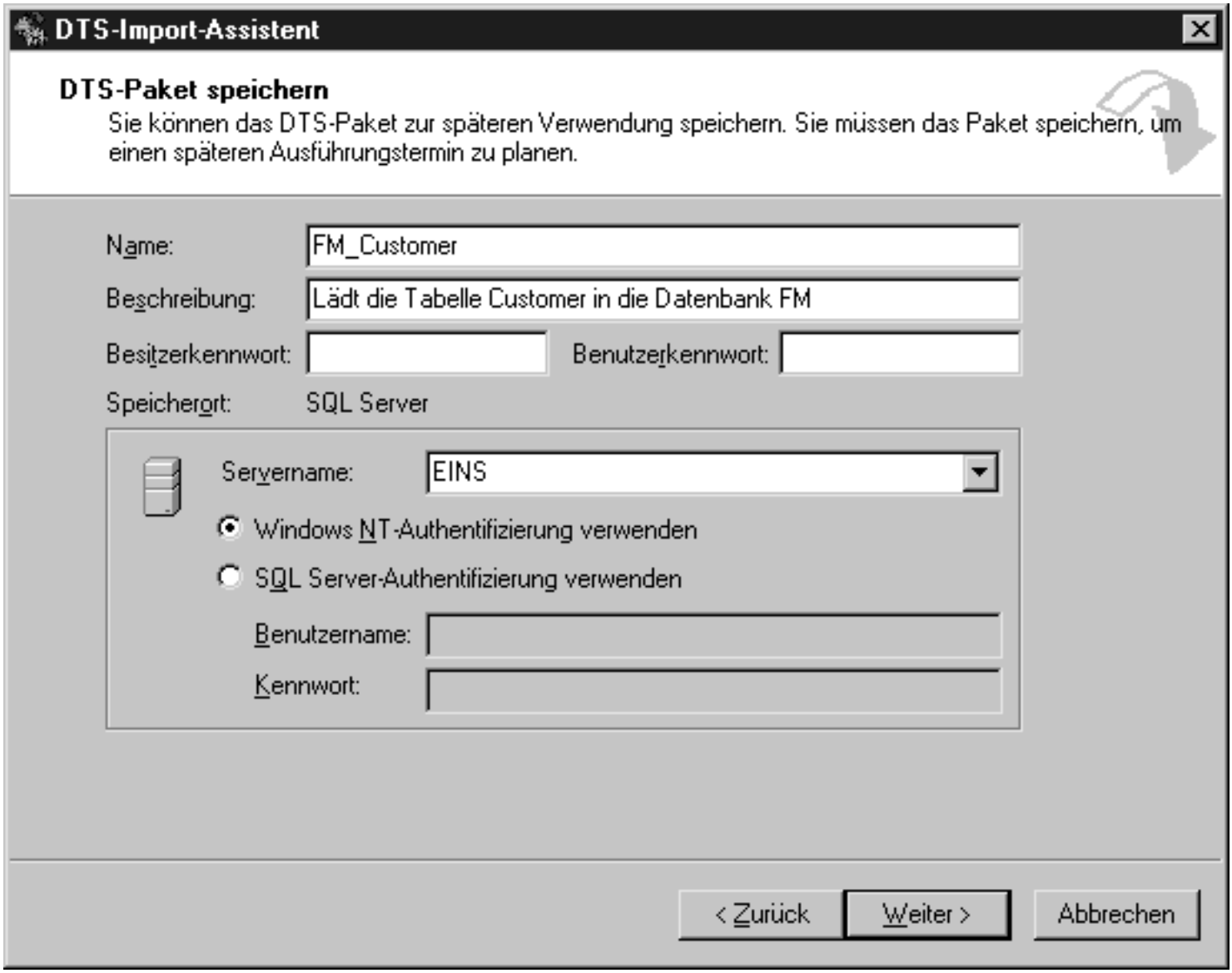


abbildung 26.7: speichern sie das paket unter einem passenden namen

8. im letzten dialogfeld des dts-assistenten erscheint eine zusammenfassung der getroffenen einstellungen. klicken sie auf **fertigstellen**. der assistent zeigt nun eine fortschrittsmeldung an (siehe abbildung 26.8). klicken sie abschließend auf **fertig**.

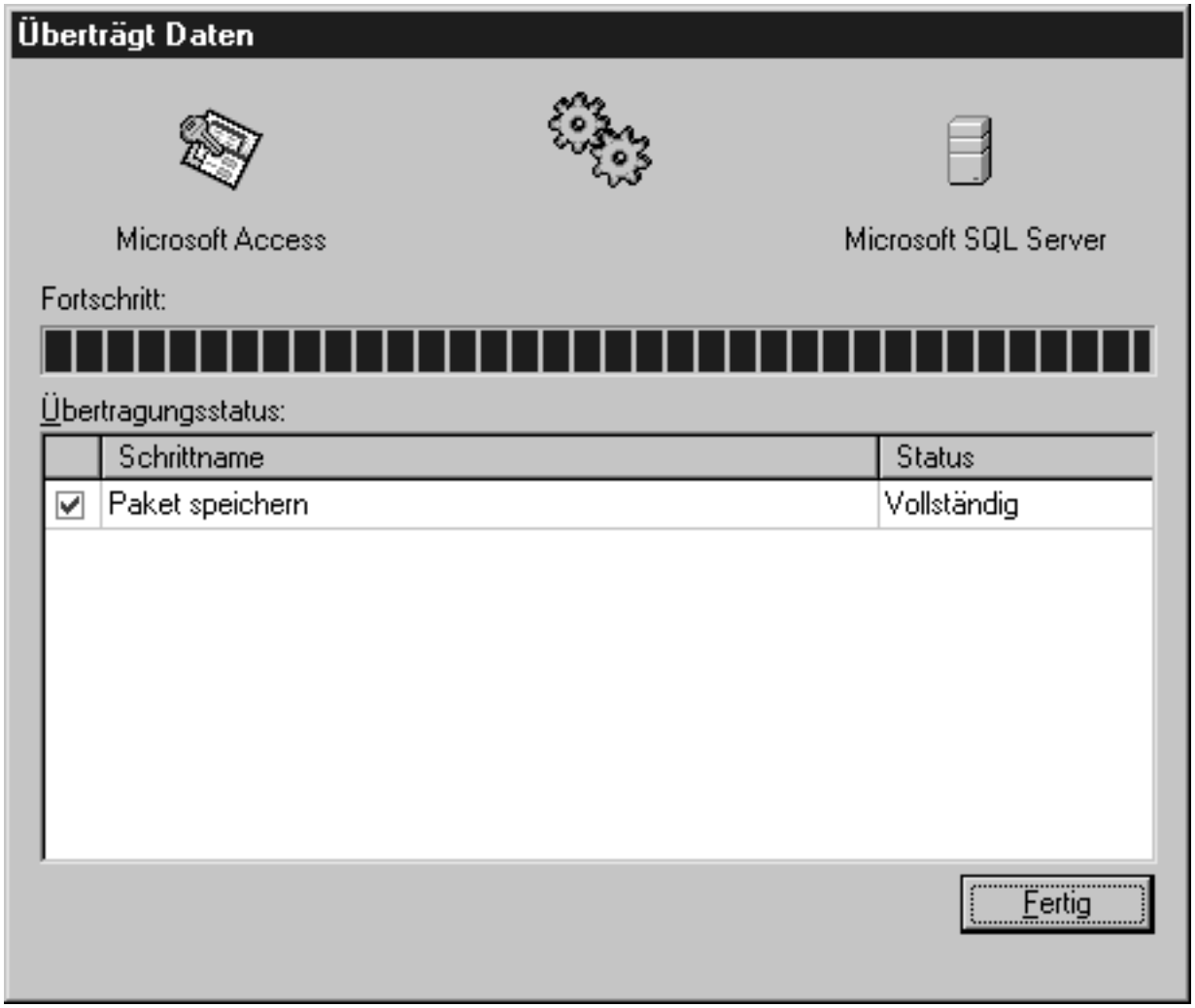


abbildung 26.8: fortschrittsmeldung beim speichern des pakets

26.6.2 dts-designer

der dts-designer ist ein werkzeug, mit dem sie pakete grafisch erstellen und bearbeiten können. um den dts-designer zu starten, erweitern sie in der konsolenstruktur des enterprise managers den ordner **data transformation services**. wenn sie auf den eintrag **lokale pakete** klicken, erscheint das im letzten abschnitt erstellte paket im detailbereich (siehe abbildung 26.9).

bild

abbildung 26.9: dts-pakete im enterprise manager

klicken sie mit der rechten maustaste im detailbereich auf das paket, und wählen sie aus dem kontextmenü den befehl **paket bearbeiten**. daraufhin wird das paket zur bearbeitung geöffnet (siehe abbildung 26.10).

bild

abbildung 26.10: paket zur bearbeitung geöffnet

die objekte können sie im arbeitsbereich des dts-designers frei anordnen und per drag&drop verschieben. wenn sie auf ein objekt doppelklicken oder mit der rechten maustaste klicken und **eigenschaften** aus dem

kontextmenü wählen, erscheint das zugehörige eigenschaftsdialogfeld.

um ein neues paket zu erstellen, klicken sie mit der rechten maustaste auf den eintrag **lokale pakete** (bzw. auf einen der anderen einträge) und wählen aus dem kontextmenü den befehl **neues paket**.

auf die vielfältigen möglichkeiten des dts-designers soll an dieser stelle nicht weiter eingegangen werden. viele elemente sind selbsterklärend, für weitere informationen sei auf die online-dokumentation verwiesen.

26.6.3 dts-pakete ausführen

vorhandene dts-pakete können sie sofort oder nach zeitplan ausführen. erweitern sie in der konsolenstruktur des enterprise managers den ordner **data transformation services**, klicken sie auf den gewünschten paketeintrag, und klicken sie dann im detailbereich mit der rechten maustaste auf das auszuführende paket. wählen sie aus dem kontextmenü den befehl **paket ausführen**, wenn sie das paket sofort starten möchten. über den befehl **termin für paket** gelangen sie zum dialogfeld **terminplan für wiederkehrende aufträge bearbeiten** (siehe abbildung 26.11).

abbildung 26.11: terminplan für ein paket festlegen

26.7 der olap-manager

der olap-manager ist wie der enterprise manager ein snap-in für die microsoft management console und unterstützt sie mit einer reihe von assistenten und editoren bei der arbeit in der olap-umgebung. den olap-manager (siehe abbildung 26.12) rufen sie über **start / programme / microsoft sql server 7.0 / olap / olap services / olap-manager** auf.

bild

abbildung 26.12: der olap-manager

dieses kapitel hat die wesentlichen schritte erläutert, wie sie ein data warehouse aus heterogenen daten erstellen können. dabei haben sie mit dem dts-assistenten ein paket erzeugt, das sich wiederholt ausführen läßt, um das data warehouse mit neuen daten zu versorgen.

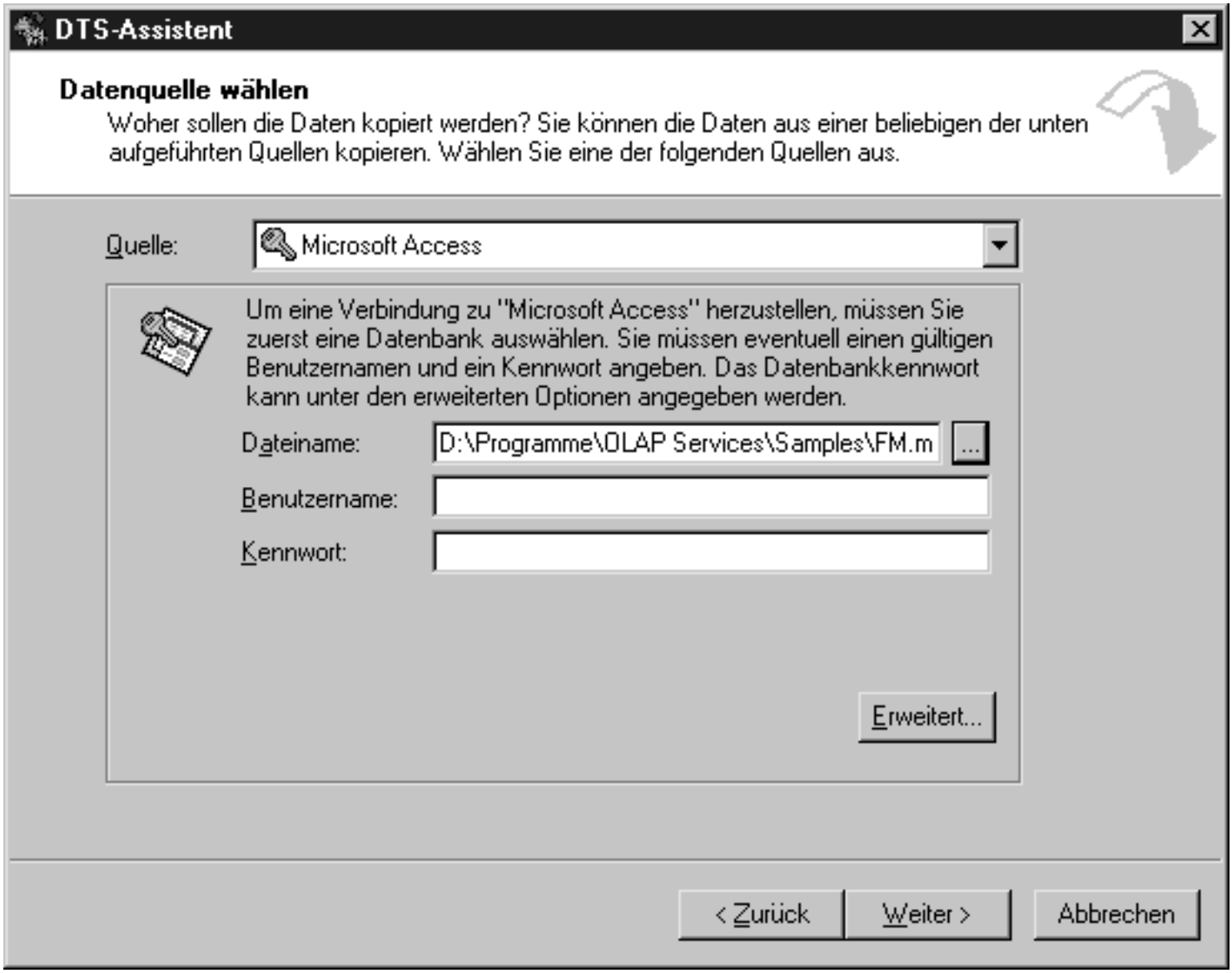
sicherlich erwarten sie nun ein praktisches beispiel für den umgang mit dem olap-manager. an dieser stelle scheint es aber angebracht, auf das ausgezeichnete lernprogramm zu verweisen, das zum olap-manager gehört.

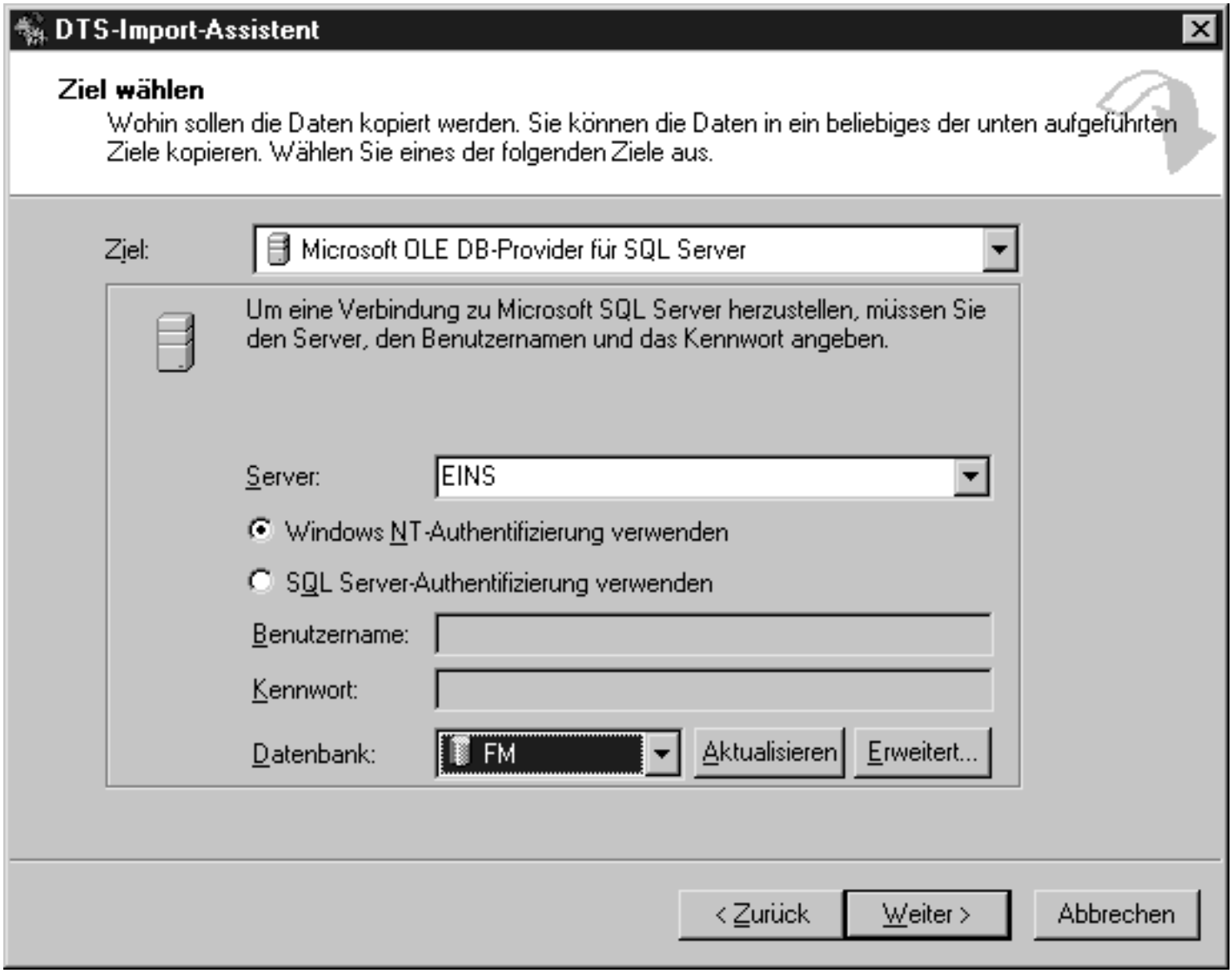
um das lernprogramm zu starten, klicken sie im detailbereich des olap-managers auf **lernprogramm *zu* olap-manager**. daraufhin wird das lernprogramm als html-dokument im internet explorer angezeigt (siehe abbildung 26.13).

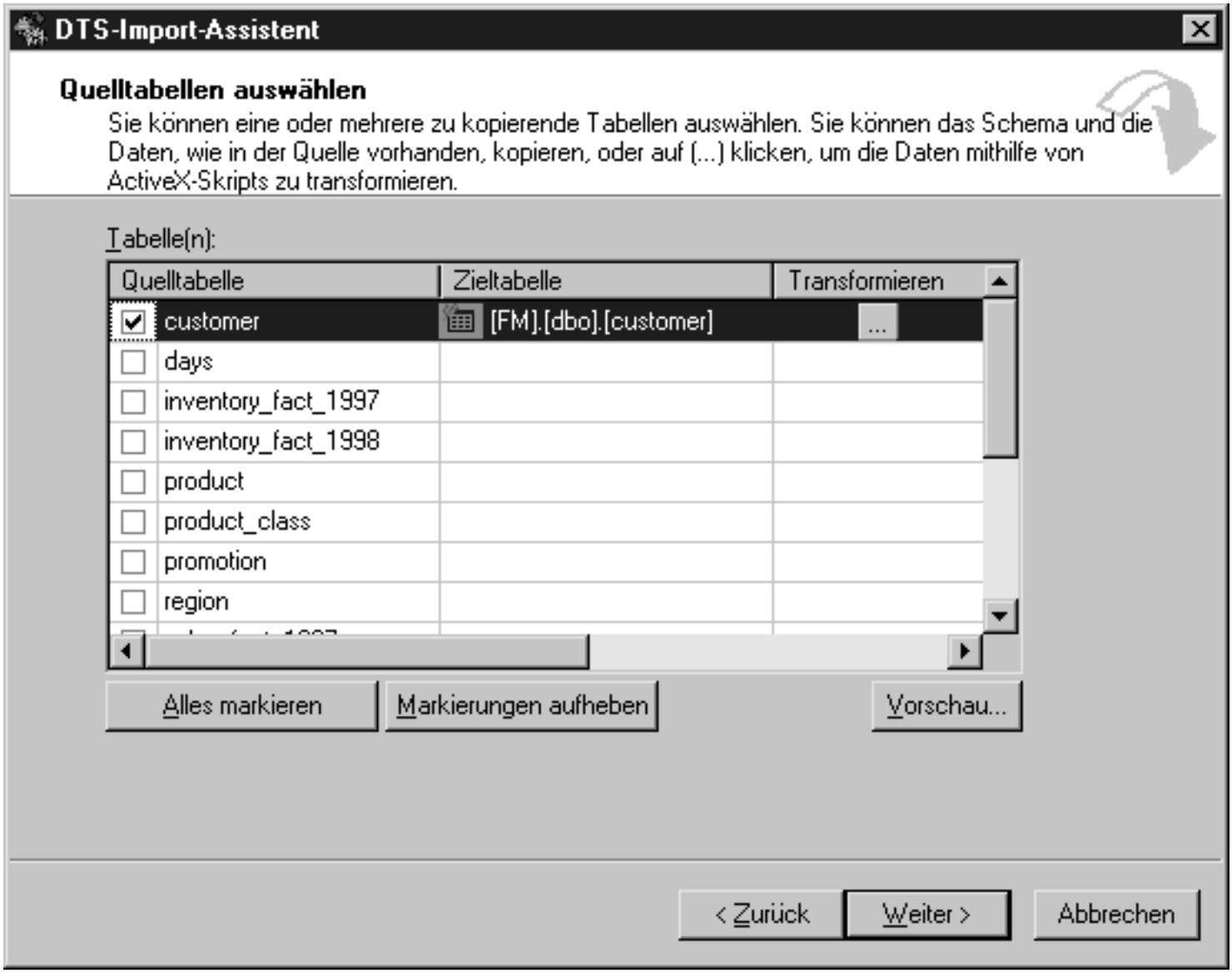
Bild

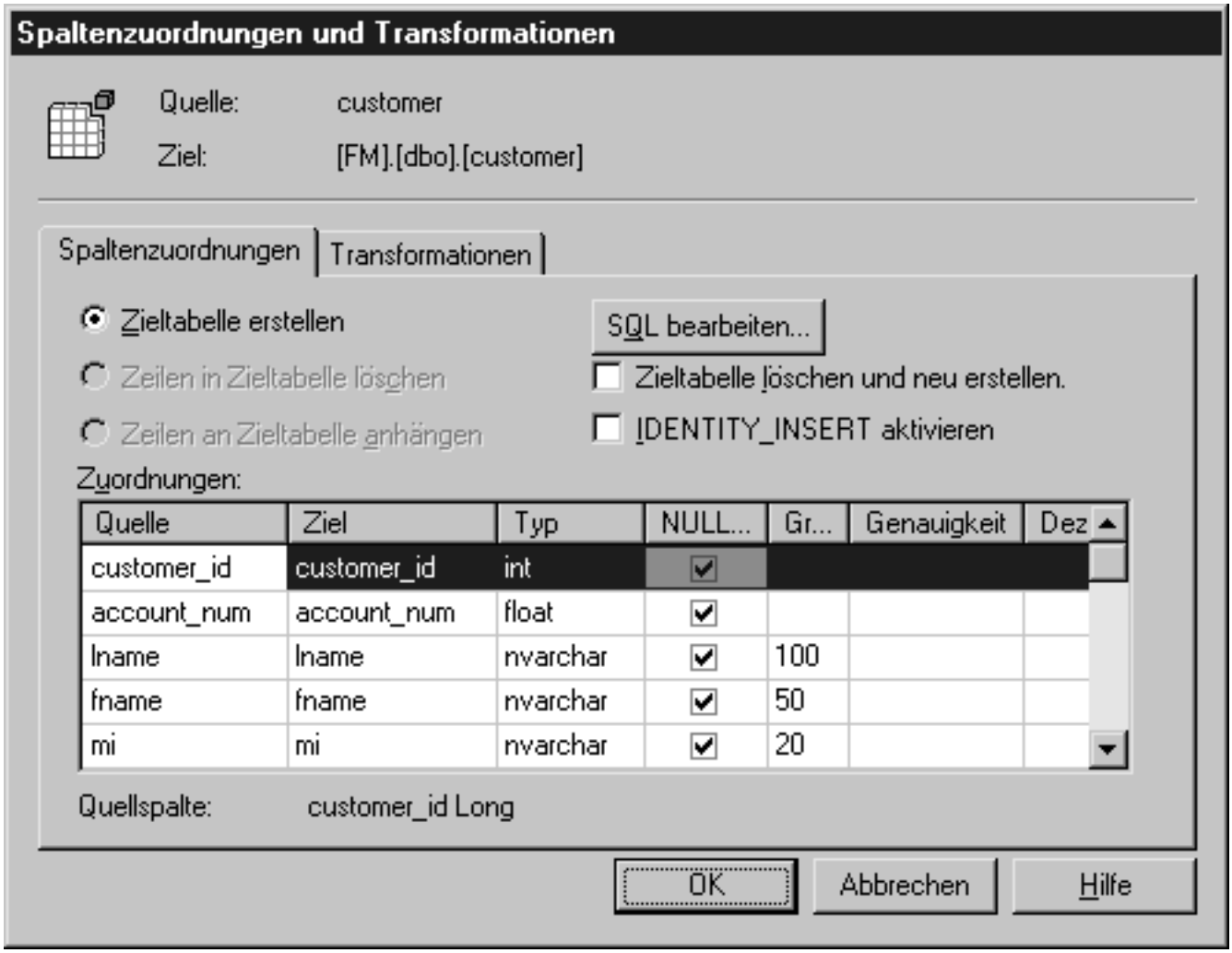
abbildung 26.13: lernprogramm für den olap-manager im internet explorer

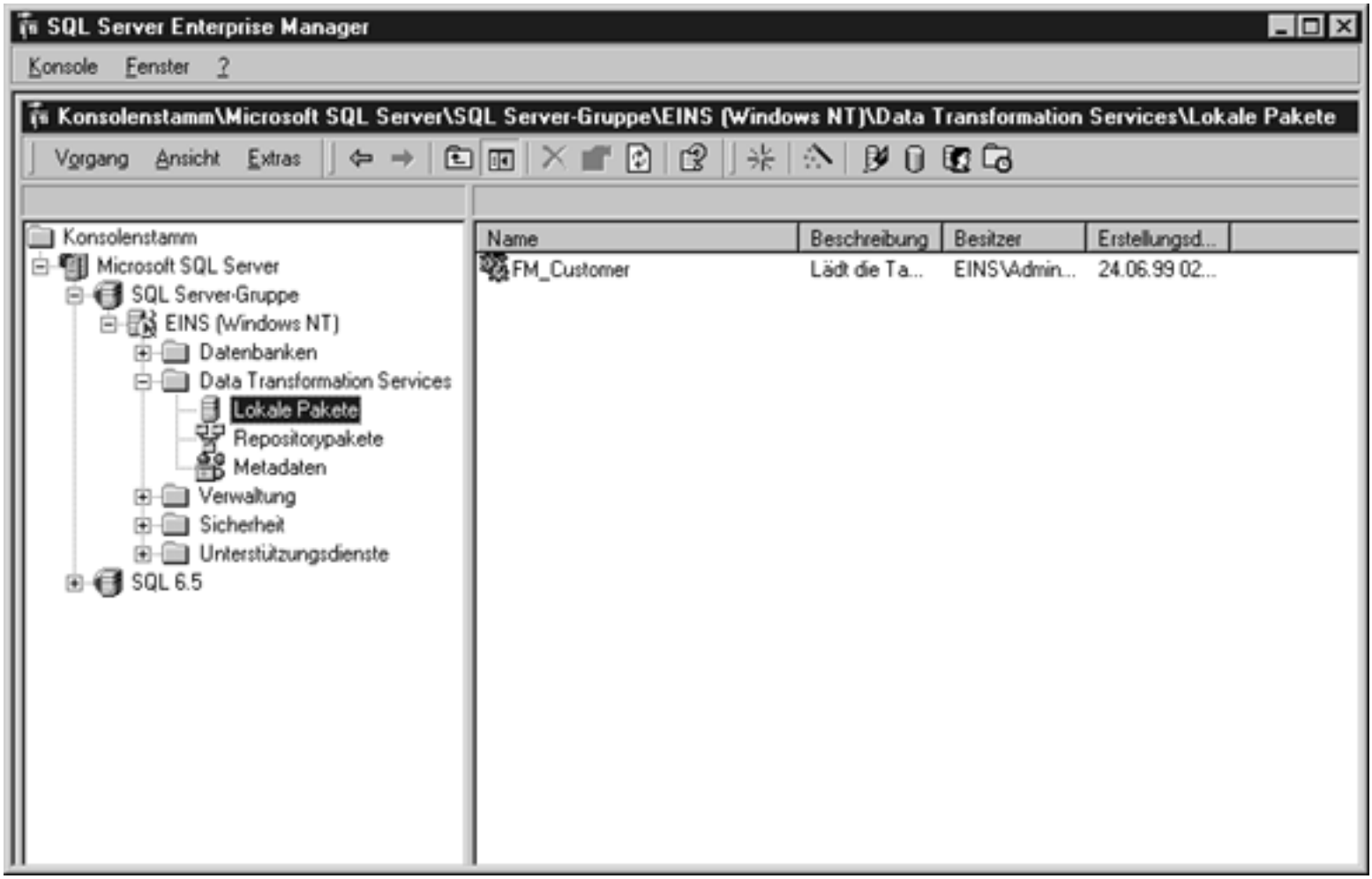
---

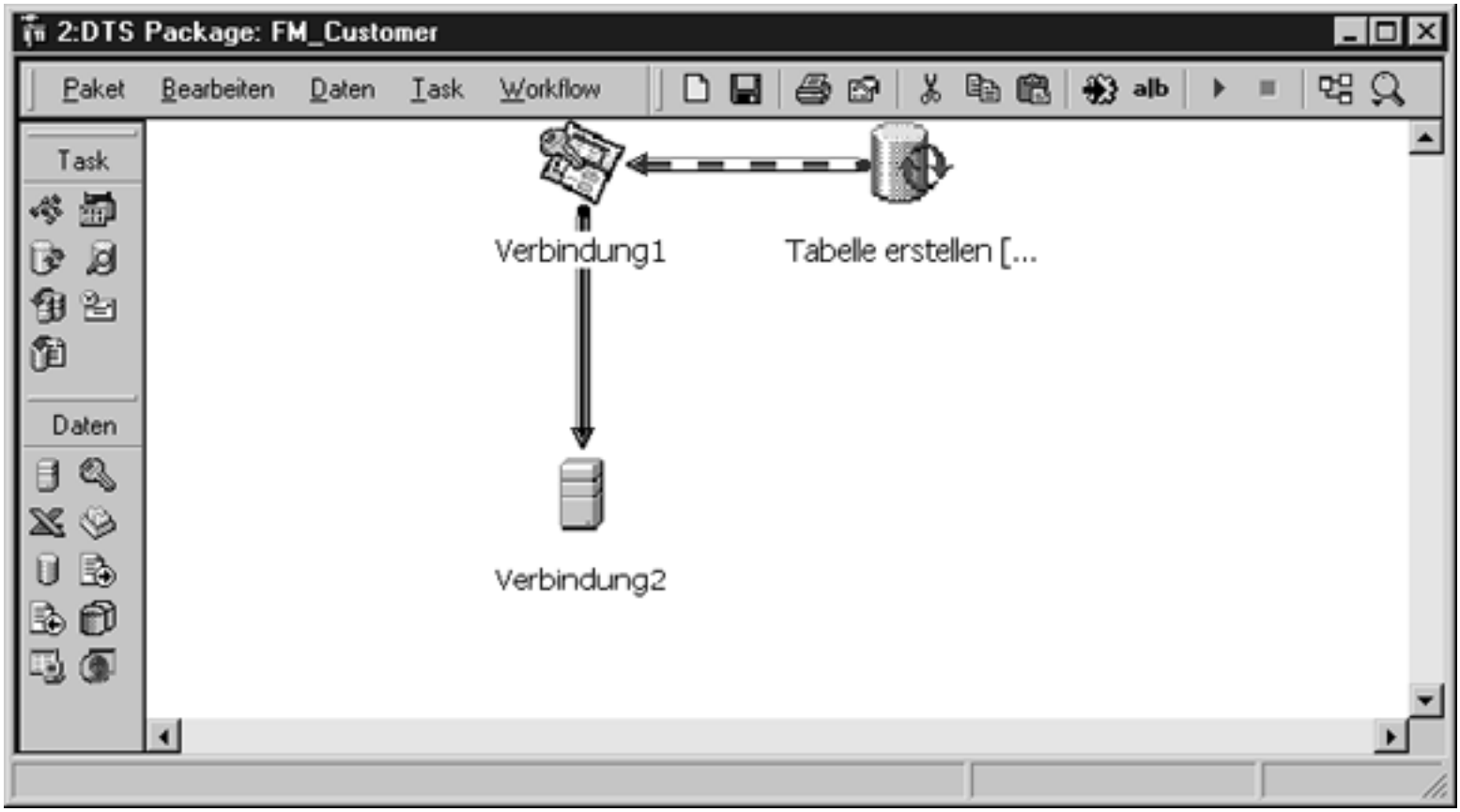

DTS-Assistent
Datenquelle wählen
Woher sollen die Daten kopiert werden? Sie können die Daten aus einer beliebigen der unten aufgeführten Quellen kopieren. Wählen Sie eine der folgenden Quellen aus.
Quelle:
Microsoft Access
Um eine Verbindung zu "Microsoft Access" herzustellen, müssen Sie zuerst eine Datenbank auswählen. Sie müssen eventuell einen gültigen Benutzernamen und ein Kennwort angeben. Das Datenbankkennwort kann unter den erweiterten Optionen angegeben werden.
Dateiname:
D:\Programme\OLAP Services\Samples\FM.m
...
Benutzername:
Kennwort:
Erweitert...
< Zurück
Weiter >
Abbrechen

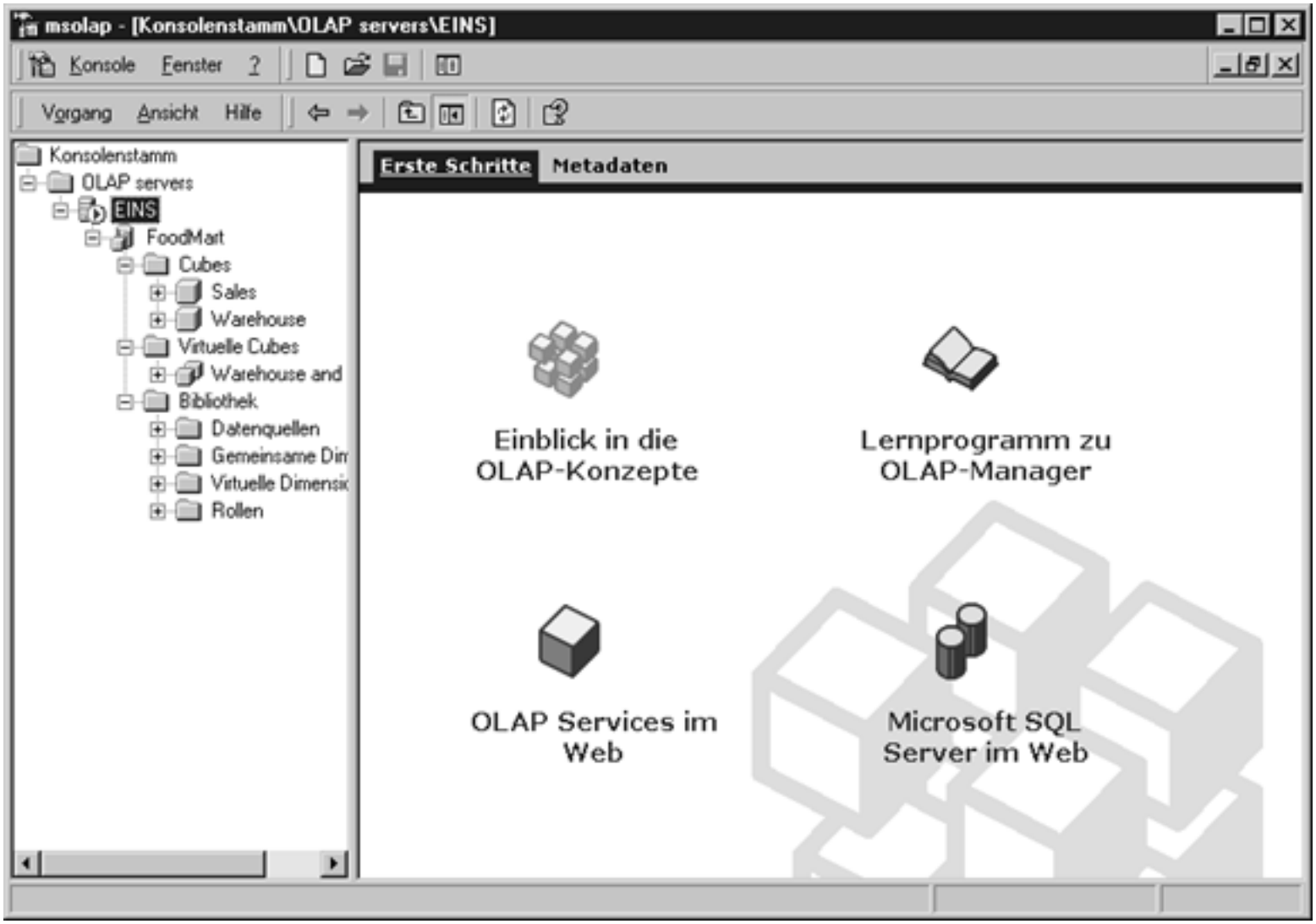

DTS-Import-Assistent
Ziel wählen
Wohin sollen die Daten kopiert werden. Sie können die Daten in ein beliebiges der unten aufgeführten Ziele kopieren. Wählen Sie eines der folgenden Ziele aus.
Ziel:
Microsoft OLE DB-Provider für SQL Server
Um eine Verbindung zu Microsoft SQL Server herzustellen, müssen Sie den Server, den Benutzernamen und das Kennwort angeben.
Server:
EINS
Windows NT-Authentifizierung verwenden
SQL Server-Authentifizierung verwenden
Benutzername:
Kennwort:
Datenbank:
FM
Aktualisieren
Erweitert...
< Zurück
Weiter >
Abbrechen

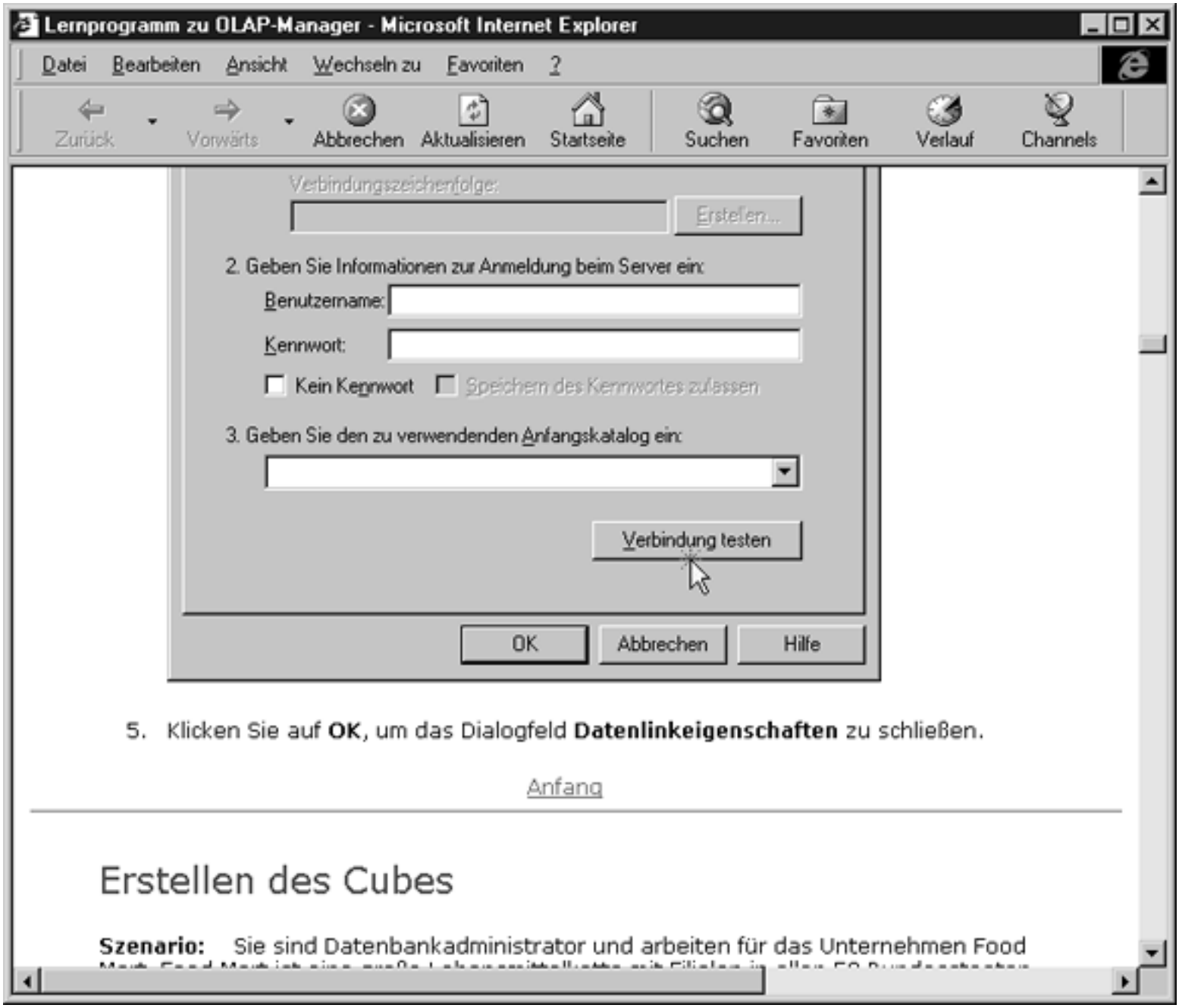

DTS-Import-Assistent
Quelltabellen auswählen
Sie können eine oder mehrere zu kopierende Tabellen auswählen. Sie können das Schema und die Daten, wie in der Quelle vorhanden, kopieren, oder auf (...) klicken, um die Daten mithilfe von ActiveX-Skripts zu transformieren.
Tabelle(n):
Quelltabelle
Zieltabelle
Transformieren
customer
[FM].[dbo].[customer]
...
days
inventory_fact_1997
inventory_fact_1998
product
product_class
promotion
region
Alles markieren
Markierungen aufheben
Vorschau...
< Zurück
Weiter >
Abbrechen


Spaltenzuordnungen und Transformationen
Quelle: customer
Ziel: [FM].[dbo].[customer]
Spaltenzuordnungen
Transformationen
Zieltabelle erstellen
Zeilen in Zieltabelle löschen
Zeilen an Zieltabelle anhängen
SQL bearbeiten...
Zieltabelle löschen und neu erstellen.
IDENTITY_INSERT aktivieren
Zuordnungen:
Quelle
Ziel
Typ
NULL...
Gr...
Genauigkeit
Dez
customer_id
customer_id
int
account_num
account_num
float
lname
lname
nvarchar
100
fname
fname
nvarchar
50
mi
mi
nvarchar
20
Quellspalte: customer_id Long
OK
Abbrechen
Hilfe


SQL Server Enterprise Manager
Konsole Fenster ?
Konsolenstamm\Microsoft SQL Server\SQL Server-Gruppe\EINS (Windows NT)\Data Transformation Services\Lokale Pakete
Vorgang Ansicht Extras
Konsolenstamm
Microsoft SQL Server
SQL Server-Gruppe
EINS (Windows NT)
Datenbanken
Data Transformation Services
Lokale Pakete
Repositorypakete
Metadaten
Verwaltung
Sicherheit
Unterstützungsdienste
SQL 6.5
Name
Beschreibung
Besitzer
Erstellungsd...
FM_Customer
Lädt die Ta...
EINS\Admin...
24.06.99 02...


2:DTS Package: FM_Customer
Paket
Bearbeiten
Daten
Task
Workflow
Task
Daten
Verbindung1
Tabelle erstellen [...
Verbindung2


msolap - [Konsolenstamm\OLAP servers\EINS]
Konsole
Fenster
?
Vorgang
Ansicht
Hilfe
Konsolenstamm
OLAP servers
EINS
FoodMart
Cubes
Sales
Warehouse
Virtuelle Cubes
Warehouse and
Bibliothek
Datenquellen
Gemeinsame Dir
Virtuelle Dimensi
Rollen
Erste Schritte
Metadaten
Einblick in die OLAP-Konzepte
Lernprogramm zu OLAP-Manager
OLAP Services im Web
Microsoft SQL Server im Web



5. Klicken Sie auf **OK**, um das Dialogfeld **Datenlinkeigenschaften** zu schließen.

Anfang

Erstellen des Cubes

Szenario: Sie sind Datenbankadministrator und arbeiten für das Unternehmen Food

kapitel 27 anwendungen programmieren

27.1 à la carte

der größte teil dieses buches beschäftigt sich ausschließlich mit den »bordmitteln« von sql server. zum beispiel haben sie abfragen bisher über dienstprogramme wie query analyzer oder osql ausgeführt. einem benutzer, der an den daten einer datenbank interessiert ist und nicht an den technischen vorgängen, wie er zu diesen daten gelangt, kann man nicht zumuten, select-anweisungen zu fuß zu formulieren. abgesehen vom datenbankadministrator und den datenbankentwicklern erfolgt der zugriff auf sql-server-datenbanken daher gewöhnlich über anwendungen, die den benutzer von den vorgängen hinter den kulissen abschirmen.

dieses kapitel bringt eine einführung in die entwicklung von anwendungen und zeigt, welche möglichkeiten der anwendungsentwickler hat, die daten von sql-server-datenbanken für den endbenutzer zugänglich zu machen.

27.2 datenbankanwendungen

anwendungen, die auf datenbanken zugreifen, realisieren in der regel folgende aufgaben:

- herstellen einer verbindung zum datenbankserver, einschließlich der anmeldeprozeduren
- abrufen von informationen aus der datenbank
- ändern der informationen durch den benutzer
- zurückschreiben der geänderten daten in die datenbank
- abmelden und trennen

darüber hinaus lassen sich mit spezialisierten anwendungen auch verwaltungsaufgaben für den datenbankserver programmieren, so daß der datenbankadministrator die routinearbeiten von einer komfortablen benutzeroberfläche aus steuern und überwachen kann.

neben der eigentlichen anwendungslogik für die benutzeroberfläche oder die dateiarbeit, greift eine datenbankanwendung auf folgende zwei komponenten zurück:

- anwendungsprogrammierschnittstelle (api): eine datenbank-api definiert, auf welche weise die anwendung eine verbindung zur datenbank herstellt und befehle an die datenbank sendet.
- datenbanksprache: definiert die syntax für die befehle, die die anwendung an die datenbank sendet. die an sql server gesendeten befehle sind transact-sql-anweisungen.

27.3 odbc-datenquelle einrichten

damit eine anwendung mit einer datenbank arbeiten und sich der odbc-client mit sql server verbinden kann, müssen sie eine odbc-datenquelle konfigurieren, die auf die gewünschte datenbank verweist. wenn sie im rahmen dieser konfiguration den namen der datenquelle (dsn) spezifizieren, benötigen sie eventuell den genauen pfad zur datenbankt. die erforderlichen angaben lassen sich aus der systemtabelle sysdatabases ermitteln. unter der annahme, daß sich die datenbank auf dem server eins befindet und sie noch mit den standardeinstellungen sa für den benutzer und ohne kennwort arbeiten, können sie folgende befehle an der eingabeaufforderung eingeben:

```
osql -usa -p -seins
1> use master
2> select name, filename from sysdatabases
3> go
```

eine odbc-datenquelle (im beispiel für die datenbank lotto) konfigurieren sie in folgenden schritten:

1. starten sie den odbc-administrator über **start / einstellungen / systemsteuerung / odbc-datenquellen (32 bit)**.
2. im dialogfeld **odbc-datenquellen-administrator** (siehe abbildung 27.1) klicken sie auf die schaltfläche **hinzufügen**. es erscheint das dialogfeld **neue datenquelle erstellen** (siehe abbildung 27.2).

bild

abbildung 27.1: der odbc-datenquellen-administrator

eine odbc-datenquelle speichert die informationen, wie eine verbindung zu einem datenprovider hergestellt wird. benutzerdatenquellen sind nur für den benutzer, der den dsn erstellt hat, verfügbar und lassen sich nur auf dem aktuellen computer verwenden. das kann in der anwendung zu problemen führen, wenn sich ein anderer benutzer am system anmeldet oder ein systemprozeß auf den datenquellennamen zugreifen muß. auf eine systemdatenquelle können dagegen alle benutzer eines computers und die nt-dienste zugreifen.

3. markieren sie den eintrag *sql server*, und klicken sie auf **fertigstellen**.
4. daraufhin gelangen sie zum dialogfeld **neue datenquelle für sql server erstellen** (siehe abbildung 27.3). im feld **name** bezeichnen sie die datenquelle - im beispiel lotto. im eingabefeld **beschreibung** können sie optional eine beschreibung eintragen. das kombinationsfeld **server** erlaubt die auswahl oder eingabe des datenbankservers. hier können sie *(lokal)* wählen, wenn auf dem client-computer auch sql server mit der datenbank installiert ist. im beispiel befindet sich die datenbank auf dem server eins. tippen sie den namen des servers ein, wenn er nicht in der liste aufgeführt ist. klicken sie auf **weiter**.

bild

abbildung 27.2: im dialogfeld neue datenquelle erstellen markieren sie den eintrag sql server

datenquelle ist nicht gleich datenbank. entsprechend der datenbankterminologie gehören zu einer datenquelle ein bestimmter satz von daten, notwendige informationen für den zugriff auf die daten und die position der eigentlichen datenquelle in form eines datenquellennamens (dsn - data source name).

bild

abbildung 27.3: in diesem dialogfeld geben sie den namen der datenquelle, eine optionale beschreibung und den server an

5. falls der server noch nicht in der liste aufgeführt ist und sie den namen eintippen, können sie im nächsten dialogfeld des assistenten einen client-konfigurationseintrag für den neuen namen erstellen (siehe abbildung 27.4). klicken sie gegebenenfalls auf die schaltfläche **clientkonfiguration**, wenn sie die netzwerkeinstellungen für die verbindung mit sql server ändern müssen. wenn sie im oberen teil des dialogfelds die zweite option (*mit sql server-authentifizierung*) gewählt haben, müssen sie im feld **login-id** den anmeldenamen für sql server (in der standardeinstellung sa) und im feld **kennwort** das dazugehörige kennwort (in der standardeinstellung ist kein kennwort erforderlich) eingeben.

bild

abbildung 27.4: in diesem dialogfeld legen sie die anmeldeeinstellungen zu sql server fest

6. klicken sie auf **weiter**, um zum nächsten dialogfeld des assistenten zu gelangen, in dem sie die standarddatenbank festlegen können (siehe abbildung 27.5). dabei handelt es sich um die datenbank, die sql server nach dem herstellen einer verbindung verwendet. im speziellen fall dieses beispiels können sie die datenbank lotto wählen.

bild

abbildung 27.5: hier können sie gegebenenfalls die standarddatenbank ändern

7. klicken sie auf **weiter**. im vierten dialogfeld des assistenten legen sie verschiedene ländereinstellungen fest (siehe abbildung 27.6).

bild

abbildung 27.6: im vierten dialogfeld legen sie ländereinstellungen und konvertierungsoptionen fest

8. klicken sie auf **fertigstellen**. der odbc-administrator zeigt nun eine übersicht an, mit welchen konfigurationen die datenquelle erstellt wird (siehe abbildung 27.7).

abbildung 27.7: die neue datenquelle wird mit den angegebenen einstellungen erstellt

9. klicken sie auf **datenquelle testen**, um die verbindung zur sql-server-datenbank zu überprüfen. der odbc-administrator sollte ein dialogfeld wie in abbildung 27.8 zeigen. andernfalls ist der gesamte vorgang zu wiederholen, wobei sie die einstellungen genau prüfen sollten.

der odbc-administrator testet die verbindung zur datenquelle und nicht zur datenbank selbst. die konkrete datenbank legen sie erst in der jeweiligen anwendung fest, die auf die datenquelle zugreift.

abbildung 27.8: der odbc-administrator zeigt an, ob der test der datenquelle erfolgreich verlaufen ist

über diese datenquelle können sie nun aus einer anwendung auf eine sql-server-datenbank zugreifen.

27.4 visual c++ 6

in diesem abschnitt erstellen sie mit visual c++ 6 eine anwendung, die auf die datenbank lotto zugreift, ziehungsergebnisse anzeigt, die eingabe einer neuen ziehung erlaubt und auch das löschen einer ziehung realisiert. die anwendung ist bewußt einfach gehalten und soll vor allem demonstrieren, wie ihnen das objekt crecordset die auf die datenbank bezogenen programmteile »mundgerecht« serviert.

anwendungsgerüst erstellen

eine neue anwendung mit datenbankunterstützung erstellen sie in folgenden schritten:

1. starten sie microsoft visual c++ 6 über **start / programme / microsoft visual studio 6.0 / microsoft visual c++ 6.0**.
2. wählen sie im menü **datei** den befehl **neu**. auf der registerkarte **projekte** markieren sie *mfc-anwendungs-assistent (exe)*. legen sie als erstes den pfad zu dem verzeichnis fest, in dem sie die anwendung speichern möchten, und geben sie dann im feld **projektname** einen passenden namen für die anwendung ein - im beispiel lottodb. klicken sie auf **ok**.

3. im dialogfeld **mfc-anwendungs-assistent - schritt 1** wählen sie die option *einzelnes dokument (sdi)*. das kontrollkästchen **unterstützung der dokument-/ansicht-architektur** muß eingeschaltet sein, da sich sonst keine datenbankanwendung erstellen läßt (siehe abbildung 27.9). klicken sie auf **weiter**.

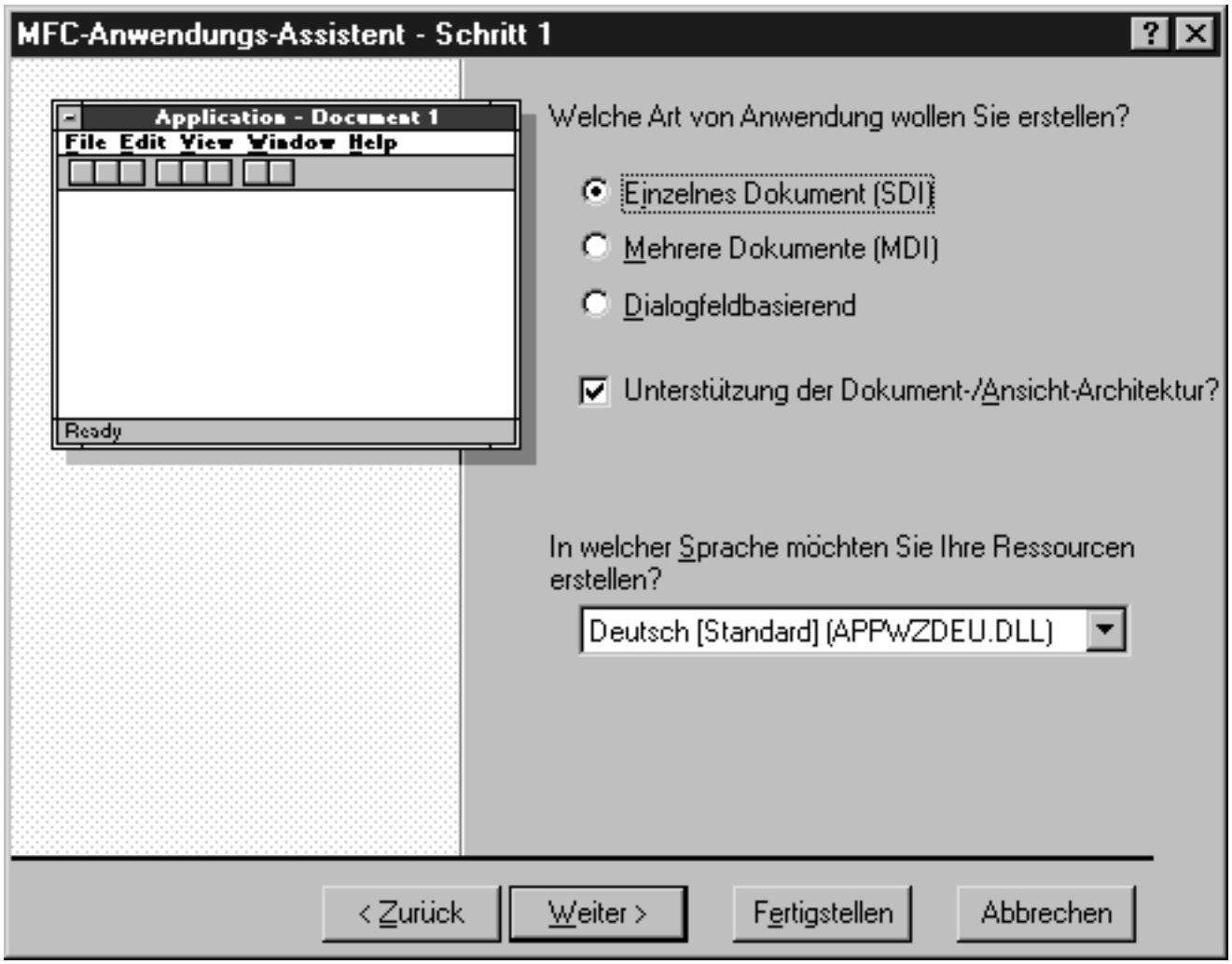


abbildung 27.9: wählen sie die option einzelnes dokument und die unterstützung der dokument-/ansicht-architektur

4. im dialogfeld **mfc-anwendungs-assistent - schritt 2 von 6** wählen sie die option *datenbankansicht mit dateiunterstützung*. wenn sie eine option für datenbankansicht wählen, wird die schaltfläche **datenquelle** aktiv. da noch keine datenquelle ausgewählt ist, müssen sie zuerst eine datenquelle festlegen. klicken sie auf die schaltfläche **datenquelle**. es erscheint das in abbildung 27.10 dargestellte dialogfeld **datenbankoptionen**.
5. im abschnitt **datenquelle** wählen sie aus dem listenfeld **odbc** eine sql-server-datenquelle aus (die sie zum beispiel nach den schritten am beginn dieses kapitels erstellt haben). als recordset-typ übernehmen sie die voreinstellung *snapshot*. klicken sie auf **ok**.

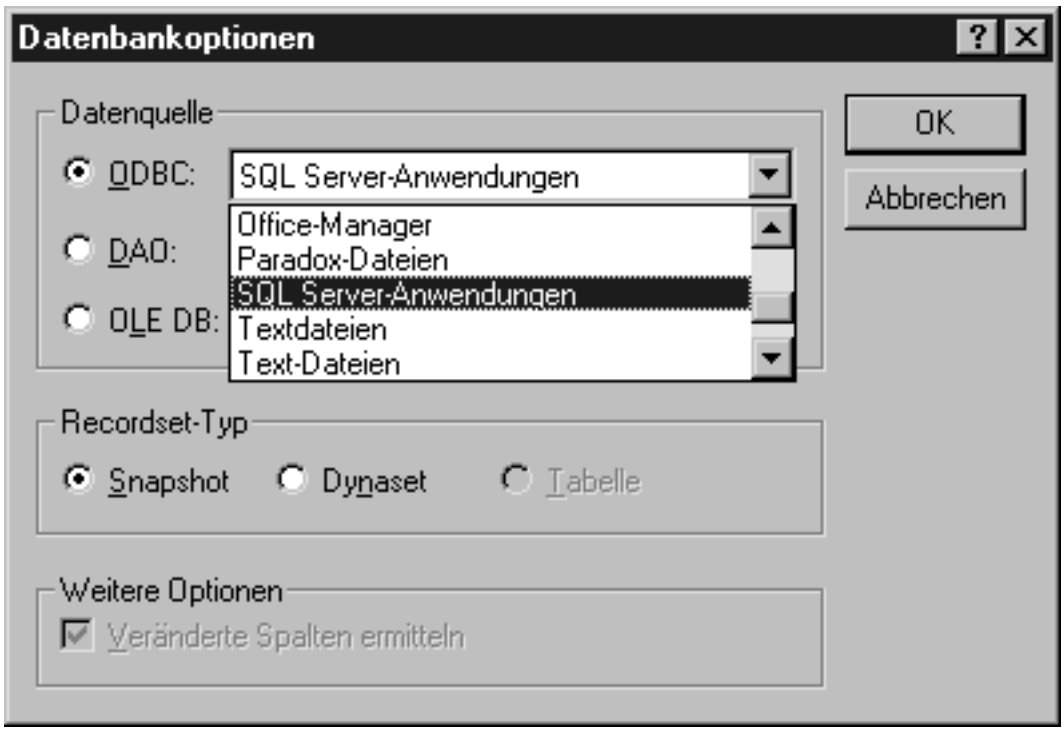


abbildung 27.10: wählen sie die sql-server-datenquelle aus

6. im dialogfeld **sql server-login** geben sie als login-id den anmeldenamen (sa in der voreinstellung) und das kennwort (in der voreinstellung ohne kennwort) ein, wenn sie die sql-server-authentifizierung verwenden. bei windows-nt-authentifizierung ist das kontrollkästchen **vertraute verbindung verwenden** einzuschalten. in diesem fall sind die felder **login-id** und **kennwort** deaktiviert.
7. auf den ersten blick ist bei diesem dialogfeld nicht zu sehen, daß hier noch eine wichtige einstellung vorzunehmen ist. klicken sie auf die schaltfläche **optionen**, um das dialogfeld zu erweitern (siehe abbildung 27.11).
8. im listenfeld **datenbank** wählen sie die datenbank aus, mit der die anwendung kommunizieren soll. die felder **anwendungsname** und **arbeitsstations-id** sind optional. diese angaben werden in master.dbo.sysprocesses in der spalte program_name bzw. hostname für diese verbindung in sql server gespeichert. klicken sie auf **ok**.

die datenbank können sie prinzipiell auch nach dem start der anwendung im anmeldedialog festlegen. wenn sie das aber vergessen, wird eine verbindung mit der standarddatenbank hergestellt. im beispiel haben sie zwar beim einrichten der odbc-datenquelle die datenbank lotto angegeben. das ist aber keine pflicht. es könnte sich genauso gut um jede andere datenbank handeln. dann hat ihre anwendung gleich mit einem fehler zu kämpfen.

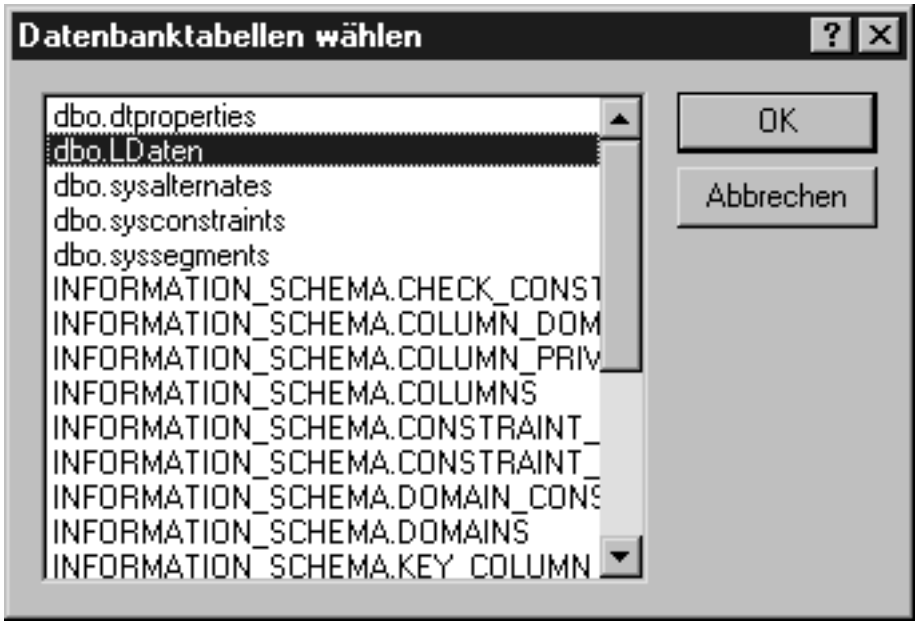


abbildung 27.11: in diesem dialogfeld wählen sie die datenbanktabelle(n) aus

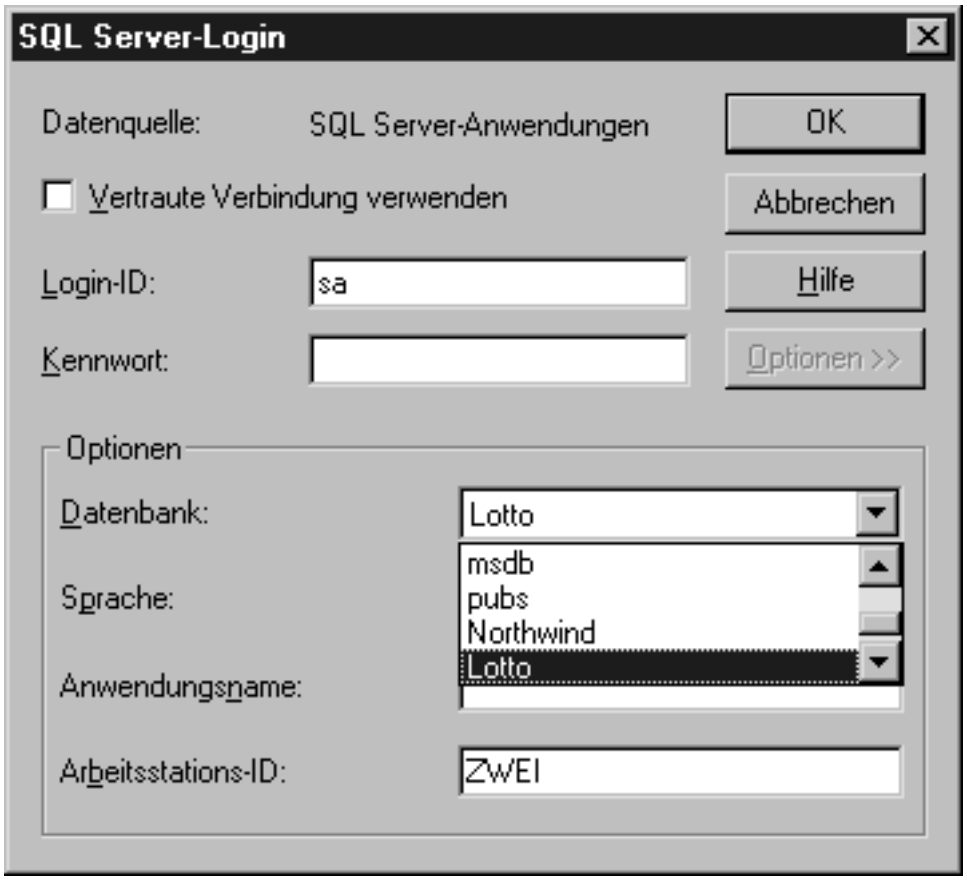


abbildung 27.12: geben sie die anmeldedaten ein, und klicken sie auf optionen, um die datenbank auszuwählen

9. im dialogfeld **datenbanktabellen wählen** (siehe abbildung 27.12) sind alle tabellen, ansichten, abfragen und gespeicherten prozeduren aufgeführt, die für die datenbank existieren und die sie in die satzgruppe übernehmen können. markieren sie den eintrag *dbo.ldaten*. (um mehrere elemente auszuwählen, müssen sie auf jedes einzeln klicken.) klicken sie auf **ok**, um zum mfc-anwendungs-assistenten zurückzukehren.

10. die übrigen einstellungen des assistenten können sie übernehmen. klicken sie auf **weiter**, bis das letzte dialogfeld (siehe abbildung 27.13) erreicht ist. es zeigt, daß der anwendungs-assistent eine zusätzliche klasse clottodbset erzeugt hat, die von crecordset abgeleitet ist. die ansichtsklasse clottodbview ist von crecordview und diese wiederum von cformview abgeleitet. in diesen klassen ist die datenbankunterstützung realisiert.

bild

abbildung 27.13: das letzte dialogfeld zeigt die vom assistenten erzeugten klassen

11. klicken sie auf **fertigstellen**, um das anwendungsgerüst erzeugen zu lassen.

formular erstellen

im formular - d.h. im hauptfenster - dieser einfachen sdi-anwendung müssen sie lediglich die felder für die ziehungsdaten vorsehen. für die navigation durch die datensätze bindet visual c++ aufgrund der im **mfc-anwendungs-assistent - schritt 2 von 6** gewählten option *datenbankansicht mit dateiunterstützung* bereits ein menü **datensatz** mit den befehlen **erster / vorheriger / nächster / letzer datensatz** und eine entsprechende symbolleiste automatisch ein. das menü und die symbolleiste erweitern sie zwar noch, um auch neue datensätze erstellen und datensätze löschen zu können, momentan aber brauchen sie sich nicht darum zu kümmern. abbildung 27.14 zeigt das formular der anwendung im entwurfsmodus.

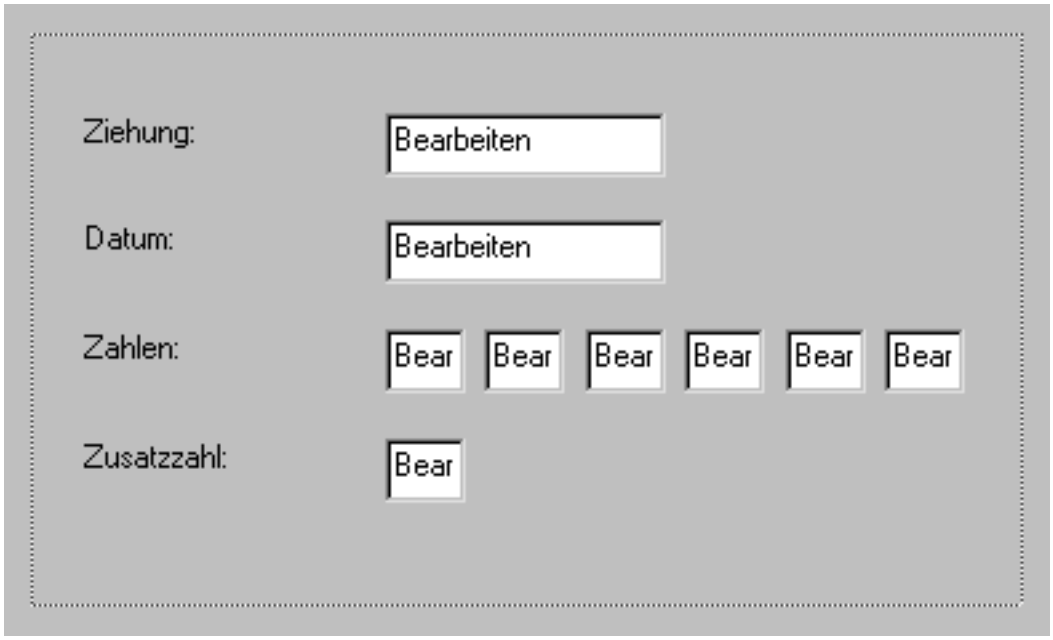


abbildung 27.14: das formular der anwendung im entwurfsmodus

tabelle 27.1 zeigt die steuerelemente, aus denen sich das hauptformular aufbaut.

objekt	eigenschaft	einstellung
text	id titel	idc_static *ziehung:*
eingabefeld	id	idc_ziehung

text	id titel	idc_static *datum:*
eingabefeld	id	idc_datum
text	id titel	idc_static *zahlen:*
eingabefeld	id	idc_z1
eingabefeld	id	idc_z2
eingabefeld	id	idc_z3
eingabefeld	id	idc_z4
eingabefeld	id	idc_z5
eingabefeld	id	idc_z6
text	id titel	idc_static *zusatzzahl:*
eingabefeld	id	idc_zz

tabelle 27.1: steuerelemente für das hauptformular der anwendung

die eingabefelder des formulars sind als nächstes mit den feldern - sprich spalten - der datenbanktabelle zu verbinden. öffnen sie dazu den klassen-assistenten, und gehen sie auf die registerkarte **member-variablen**. vergewissern sie sich, daß im feld **klassenname** die klasse clottodbview (die sichtklasse der anwendung) ausgewählt ist. klicken sie auf die schaltfläche **variable hinzufügen**. im dialogfeld **member-variable hinzufügen** (siehe abbildung 27.15) finden sie im drop-down-kombinationsfeld **name der member-variablen** eine liste von variablennamen für die felder des recordsets. die variablen verbinden sie mit den eingabefeldern des formulars. tabelle 27.2 zeigt die zuordnung der steuerelemente zu den variablennamen. fügen sie die entsprechenden variablen hinzu.

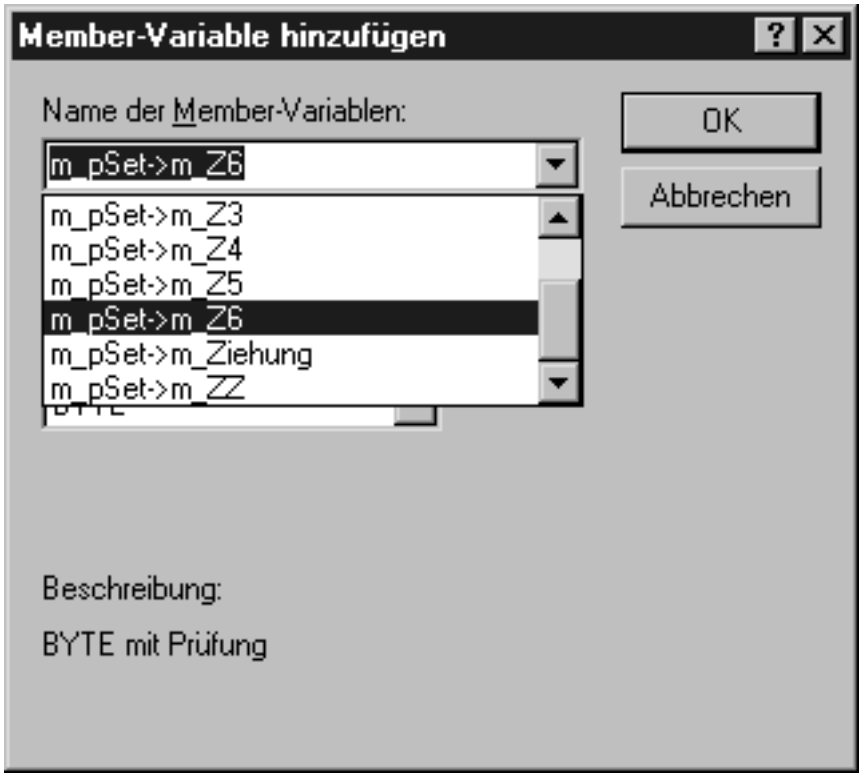


abbildung 27.15: das dialogfeld member-variable hinzufügen

objekt-id	name der member-variablen
idc_ziehung	m_pset->m_ziehung
idc_z1	m_pset->m_z1
idc_z2	m_pset->m_z2
idc_z3	m_pset->m_z3
idc_z4	m_pset->m_z4
idc_z5	m_pset->m_z5
idc_z6	m_pset->m_z5
idc_zz	m_pset->m_z6

tabelle 27.2: zuordnung der variablen zu den eingabefeldern des formulars

wo aber erscheint das datumsfeld? in der liste der member-variablen ist es nicht aufgeführt. gehen sie im klassen-assistenten auf die registerkarte **member-variablen**, und wählen sie die klasse clottodbset aus. in der liste ist für den spaltennamen [datum] eine variable m_datum vom typ ctime eingetragen. visual c++ stellt aber weder ein makro noch eine funktion bereit, um eine variable vom typ ctime mit einem steuerelement zu verbinden.

abhilfe schafft die deklaration der variablen für die spalte datum als typ coledatetime. jetzt müssen sie etwas in die trickkiste greifen. der klassen-assistent beansprucht die in //{{afx... //}}afx eingeschlossenen codeabschnitte für sich allein. wenn sie änderungen in diesen abschnitten vornehmen, kommt der klassen-assistent eventuell nicht mehr klar, oder er überschreibt ihre änderungen durch eigene daten. hier

müssen sie aber einmal vom normalen weg abweichen und die änderung des datentyps für das datum selbst in die hand nehmen. listing 27.1 zeigt den entsprechenden ausschnitt aus der datei lottodbset.h.

die vollständigen quelldateien und ressourcen für die anwendung lottodb finden sie auf der begleit-cd.

listing 27.1 - ausschnitt aus der datei lottodbset.h

```
class clottodbset : public crecordset
{
public:
    long getmaxid();
    clottodbset(cdatabase* pdatabase = null);
    declare_dynamic(clottodbset)

// feld-/parameterdaten
    //{{afx_field(clottodbset, crecordset)
    long    m_ziehung;
    coledatetime    m_datum;
    byte    m_z1;
    byte    m_z2;
    byte    m_z3;
    byte    m_z4;
    byte    m_z5;
    byte    m_z6;
    byte    m_zz;
    //}}afx_field
```

in der datei lottodbview.cpp müssen sie ebenfalls eine diesbezügliche änderung vornehmen. öffnen sie zunächst den klassen-assistenten, gehen sie auf die registerkarte **member-variablen**, wählen sie die klasse clottodbview aus, löschen sie die für das steuerelement idc_datum eingetragene variable, und klicken sie auf **variable hinzufügen**.

im dialogfeld **member-variable hinzufügen** geben sie als namen der variablen zum beispiel m_oledtdatum ein und wählen als variablentyp coledatetime aus.

als nächstes ändern sie die funktion dodataexchange der klasse clottodbview in der datei lottodbview.cpp gemäß listing 27.2 ab. achten sie darauf, daß die eingefügten zeilen außerhalb des afx-abschnitts stehen.

listing 27.2 - ausschnitt aus der datei lottodbview.cpp

```
void clottodbview::dodataexchange(cdataexchange* pdx)
{
    crecordview::dodataexchange(pdx);
    if (pdx->m_bsaveandvalidate == false)
```

```
        m_oledtdatum = m_pset->m_datum;
    //{{afx_data_map(clottodbview)
    ddx_fieldtext(pdx, idc_z1, m_pset->m_z1, m_pset);
    ddx_fieldtext(pdx, idc_z2, m_pset->m_z2, m_pset);
    ddx_fieldtext(pdx, idc_z3, m_pset->m_z3, m_pset);
    ddx_fieldtext(pdx, idc_z4, m_pset->m_z4, m_pset);
    ddx_fieldtext(pdx, idc_z5, m_pset->m_z5, m_pset);
    ddx_fieldtext(pdx, idc_z6, m_pset->m_z6, m_pset);
    ddx_fieldtext(pdx, idc_ziehung, m_pset->m_ziehung, m_pset);
    ddx_fieldtext(pdx, idc_zz, m_pset->m_zz, m_pset);
    ddx_text(pdx, idc_datum, m_oledtdatum);
    //}}afx_data_map
    if (pdx->m_bsaveandvalidate == true)
        m_pset->m_datum = m_oledtdatum;

}
```

schließlich kommentieren sie noch die initialisierung der variablen m_datum im konstruktor der recordset-klasse aus, wie es listing 27.3 zeigt.

listing 27.3 - ausschnitt aus der datei lottodbset.cpp

```
clottodbset::clottodbset(cdatabase* pdb)
    : crecordset(pdb)
{
    //{{afx_field_init(clottodbset)
    m_ziehung = 0;
    //m_datum = 0;
    m_z1 = 0;
    m_z2 = 0;
    m_z3 = 0;
    m_z4 = 0;
    m_z5 = 0;
    m_z6 = 0;
    m_zz = 0;
    m_nfields = 9;
    //}}afx_field_init
    m_ndefaulttype = snapshot;
}
```

die anwendung ist zwar in diesem stadium bereits funktionsfähig, bietet aber noch keine möglichkeit, einen neuen datensatz hinzuzufügen oder einen datensatz zu löschen.

um einen neuen datensatz hinzuzufügen, ermitteln sie zunächst die nummer der letzten ziehung im recordset und addieren 1 dazu. dann fügen sie mit addnew einen neuen datensatz hinzu, setzen das feld der ziehungsnummer auf den neuen wert und übernehmen den neuen datensatz durch aufruf der funktion update. die ergebnismenge - d.h. den recordset - aktualisieren sie schließlich mit der funktion requery.

aktivieren sie im arbeitsbereich die registerkarte **klassen**, klicken sie mit der rechten maustaste auf die klasse clottodbset, und wählen sie aus dem kontextmenü den befehl **member-funktion hinzufügen**. als funktionstyp geben sie long und als funktionsdeklaration getmaxziehung ein. den zugriffsstatus übernehmen sie mit public.

übernehmen sie den code aus listing 27.4 in die neue funktion getmaxziehung.

listing 27.4 - die funktion getmaxziehung der klasse clottodbset

```
long clottodbset::getmaxziehung()
{
    movelast();
    return m_ziehung;
}
```

für das einfügen und löschen von datensätzen fügen sie entsprechende menübefehle in das menü **datensatz** ein. gehen sie im arbeitsbereich auf die registerkarte **ressourcen**, und öffnen sie das menü idr_mainframe (siehe abbildung 27.16). daraufhin erscheint das menü im rechten fensterbereich. öffnen sie das menü **datensatz**, und fügen sie die menübefehle **neuer datensatz** und **datensatz löschen** am unteren rand des menüs hinzu. markieren sie dazu das jeweils leere feld im menü, klicken sie mit der rechten maustaste, und wählen sie aus dem kontextmenü den befehl **eigenschaften**. im dialogfeld **menübefehl eigenschaften** geben sie die id, den titel und den statuszeilentext ein, wie es abbildung 27.16 für den befehl **datensatz löschen** zeigt. tabelle 27.3 faßt die einstellungen für beide menübefehle zusammen.

bild

abbildung 27.16: menübefehle für das menü datensatz hinzufügen

objekt	eigenschaft	einstellung
menübefehl	id titel statuszeilentext	idm_record_new neuer &datensatz fügt einen neuen datensatz hinzu\nneuer datensatz
menübefehl	id titel statuszeilentext	idm_record_delete datensatz l&öschen löscht den aktuellen datensatz\ndatensatz löschen

tabelle 27.3: eigenschaften für die beiden menübefehle

passend zu den menübefehlen nehmen sie noch zwei schaltflächen in die navigationssymbolleiste auf. öffnen sie im arbeitsbereich den zweig **toolbar**, doppelklicken sie auf idr_mainframe, und fügen sie zwei schaltflächen entsprechend abbildung 27.17 in die symbolleiste ein.

Bild

abbildung 27.17: in die symbolleiste nehmen sie zwei schaltflächen für neuen datensatz und datensatz löschen auf

wenn sie auf das bild der kleinen schaltfläche doppelklicken, öffnet sich das zugehörige dialogfeld **schaltfläche für symbolleiste eigenschaften**. hier tragen sie die gleichen ids ein, die sie bereits für die menübefehle verwendet haben (siehe tabelle 27.3).

ohne funktionalität nützen die schönsten menübefehle und schaltflächen nichts. es sind also noch die entsprechenden behandlungsroutinen zu erstellen. öffnen sie den klassen-assistenten, aktivieren sie die registerkarte **nachrichtenzuordnungstabellen**, und wählen sie den klassennamen clottodbview. suchen sie in der liste **objekt-ids** den eintrag idm_record_new. in der liste nachrichten markieren sie command. klicken sie auf **funktion hinzufügen**. daraufhin erscheint das dialogfeld **member-funktion hinzufügen**. den vorgegebenen namen können sie übernehmen. in die funktion onrecordnew geben sie den code entsprechend listing 27.5 ein. wiederholen sie die gleichen schritte für den eintrag idm_record_delete. in die funktion onrecorddelete übernehmen sie den code aus listing 27.6. der code für das löschen einer ziehung enthält eine sicherheitsabfrage, ob der benutzer die ziehung wirklich löschen will. nur wenn der benutzer mit **ja** bestätigt, wird die ziehung tatsächlich gelöscht.

listing 27.5 - die funktion onrecordnew der klasse clottodbview

```
void clottodbview::onrecordnew()
{
    // todo: code für befehlsbehandlungsroutine hier einfügen
    crecordset* pset = ongetrecordset();
    if (pset->canupdate() && !pset->isdeleted())
    {
        pset->edit();
        if (!updatedata())
            return;
        pset->update();
    }
    long m_lnewid = m_pset->getmaxziehung() +1;
    m_pset->addnew();
    m_pset->m_ziehung = m_lnewid;
    m_pset->update();
    m_pset->requery();
    m_pset->movelast();
    updatedata(false);
}
```

listing 27.6 - die funktion onrecorddelete der klasse clottodbview

```
void clottodbview::onrecorddelete()
{
    // todo: code für befehlsbehandlungsroutine hier einfügen
    if (messagebox("möchten sie diese ziehung wirklich
            löschen?", "diese ziehung löschen?", mb_yesno |
                                    mb_iconquestion) == idyes)
    {
        m_pset->delete();
        m_pset->moveprev();
        updatedata(false);
    }
}
```

jetzt können sie die anwendung kompilieren und ausführen. beim start erscheint zunächst das anmeldedialogfeld (siehe abbildung 27.18), das sie bereits bei der auswahl der datenquelle im anwendungs-assistenten kennengelernt haben (siehe abbildung 27.11).

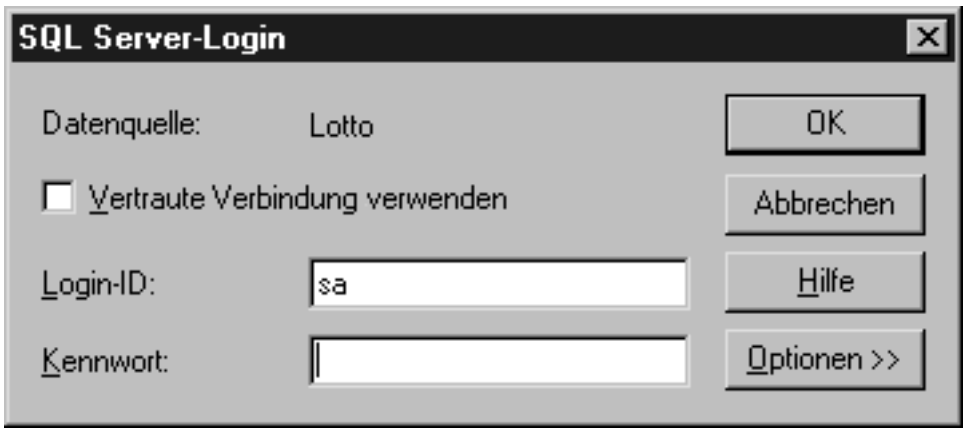


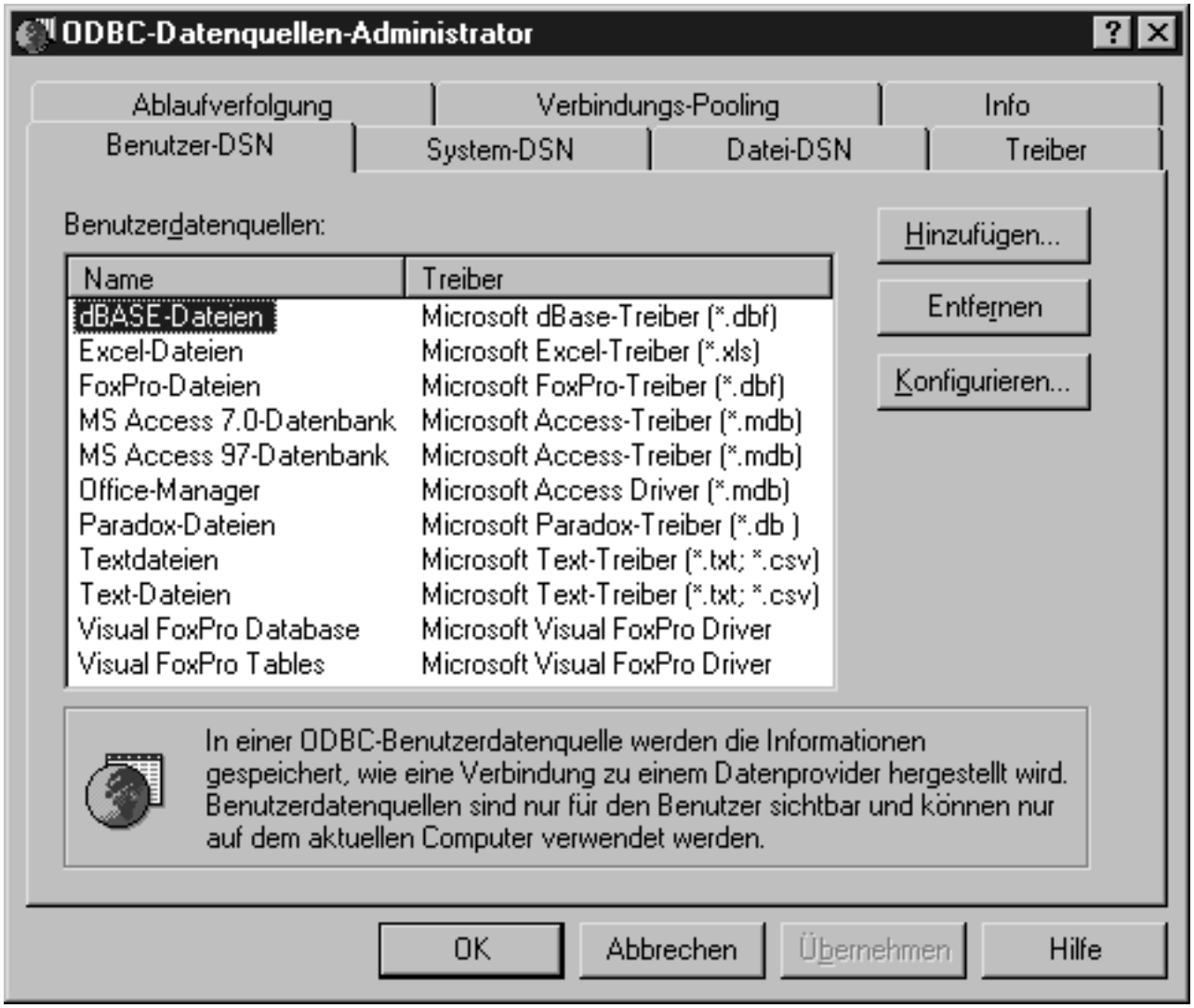
abbildung 27.18: anmeldung bei sql server nach start der anwendung

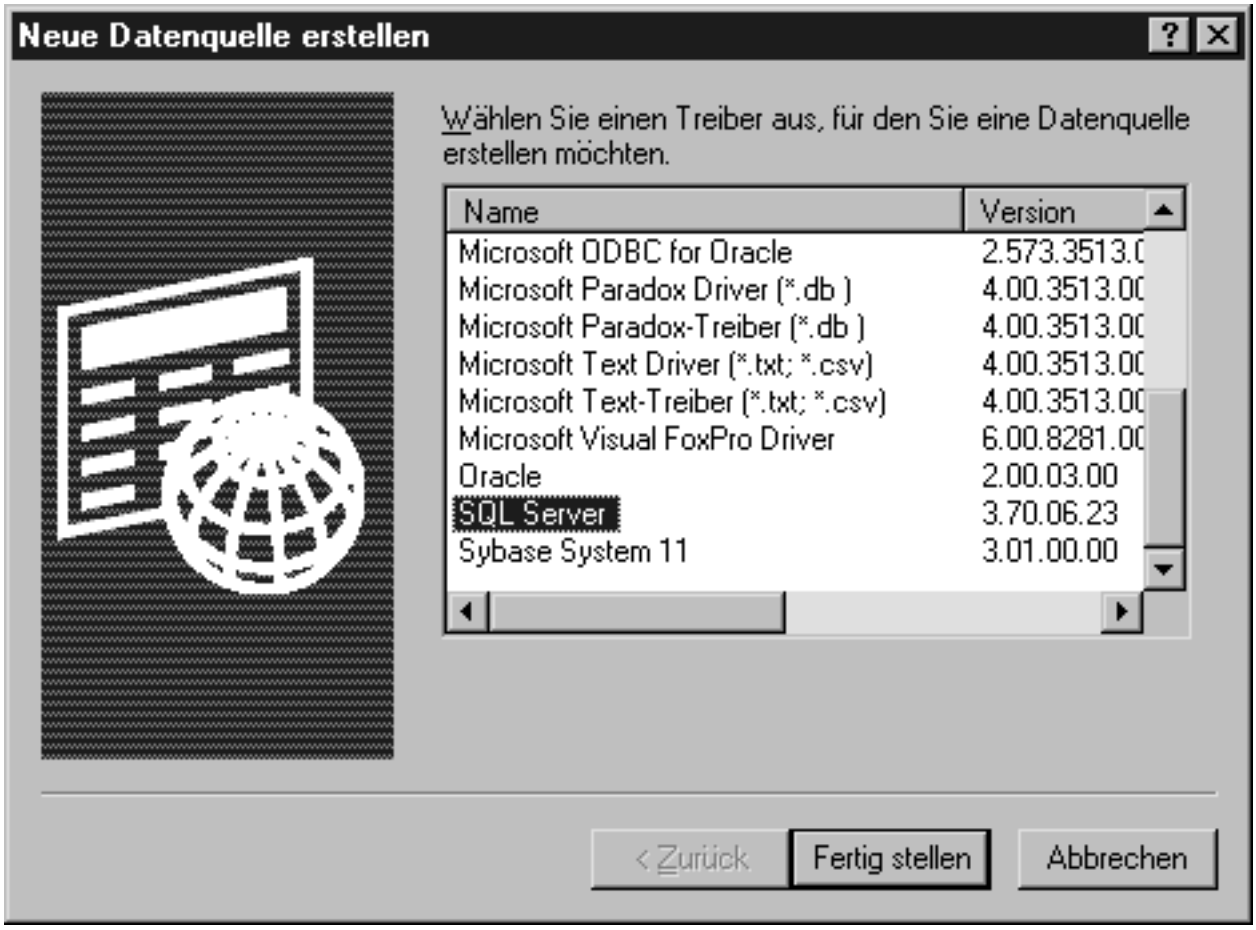
klicken sie auf **ok**, um die verbindung zu sql server herzustellen. die anwendung ruft nun die daten aus der tabelle ldaten der datenbank lotto ab und zeigt nach kurzer zeit die ergebnisse an (siehe abbildung 27.19).

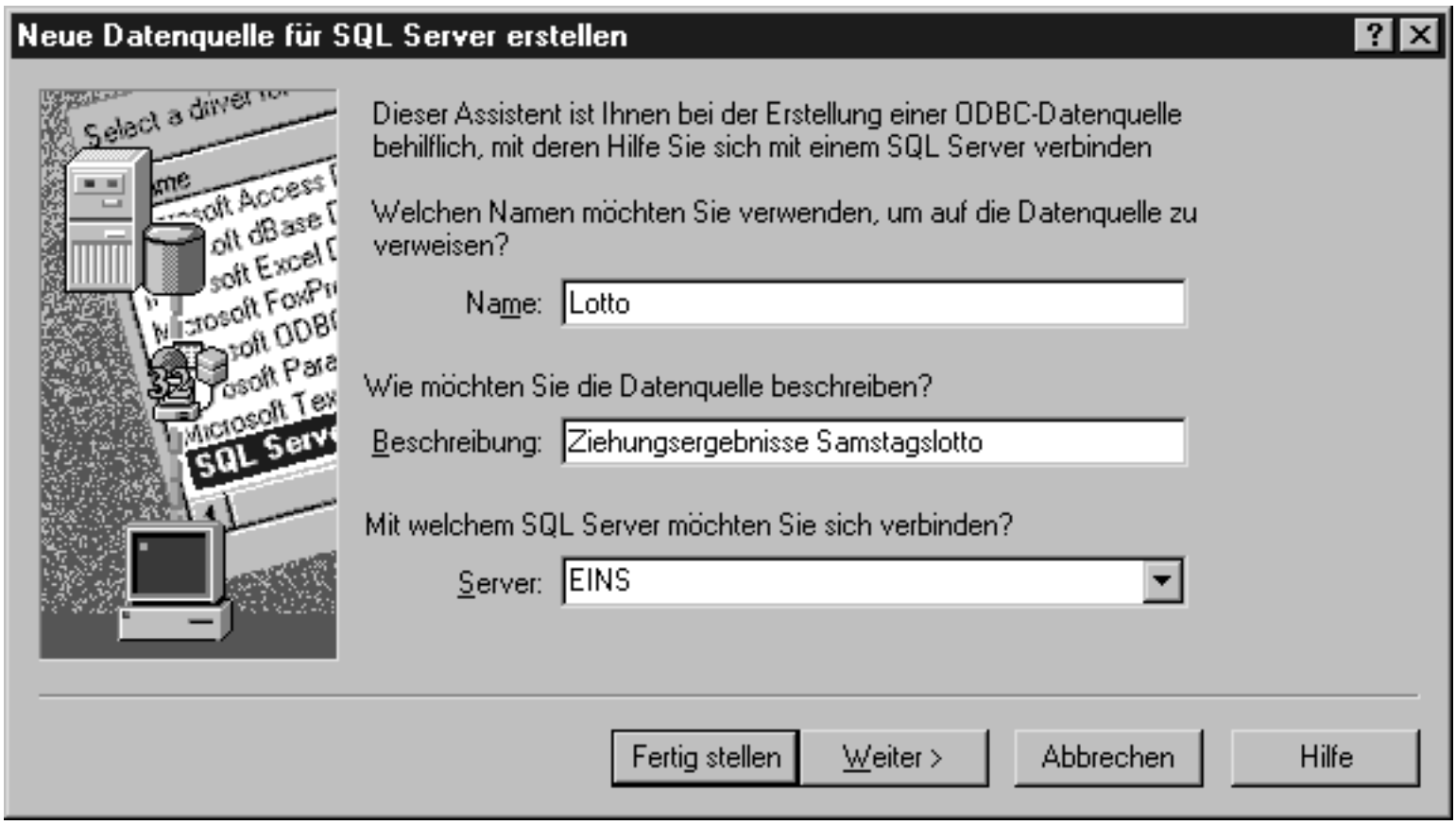
abbildung 27.19: die laufende anwendung lottodb

über die schaltflächen der symbolleiste oder die menübefehle können sie jetzt zum ersten, letzten, vorherigen oder nächsten datensatz navigieren. außerdem läßt sich mit den zusätzlich eingebauten befehlen ein neuer datensatz einfügen oder ein vorhandener löschen. probieren sie es aus.

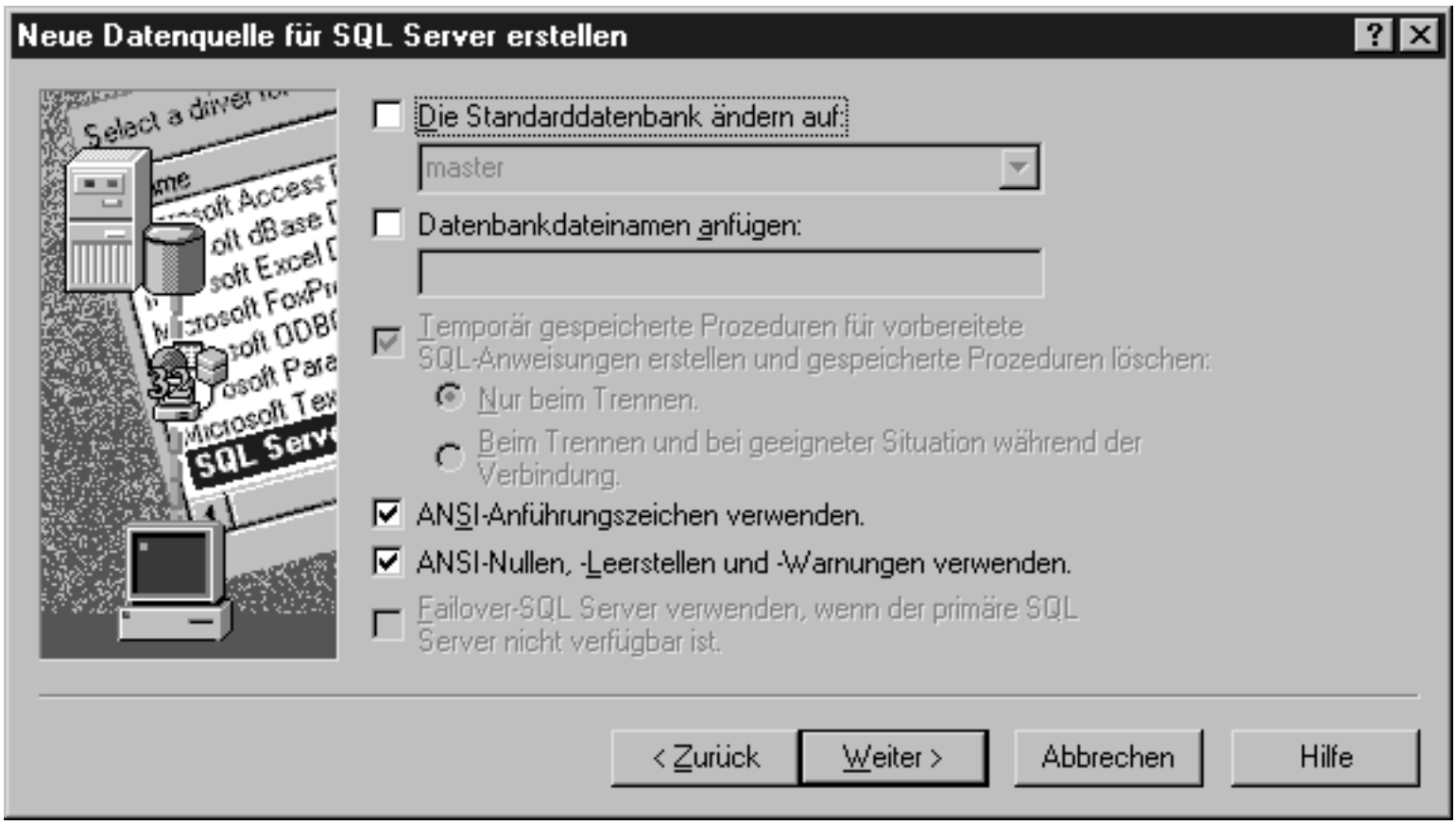
wundern sie sich nicht, wenn nach dem start der anwendung im feld **zusatzzahl** eine 0 erscheint. die zusatzzahl wurde erst mit der 37. ziehung am 17. juni 1956 eingeführt.


ODBC-Datenquellen-Administrator
Ablaufverfolgung
Verbindungs-Pooling
Info
Benutzer-DSN
System-DSN
Datei-DSN
Treiber
Benutzerdatenquellen:
Hinzufügen...
Entfernen
Konfigurieren...
Name
Treiber
dBASE-Dateien
Microsoft dBase-Treiber (*.dbf)
Excel-Dateien
Microsoft Excel-Treiber (*.xls)
FoxPro-Dateien
Microsoft FoxPro-Treiber (*.dbf)
MS Access 7.0-Datenbank
Microsoft Access-Treiber (*.mdb)
MS Access 97-Datenbank
Microsoft Access-Treiber (*.mdb)
Office-Manager
Microsoft Access Driver (*.mdb)
Paradox-Dateien
Microsoft Paradox-Treiber (*.db)
Textdateien
Microsoft Text-Treiber (*.txt; *.csv)
Text-Dateien
Microsoft Text-Treiber (*.txt; *.csv)
Visual FoxPro Database
Microsoft Visual FoxPro Driver
Visual FoxPro Tables
Microsoft Visual FoxPro Driver
In einer ODBC-Benutzerdatenquelle werden die Informationen gespeichert, wie eine Verbindung zu einem Datenprovider hergestellt wird. Benutzerdatenquellen sind nur für den Benutzer sichtbar und können nur auf dem aktuellen Computer verwendet werden.
OK
Abbrechen
Übernehmen
Hilfe

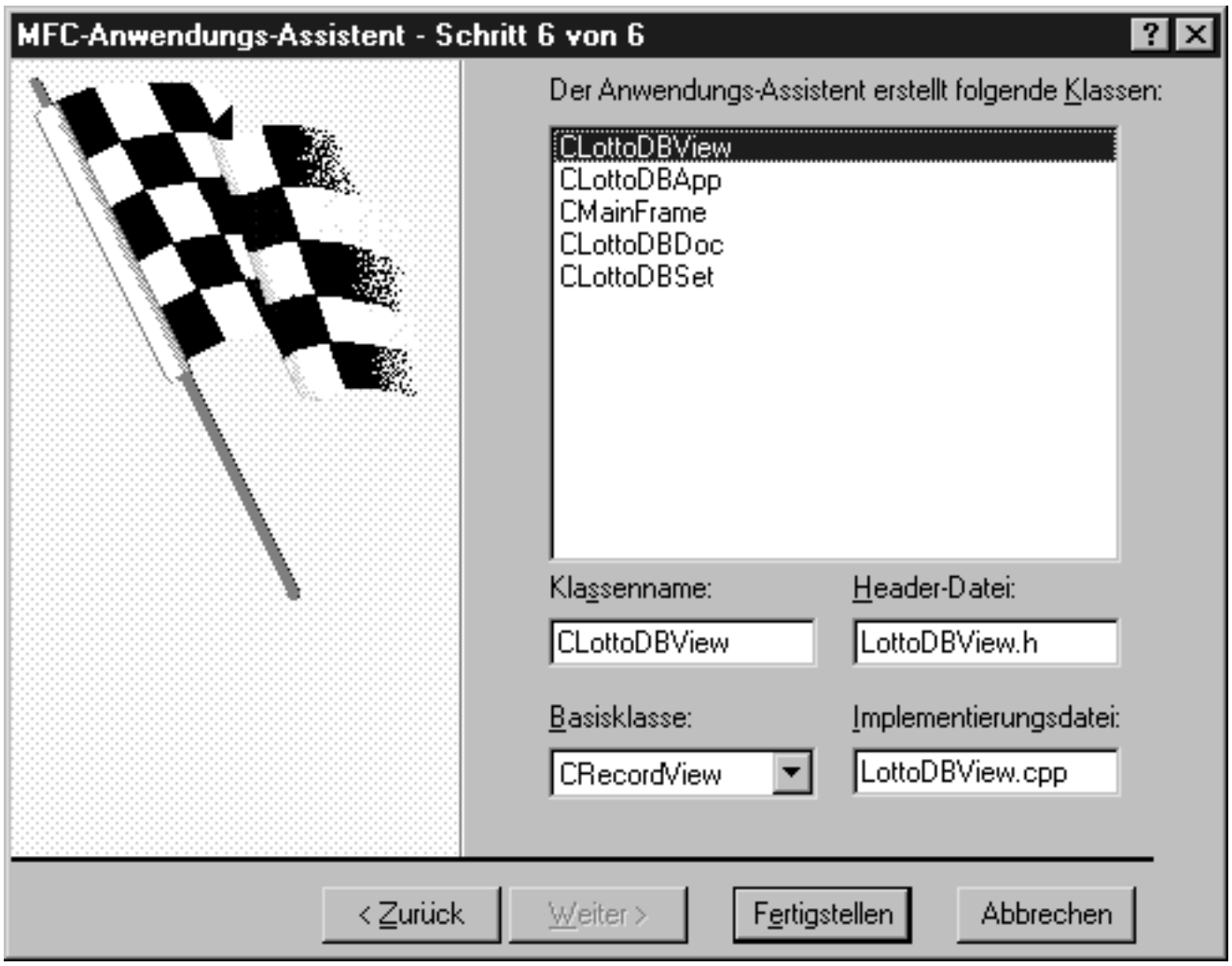

Neue Datenquelle erstellen
Wählen Sie einen Treiber aus, für den Sie eine Datenquelle erstellen möchten.
Name
Version
Microsoft ODBC for Oracle 2.573.3513.0
Microsoft Paradox Driver (*.db) 4.00.3513.00
Microsoft Paradox-Treiber (*.db) 4.00.3513.00
Microsoft Text Driver (*.txt; *.csv) 4.00.3513.00
Microsoft Text-Treiber (*.txt; *.csv) 4.00.3513.00
Microsoft Visual FoxPro Driver 6.00.8281.00
Oracle 2.00.03.00
SQL Server 3.70.06.23
Sybase System 11 3.01.00.00
< Zurück
Fertig stellen
Abbrechen

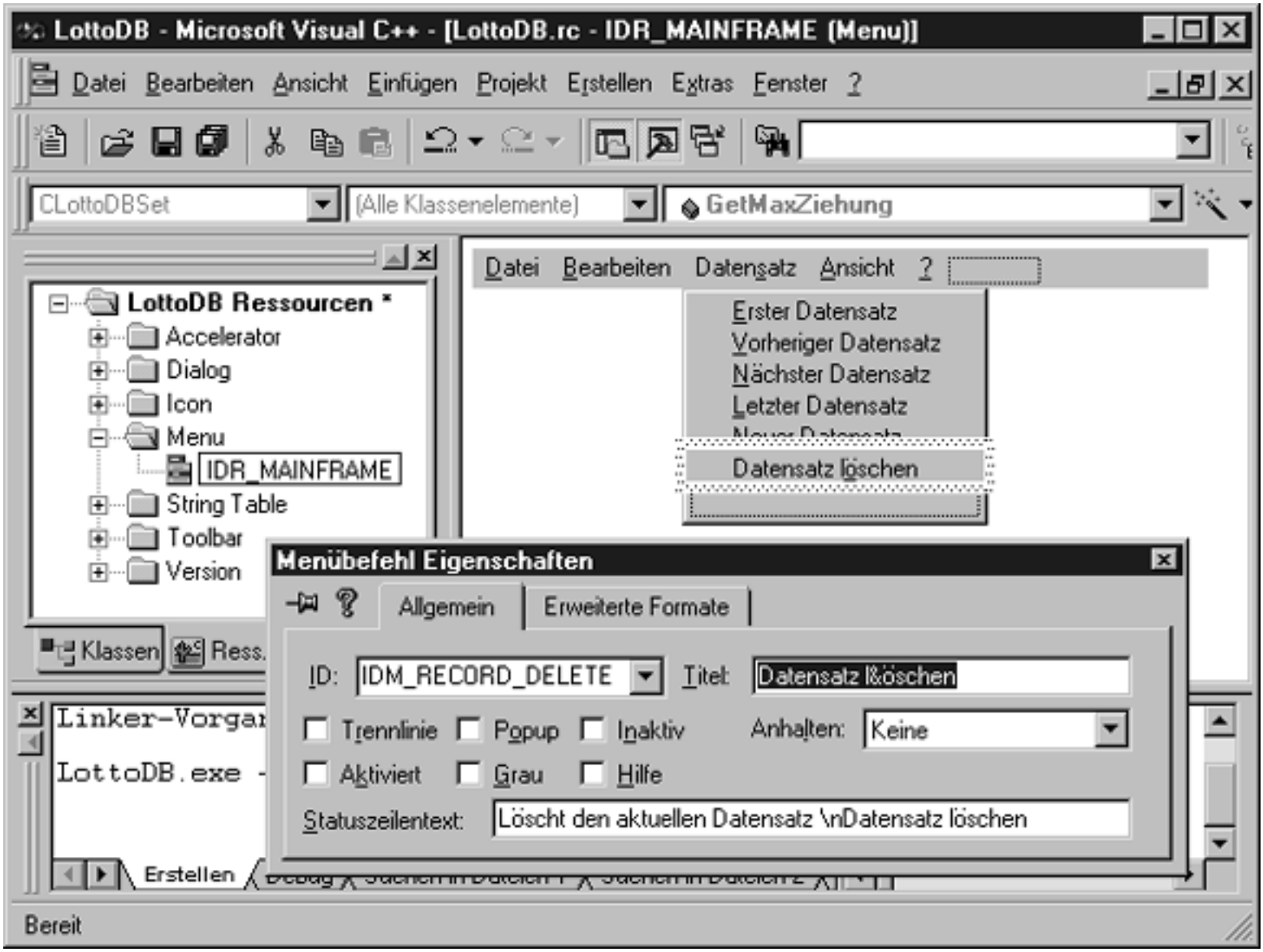

Neue Datenquelle für SQL Server erstellen
Dieser Assistent ist Ihnen bei der Erstellung einer ODBC-Datenquelle behilflich, mit deren Hilfe Sie sich mit einem SQL Server verbinden
Welchen Namen möchten Sie verwenden, um auf die Datenquelle zu verweisen?
Name: Lotto
Wie möchten Sie die Datenquelle beschreiben?
Beschreibung: Ziehungsergebnisse Samstagslotto
Mit welchem SQL Server möchten Sie sich verbinden?
Server: EINS
Fertig stellen
Weiter >
Abbrechen
Hilfe

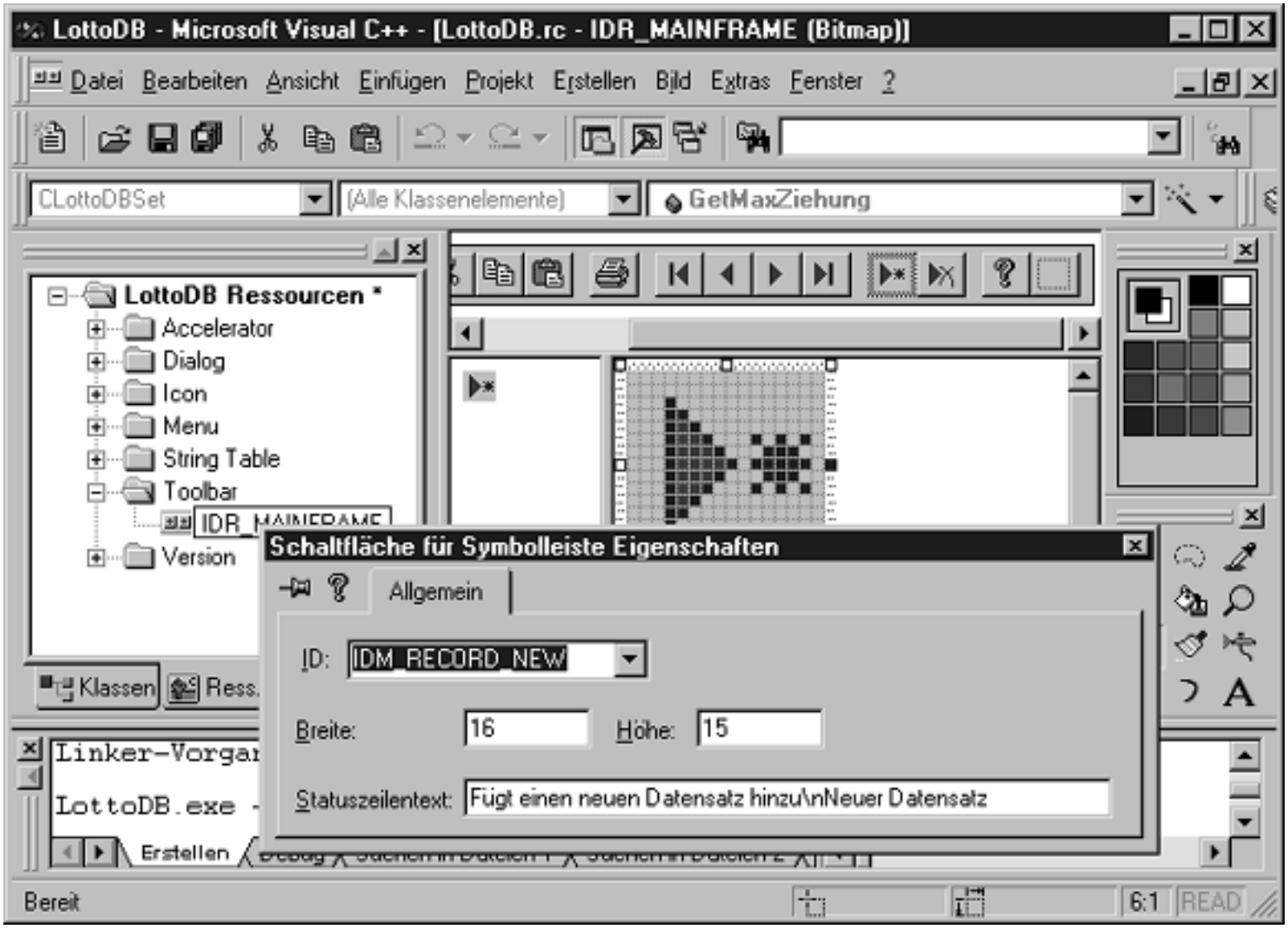

Neue Datenquelle für SQL Server erstellen
Wie soll der SQL Server die Authentizität der Login-ID bestätigen?
Mit Windows NT-Authentifizierung durch die Login-ID des Netzwerks.
Mit SQL Server-Authentifizierung durch Login-ID und Kennwort eingegeben vom Benutzer.
Um die Netzwerkbibliothek für die Kommunikation mit dem SQL Server zu ändern, klicken Sie auf Clientkonfiguration.
Clientkonfiguration...
Zum SQL Server verbinden, um Standardeinstellungen für die zusätzlichen Konfigurationsoptionen zu erhalten.
Login-ID:
sa
Kennwort:
< Zurück
Weiter >
Abbrechen
Hilfe


Neue Datenquelle für SQL Server erstellen
Die Standarddatenbank ändern auf:
master
Datenbankdateinamen anfügen:
Temporär gespeicherte Prozeduren für vorbereitete SQL-Anweisungen erstellen und gespeicherte Prozeduren löschen:
Nur beim Trennen.
Beim Trennen und bei geeigneter Situation während der Verbindung.
ANSI-Anführungszeichen verwenden.
ANSI-Nullen, -Leerstellen und -Warnungen verwenden.
Failover-SQL Server verwenden, wenn der primäre SQL Server nicht verfügbar ist.
< Zurück
Weiter >
Abbrechen
Hilfe


Neue Datenquelle für SQL Server erstellen
Sprache der SQL Server-Systemmeldungen ändern auf:
German
Konvertierung für Zeichen durchführen
Regionale Einstellungen verwenden für die Ausgabe von Währung, Zahlen, Datum- und Zeitangaben.
Abfragen mit langer Laufzeit speichern in der Protokolldatei:
C:\QUERY.LOG
Durchsuchen...
Mindestlaufzeit für Protokollierung (in Millisekunden):
30000
ODBC-Treiberstatistik protokollieren in der Protokolldatei:
C:\STATS.LOG
Durchsuchen...
< Zurück
Fertig stellen
Abbrechen
Hilfe


MFC-Anwendungs-Assistent - Schritt 6 von 6
Der Anwendungs-Assistent erstellt folgende Klassen:
CLottoDBView
CLottoDBApp
CMainFrame
CLottoDBDoc
CLottoDBSet
Klassenname:
CLottoDBView
Header-Datei:
LottoDBView.h
Basisklasse:
CRecordView
Implementierungsdatei:
LottoDBView.cpp
< Zurück
Weiter >
Fertigstellen
Abbrechen


LottoDB - Microsoft Visual C++ - [LottoDB.rc - IDR_MAINFRAME (Menu)]
Datei Bearbeiten Ansicht Einfügen Projekt Erstellen Extras Fenster ?
CLottoDBSet
(Alle Klassenelemente)
GetMaxZiehung
LottoDB Ressourcen *
Accelerator
Dialog
Icon
Menu
IDR_MAINFRAME
String Table
Toolbar
Version
Klassen
Ress.
Datei Bearbeiten Datensatz Ansicht ?
Erster Datensatz
Vorheriger Datensatz
Nächster Datensatz
Letzter Datensatz
Datensatz löschen
Menübefehl Eigenschaften
Allgemein
Erweiterte Formate
ID: IDM_RECORD_DELETE
Titel: Datensatz l&öschen
Trennlinie
Popup
Inaktiv
Anhalten: Keine
Aktiviert
Grau
Hilfe
Statuszeilentext: Löscht den aktuellen Datensatz \nDatensatz löschen
Linker-Vorgar
LottoDB.exe
Erstellen
Bereit


LottoDB - Microsoft Visual C++ - [LottoDB.rc - IDR_MAINFRAME (Bitmap)]
Datei Bearbeiten Ansicht Einfügen Projekt Erstellen Bild Extras Fenster ?
CLottoDBSet
(Alle Klassenelemente)
GetMaxZiehung
LottoDB Ressourcen *
Accelerator
Dialog
Icon
Menu
String Table
Toolbar
Version
Klassen
Schaltfläche für Symbolleiste Eigenschaften
Allgemein
ID:
IDM_RECORD_NEW
Breite:
16
Höhe:
15
Statuszeilentext:
Fügt einen neuen Datensatz hinzu\nNeuer Datensatz
Linker-Vorgan
LottoDB.exe -
Erstellen
Bereit
6:1
READ

kapitel 28 im internet veröffentlichen

28.1 daten für die welt

in den letzten jahren sind viele unternehmen dazu übergegangen, firmenspezifische daten im web zu veröffentlichen. vornehmlich sind das produktinformationen, die zudem noch einer ständigen veränderung unterliegen. die zu veröffentlichenden daten stehen meist in einer datenbank zu verfügung. was liegt also näher, als die daten aus der datenbank in eine webseite zu übertragen und dort regelmäßig zu aktualisieren?

sql server bietet von haus aus die erforderliche unterstützung, um daten automatisch auf webseiten zu veröffentlichen. dieses kapitel zeigt, wie sie eine webseite mit dem web-assistenten erstellen. dann geht es um die gespeicherte prozedur sp_makewebtask, für die der assistent eine schnittstelle bereitstellt.

28.2 html-dokumente für webseiten

die zu veröffentlichenden daten werden auf einer - statischen - webseite präsentiert. dabei sollte man auf den umfang der seite achten, damit die zeiten für den download nicht ins unermeßliche gehen. als richtwert kann man 50 kbyte annehmen. grafiken verlangsamen naturgemäß den download-vorgang. hinsichtlich der gestaltungsmöglichkeiten einer webseite schränkt sie der web-assistent in keiner weise ein. wenn sie mit den gebotenen standardformaten nicht zufrieden sind, können sie die webseite problemlos auf einer von ihnen bereitgestellten vorlagendatei aufbauen lassen.

28.3 web-assistent und sp_makewebtask

eine webseite können sie sowohl mit der gespeicherten prozedur sp_makewebtask als auch mit dem web-assistenten erstellen. beide mechanismen arbeiten hand in hand. der web-assistent stellt eine schnittstelle - oder benutzeroberfläche - für die gespeicherte prozedur sp_makewebtask bereit. der web-assistent ist ein mächtiges werkzeug und gehört wohl zu den umfangreichsten assistenten im repertoire von sql server. der nächste abschnitt zeigt die - immerhin recht umfangreiche - schrittfolge, die sie mit dem web-assistenten durchlaufen, um die daten aus der tabelle titles der datenbank pubs in das format einer webseite zu bringen. danach haben sie einen überblick über die zahlreichen optionen, die sich beim erstellen eines webauftrags festlegen lassen. der sich anschließende abschnitt zur gespeicherten prozedur sp_makewebtask baut darauf auf und erläutert die wichtigsten parameter dieser prozedur.

28.3.1 web-assistent

um den web-assistenten aufzurufen, erweitern sie die konsolenstruktur bis zum ordner **datenbanken**, wählen dann aus dem menü des enterprise managers den befehl **assistenten** oder klicken auf der symbolleiste auf die schaltfläche **assistenten auswählen**. im dialogfeld **assistent auswählen** erweitern

sie den zweig **verwaltung**, markieren *sql server web-assistent* und klicken auf **ok**. führen sie die folgenden schritte aus, um die webseite mit dem web-assistenten zu erstellen:

1. klicken sie im startdialogfeld des assistenten auf **weiter**. im zweiten dialogfeld (siehe abbildung 28.1) wählen sie die datenbank aus, deren daten sie auf der webseite veröffentlichen wollen.

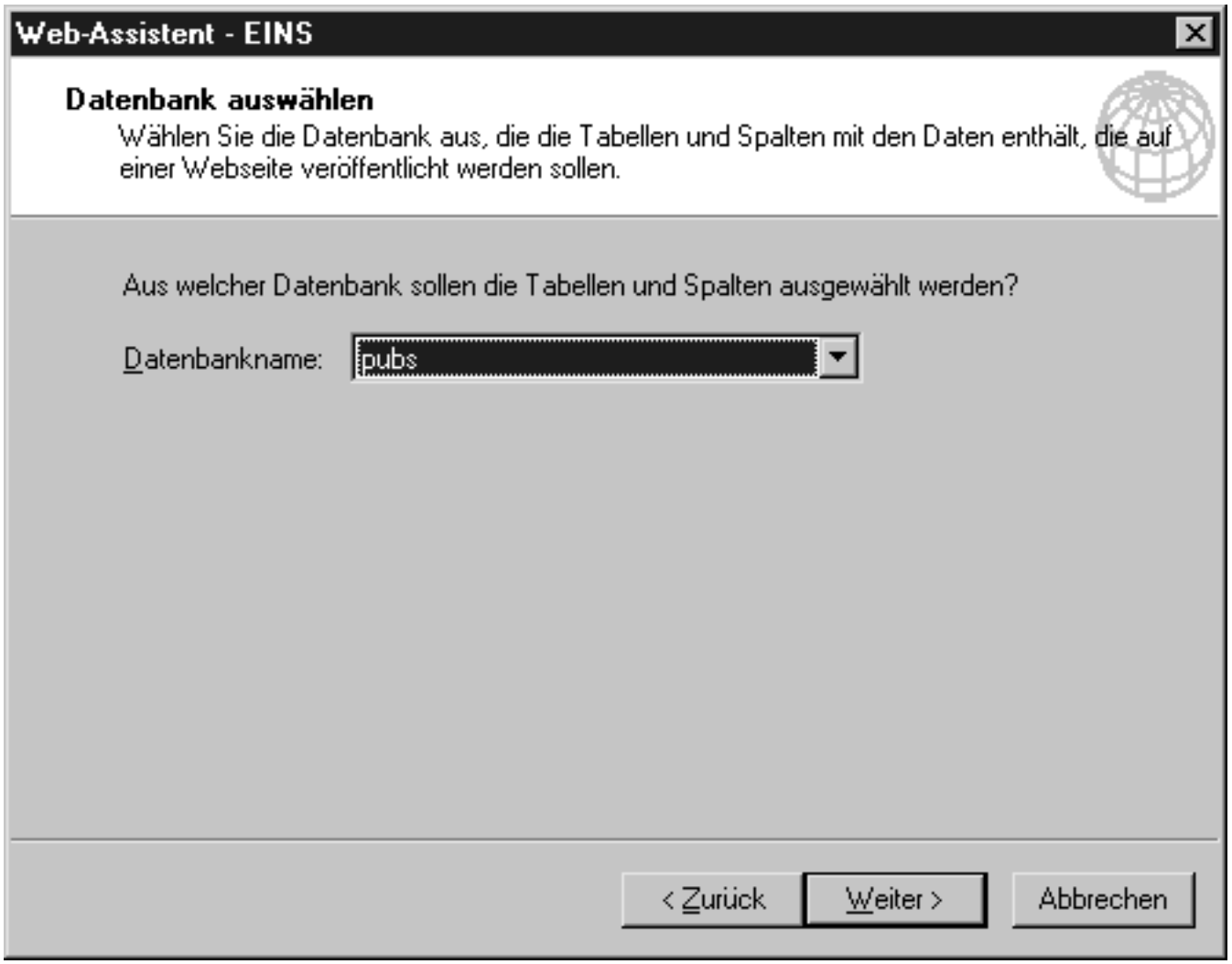


abbildung 28.1: im zweiten dialogfeld wählen sie die datenbank aus

2. im dialogfeld **einen neuen auftrag für den web-assistenten erstellen** (siehe abbildung 28.2) geben sie zunächst einen namen für den auftrag ein (im beispiel »webseite titel«). der assistent gibt als name '*datenbank* webseite' vor.

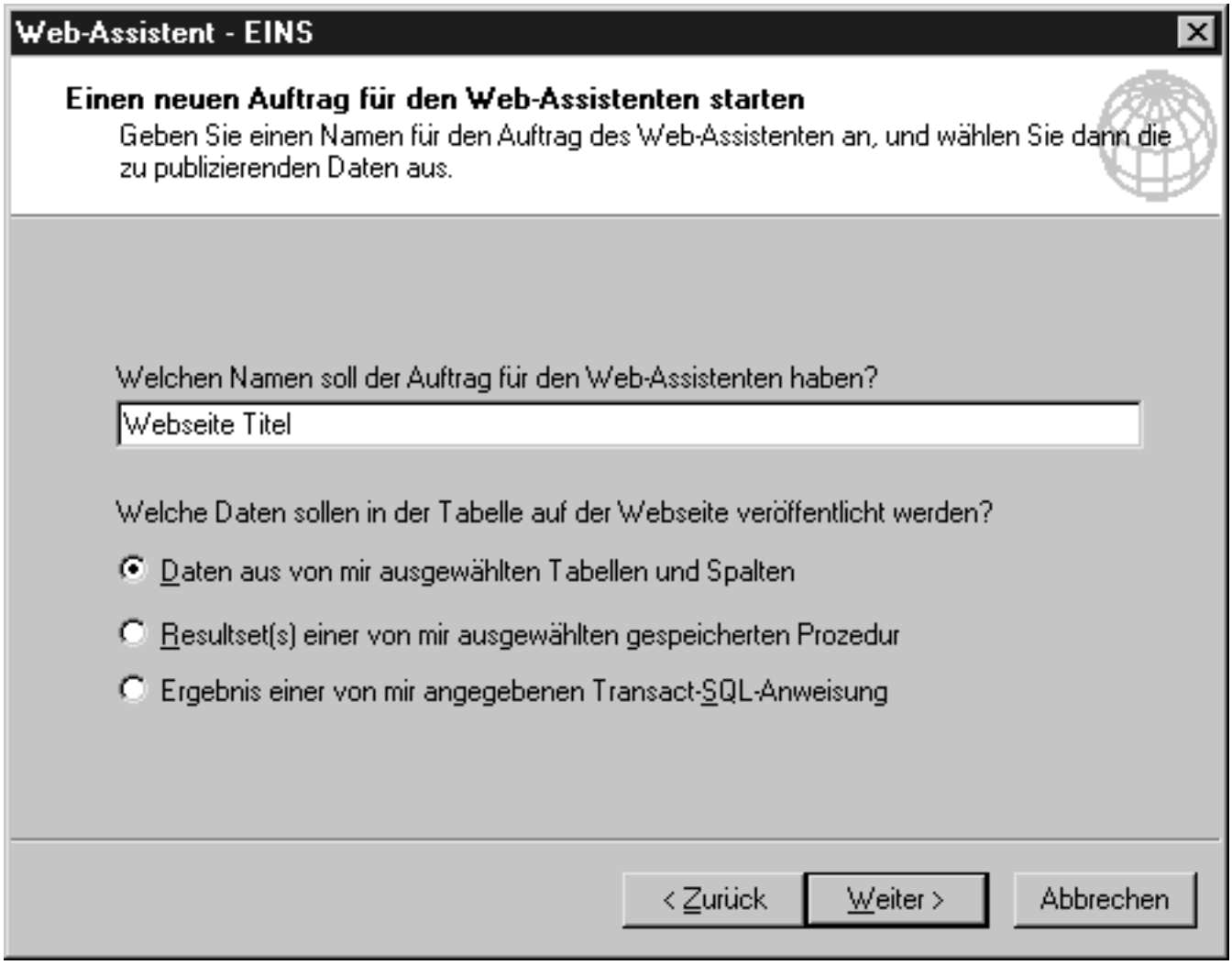


abbildung 28.2: einen neuen auftrag für den web-assistenten erstellen

für die auswahl der daten, die auf der webseite erscheinen sollen, stehen ihnen drei optionen zur verfügung. die nachfolgenden schritte 2 und 3 sind für die erste, die schritte 4 und 5 für die zweite und der schritt 6 für die dritte option gültig. anschließend geht es bei allen optionen mit schritt 7 weiter.

3. *daten aus von mir ausgewählten tabellen und spalten*: bei dieser option durchlaufen sie zwei weitere dialogfelder. im ersten (siehe abbildung 28.3) wählen sie eine tabelle im drop-down-listenfeld **verfügbare tabellen** aus. daraufhin erscheinen die zugehörigen spalten im linken listenfeld **tabellenspalten**. klicken sie auf die schaltfläche **alle hinzufügen**, wenn sie alle spalten der tabelle anzeigen möchten. sie können auch einzelne spalten im linken listenfeld markieren und mit der schaltfläche **hinzufügen** in das rechte listenfeld **ausgewählte spalten** übernehmen. klicken sie auf **weiter**.

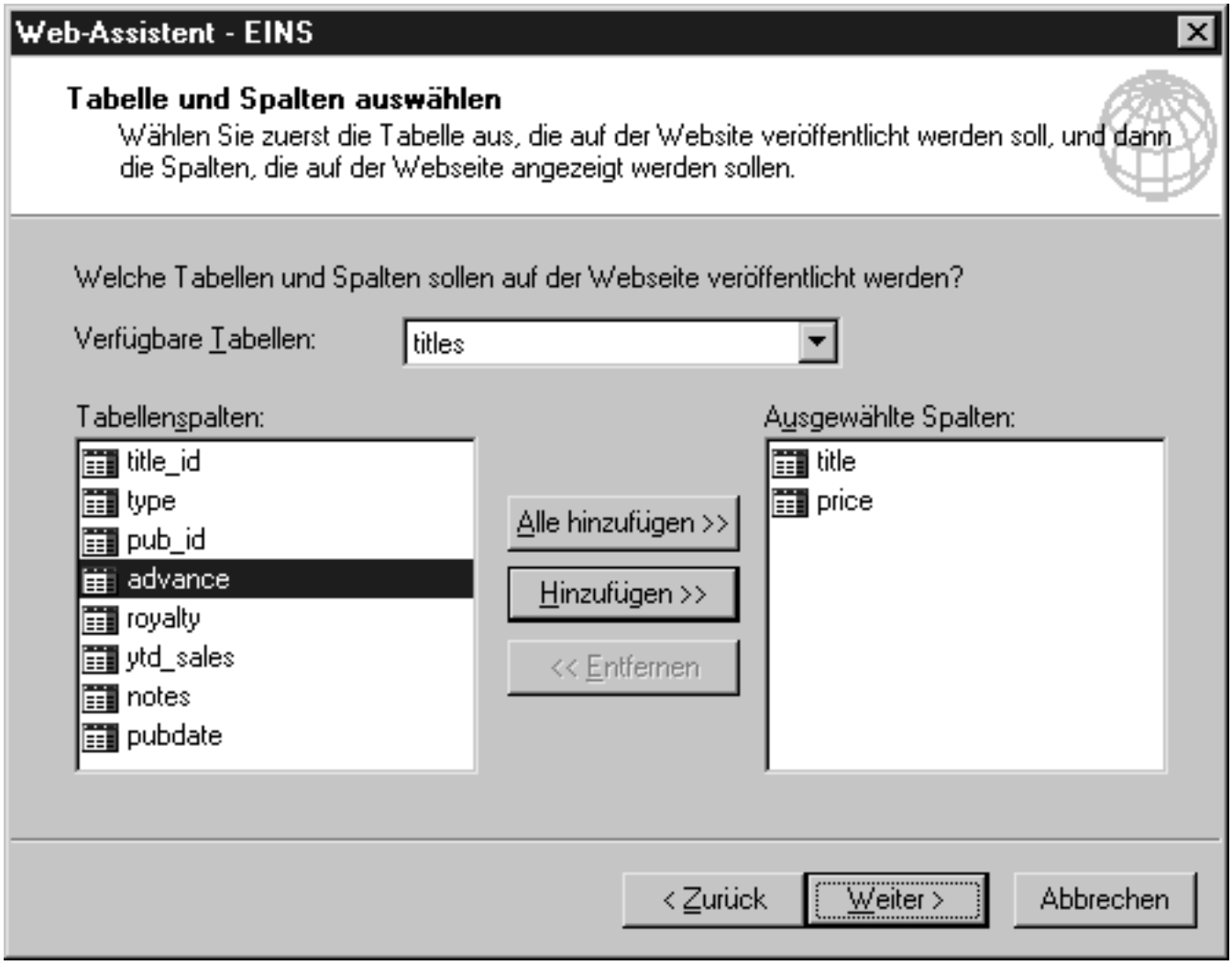


abbildung 28.3: in diesem dialogfeld wählen sie eine tabelle und die spalten aus

4. im dialogfeld **zeilen auswählen** (siehe abbildung 28.4) legen sie fest, welche tabellenzeilen auf der webseite veröffentlicht werden sollen. voreingestellt ist die option *alle zeilen*. bei der zweiten option können sie bestimmte kriterien festlegen und mit *und* bzw. *oder* verknüpfen. wenn sie eine komplexere bedingung formulieren möchten, wählen sie die dritte option und geben eine where-klausel an. im beispiel ist als bedingung is not null definiert, so daß nur titel auf der webseite erscheinen, für die auch ein preis vorhanden ist.

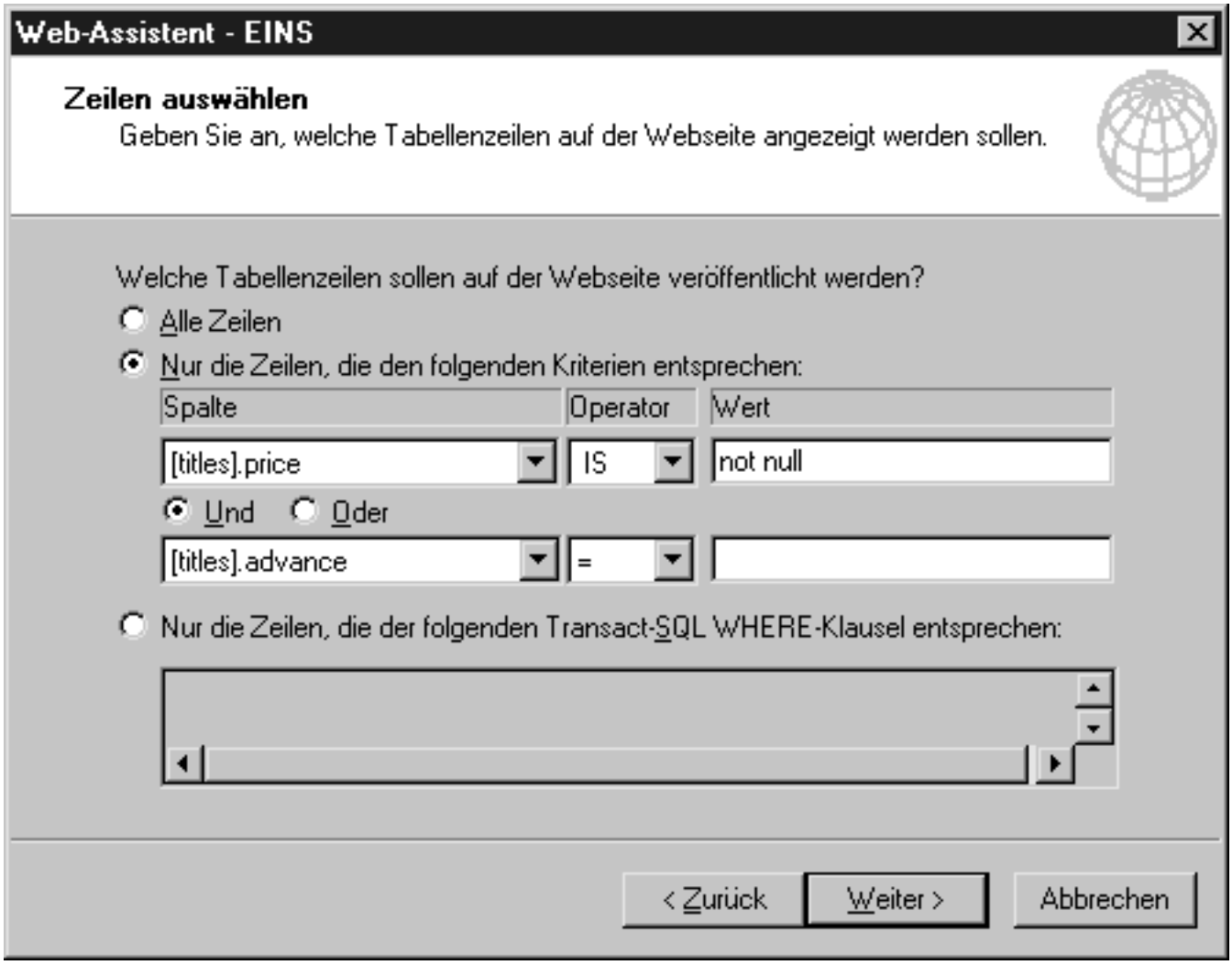


abbildung 28.4: bei der auswahl der anzuzeigenden zeilen können sie bedingungen formulieren

klicken sie auf **weiter**, und gehen sie zu schritt 7.

5. *resultset(s) einer von mir ausgewählten gespeicherten prozedur*: bei dieser option legen sie zuerst im dialogfeld **gespeicherte prozedur auswählen** die prozedur fest, deren ergebnis auf der webseite erscheinen soll (siehe abbildung 28.5).

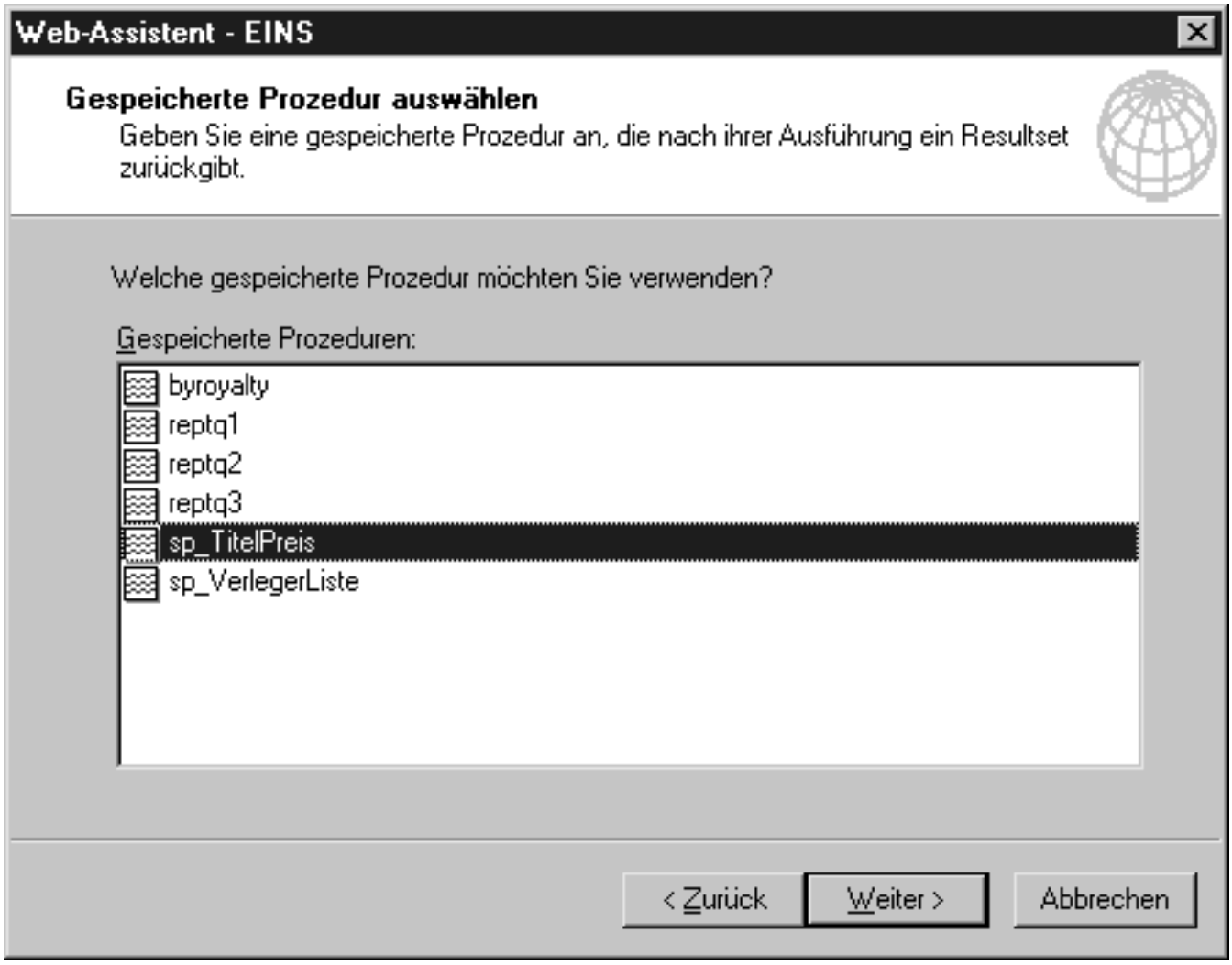


abbildung 28.5: eine gespeicherte prozedur auswählen

die hier ausgewählte prozedur wurde bereits vorher erstellt und verlangt als parameter eine obere preisgrenze, bis zu der titel angezeigt werden. die prozedur wurde mit folgenden anweisungen erstellt:

```
use pubs
go
create procedure sp_titelpreis (@limit money) as
  select title, price from titles
  where price <= @limit and price is not null
go
```

6. wenn an die prozedur parameter zu übergeben sind, bringt sie der web-assistent in ein weiteres dialogfeld, in dem sie werte für die parameter festlegen können (siehe abbildung 28.6). da sich der wert nicht dynamisch bereitstellen läßt, müssen sie einen wert fest eintragen. um alle bücher anzuzeigen, wählen sie 200 (in der tabelle titles der datenbank pubs sind momentan bücher bis maximal $ 22.95 erfaßt).

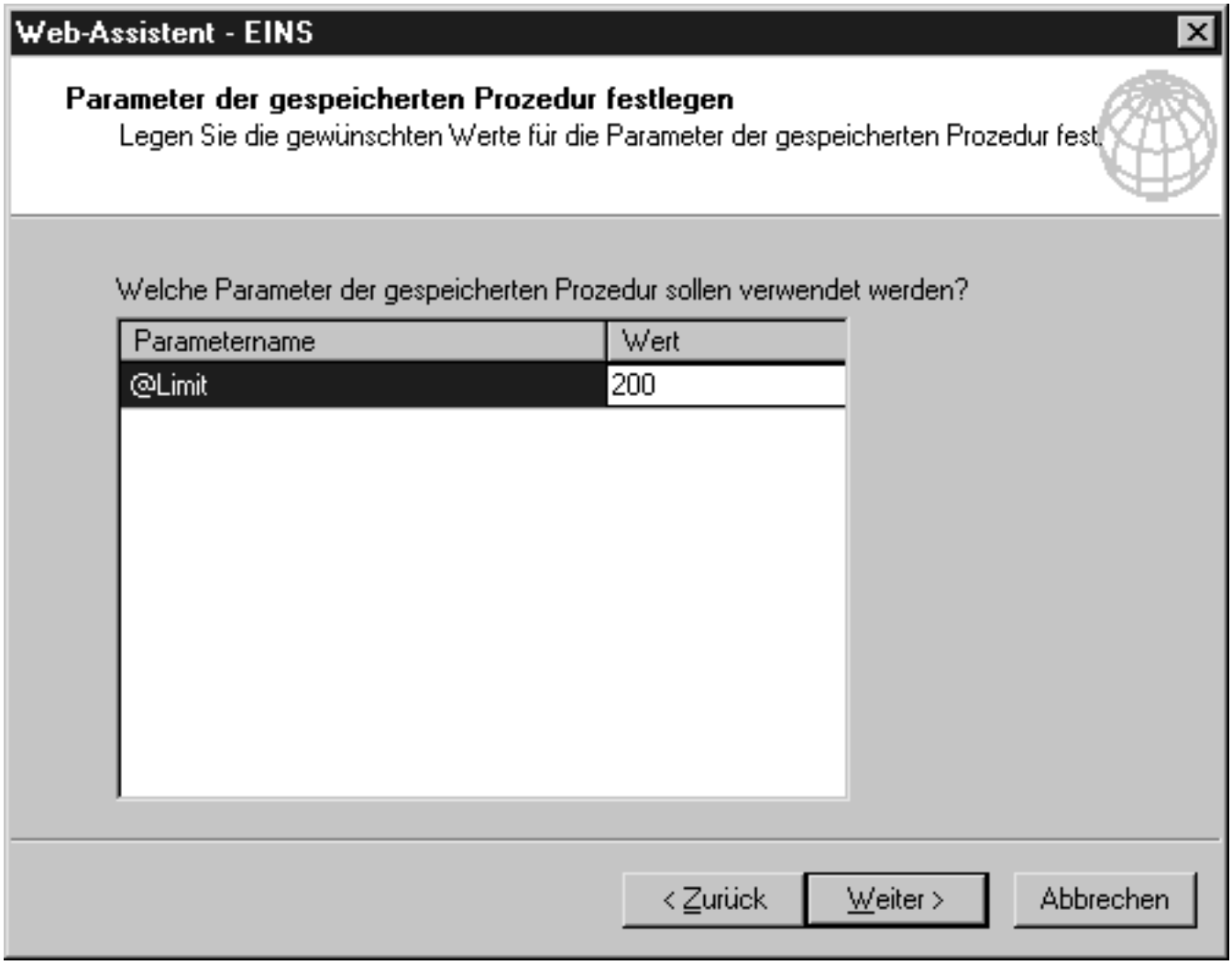


abbildung 28.6: wenn die gespeicherte prozedur parameter erfordert, legen sie die werte in diesem dialogfeld fest

klicken sie auf **weiter**, und gehen sie zu schritt 7.

7. *ergebnis einer von mir angegebenen transact-sql-anweisung*: nur bei dieser dritten option ist es möglich, daten aus verschiedenen tabellen auszuwählen. außerdem können sie mehrere ergebnistabellen auf der webseite präsentieren, indem sie einfach die entsprechende anzahl von select-anweisungen schreiben. die in abbildung 28.7 gezeigte anweisung entspricht der im schritt 4 wiedergegebenen prozedur, allerdings ohne bedingung für die obergrenze eines preises. klicken sie auf **weiter**.

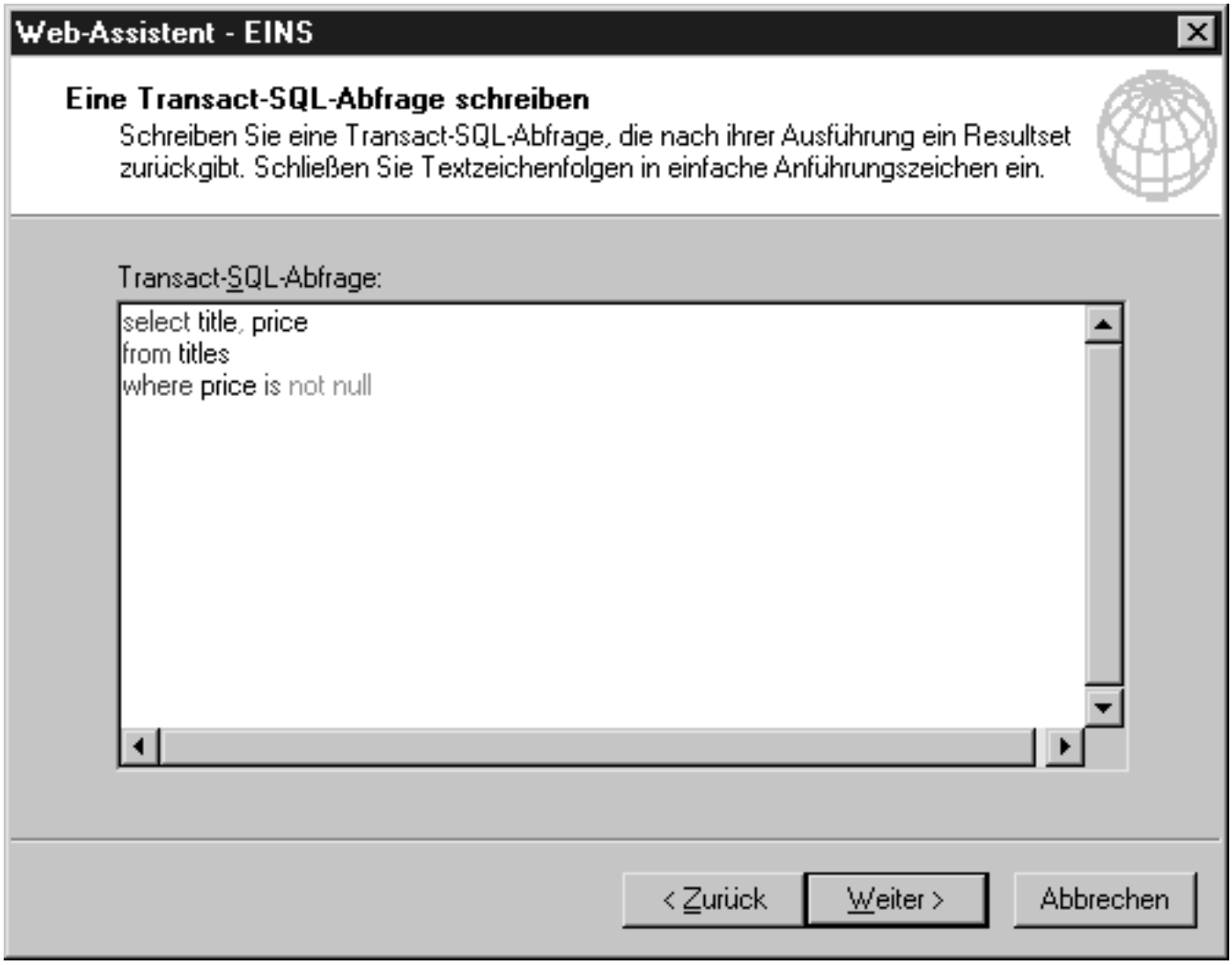


abbildung 28.7: in diesem dialogfeld geben sie eine vollständige transact-sql-anweisung ein

8. im dialogfeld **termin für auftrag des web-assistenten planen** (siehe abbildung 28.8) legen sie fest, wann die webseite zu erstellen ist und wie häufig die aktualisierungen stattfinden sollen.

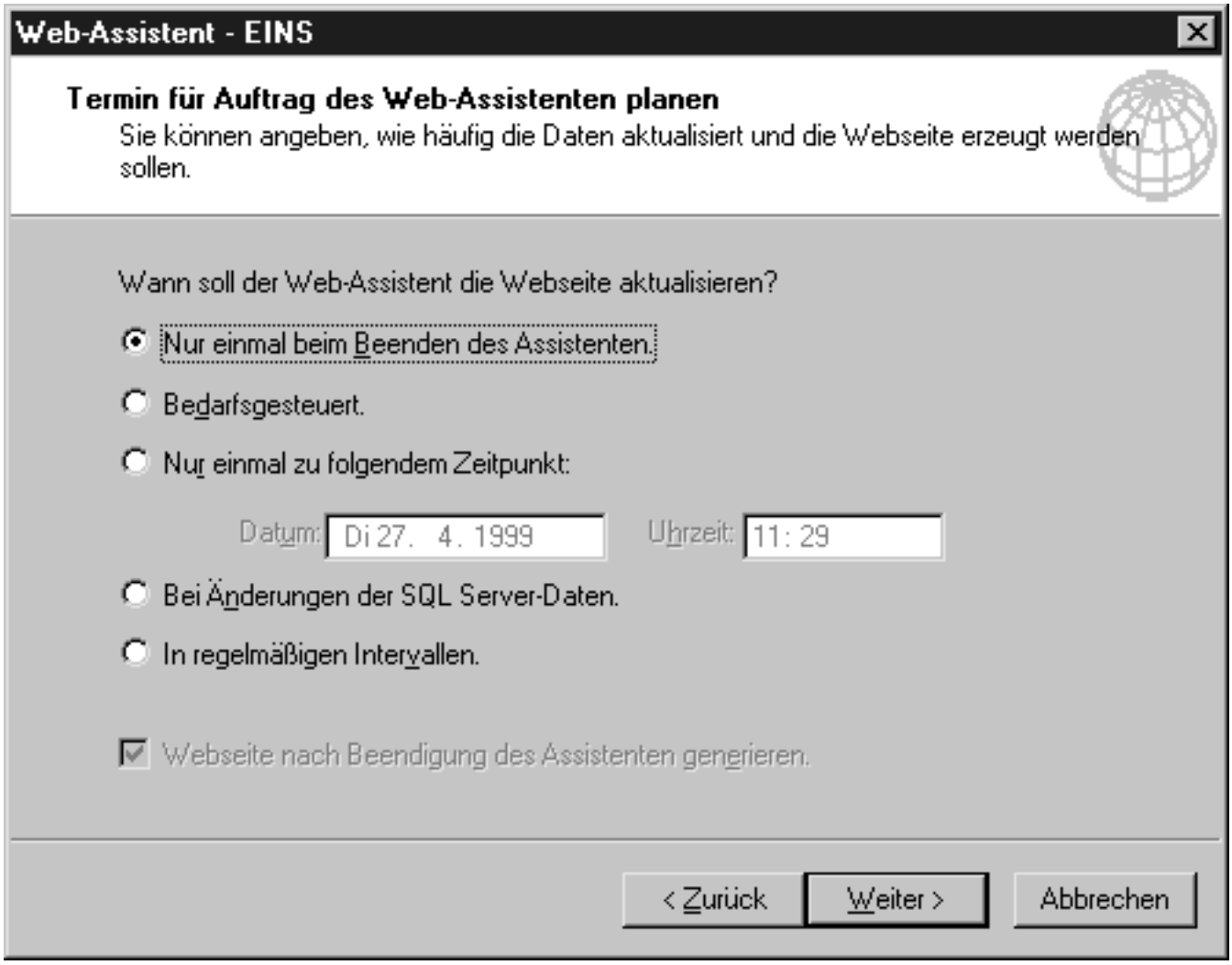


abbildung 28.8: das dialogfeld termin für auftrag planen

es stehen folgende optionen zur auswahl:

nur einmal beim beenden des assistenten: sql server erzeugt den webauftrag, führt ihn sofort aus und löscht ihn anschließend.

bedarfsgesteuert: die webseite wird nur auf anforderung hin erstellt. die generierte systemprozedur (sp_makewebtask) enthält keine klauseln für einen terminplan. wenn sie keine anfängliche webseite erstellen, müssen sie später mit der systemprozedur sp_runwebtask die webseite erzeugen und aktualisieren. in jedem fall ist bei der option *bedarfsgesteuert* der webauftrag mit der systemprozedur sp_runwebtask zu starten und mit der systemprozedur sp_dropwebtask explizit zu löschen.

nur einmal zu folgendem zeitpunkt: der web-assistent erstellt die prozedur sp_makewebtask sofort und führt den webauftrag zum angegebenen zeitpunkt aus. der auftrag wird nach der ausführung automatisch gelöscht.

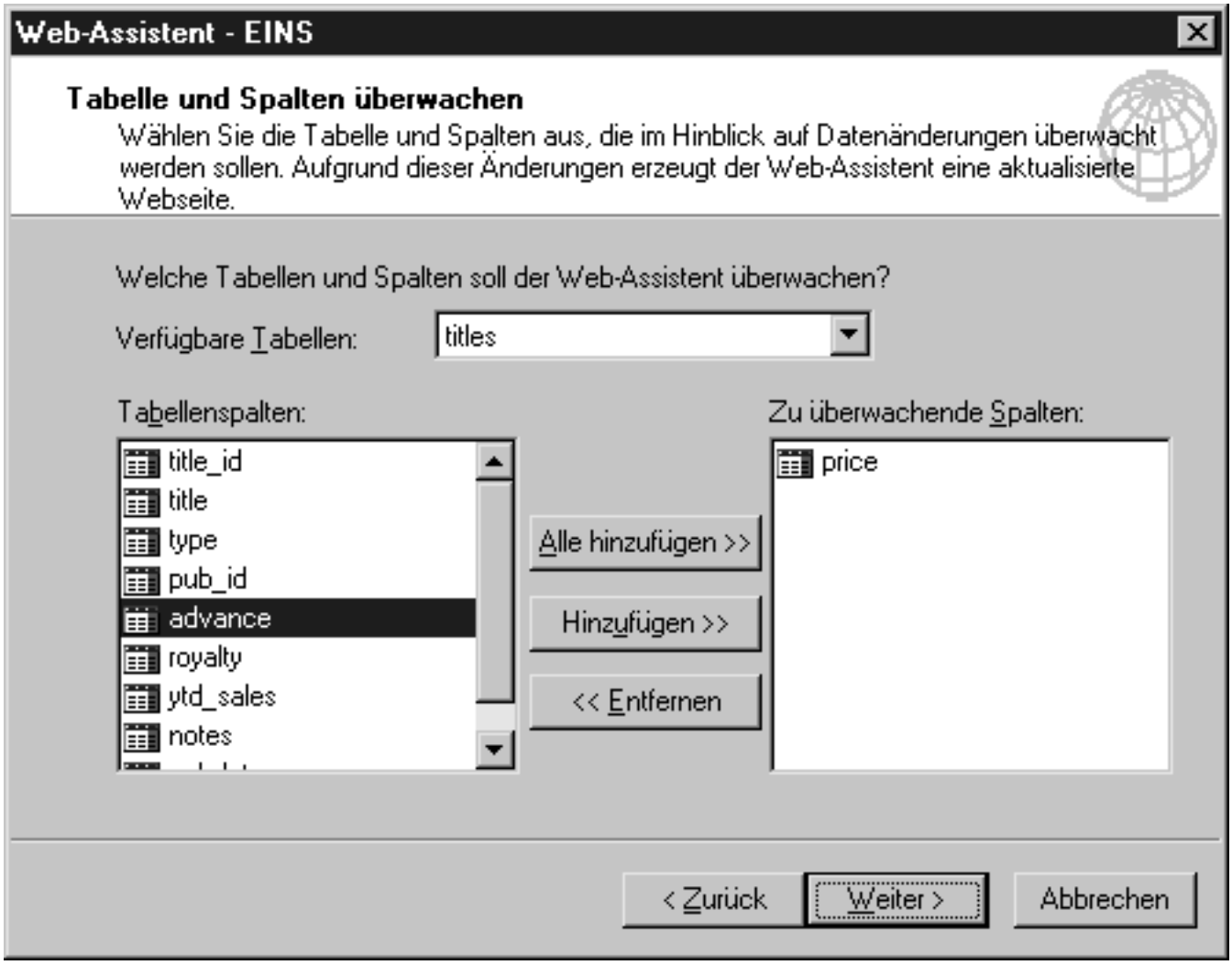


abbildung 28.9: das dialogfeld tabelle und spalten überwachen

bei änderungen der sql server-daten: der web-assistent erstellt die prozedur sp_makewebtast sofort. der auftrag wird einmal sofort und später bei jeder änderung von tabellendaten ausgeführt. datenänderungen lösen einen trigger für die anweisungen insert, update oder delete aus. der trigger führt seinerseits die prozedur sp_runwebtask aus, um die webseite zu aktualisieren. wenn sie diese option gewählt haben, gelangen sie nach klicken auf **weiter** zum dialogfeld **tabelle und spalten überwachen** (siehe abbildung 28.9). hier wählen sie die tabelle und die spalten aus, die zu überwachen sind. klicken sie auf **weiter**, und gehen sie zu schritt 8.

in regelmäßigen intervallen: wenn sie diese option wählen, gelangen sie als nächstes zum dialogfeld **aktualisierungsintervall planen** (siehe abbildung 28.10). der assistent erstellt die prozedur sp_makewebtask sofort und führt den webauftrag regelmäßig zu den angegebenen zeitpunkten aus.

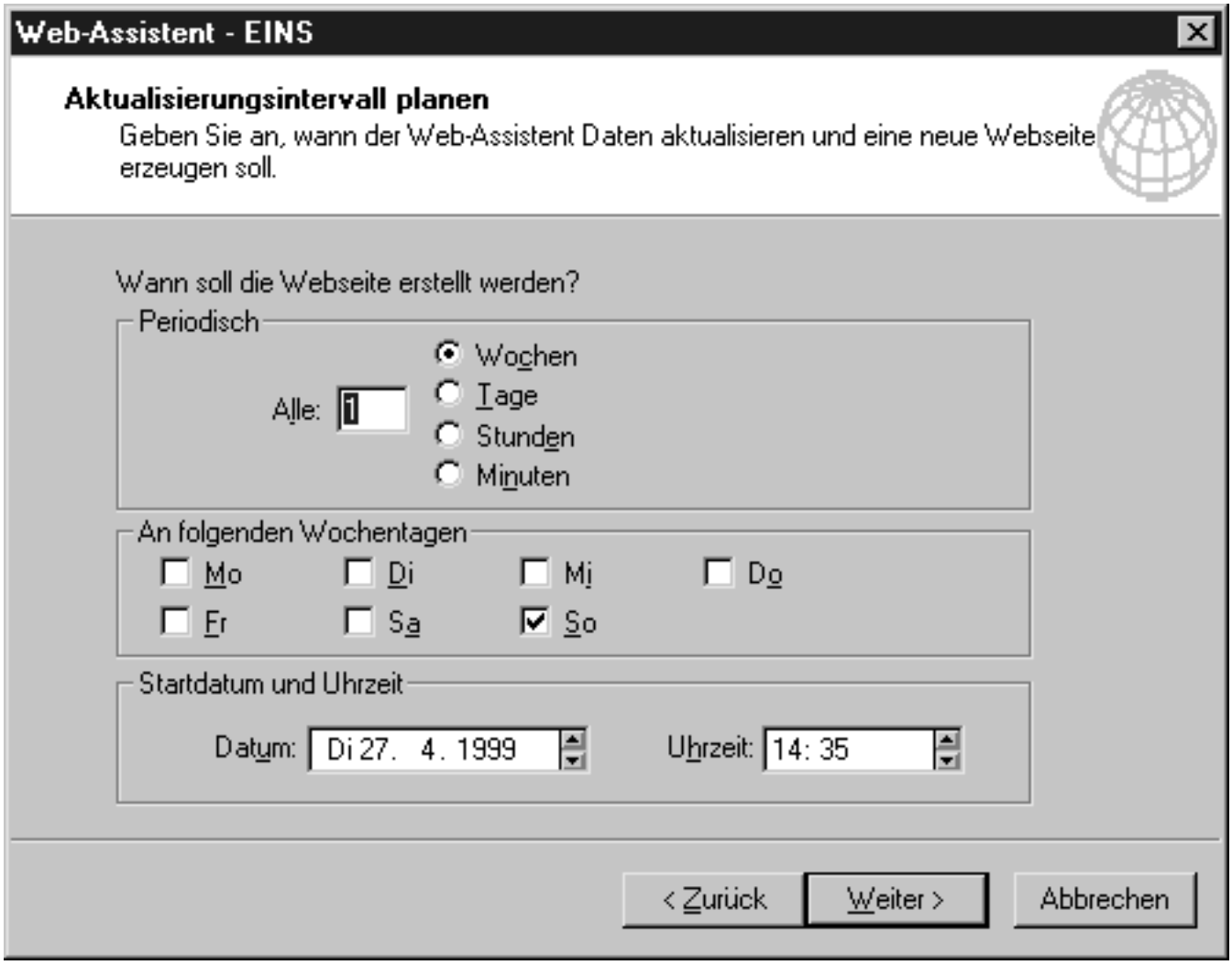


abbildung 28.10: das dialogfeld aktualisierungsintervall planen

9. in den bisherigen schritten ging es um die quelldaten, die sie auf einer webseite veröffentlichen wollen. jetzt kommen sie in die gefilde, die das internet - also das ziel - betreffen. im dialogfeld **webseite veröffentlichen** (siehe abbildung 28.11) legen sie den standort fest, wo der assistent die webseite ablegen soll. dabei kann es sich um ein festplattenverzeichnis, ein netzwerkverzeichnis oder einen http/ftp-pfad handeln, vorausgesetzt, sql server kann auf diesen standort zugreifen.

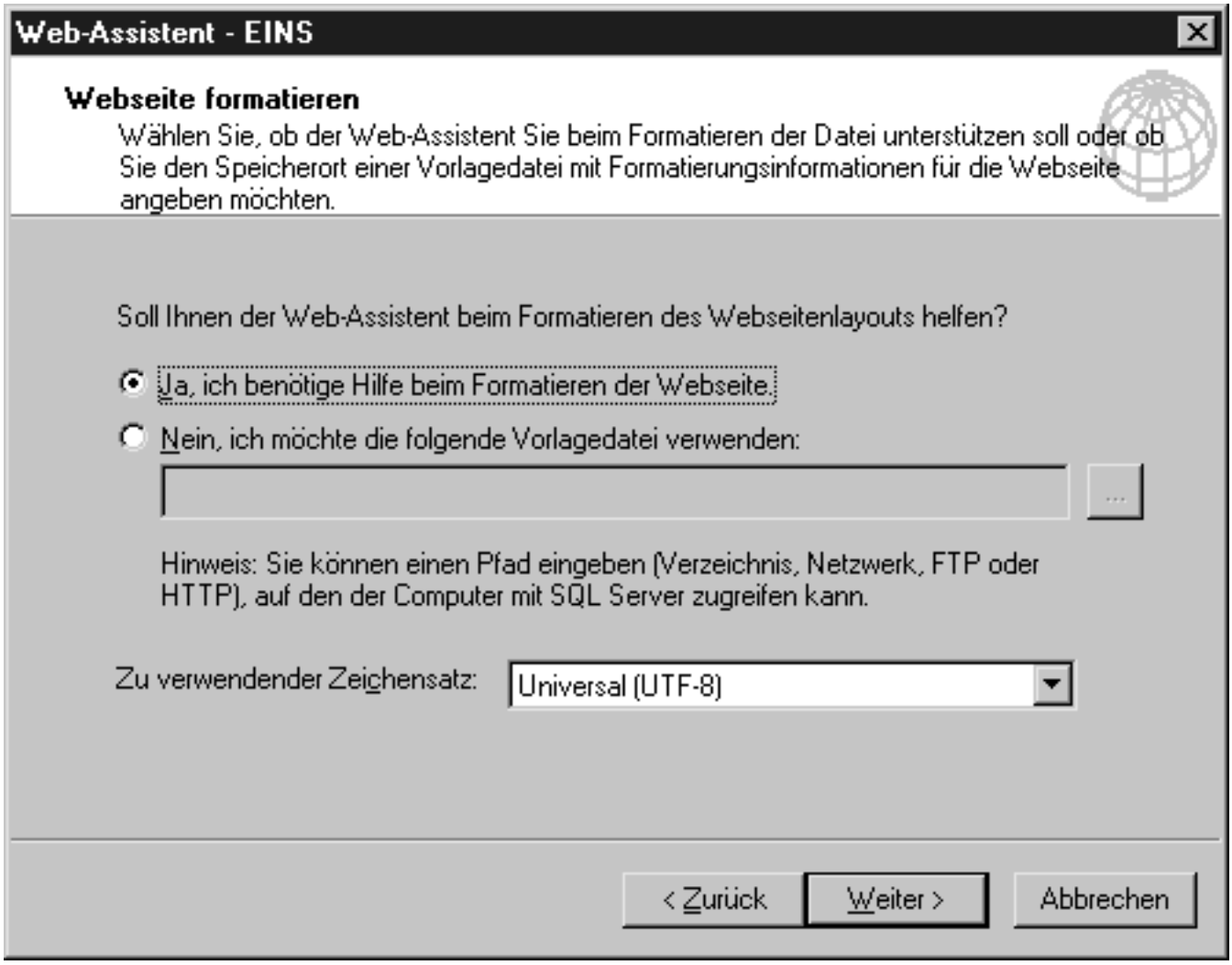


abbildung 28.11: das dialogfeld webseite formatieren

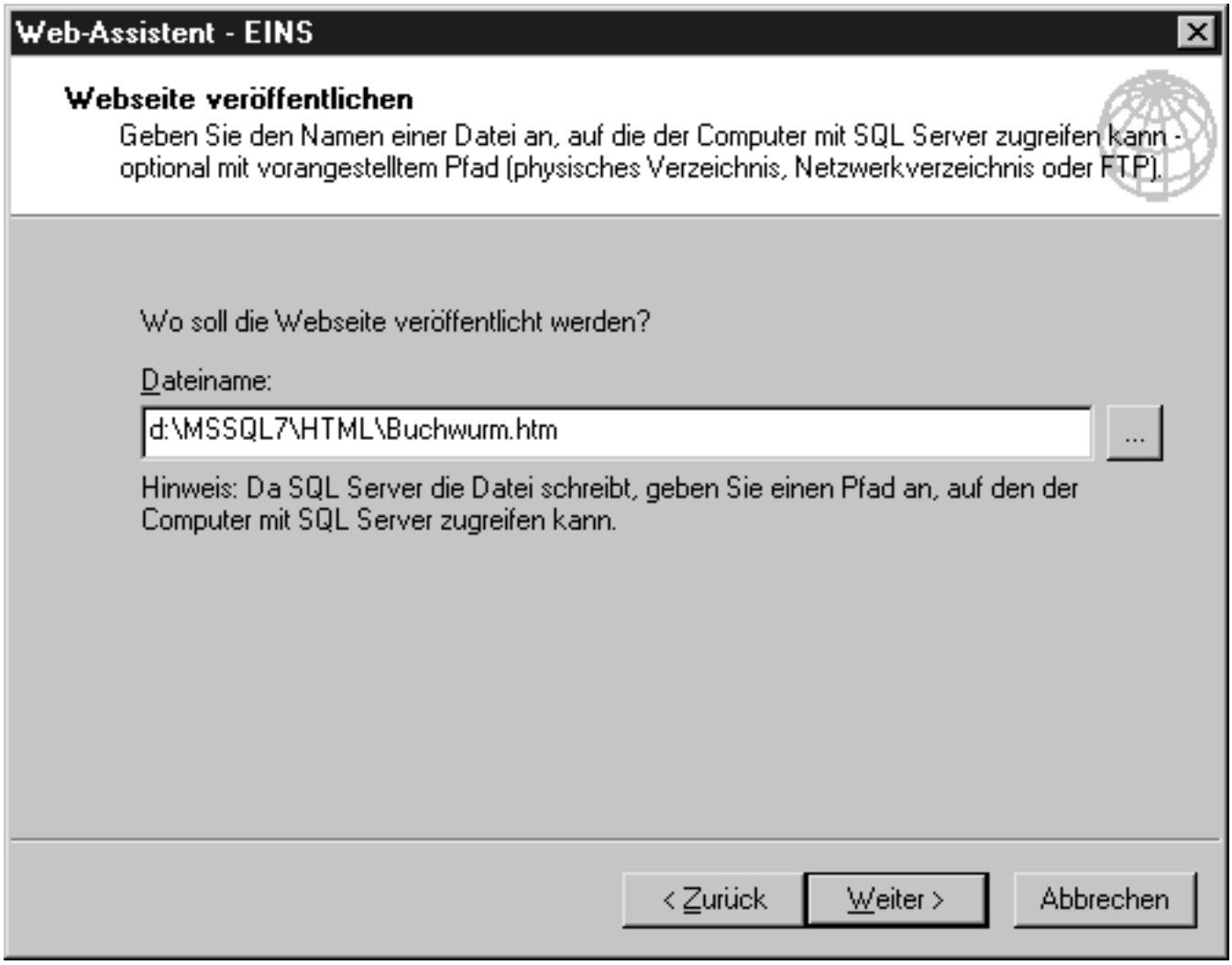


abbildung 28.12: im dialogfeld webseite veröffentlichen geben sie den pfad zur html-seite an

10. zu einer webseite gehört natürlich mehr, als nur eine tabelle ins internet zu stellen. mit dem dialogfeld nach abbildung 28.12 beginnt die formatierung der webseite. hier wählen sie, ob sie der assistent durch die formatierung führen soll oder ob sie eine vorlagedatei verwenden wollen. wenn sie eine vorlagedatei wählen und auf **weiter** klicken, gelangen sie direkt zum dialogfeld **zeilenanzahl beschränken** in schritt 13. andernfalls geht es mit schritt 10 weiter.
11. im dialogfeld **titel festlegen** (siehe abbildung 28.13) geben sie die gewünschte überschrift der webseite und den titel der datentabelle ein. außerdem können sie hier den schriftgrad für die html-tabelle spezifizieren.

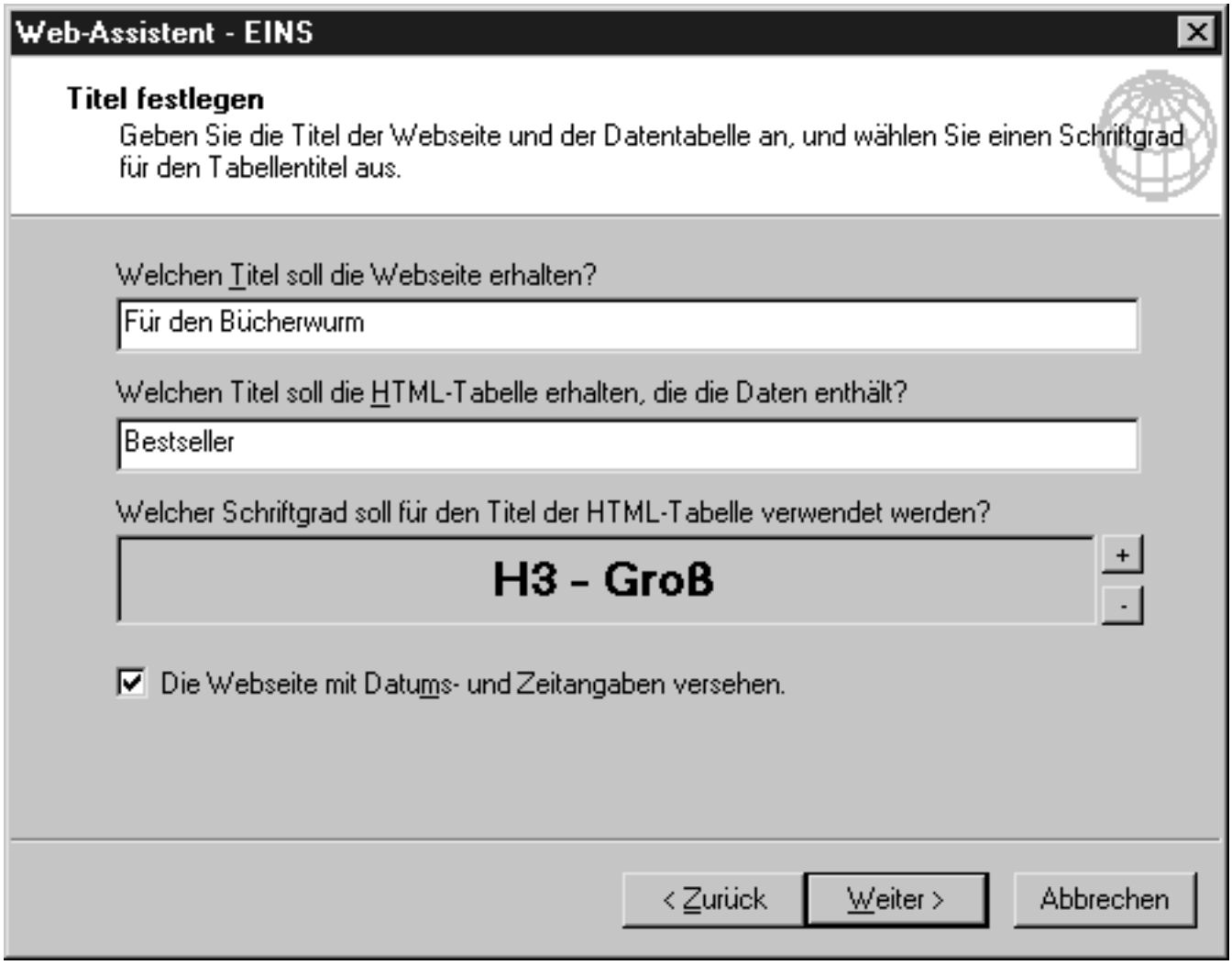


abbildung 28.13: das dialogfeld titel festlegen

12. im dialogfeld **tabelle formatieren** (siehe abbildung 28.14) legen sie fest, ob die spaltennamen auf der webseite erscheinen sollen, mit welchen schriftattributen die tabellendaten zu versehen sind und ob die tabelle mit rahmenlinien auszustatten ist.

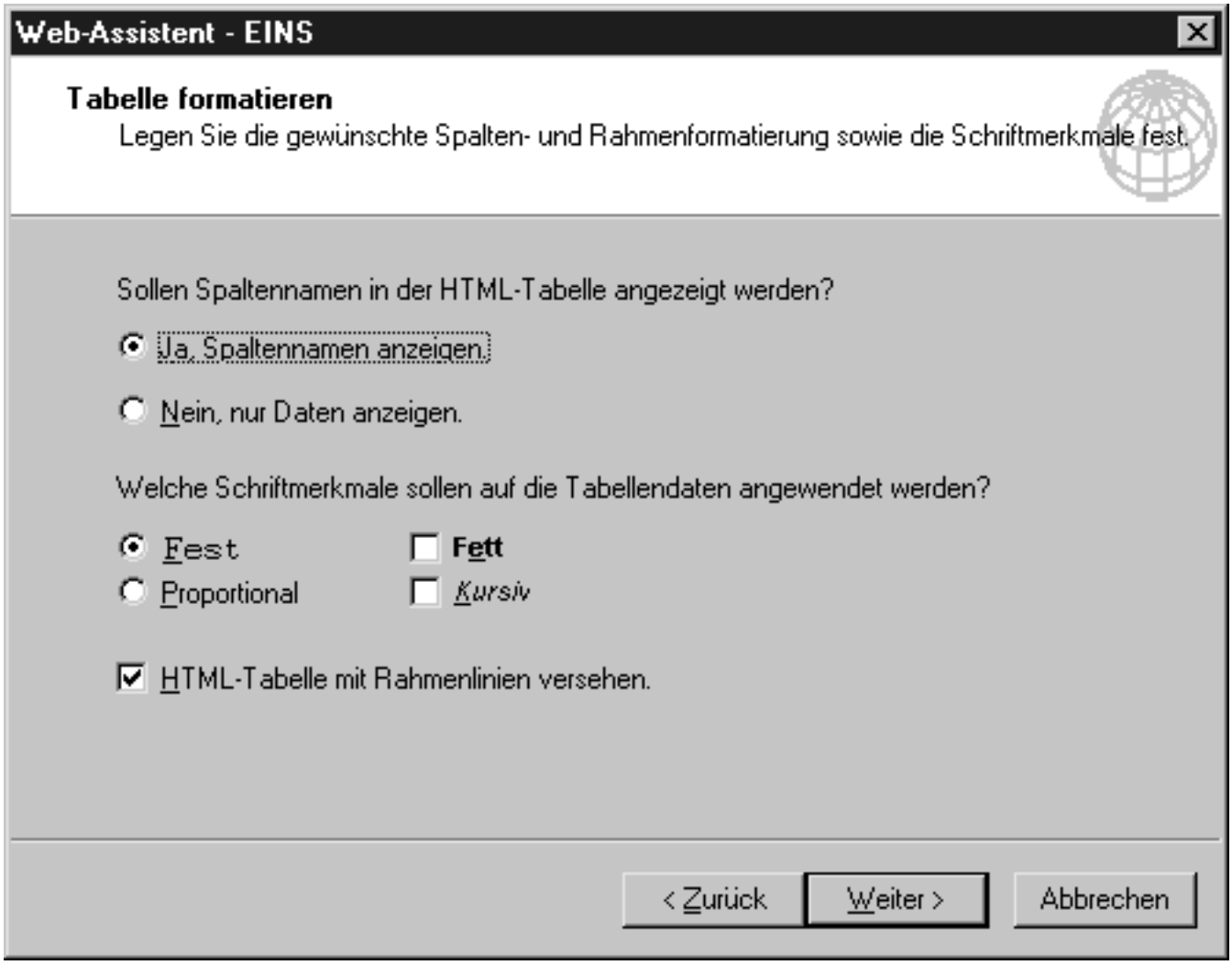


abbildung 28.14: das dialogfeld tabelle formatieren

13. im dialogfeld **hyperlinks zur webseite hinzufügen** (siehe abbildung 28.15) können sie auf ihrer webseite verweise zu anderen webseiten einbinden. wenn sie nur einen einzigen hyperlink angeben wollen, wählen sie die zweite option *ja, einen hyperlink hinzufügen* und geben die webadresse sowie eine bezeichnung in die dafür vorgesehenen bearbeitungsfelder ein. bei wahl der dritten option wird das darunter befindliche textfeld aktiviert. hier können sie eine transact-sql-abfrageanweisung für eine tabelle eingeben, in der sie die hyperlinks und die zugeordneten bezeichnungen gespeichert haben.

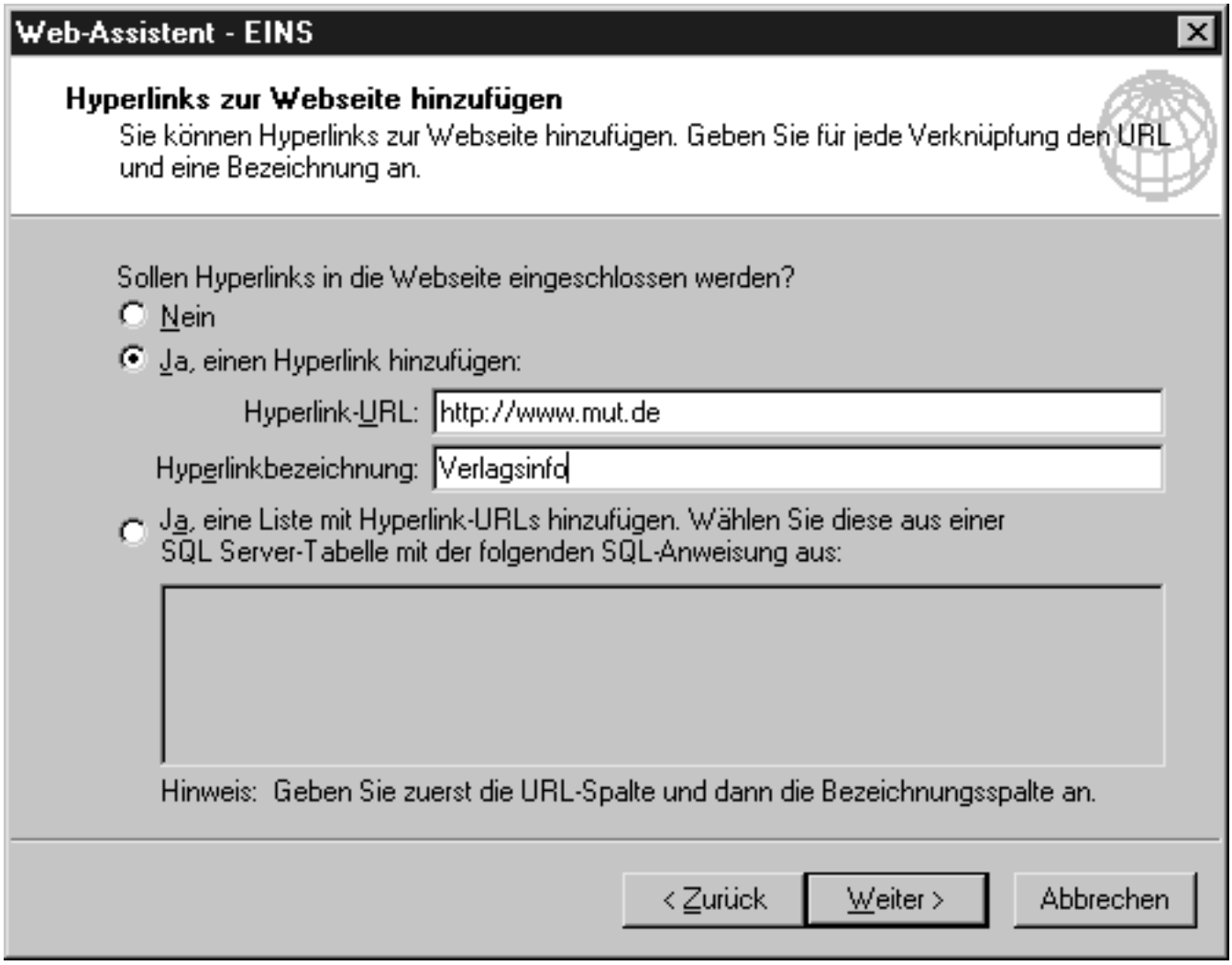


abbildung 28.15: das dialogfeld hyperlinks zu webseite hinzufügen

14. das dialogfeld **zeilenanzahl beschränken** (siehe abbildung 28.16) bietet zwei grundsätzliche optionen: beschränken der von sql server gelieferten ergebnismenge und beschränken der auf der webseite zu präsentierenden informationen. die beispieltabelle titles liefert lediglich 16 zeilen (die beiden null-zeilen werden laut bedingung in schritt 3 ausgeblendet). bei wesentlich umfangreicheren tabellen empfiehlt es sich, die von sql server zurückgegebenen informationen zu beschränken, damit eine aktualisierung nicht zu lange dauert. wenn die zeilenanzahl dann immer noch so groß ist, daß der benutzer einen bildlauf auf der webseite durchführen müßte, um alle zeilen einzusehen, können sie die ergebnisse auch auf mehrere webseiten aufteilen. die einzelnen seiten sind dann durch hyperlinks miteinander verknüpft und erhalten fortlaufende nummern.

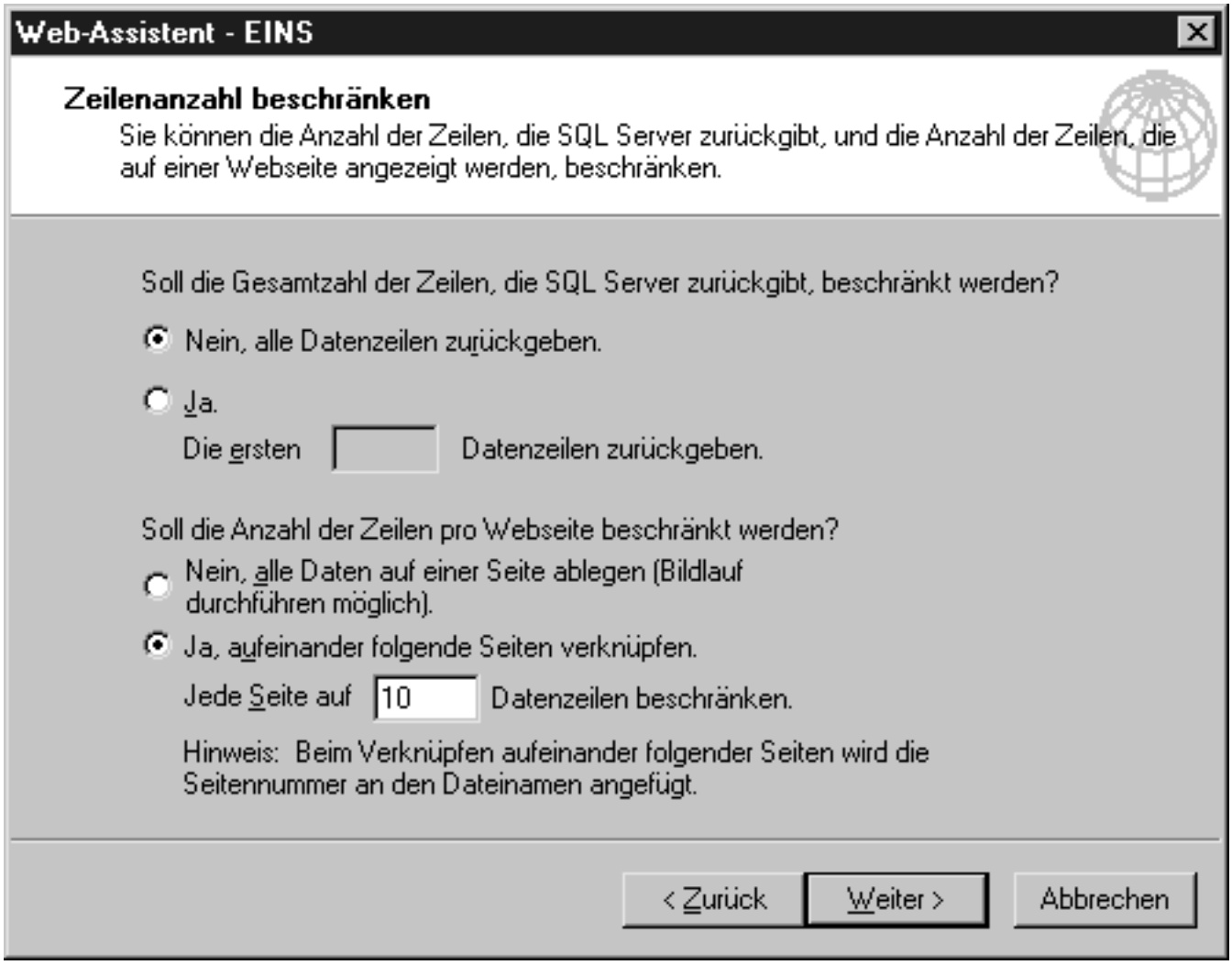


abbildung 28.16: das dialogfeld zeilenanzahl beschränken

15. nach diesem ausgedehnten trip durch das erstellen einer webseite kommen sie nun zum letzten dialogfeld des assistenten, das ihre angaben zusammenfaßt (siehe abbildung 28.17). über die schaltfläche **transact-sql in datei schreiben** können sie das erzeugte skript separat speichern. der abschnitt zu sp_makewebtask weiter unten in diesem kapitel erläutert anhand dieses skripts die parameter der gespeicherten prozedur.

bild

abbildung 28.17: das dialogfeld web-assistenten beenden

16. nachdem sie auf **fertigstellen** geklickt haben, erstellt der web-assistent die html-seite und bringt abschließend die meldung: *der web-assistent hat den task erfolgreich abgeschlossen.*

Bild

abbildung 28.18: die vom web-assistenten generierte webseite

wenn sie den generierten auftrag starten wollen, erweitern sie im enterprise manager den ordner **verwaltung** und klicken auf **webpublishing**. im detailbereich ist der auftrag *webseite titel* aufgeführt. das setzt natürlich voraus, daß sie im schritt 7 nicht die option *nur einmal beim beenden des assistenten* gewählt haben. klicken sie mit der rechten maustaste auf den auftrag, und wählen sie aus dem

kontextmenü den befehl **auftrag des web-assistenten starten**. sql server führt daraufhin den auftrag aus und zeigt die meldung *der webtask 'webseite titel' wurde gestartet* an.

der web-assistent legt die generierte html-seite entsprechend der einstellung von schritt 8 unter dem namen buchwurm.htm im ordner \mssql7\html ab. gehen sie über den **arbeitsplatz** in dieses verzeichnis, und doppelklicken sie auf buchwurm.htm. abbildung 28.18 zeigt die vom web-assistenten generierte webseite.

28.3.2 sp_makewebtask

die gespeicherte prozedur sp_makewebtask erzeugt einen webauftrag, der seinerseits eine html-seite erstellt oder aktualisiert. die syntax der gespeicherten prozedur sp_makewebtask sieht folgendermaßen aus:

syntax:

```
sp_makewebtask [@outputfile =] 'ausgabedatei',
  [@query =] 'abfrage'
  [,[@fixedfont =] festbreitenschrift]
  [,[@bold =] fett]
  [,[@italic =] kursiv]
  [,[@colheaders =] spaltennamen]
  [,[@lastupdated =] zuletztaktualisiert]
  [,[@htmlheader =] formatcode]
  [,[@username =] benutzername]
  [,[@dbname =] datenbankname]
  [,[@templatefile =] vorlagedatei]
  [,[@webpagetitle =] titelderwebseite]
  [,[@resultstitle =] ergebnistitel]
  [
    [,[@url =] 'hyperlink',
      [@reftext =] 'bezeichnung'] |
    [,[@table_urls =] hyperlinkausabfrage,
      [@url_query =] 'urlabfrage']
  ]
  [,[@whentype = terminplan]]
  [,[@targetdate =] datum]
  [,[@targettime =] uhrzeit]
  [,[@dayflags =] wochentag]
  [,[@numunits =] anzahleinheiten]
  [,[@unittype =] einheit]
  [,[@procname =] prozedurname]
  [,[@maketask =] taskfürprozedur]
  [,[@rowcnt =] ergebniszeilen]
  [,[@tabborder =] rahmen]
```

```
[,[@singlerow =] einzelzeile]
[,[@blobfmt =] blobformat]
[,[@rowsperpage =] n]
[,[@datachg =] tabellespaltenliste]
[,[@charset =] zeichensatz]
[,[@codepage =] codeseite]
```

argumente:

wenn sie die schritte des web-assistenten durchgearbeitet haben, werden sie zweifellos erkennen, welches argument in welchem schritt des assistenten »mit leben erfüllt« wird. zur besseren übersicht zeigt tabelle 28.1 die argumente der prozedur sp_makewebtask mit einer kurzen erläuterung.

argument	beschreibung
[@outputfile =] 'ausgabedatei'	speicherort der generierten html-datei. kein standardwert
[@query =] 'abfrage'	transact-sql-anweisungen, die die eigentlichen daten liefern. wenn sie mehrere select-abfragen angeben, erscheinen mehrere tabellen in der ausgabedatei. kein standardwert
[,[@fixedfont =] festbreitenschrift]	schriftattribut für abfrageergebnisse: 0 - proportionalschrift 1 - festbreitenschrift standardwert: 0
[,[@bold =] fett]	schriftattribut für abfrageergebnisse: 0 - normal 1 - fettschrift standardwert: 0
[,[@italic =] kursiv]	schriftattribut für abfrageergebnisse: 0 - nicht kursiv 1 - kursiv standardwert: 0
[,[@colheaders =] spaltennamen]	0 - keine spaltennamen in der html-tabelle 1 - spaltennamen in der html-tabelle standardwert: 1
[,[@lastupdated =] zuletztaktualisiert]	gibt an, ob auf der html-seite das datum und die uhrzeit der letzten aktualisierung vor der tabelle mit den abfrageergebnissen erscheinen soll (1) oder nicht (0). standardwert: 1

[,[@htmlheader =] formatcode]	schriftgrad für den titel der html-tabelle als html-tag: 1 - h1 2 - h2 3 - h3 4 - h4 5 - h5 6 - h6 standardwert: 3
[,[@username =] benutzername]	benutzername zum ausführen der abfrage. standardwert: aktueller benutzer
[,[@dbname =] datenbankname]	name der datenbank, auf die sich die abfrage bezieht. standardwert: aktuelle datenbank
[,[@templatefile =] vorlagedatei]	pfad zur vorlagedatei für das zu erstellende html-dokument
[,[@webpagetitle =] titelderwebseite]	titel des html-dokuments. wenn das dokument keinen titel erhalten soll, geben sie zwei leerzeichen ein. standardwert: microsoft sql server-web-assistent
[,[@resultstitle =] ergebnistitel]	titel, der über der ergebnistabelle erscheint. standardwert: abfrageergebnis
[,[@url =] 'hyperlink'	hyperlink zu einem anderen html-dokument, erscheint nach den abfrageergebnissen und am ende des html-dokuments.
[@reftext =] 'bezeichnung']	erforderliche beschreibung des hyperlinks, wenn @url angegeben ist.
[,[@table_urls =] hyperlinkausabfrage	1 - dokument enthält hyperlinks, die über eine select-abfrage von sql server bereitgestellt werden. @url und @reftext sind dann nicht zulässig. standardwert: 0
[@url_query =] 'urlabfrage']	select-anweisung, die den url und die zugehörige bezeichnung aus einer sql-server-tabelle abruft
[,[@whentype = terminplan]]	dieser parameter legt den zeitpunkt fest, zu dem der webauftrag auszuführen ist. die möglichen werte zeigt tabelle 28.2. standardwert: 1
[,[@targetdate =] datum]	das datum, an dem die webseite erzeugt werden soll. dieser parameter ist bei @whentype = 2, 3, 4 und 6 erforderlich. datumsformat: yyyymmdd standardwert: aktuelles datum

[,[@targettime =] uhrzeit]	zeitpunkt, zu dem die webseite erzeugt werden soll. zeitformat: hhmmss standardwert: 12:00 mittags
[,[@dayflags =] wochentag]	wochentag, an dem die webseite aktualisiert werden soll. dieser parameter ist bei @whentype = 3 und 7 erforderlich. die wochentage sind als bitmuster kodiert, so daß sich durch addition der einzelnen werte mehrere wochentage spezifizieren lassen: 1 - sonntag 2 - montag 4 - dienstag 8 - mittwoch 16 - donnerstag 32 - freitag 64 - samstag standardwert: 1
[,[@numunits =] anzahleinheiten]	gibt die anzahl der mit @unittype spezifizierten zeiteinheiten für eine aktualisierung bei @whentype = 4 und 8 an.
[,[@unittype =] einheit]	spezifiziert die zeiteinheit für eine aktualisierung bei @whentype = 4 und 8: 1 - wochen 2 - tage 3 - stunden 4 - minuten standardwert: 1
[,[@procname =] prozedurname]	prozedur- oder taskname für das html-dokument. wird für gespeicherte prozeduren, aufträge des sql server-agenten und aufträge des web-assistenten verwendet.
[,[@maketask =] taskfürprozedur]	0 - erstellt eine unverschlüsselte gespeicherte prozedur, aber keinen task 1 - erstellt eine verschlüsselte gespeicherte prozedur und den task zur ausführung dieser prozedur 2 - erstellt eine unverschlüsselte gespeicherte prozedur und den zugehörigen task standardwert: 2
[,[@rowcnt =] ergebniszeilen]	legt die maximalanzahl der von sql server zurückgegebenen zeilen für die ergebnismenge fest. standardwert: 0 (d.h. alle zeilen)

[,[@tabborder =] rahmen]	0 - zeichnet keinen rahmen 1 - zeichnet einen rahmen um die ergebnistabelle standardwert: 1
[,[@singlerow =] einzelzeile]	0 - alle zeilen der ergebnismenge werden auf derselben seite und in derselben tabelle angezeigt 1 - für jede einzelne zeile der ergebnismenge wird eine neue html-seite (mit fortlaufender numerierung) erstellt standardwert: 0
[,[@blobfmt =] blobformat]	null - spalten des typs *ntext* oder *image* werden in dieselbe ergebnisseite eingebunden. andernfalls werden diese spalten auf einer anderen seite gespeichert und über einen url mit dem hauptdokument verknüpft: "%n%file=ausgabedatei tplt=vorlagedatei url=hyperlink..."
[,[@rowsperpage =] n]	die ergebnismenge wird auf mehrere seiten mit jeweils n zeilen verteilt. standardwert: 0 (alle zeilen auf derselben seite)
[,[@datachg =] tabellespaltenliste]	liste mit tabellennamen und optionalen spaltennamen. löst das erstellen einer neuen seite aus, wenn sich die zugrundeliegenden daten ändern. der parameter ist bei @whentype = 10 erforderlich und hat folgendes format: {table=name[column=name]}[,...]
[,[@charset =] zeichensatz]	ein aliasname für den zeichensatz. standardwert: n'utf-8'
[,[@codepage =] codeseite]	numerischer wert für den zeichensatz. standardwert: 65001 (= utf-8)

tabelle 28.1: argumente der prozedur sp_makewebtask

die werte des parameters @whentype zeigt tabelle 28.2.

zeitpunkt für ausführung des webauftrags	webseite anfänglich erstellen, nicht aktualisieren, webauftrag löschen	webseite nicht sofort erstellen, aber später aktualisieren	webseite sofort erstellen und später aktualisieren
nur einmal beim beenden des assistenten	1		
bedarfsgesteuert		5	9
nur einmal zum angegebenen zeitpunkt		2	6

bei änderungen der sql server-daten			10
an einem bestimmten wochtentag		3	7
alle n minuten, stunden, tage oder wochen		4	8

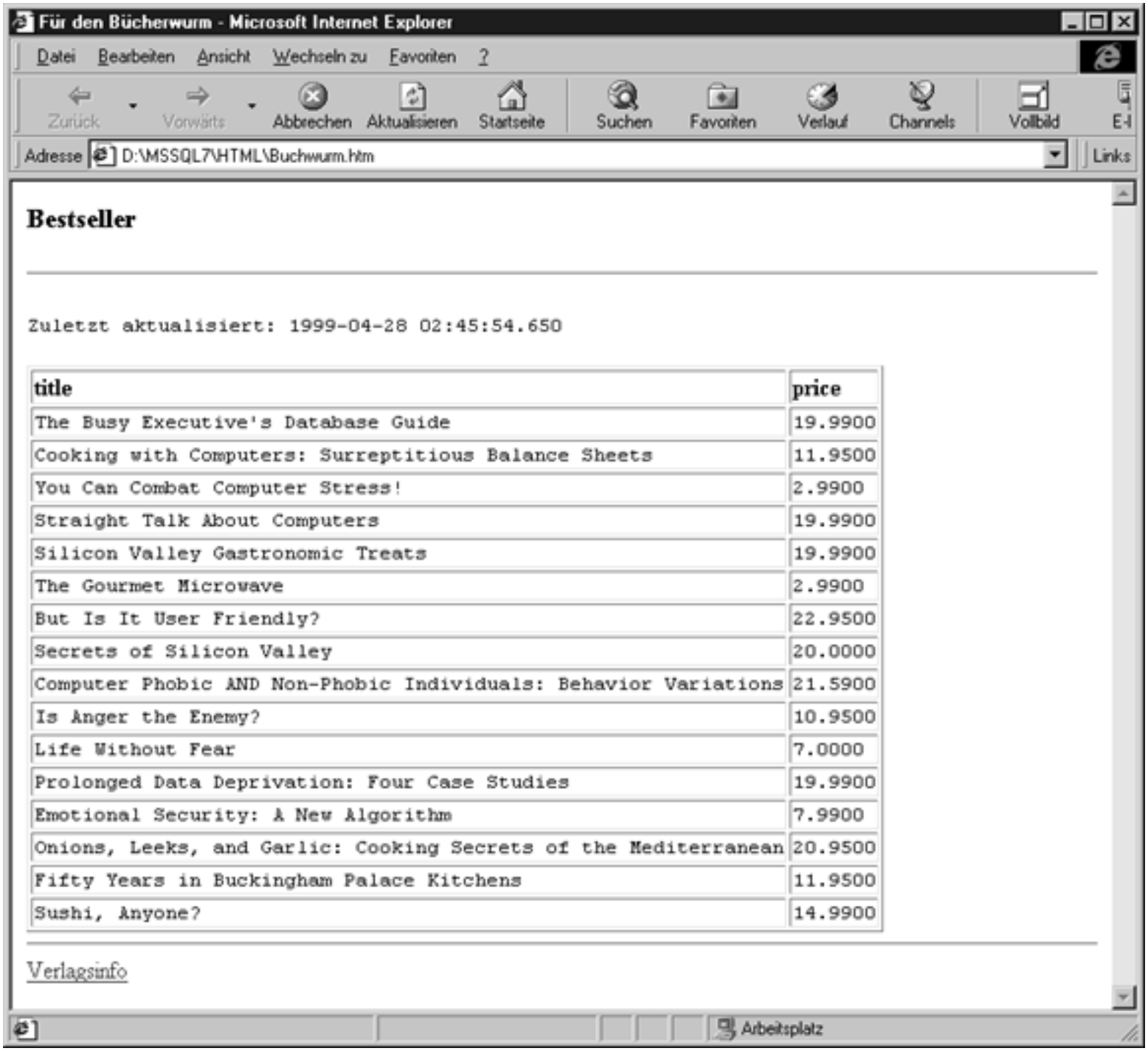
tabelle 28.2: mögliche werte des parameters @whentype

wenn sie im letzten schritt des web-assistenten die schaltfläche **transact-sql in datei schreiben** gewählt haben, können sie dieses skript studieren und verfolgen, welche argumente der assistent verwendet, wie er sie gesetzt hat und wie die gespeicherte prozedur sp_makewebtask aufgerufen wird.

das im beispiel mit dem web-assistenten generierte transact-sql-skript hat folgendes aussehen:

```
execute sp_makewebtask
@outputfile = n'd:\mssql7\html\buchwurm.htm',
@query=n'select title, price
         from titles
         where price is not null',
@fixedfont=1,
@htmlheader=3,
@webpagetitle=n'für den bücherwurm',
@resultstitle=n'bestseller',
@url=n'http://www.mut.de',
@reftext=n'verlagsinfo',
@dbname=n'pubs',
@whentype=10,
@datachg=n'table=titles column=price',
@procname=n'webseite titel',
@codepage=65001,
@charset=n'utf-8'
```


Web-Assistent - EINS
Web-Assistenten beenden
Sie haben die Schritte beendet, die zum Erstellen einer Webseite mittels SQL Server-Daten erforderlich sind. Der Assistent führt die folgenden Tasks aus:
AUSGABEDATEI: d:\MSSQL7\HTML\Buchwurm.htm
ABFRAGE:select title, price from titles where price is not r
DATENBANK: pubs
VORLAGEDATEI: keine, die angegebenen Layoutformate
WANN: jetzt und bei jeder Änderung der SQL Server-Datei
Transact-SQL in Datei schreiben...
< Zurück
Fertig stellen
Abbrechen


Für den Bücherwurm - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Wechseln zu Favoriten ?
Zurück Vorwärts Abbrechen Aktualisieren Startseite Suchen Favoriten Verlauf Channels Vollbild
Adresse D:\MSSQL7\HTML\Buchwurm.htm
Links
Bestseller
Zuletzt aktualisiert: 1999-04-28 02:45:54.650
title price
The Busy Executive's Database Guide 19.9900
Cooking with Computers: Surreptitious Balance Sheets 11.9500
You Can Combat Computer Stress! 2.9900
Straight Talk About Computers 19.9900
Silicon Valley Gastronomic Treats 19.9900
The Gourmet Microwave 2.9900
But Is It User Friendly? 22.9500
Secrets of Silicon Valley 20.0000
Computer Phobic AND Non-Phobic Individuals: Behavior Variations 21.5900
Is Anger the Enemy? 10.9500
Life Without Fear 7.0000
Prolonged Data Deprivation: Four Case Studies 19.9900
Emotional Security: A New Algorithm 7.9900
Onions, Leeks, and Garlic: Cooking Secrets of the Mediterranean 20.9500
Fifty Years in Buckingham Palace Kitchens 11.9500
Sushi, Anyone? 14.9900
Verlagsinfo
Arbeitsplatz

kapitel 29 versionsumstellung

29.1 natura non facit saltus

die natur macht keine sprünge. dieses leibnizsche prinzip mag wohl für die natur gelten, für sql server scheinbar nicht. denn die version 7.0 bedeutet einen großen sprung nach vorn. dennoch bleibt die kompatibilität zum ansi-standard und auch zu älteren versionen von sql server erhalten.

29.2 was ist neu bei sql server 7.0?

zunächst einmal ist festzustellen, daß sich die technologie von sql server 7.0 gegenüber den vorgängerversionen grundsätzlich geändert hat. daraus ergeben sich wesentliche verbesserungen, die unter anderem folgende punkte betreffen:

- architektur
- administration
- sicherheit
- replikation
- datentransformationsdienste
- data warehousing

alle verbesserungen und erweiterungen aufzuzählen und zu erläutern, hätte ein eigenes buch gefüllt. die folgenden abschnitte konzentrieren sich deshalb auf einige der hervorstechendsten merkmale. für spezielle details, die zum beispiel einzelne befehle oder die abwärtskompatibilität betreffen, sei auf die online-dokumentation verwiesen.

architekturverbesserungen

die vorgängerversionen von sql server 7.0 verwendeten sogenannte *datenbankgeräte*. sql server 7.0 speichert jetzt datenbank- und transaktionsprotokolldateien auf betriebssystemebene. damit vereinfacht sich das erstellen und die verwaltung von datenbanken und den zugehörigen transaktionsprotokollen.

damit entfällt gleichzeitig auch die notwendigkeit, die größe von datenbanken und transaktionsprotokollen zu überwachen und gegebenenfalls anzupassen. sql server 7.0 führt die dynamische größenanpassung ein. datenbanken und transaktionsprotokolle können nun ohne mitwirkung des datenbankadministrators automatisch wachsen (oder auch schrumpfen). durch diese möglichkeit gehören fehler aufgrund einer überfüllung des transaktionsprotokolls der vergangenheit an.

in die kategorie der automatischen anpassung fällt auch die dynamische speicherverwaltung durch sql server.

die für die speicherung von daten und indizes verwendeten seiten wurden von 2 kbyte auf 8 kbyte vergrößert. damit erhöht sich auch die kapazität einer tabellenzeile von 1962 byte auf 8060 byte. weitere technische daten finden sie in kapitel 2.

die grundlegende überarbeitung des abfrageprozessors schlägt sich in einer höheren leistung nieder.

zu den wohl wichtigsten architekturverbesserungen gehört die skalierbarkeit von sql server. mit ein und derselben codebasis läßt sich sql server sowohl unter windows nt als auch unter windows 95/98 installieren. damit öffnet sich die welt der »großen datenbanken« auch für benutzer von mobilen computern.

vereinfachte administration

die microsoft management console (mmc) dient als einheitliche benutzeroberfläche und umgebung für microsoft-backoffice-server. die konsole für sql server ist der enterprise manager, der gegenüber der vorgängerversion vollkommen überarbeitet wurde.

zur grafischen benutzeroberfläche gehören auch die zahlreichen assistenten, die ihnen bei fast jeder aufgabe in sql server zur seite stehen. kapitel 3 hat bereits einen überblick über die assistenten gebracht. im verlauf dieses buches konnten sie mit einigen wichtigen vertretern dieser spezies bekanntschaft schließen.

die verbesserungen betreffen auch den umfang der sprache transact-sql. im hinblick auf die administration sei hier als beispiel die anweisung zum ändern von tabellen (alter table) genannt. während sie bisher eine tabellendefinition nur ändern konnten, indem sie die daten gesichert, die tabelle gelöscht, die tabelle neu erstellt und die daten wieder geladen haben, können sie jetzt problemlos spalten hinzufügen oder die datentypen von spalten ändern. die berechtigungen für die tabelle bleiben dabei erhalten.

sicherheit

das sicherheitsmodell von sql server ist jetzt noch enger mit windows nt verbunden. sql server 7.0 führt das konzept der rollen ein, die leistungsfähiger und flexibler als die gruppen der vorgängerversionen sind. kapitel 21 beschäftigt sich ausführlich mit diesem aspekt von sql server.

replikation

die replikation hat sich zwar nicht grundsätzlich geändert, bietet aber einige wesentliche erweiterungen und verbesserungen. hier ist insbesondere die aktualisierungsreplikation zu nennen. die abonnenten können die replizierten daten ändern und beim verleger zusammenführen.

darüber hinaus bietet sql server eine reihe von assistenten, die sie bei den wichtigsten arbeiten im zusammenhang mit der replikation unterstützen.

datentransformationsdienste

die datentransformationsdienste (dts) sind eine echte neuerung in sql server 7.0. damit lassen sich daten unterschiedlichster formate nach sql server importieren und aus sql server exportieren. als datenquellen

eignen sich nicht nur produkte von microsoft, sondern auch datenbankdateien von namhaften anbietern wie dbase, interbase, oracle oder sybase. weiterhin können sie daten aus tabellenkalkulationen oder auch einfachen textdateien übertragen.

die dts stellen damit nicht nur einen ersatz des dienstprogramms bcp zum massenkopieren dar, sondern sind wesentlich flexibler und einfacher einzusetzen. vor allem aber sind die dts für das data warehousing prädestiniert.

data warehousing

zum lieferumfang von sql server gehören die olap services für die analytische online-verarbeitung. es handelt sich zwar um ein selbständiges produkt, das aber eng mit sql server verbunden ist und sich über die gleiche oberfläche installieren läßt. die olap services unterstützen den aufbau von data warehouses. eine wichtige rolle spielen dabei die bereits erwähnten dts, mit denen sich daten aus mehreren heterogenen quellen manuell oder nach terminplan importieren und transformieren lassen.

29.3 versionsumstellung

viele firmen, die sql server 7.0 einsetzen werden, arbeiten bereits mit den vorgängerversionen von sql server und haben zeit und kosten in ihre datenbanken investiert. für diesen kundenkreis ist die umstellung der 6.x-datenbanken auf das format von sql server 7.0 eine der vorrangigen aufgaben. dabei muß gleichzeitig sichergestellt sein, daß die aktuellen betriebsabläufe durch die umstellung weder gestört noch beeinträchtigt werden. falls sich die versionsumstellung zum beispiel nicht in einem vorgegebenen zeitfenster realisieren läßt, muß die rückkehr zur alten version möglich sein.

um diese probleme in den griff zu bekommen, hat microsoft mit der sql-server-versionsumstellung ein instrument geschaffen, das praktisch die beibehaltung der vorhandenen 6.x-datenbanken erlaubt, den übergang zur version 7.0 gefahrlos möglich macht und auch noch eine umschaltung zwischen sql server 6.x und 7.0 bietet.

dieser komfort ist natürlich nicht ganz umsonst zu haben. will man die alten daten beibehalten, muß man über genügend festplattenplatz verfügen. bei sehr großen datenbanken und raid-5-systemen ist das unter umständen nicht gegeben. in solchen fällen besteht immerhin noch die möglichkeit, auf netzwerklaufwerke oder bandsicherungen auszuweichen.

29.3.1 checkliste

bevor sie die versionsumstellung von sql server 6.x auf 7.0 durchführen können, müssen sie folgende vorbereitungen treffen:

- durchführen von konsistenzprüfungen für alle datenbanken der version 6.x
- sicherung aller datenbankdateien und der datenbank master der version 6.x
- datenbank tempdb der version 6.x auf 10 mbyte vergrößern (empfohlen 25 mbyte)
- sql server service pack 3 oder höher installieren
- installation von sql server 7.0
- kennwort für administrator der versionen 6.x und 7.0 bereithalten

- außerdem sollten sie folgende fragen klären:
- ist bei einer umstellung auf demselben computer genügend platz für die alten und neuen versionen der datenbanken verfügbar oder ist eine bandsicherung erforderlich?
- sind der zeichensatz und/oder die sortierreihenfolge umzustellen?

stellen sie sicher, daß keine benutzer angemeldet sind bzw. keine anwendungen auf sql server zugreifen. deaktivieren sie autostart-prozeduren, da diese die umstellung blockieren können.

nachdem sie diese vorbereitungen abgeschlossen haben, können sie den im nachfolgenden abschnitt beschriebenen sql-server-aktualisierungs-assistenten ausführen.

29.3.2 sql-server-aktualisierungs-assistent

verglichen mit den vorherigen versionen von sql server bietet der sql-server-aktualisierungs-assistent der version 7.0 möglichkeiten der versionsumstellung, die in den älteren versionen nicht annähernd so komfortabel waren.

sql server 7.0 unterstützt keine upgrades von systemen der version 4.2x. falls sie von derartigen systemen auf sql server 7.0 umstellen wollen, müssen sie den zwischenschritt über ein upgrade auf die version 6.5 gehen.

den umstieg von der version 6.x auf 7.0 führen sie mit hilfe des sql-server-aktualisierungs-assistenten in folgenden schritten aus:

1. wählen sie **start / programme / microsoft sql server - versionsumstellung / sql server-aktualisierungs-assistent**. es erscheint das in abbildung 29.1 dargestellte startdialogfeld. klicken sie auf **weiter**.

bild

abbildung 29.1: startdialogfeld des sql server-aktualisierungs-assistenten

2. im dialogfeld **daten- und objektübertragung** (siehe abbildung 29.2) wählen sie die optionen für die aktualisierung.

bild

abbildung 29.2: das dialogfeld daten- und objektübertragung

die kontrollkästchen für export und import im oberen teil des dialogfelds haben lediglich hinweischarakter und lassen sich nicht ausschalten. (was wollen sie auch sonst exportieren bzw. importieren?)

als datenübertragungsmethode ist *named pipe* voreingestellt. das ist auch die einzige option, wenn kein windows-nt-bandtreiber installiert ist. selbst wenn sie die wahl haben, sollten sie mit named pipe arbeiten, da es sich hierbei um die zuverlässigste und schnellste methode handelt. allerdings setzt das entsprechend viel festplattenplatz (für die ursprünglichen dateien der version 6.x und die neu anzulegenden dateien der version 7.0) voraus.

die überprüfungsoptionen sind per vorgabe deaktiviert. inkompatible oder fehlerhafte objekte, die sich nicht übertragen lassen, führt der aktualisierungs-assistent im ausgabeprotokoll auf. wenn sie das

kontrollkästchen **erfolgreiche übertragung der objektdateien überprüfen** einschalten, untersucht der assistent die 6.x-datenbanken *vor* und die 7.0-datenbanken *nach* der aktualisierung, und er erstellt eine liste aller objekte und meldet alle abweichungen. bei zusätzlich aktivierter option **ausführliche integritätsprüfung** bildet der assistent für alle tabellenspalten prüfsummen vor und nach der aktualisierung, um eine veränderung der datenbestände ermitteln zu können. sobald sie das kontrollkästchen einschalten, weist sie der assistent darauf hin, daß die überprüfung mit dieser option wesentlich länger dauert (siehe abbildung 29.3).

bild

abbildung 29.3: bestätigung der ausführlichen datenintegritätsprüfung

3. im dialogfeld **anmelden** (siehe abbildung 29.4) tragen sie die kennwörter - nicht die benutzernamen - des quell- und zielservers ein. weiterhin legen sie hier den exportserver der 6.x-dateien fest. per vorgabe ist das der server, auf dem sie den assistenten ausführen. der importserver ist immer der server, auf dem der assistent läuft. als optionale startargumente können sie zum beispiel ablaufverfolgungsflags angeben.

abbildung 29.4: das dialogfeld anmelden für export- und importserver

4. nachdem sie auf **weiter** geklickt haben, erscheint die in abbildung 29.5 gezeigte warnung.

vergewissern sie sich, daß während der versionsumstellung keine benutzer mit ihrem server verbunden sind. klicken sie dann auf **ja**. (die warnung erscheint in jedem fall, auch wenn alle sql-server-dienste beendet sind.)

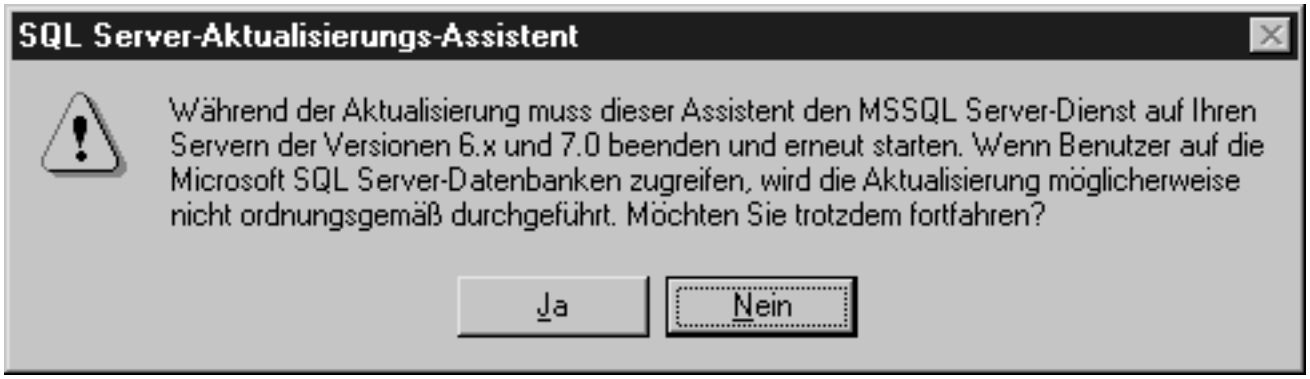


abbildung 29.5: warnung, die sich auf eventuell bestehende verbindungen bezieht

5. der aktualisierungs-assistent wechselt nun mehrfach zwischen den 6.x- und 7.0-servern, sammelt informationen und startet dienste neu. gegebenenfalls erscheinen dabei die in abbildung 29.6 und abbildung 29.7 gezeigten hinweise und aufforderungen.

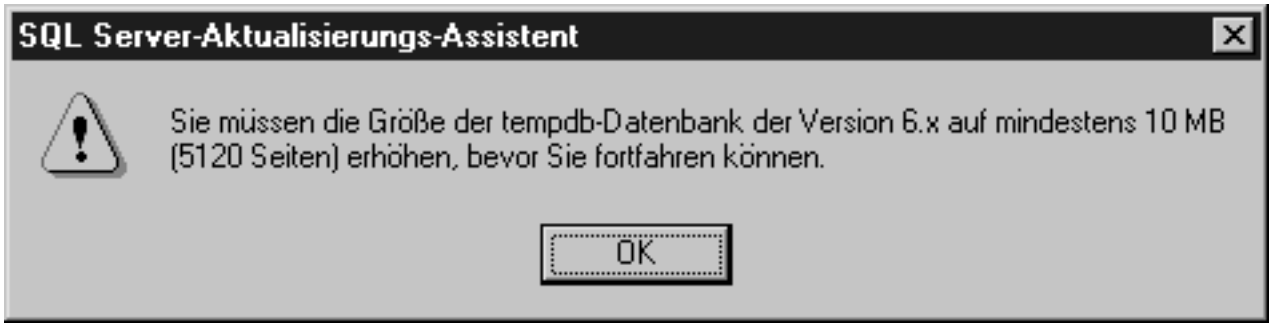


abbildung 29.6: aufforderung, die datenbank tempdb der version 6.x zu vergrößern

abbildung 29.7: hinweis auf laufende programme, die eine ordnungsgemäße ver-sionsumstellung

verhindern können

6. schließen sie die genannten programme. um die datenbank tempdb zu vergrößern, starten sie den enterprise manager der version 6.x über **start / programme / microsoft sql server 6.x / enterprise manager**. hier wählen sie die datenbank tempdb aus und passen die größe entsprechend an.

zu diesem zeitpunkt in der versionsumstellung enthält das menü **start** nicht mehr den eintrag **microsoft sql server 7.0**, sondern die entsprechenden einträge für die version 6.x. um wieder zur version 7.0 zu wechseln (beispielsweise beim abbruch der versionsumstellung), wählen sie **start / programme / microsoft sql server - versionsumstellung / microsoft sql server 7.0**.

7. als nächstes erscheint das dialogfeld **auswahl der codepage** (siehe abbildung 29.8). normalerweise können sie die vorgegebene einstellung übernehmen. klicken sie auf **weiter**.

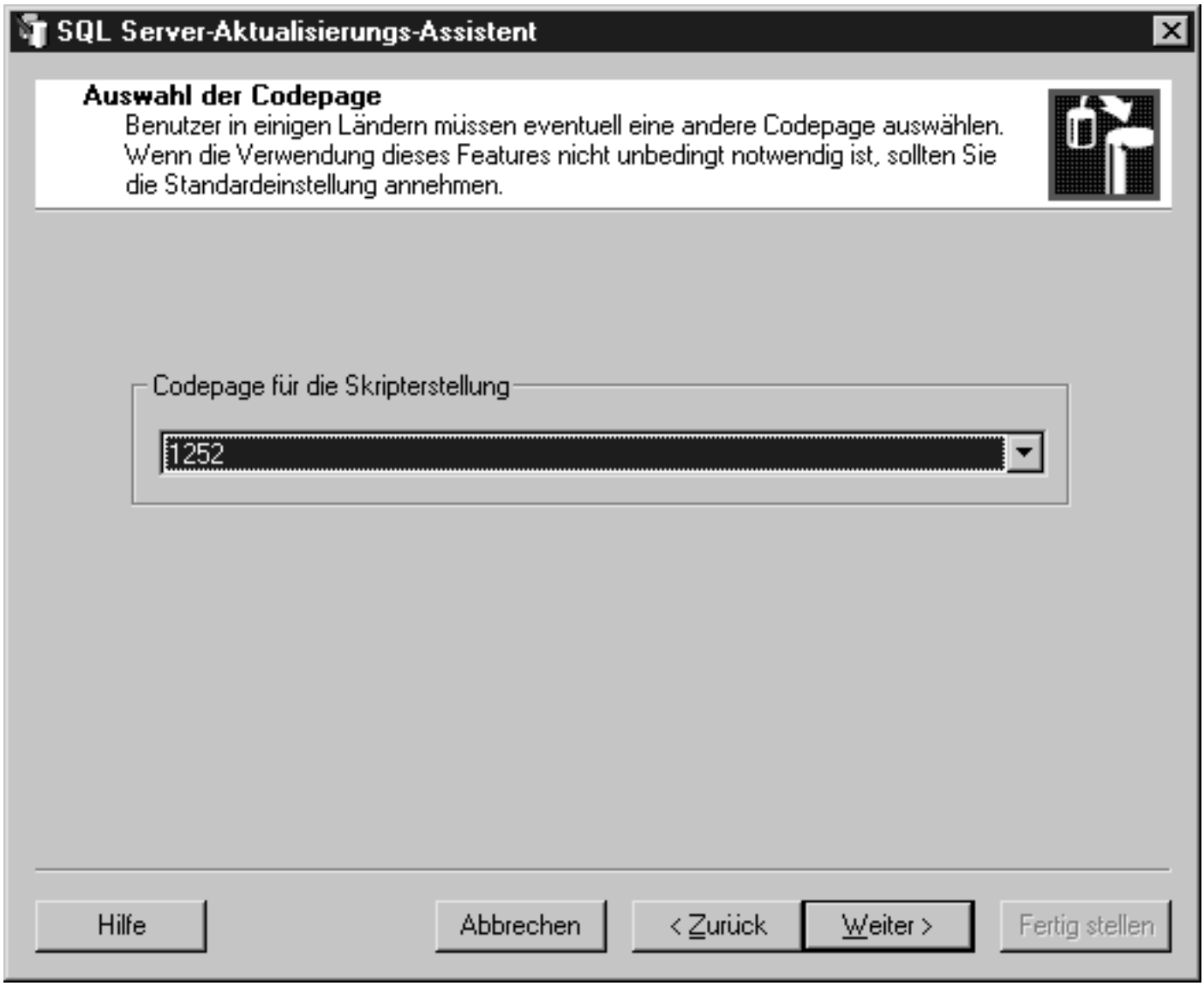


abbildung 29.8: das dialogfeld auswahl der codepage

8. im dialogfeld **datenbanken auf sql server 7.0 aktualisieren** können sie bestimmte datenbanken von der aktualisierung ausschließen bzw. einschließen. es empfiehlt sich, die standardeinstellung zu übernehmen. klicken sie auf **weiter**.
9. im dialogfeld **datenbankerstellung** (siehe abbildung 29.9) legen sie fest, auf welche weise die

sql-server-7.0-datenbanken erstellt werden sollen. wenn sie eine andere option wählen, blendet der assistent ein meldungsfeld mit dem hinweis ein, daß die verwendung der betreffenden option nicht empfohlen wird, und zwar einfach deshalb, weil die anderen verfahren einerseits fehleranfälliger sind und sie andererseits mehr - eigentlich unnötige - arbeit damit haben.

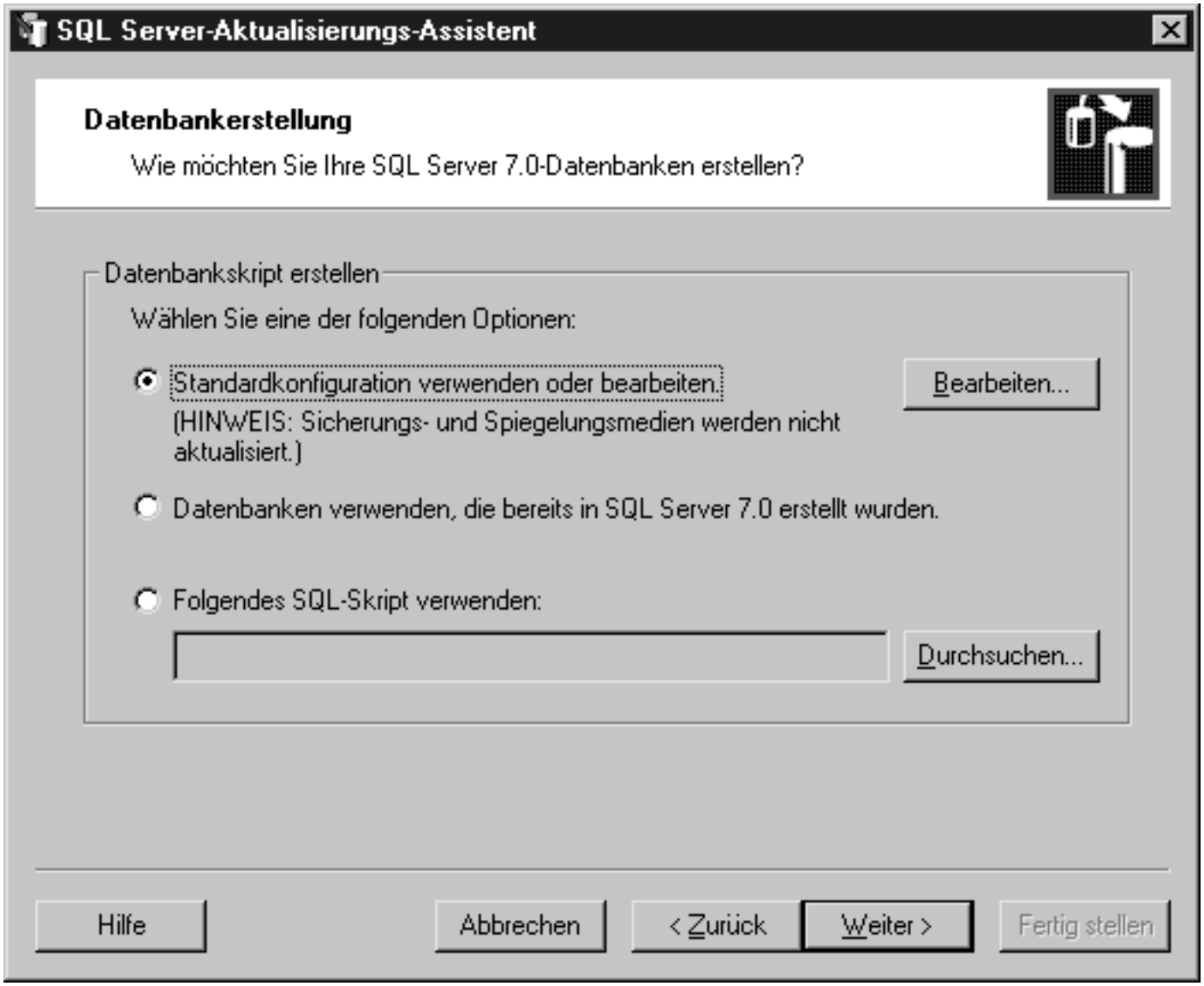


abbildung 29.9: das dialogfeld datenbankerstellung

10. wenn sie die verzeichnisstruktur der datenbanken ändern wollen, klicken sie auf **bearbeiten**. daraufhin erscheint ein dialogfeld zur bearbeitung des datenbanklayouts (siehe abbildung 29.10). hier können sie dateigruppen hinzufügen und entfernen sowie datenträgerdateien hinzufügen, entfernen und deren eigenschaften bearbeiten. wenn sie getroffene änderungen übernehmen möchten, klicken sie auf **annehmen**. klicken sie dann im dialogfeld **datenbankerstellung** auf **weiter**.

bild

abbildung 29.10: in diesem dialogfeld können sie das datenbanklayout ändern

11. im dialogfeld **systemkonfiguration** (siehe abbildung 29.11) legen sie die zu übertragenden systemobjekte fest. bei aktiviertem kontrollkästchen **serverkonfiguration** überträgt der assistent alle benutzernamen und die optionen der serverkonfiguration, die für die version 7.0 zutreffen.

wurde die version 6.x für die replikation konfiguriert , ist das kontrollkästchen **replikationseinstellungen** verfügbar. der assistent übernimmt dann alle artikel, publikationen und abonnements der ausgewählten datenbanken und überträgt auch - falls erforderlich - die verteilungsdatenbank. wenn sie das kontrollkästchen **sql executive-einstellungen** einschalten, werden die von der sql executive der version 6.x geplanten tasks in den sql server-agenten der version 7.0 übernommen und aktualisiert.

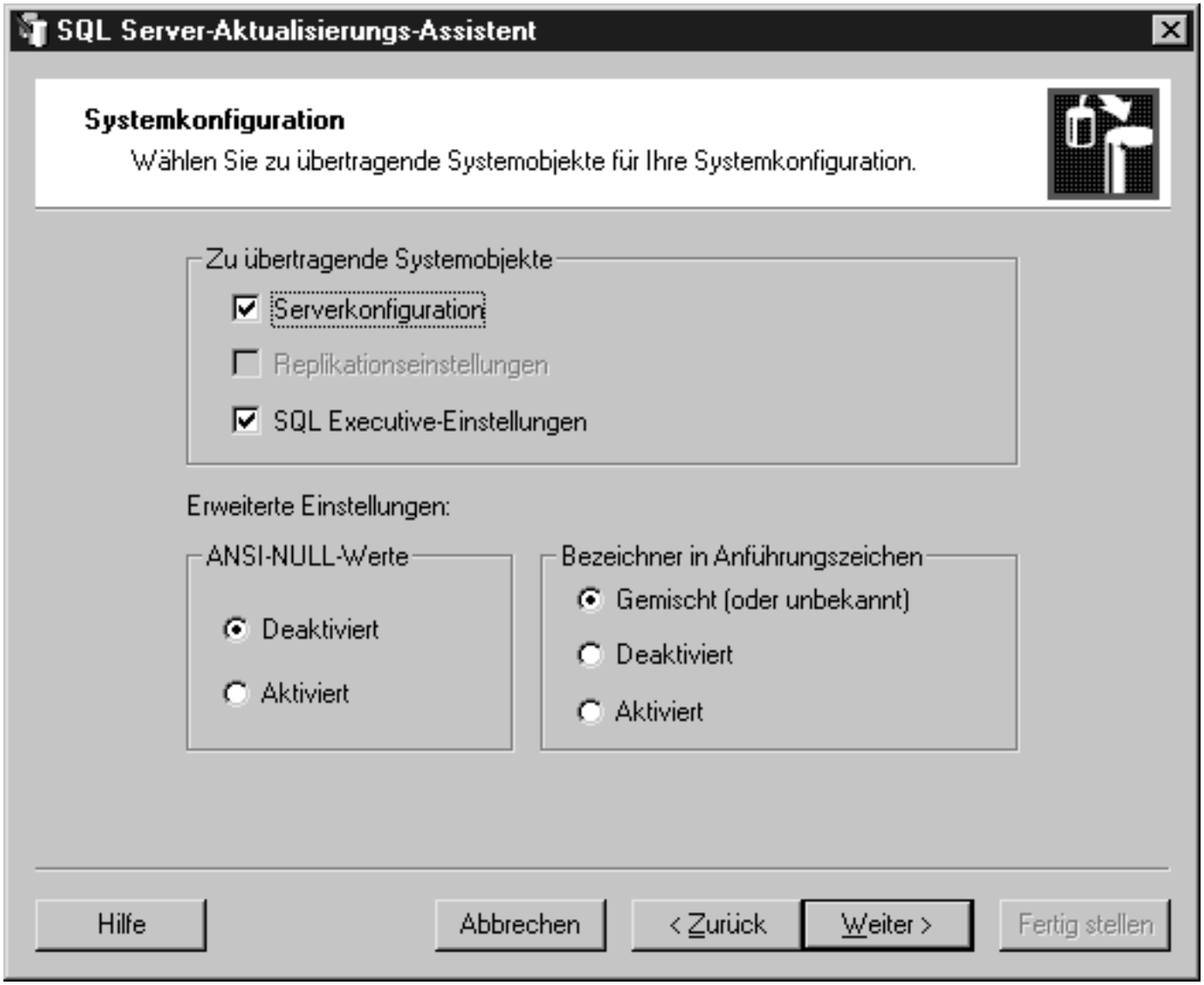


abbildung 29.11: das dialogfeld systemkonfiguration

12. im letzten dialogfeld (siehe abbildung 29.12) können sie über die schaltfläche **warnungen und auswahl werden im editor angezeigt** die datei summary.txt im editor öffnen und kontrollieren (siehe abbildung 29.13).

bild

abbildung 29.12: warnungen und auswahl im editor

bild

abbildung 29.13: letztes dialogfeld des aktualisierungs-assistenten

13. klicken sie im letzten dialogfeld auf **fertigstellen**, um die aktualisierung zu starten. es erscheint das dialogfeld **skriptinterpreter für sql server-aktualisierung** (siehe abbildung 29.14). wenn sie die einzelnen schritte verfolgen wollen, klicken sie auf die schaltfläche **zwischen schritten anhalten**. nach jedem schritt erscheint dann ein dialogfeld wie in abbildung 29.15 gezeigt.

bild

abbildung 29.14: im schrittbetrieb können sie fortschritt oder abbruch wählen

Bild

abbildung 29.15: der fortschritt der aktualisierung erscheint im dialogfeld skriptinterpreter für sql-server-aktualisierung

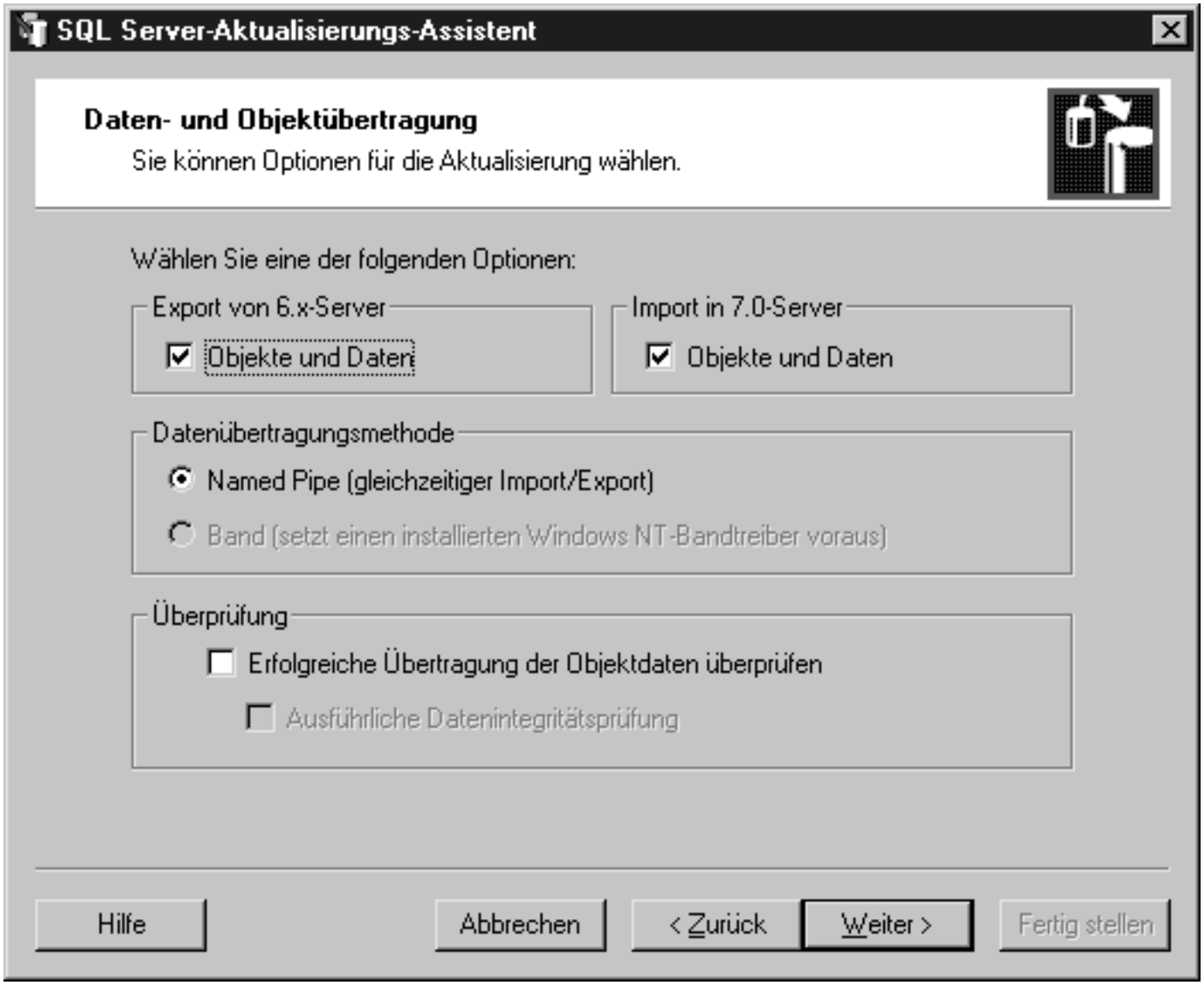
am ende erscheint (hoffentlich) die meldung *aktualisierung vollständig beendet*. klikken sie im meldungsfeld auf **ok**, im dialogfeld **skripterstellung** auf **schliessen** und im meldungsfeld mit der frage, ob sie die anwendung wirklich schließen möchten, auf **ja**. nachdem der assistent noch einige aufräumarbeiten erledigt hat, ist die versionsumstellung bzw. aktualisierung abgeschlossen.

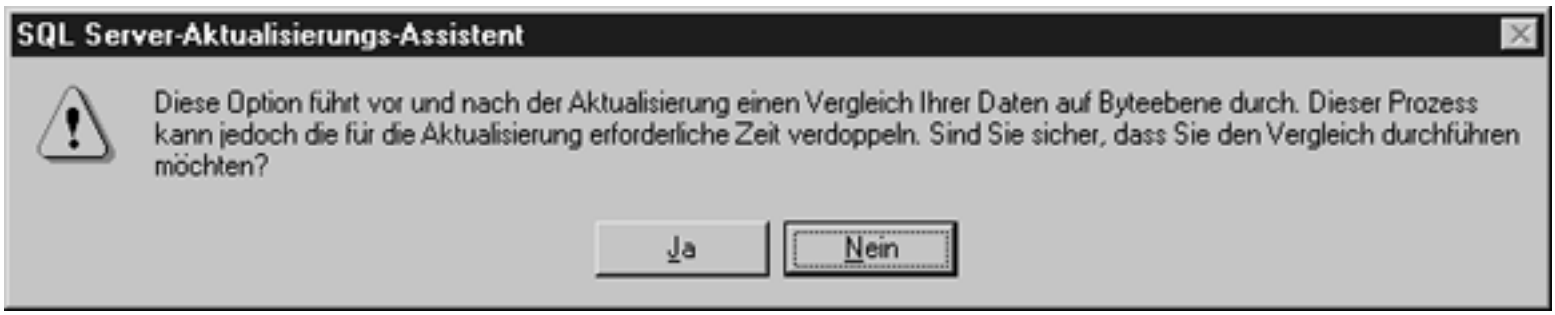
29.4 sichern/wiederherstellen

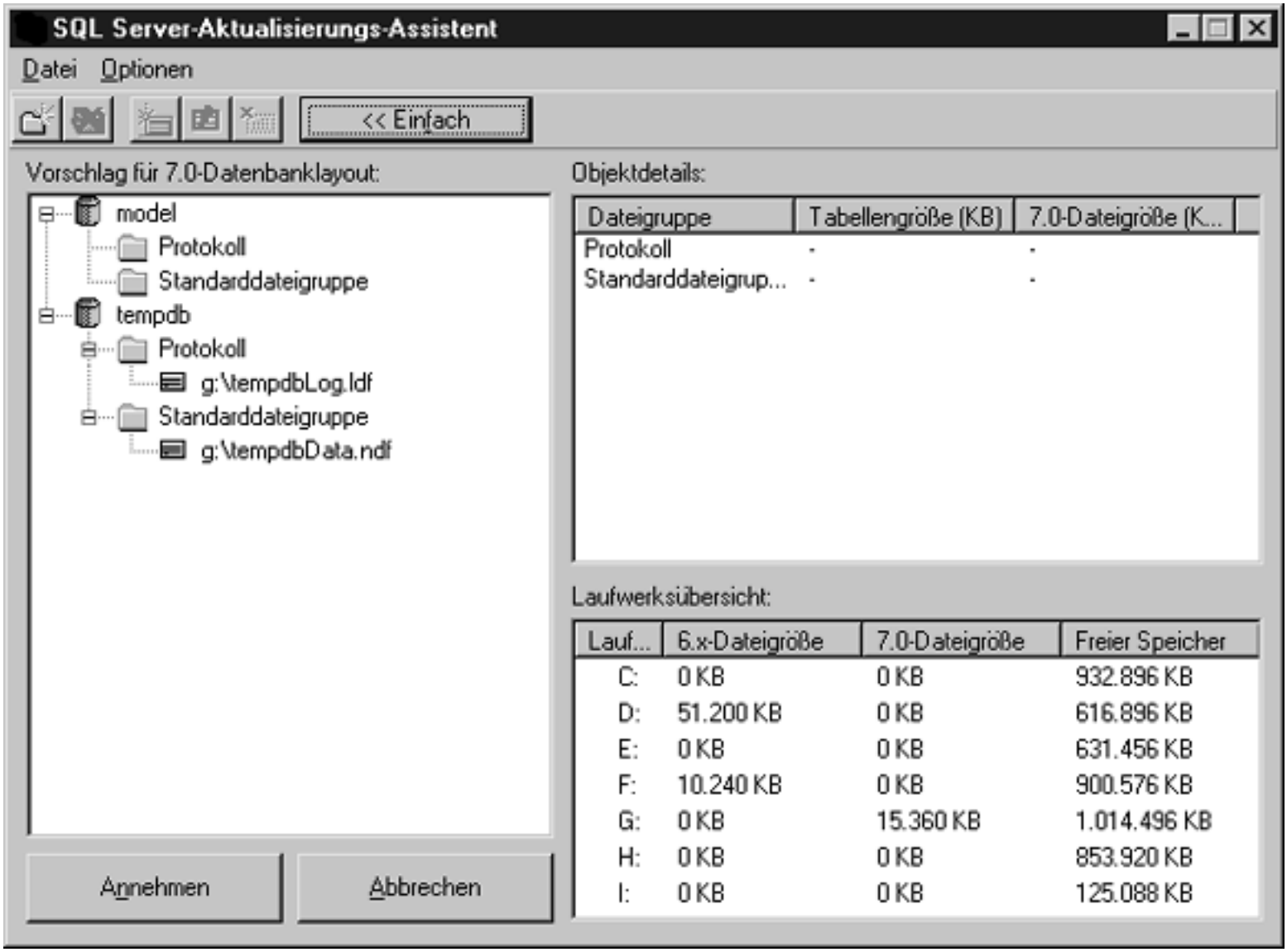
da die sicherungsformate früherer versionen von sql server nicht zur version 7.0 kompatibel sind, lassen sich datenbanksicherungen nicht direkt in sql server 7.0 wiederherstellen. um ältere datenbanken nach sql server 7.0 zu konvertieren, können sie folgende wege einschlagen:

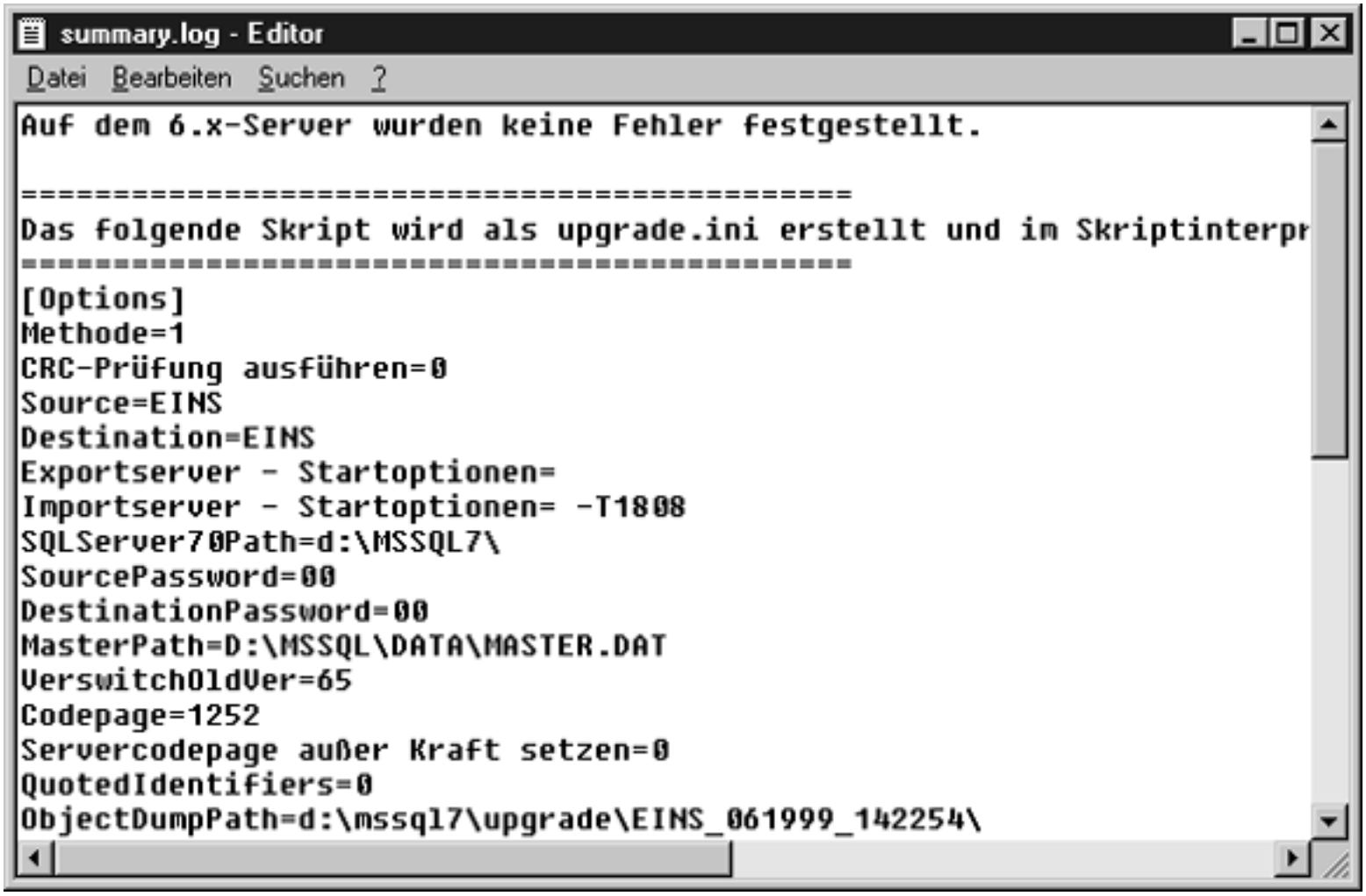
- die umstellung mit dem sql server-aktualisierungs-assistenten vornehmen,
- die daten mit hilfe der datentransformationsdienste (dts) von einem computer, auf dem sql server läuft, direkt auf einen anderen kopieren,
- die daten mit dem dienstprogramm bcp von einem computer, auf dem sql server 6.5 läuft, in eine datendatei kopieren und anschließend die daten aus der datendatei auf einen computer, auf dem sql server 7.0 läuft, kopieren.

in diesem kapitel haben sie den sql-server-aktualisierungs-assistenten kennengelernt. die erforderlichen schritte für die beiden letzten optionen wurden im kapitel 9 dargestellt.


SQL Server-Aktualisierungs-Assistent
Daten- und Objektübertragung
Sie können Optionen für die Aktualisierung wählen.
Wählen Sie eine der folgenden Optionen:
Export von 6.x-Server
Objekte und Daten
Import in 7.0-Server
Objekte und Daten
Datenübertragungsmethode
Named Pipe (gleichzeitiger Import/Export)
Band (setzt einen installierten Windows NT-Bandtreiber voraus)
Überprüfung
Erfolgreiche Übertragung der Objektdaten überprüfen
Ausführliche Datenintegritätsprüfung
Hilfe
Abbrechen
< Zurück
Weiter >
Fertig stellen


SQL Server-Aktualisierungs-Assistent
Diese Option führt vor und nach der Aktualisierung einen Vergleich Ihrer Daten auf Byteebene durch. Dieser Prozess kann jedoch die für die Aktualisierung erforderliche Zeit verdoppeln. Sind Sie sicher, dass Sie den Vergleich durchführen möchten?
Ja
Nein


SQL Server-Aktualisierungs-Assistent
Datei Optionen
<< Einfach
Vorschlag für 7.0-Datenbanklayout:
model
Protokoll
Standarddateigruppe
tempdb
Protokoll
g:\tempdbLog.ldf
Standarddateigruppe
g:\tempdbData.ndf
Annehmen
Abbrechen
Objektdetails:
Dateigruppe
Tabellengröße (KB)
7.0-Dateigröße (K...
Protokoll
-
-
Standarddateigrup...
-
-
Laufwerksübersicht:
Lauf...
6.x-Dateigröße
7.0-Dateigröße
Freier Speicher
C: 0 KB 0 KB 932.896 KB
D: 51.200 KB 0 KB 616.896 KB
E: 0 KB 0 KB 631.456 KB
F: 10.240 KB 0 KB 900.576 KB
G: 0 KB 15.360 KB 1.014.496 KB
H: 0 KB 0 KB 853.920 KB
I: 0 KB 0 KB 125.088 KB


summary.log - Editor
Datei Bearbeiten Suchen ?
Auf dem 6.x-Server wurden keine Fehler festgestellt.
==========
Das folgende Skript wird als upgrade.ini erstellt und im Skriptinterpr
==========
[Options]
Methode=1
CRC-Prüfung ausführen=0
Source=EINS
Destination=EINS
Exportserver - Startoptionen=
Importserver - Startoptionen= -T1808
SQLServer70Path=d:\MSSQL7\
SourcePassword=00
DestinationPassword=00
MasterPath=D:\MSSQL\DATA\MASTER.DAT
VerswitchOldVer=65
Codepage=1252
Servercodepage außer Kraft setzen=0
QuotedIdentifiers=0
ObjectDumpPath=d:\mssql7\upgrade\EINS_061999_142254\


SQL Server-Aktualisierungs-Assistent
SQL Server-Aktualisierungs-Assistenten beenden
Sie haben den SQL Server-Aktualisierungs-Assistenten erfolgreich beendet. Ihre Auswahl für die Aktualisierung wird im Folgenden angezeigt.
Zusammenfassung der Warnungen (sofern vorhanden) und der Auswahl:
Auf dem 6.x-Server wurden keine Fehler festgestellt.
==========
Das folgende Skript wird als upgrade.ini erstellt und im Skr
==========
[Options]
Methode=1
CRC-Prüfung ausführen=0
Warnungen und Auswahl werden im Editor angezeigt...
Klicken Sie auf 'Fertig stellen', um mit der Aktualisierung zu beginnen.
Hilfe
Abbrechen
< Zurück
Weiter >
Fertig stellen


Skriptinterpreter für SQL Server-Aktualisierung
Wählen Sie 'OK', um den nächste Task [Datenbankbesitzer exportieren] auszuführen, oder 'Abbrechen', um die derzeit ausgeführten Tasks zu beenden.
OK
Abbrechen


Skriptinterpreter für SQL Server-Aktualisierung
Führt Benutzernamen exportieren aus
Task
Status
Gestartet
Beende
Bereitet SQL-DMO für Aktualisierung vor
Vollständig
15:23:59
15:25:1
Servereinstellungen vom Masterserver exportieren
Vollständig
15:25:37
15:25:3
Benutzernamen exportieren
Wird ausgeführt
15:27:26
Task anhalten
Task abbrechen
Task wiederholen
Schließen
Zwischen Schritten anhalten

Anhang A Glossar

ieses Glossar soll keine Konkurrenz zum ausgezeichneten und recht umfangreichen Glossar der Online-Dokumentation von SQL Server sein. Es konzentriert sich vielmehr auf die - durchweg englischen - Abkürzungen, denen Sie auf Schritt und Tritt bei Ihrer Arbeit mit SQL Server begegnen. Wenn Sie doch einmal einen bestimmten Begriff suchen, finden Sie ihn sicherlich im Index und können dann die entsprechende Passage im Hauptteil dieses Buches nachlesen.

ACID

Akronym für Atomicity (Unteilbarkeit), Consistency (Konsistenz), Isolation und Durability (Beständigkeit). Eine logische Arbeitseinheit, die aus einer Folge von Operationen besteht, muß diese vier ACID-Eigenschaften aufweisen, um als Transaktion zu gelten.

ADO - ActiveX Data Objects

Nachfolger von DAO/RDO. Ein Objektmodell für den Zugriff auf Datenbanken. Der momentane Entwicklungsstand von ADO erlaubt es noch nicht, alle Anwendungen zum Beispiel von DAO auf ADO umzustellen. ADO MD ist eine Erweiterung für multidimensionale Datenanwendungen.

ANSI - American National Standards Institute

Ausschuß der US-amerikanischen Industrie und verschiedener Unternehmensgruppen, der sich mit Handels- und Kommunikationsstandards beschäftigt. ANSI ist Mitglied der International Organization for Standardization (ISO) und der International Electrotechnical Commission (IEC).

API - Application Programming Interface

Schnittstelle für Anwendungsprogrammierung. Eine festgelegte Gruppe von Funktionen, über die eine Anwendung systemnahe Dienste anfordern und aufrufen kann. Das zugrundeliegende Betriebssystem stellt diese Dienste bereit.

ASP - Active Server Pages

Eine Umgebung für serverseitige Skripten zum Erstellen dynamischer Webseiten und leistungsfähiger Webanwendungen.

CLI - Call Level Interface

Eine Bibliothek mit Funktionsaufrufen, die die SQL-Anweisungen unterstützen und der Call Level Interface-Spezifikation der SQL Access Group entsprechen. Beispiel für eine derartige Schnittstelle ist ODBC.

COM - Component Object Model

Eine Spezifikation, die alle Interaktionen zwischen Objekten definiert. COM beschreibt eine offene Architektur für die plattformübergreifende Entwicklung von Client-Server-Anwendungen. Es ist die ursprüngliche Spezifikation, auf der OLE basiert.

CPU - Central Processing Unit

Zentrale Verarbeitungseinheit. Im engeren Sinn der Mikroprozessor, der die Befehle eines Programms interpretiert und ausführt sowie periphere Elemente steuert.

DAO - Data Access Objects

Datenzugriffsobjekte. Eine hierarchisch organisierte Gruppe von Objekten, die den Entwickler einer Datenbankanwendung von den zugrundeliegenden Details beim Lesen und Schreiben der Datensätze abschirmt.

DAP - Data Access Pages

Von *MSDN Online* veröffentlichte Seiten als zentrale Informationsstelle für Themen, die sich auf Microsoft-Technologien des Datenzugriffs beziehen (http://www.microsoft.com/data/).

DBCC - Database Consistency Checker

Datenbankkonsistenzprüfer. Die Sprache Transact-SQL stellt verschiedene DBCC-Anweisungen bereit. Damit läßt sich zum Beispiel die physische und logische Konsistenz einer Datenbank oder die Speicherverwendung prüfen, eine Datenbank verkleinern oder die Leistungsstatistik überprüfen.

DBCS - Double-Byte Character Set

Doppelbyte-Zeichensatz. Ein Zeichensatz, der die Zeichen mit ein oder zwei Byte darstellt. DBCS ist nicht mit Unicode identisch.

DBMS - Database Management System

Datenbank-Managementsystem. Ein Programm, das die Zugriffe von Benutzern auf Datenbanken verwaltet.

DBO - Database Owner

Datenbankbesitzer. Jedes Mitglied der von sysadmin fixierten Server-Rolle wird auf einen bestimmten Benutzer innerhalb jeder Datenbank namens dbo abgebildet. Jedes Objekt, das ein Mitglied der durch sysadmin fixierten Server-Rolle erzeugt, gehört automatisch zu dbo.

DBOO - Database Object Owner

Der Datenbankobjekteigentümer (DBOO) ist der Benutzer, der das betreffende Datenbankobjekt erzeugt hat.

DCL - Data Control Language

Datensteuerungssprache. Mit den Befehlen dieser Teilmenge der Sprache SQL lassen sich die Berechtigungen für Datenbankobjekte steuern. In diese Kategorie fallen SQL-Anweisungen wie GRANT und REVOKE.

DDL - Data Definition Language

Datendefinitionssprache, auch als Datenbankentwurfssprache oder Datenentwurfssprache bezeichnet. Mit den Befehlen der DDL wird die Struktur einer Datenbank festgelegt. In diese Kategorie fallen SQL-Anweisungen wie CREATE TABLE, CREATE INDEX und DROP TABLE.

DLL - Dynamic Link Library

Dynamische Link-Bibliothek. Eine spezielle ausführbare Datei, die sich von mehreren Programmen gemeinsam verwenden läßt und erst bei Bedarf geladen wird. Die in einer DLL-Datei definierten Routinen werden dynamisch zur Laufzeit gebunden.

DML - Data Manipulation Language

Datenmanipulationssprache. Diejenige Teilmenge der Sprache SQL, mit der sich Daten abrufen und modifizieren lassen. Hierzu gehören Anweisungen wie SELECT, UPDATE, INSERT und DELETE.

DMF - Distributed Management Framework

Siehe SQL DMF.

DMO - Distributed Management Objects

Siehe SQL-DMO.

DNA - Distributed interNet Applications Architecture

Eine Architektur für die Integration von Web- und Client-Server-Modellen. Dieses umfassende Konzept definiert die Zusammenarbeit unterschiedlicher Produkte und Technologien von Microsoft.

DRI - Declarative Referential Integrity

Deklarative referentielle Integrität. Die von SQL Server bereitgestellte Funktionalität, mit der sich die Datenintegrität von verbundenen Tabellen durchsetzen läßt.

DSN - Data Source Name

Name einer Datenquelle. Ein registrierter Name, über den Anwendungen eine Verbindung zur Datenquelle - beispielsweise über eine ODBC-Schnittstelle - anfordern.

DTC - Siehe MS DTC

DTS - Data Transformation Services

Datentransformationsdienste. Mit den DTS lassen sich Data-Warehouse-Umgebungen einfacher erstellen, betreiben und warten. Zu diesem Zweck stellen die DTS komfortable Funktionen bereit, mit denen man Daten aus beliebigen heterogenen Datenquellen importieren, exportieren und transformieren kann.

EDI - Electronic Data Interchange

Der elektronische Datenaustausch.

ERD - Entity Relationship Diagram

Datenbankdiagramm. Eine Darstellung, die die Beziehungen zwischen Entitäten verdeutlicht.

FAT - File Allocation Table

Dateizuordnungstabelle. Das danach benannte FAT-Dateisystem verfolgt anhand der Einträge in der Dateizuordnungstabelle, wie der Speicherplatz auf dem Datenträger belegt ist und verwendet werden kann.

GUID - Global(ly) Unique Identifier

Global eindeutiger Bezeichner. SQL Server stellt im Datentyp uniqueidentifier einen GUID als Hexadezimalzahl mit 16 Byte Länge dar.

HCL - Hardware Compatibility List

Hardwarekompatibilitätsliste. Eine Liste mit Geräten, die den Kompatibilitätstest für das jeweilige Betriebssystem (zum Beispiel Windows NT, Windows 95/98) bestanden haben.

HOLAP - Hybrid OLAP

Speicherung von multidimensionalen Daten als Kombination von multidimensionalen Datenstrukturen (für Aggregationen) und relationalen Datenbanktabellen (für Fakten).

IDC - Internet Database Connector

Eine ISAPI-basierte Anwendung von Microsoft, die einfache, aber leistungsfähige Datenbankunterstützung für Webserver-basierte Anwendungen bietet. Damit kann ein Webbrowser auf beliebige ODBC-fähige Serverdatenbanken zugreifen.

IEC - International Electrotechnical Commission

Internationale Vereinigung, die sich unter anderem mit der Entwicklung von Standards im Bereich der Datenkommunikation beschäftigt. Siehe auch ANSI, ISO.

IIS - Internet Information Server

Siehe MIIS.

IPC - Interprocess Communication

Prozeßübergreifende Kommunikation. Ein System, durch das Threads und Prozesse Daten und Nachrichten untereinander austauschen können. Mit IPC lassen sich anderen Programmen Dienste zur Verfügung stellen oder von anderen Programmen übernehmen.

ISAM - Indexed Sequential Access Method

Indexsequentielle Zugriffsmethode. Ein Schema, nach dem sich Datensätze in großen Datenbanken schneller auffinden lassen. Ein Feld des Datensatzes dient als eindeutiger Schlüssel, der auf den Datensatz verweist.

ISAPI - Internet Server Application Programming Interface

Eine Anwendungsprogrammierschnittstelle für Internet-Server wie Windows NT Server, auf dem Microsoft Internet Information Server ausgeführt wird.

ISO - International Organization for Standardization

Internationale Vereinigung, die sich unter anderem mit der Entwicklung von Standards im Bereich der Datenkommunikation beschäftigt. Siehe auch ANSI, IEC.

ISV - Independent Software Vendors

Unabhängige Software-Anbieter.

JAD - Joint Application Development

Gemeinsame Anwendungsentwicklung.

MAPI - Messaging Application Programming Interface

Eine Schnittstelle für die Programmierung von Mail-fähigen Anwendungen.

MDAC - Microsoft Data Access Components

Eine Gruppe von Technologien, die Universal Data Objects implementieren und aus neuen, synchronisierten Versionen von ActiveX Data Objects (ADO), OLE DB und ODBC besteht.

MIIS - Microsoft Internet Information Server

In Windows NT integrierter Webserver, der FTP-, HTTP- und Gopher-Dienste bietet.

MMC - Microsoft Management Console

Eine einheitliche Benutzeroberfläche und Umgebung für Microsoft BackOffice-Server. Die Konsole für SQL Server ist der Enterprise Manager.

MOLAP - Multidimensional OLAP

Speicherung von Fakten und Aggregationen in einer proprietären multidimensionalen Struktur.

MSCS - Microsoft Cluster Server

Ein Funktionsmerkmal von Microsoft Windows NT Server Enterprise Edition. In der MSCS-Architektur lassen sich mehrere Server, oder Knoten, zu einem Cluster verbinden, um Verfügbarkeit, Zuverlässigkeit und Verwaltbarkeit von Daten und Anwendungen zu verbessern.

MSDE - Microsoft Data Engine

Eine in Microsoft Access integrierte Datenbank-Technologie. Der Client kann damit Daten, die voll kompatibel zu SQL Server sind, lokal vorhalten. Innerhalb von Access 2000 kann der Benutzer wahlweise entweder die Jet-Technologie oder die MSDE verwenden. Praktisch handelt es sich bei MSDE um eine abgespeckte Version des SQL Server 7.0.

MS DTC - Microsoft Distributed Transaction Coordinator

Ein Koordinationswerkzeug für verteilte Transaktionen. Die an einer Transaktion beteiligten Datenquellen können auf verschiedenen Computern untergebracht sein.

MTS - Microsoft Transaction Server

Auf Komponenten basierendes Transaktionsverarbeitungssystem, mit dem sich leistungsfähige, skalierbare und robuste Server-Anwendungen entwickeln, verteilen und verwalten lassen.

NTFS - NT File System

Dateisystem von Windows NT. Dieses Dateisystem realisiert umfangreiche Sicherheitsfunktionen, kann sehr große Speichermedien verwalten und bietet die Möglichkeit zur Wiederherstellung des Dateisystems.

ODBC - Open Database Connectivity

Eine offene und herstellerneutrale Schnittstelle für den Zugriff auf Datenbanken.

ODS - Open Data Services

Eine API für die Server-Seite eines Client-Server-Systems.

OLAP - Online Analytical Processing

Mit mehrdimensionalen Strukturen arbeitendes Verfahren, das schnellen Zugriff auf Daten für Analysen bietet.

OLE - Object Linking and Embedding

Objekte verknüpfen und einfügen. Eine API, die es mehreren Anwendungen gestattet, Objekte gemeinsam zu nutzen.

OLE DB

Auf dem COM-Modell basierende API für den Datenzugriff.

OLTP - Online Transaction Processing

Online-Transaktionsverarbeitung. Ein Datenbank-Managementsystem, in dem vorwiegend Datenänderungen - d.h. Transaktionen - von einer großen Anzahl gleichzeitiger Benutzer durchgeführt werden.

PDF - Package Definition Format (file)

Ein Dateiformat, das Microsoft Systems Management Server (SMS) verarbeiten kann, um SQL Server automatisch zu installieren.

RAID - Redundant Array of Inexpensive Disks

Fehlertolerante Festplattensysteme. Nach dem Niveau der Fehlertoleranz unterscheidet man die Systeme RAID 0 bis RAID 5, wobei RAID 5 die höchste Fehlertoleranz bietet, aber auch das teuerste System darstellt.

RDBMS - Relationales Datenbank-Managementsystem

Ein Datenbank-Managementsystem, das Datenbanken nach dem relationalen Modell verwaltet.

RDO - Remote Data Objects

Ein Objektmodell für den Zugriff auf relationale Remote-ODBC-Datenquellen.

ROLAP - Relational OLAP

Speicherung multidimensionaler Datenstrukturen in Tabellen einer relationalen Datenbank.

RPC - Remote Procedure Calls

Remoteprozeduraufrufe. Von einem Server ausgelöste Aufrufe gespeicherter Prozeduren auf einem Remoteserver.

SA - System Administrator

Kurzzeichen bzw. Benutzername für den Systemadministrator, der über alle Berechtigungen in einer Datenbank verfügt.

SID - Security Identifier

Sicherheits-ID. Unter Windows NT eine Benutzer-ID, die garantiert für jeden Benutzer in einer Windows-NT-Domäne eindeutig ist (GUID). SQL Server übernimmt automatisch die SID-Nummer für Benutzer und Gruppen von Windows NT, um eine SID für SQL-Server-Benutzernamen zu generieren.

SMP - Symmetric Multiprocessing

SQL Server unterstützt bis zu acht Prozessoren bei symmetrischem Mehrprozessorbetrieb.

SMS - (Microsoft) Systems Management Server

Ein Mitglied der BackOffice-Familie. Mit SMS können Sie Microsoft SQL Server automatisch auf mehreren Servern, die unter Microsoft Windows NT laufen, installieren.

SNA - Systems Network Architecture

Microsoft SNA Server erlaubt Client-Arbeitsstationen den Zugriff auf Großrechnerressourcen. Die SNA-Komponenten realisieren den Zugriff auf Dateien und Datenbanken sowie die Integration mit Transaktionen auf Großrechnern.

SNMP - Simple Network Management Protocol

Ein Anwendungsprotokoll, das Dienste zur Verwaltung von Netzwerken bietet. Mittels SNMP kann man Microsoft SQL Server über verschiedenartige Hardware-Plattformen (beispielsweise Microsoft Windows NT, Unix und HP) hinweg überwachen.

SPID - Systemprozeß-ID

Eine eindeutige, ganzzahlige Kennummer, die einer Benutzerverbindung für die Dauer der Verbindung zu SQL Server zugewiesen wird.

SQL - Structured Query Language

Strukturierte Abfragesprache. Eine Datenbanksprache, mit der sich relationale Datenbanken abfragen, aktualisieren und verwalten lassen.

SQL-DMF - SQL Distributed Management Framework

Eine integrierte Umgebung, die Objekte, Dienste und Komponenten zur Verwaltung von SQL Server bereitstellt. Die unterste Ebene bietet mit Hilfe von Transact-SQL direkten Zugriff auf das SQL-Server-Modul und die SQL-Server-Dienste. Die mittlere Ebene realisiert mit DMO eine Objektschnittstelle. Auf der obersten Ebene präsentiert sich die Umgebung in Form des grafischen Werkzeugs SQL Server Enterprise Manager.

SQL-DMO - SQL Distributed Management Objects

Mit der OLE-Automatisierung kompatible 32-Bit-Objekte des Component Object Model (COM) für Windows 95/98 und Windows NT. Mit den Objekten, Eigenschaften, Methoden und Auflistungen dieses Objektmodells lassen sich Programme zur Verwaltung mehrerer Computer mit SQL Server in einem Netzwerk schreiben.

SQL-NS - SQL Namespace

Eine API, deren Objekte die Elemente der Benutzeroberfläche von SQL Server Enterprise Manager kapseln.

TCP/IP - Transmission Control Protocol/Internet Protocol

Protokoll für die Kommunikation von Netzwerken im Internet.

TDS - Tabular Data Stream

SQL-Server-spezifisches Protokoll auf Anwendungsebene, mit dem Clients SQL-Anweisungen senden.

UDT - User Defined Data Type

Benutzerdefinierter Datentyp. In SQL Server ein Aliasname für einen Systemdatentyp.

UNC - Universal Naming Convention

Eine Konvention zur Benennung von Objekten in einer Netzwerkumgebung. Das Format eines UNC-Namens lautet: \\Servername\Freigabename\Pfad\Dateiname.

VLDB - Very Large Database

Sehr große Datenbank, etwa in der Größenordnung von 100 Gbyte an aufwärts.

Anhang B Transact-SQL in der Praxis

B.1 Warum in die Ferne schweifen ...

... wenn SQL Server bereits ein umfangreiches Praxisbeispiel frei Haus liefert? Im Skript InstPubs.sql finden Sie einen großen Teil der im Buch behandelten Befehle und Elemente von Transact-SQL. Das Setup-Programm installiert das Skript per Vorgabe im Verzeichnis \MSSQL7\Install\.

Im einzelnen führt das Skript folgende Aufgaben aus:

- Löschen der Datenbank pubs
- Neuerstellen dieser Datenbank
- Definieren aller Objekte der Datenbank
- Füllen der Datenbank mit Datensätzen

Die Erläuterung der Transact-SQL-Befehle anhand von InstPubs.sql bringt Ihnen außerdem den Vorteil, daß Sie die Kommentare in einer deutschen Version lesen können.

B.2 InstPubs.sql

Die Befehle im Skript InstPubs.sql sind in mehrere Teile gegliedert, die als separate Stapel ausgeführt werden. Die Endekennzeichnung der Stapel erfolgt mit dem Schlüsselwort GO.

Die Zeilennummern dienen lediglich der besseren Orientierung und sind nicht Bestandteil der Datei InstPubs.sql. Zeilen ohne Zeilennummern gehören zur jeweils vorhergehenden Zeile. Im Originalskript stehen die betreffenden Anweisungen also in ein und derselben Zeile.

Die folgende Analyse der Datei erfolgt in einzelnen Abschnitten, die meistens den Ausführungsstapeln entsprechen. An manchen Stellen sind Hinweise auf die einschlägigen Kapitel angegeben.

```
1:  /*                                                      */
2:  /*              InstPubs.SQL - Creates the Pubs database */
3:  /*                                                      */
4:
5:  GO
6:
```

Am Anfang des Skripts stehen Kommentarzeilen, die den Namen der Datei angeben. Die Kommentare beginnen mit /* und enden mit */. Dieses Format eignet sich auch für mehrzeilige Kommentare. Einzeilige Kommentare beginnen mit zwei Minuszeichen und erstrecken sich nur bis zum Ende der

aktuellen Zeile. Kommentare bewirken keine Operationen und sind vor allem für die Erläuterung des Programmablaufs nützlich.

Das GO am Ende der Kommentarzeilen dient lediglich einem sauberen Abschluß dieses Teils.

```
7:  set nocount    on
8:  set dateformat mdy
9:
10: USE master
11:
12: declare @dttm varchar(55)
13: select  @dttm=convert(varchar,getdate(),113)
14: raiserror('Beginning InstPubs.SQL at %s ....',1,1,@dttm)
    with nowait
15:
16: GO
```

Zeile 7 unterdrückt die Ausgabe der Meldung, wie viele Zeilen von einem Befehl betroffen sind. Zeile 8 legt die Reihenfolge der Datumsbestandteile (hier Monat, Tag, Jahr) fest. Diese Einstellung wird beim Konvertieren von Zeichenfolgen in Datumswerte verwendet. Auf die Anzeige des Datums hat sie keinen Einfluß.

In Zeile 10 wird die Systemdatenbank master als aktuelle Datenbank ausgewählt. Auf die jeweils aktuelle Datenbank beziehen sich alle weiteren Befehle, ohne daß die Datenbank explizit zu erwähnen ist. Diese Einstellung bleibt auch über den aktuellen Stapel hinaus gültig.

Zeile 12 deklariert die Variable @dttm vom Typ varchar mit einer Länge von 55 Zeichen. Benutzerdefinierte Variablen müssen mit einem At-Zeichen beginnen. Die Variable ist nur im aktuellen Stapel gültig, d.h. bis zur Ausführung des GO in Zeile 16.

Zeile 13 ruft mit der Funktion getdate() das aktuelle Systemdatum ab. Die Funktion convert wandelt das Datum in eine Zeichenfolge um. Dabei wird das Format mit der Codenummer 113 verwendet, das europäische Standardformat in der Form Tag, Monat, Jahr, wobei Tag und Monat zweistellig und das Jahr vierstellig (mit Jahrhundert) ausgegeben werden.

Die Anweisung in Zeile 14 liefert eine benutzerdefinierte Meldung. Eigentlich dient raiserror zur Ausgabe einer Fehlermeldung. Man könnte hier genauso gut mit print arbeiten. Durch die Konvertierung des Datums in eine Zeichenfolge erscheint die Ausgabe im gewohnten Format 22 Mär 1999 (mit ein paar Schönheitsfehlern). Bei der Datumsbehandlung kommt auch das Jahr-2000-Problem ins Spiel, das bei SQL Server eigentlich ein Jahr-1950- bis 2049-Problem ist (siehe dazu die Online-Dokumentation).

Zeile 16 beendet den Stapel.

```
17:
18: if exists (select * from sysdatabases where name='pubs')
19: begin
```

```
20:   raiserror('Dropping existing pubs database ....',0,1)
21:   DROP database pubs
22: end
23: GO
```

Zeile 17 testet mit der Anweisung if exists, ob die Datenbank pubs vorhanden ist, d.h., ob die in Klammern folgende Unterabfrage mit SELECT mindestens eine Zeile zurückgibt.

Die Systemtabelle sysdatabases enthält eine Zeile für jede Datenbank in SQL Server. Ist die Datenbank vorhanden, wird der in begin und end eingeschlossene Block in den Zeilen 19 bis 22 ausgeführt. Zeile 20 gibt mit der Anweisung raiserror die Meldung aus, daß die vorhandene Datenbank pubs gelöscht wird, und Zeile 21 löscht schließlich die Datenbank.

```
24:
25: CHECKPOINT
26: go
```

Die Transact-SQL-Anweisung CHECKPOINT bezeichnet einen Prüfpunkt. Dieser garantiert, daß an diesem Punkt alle an einer Datenbank vorgenommenen Änderungen auf den Datenträger geschrieben wurden. Damit läßt sich die Zeit bei Wiederherstellungen verkürzen. Hier dient CHECKPOINT dazu, das Löschen der Datenbank pubs zu besiegeln.

```
27:
28: raiserror('Creating pubs database....',0,1)
29: go
```

Zeile 28 gibt die Meldung aus, daß die Datenbank pubs erstellt wird.

Wenn Sie beim Befehl raiserror eine Fehlermeldung wie »Falsche Syntax in der Nähe von 'Text'« erhalten, haben Sie sicherlich ein »e« zuviel geschrieben. Denn wenn Ihnen die englischen Wörter raise und error geläufig sind, setzen Sie den Befehl vielleicht intuitiv aus diesen beiden Einzelwörtern zusammen: raiseerror, es muß aber nur raiserror heißen.

```
30: /*
31:    If SQL Server 4.2, 6.0, or 6.5, create a 3MB database.
32:    Use default size with autogrow if SQL Server 7.0 or
       later.
33: */
34:
35: IF (CHARINDEX('4.2', @@version) > 0 OR
36:     CHARINDEX('6.00', @@version) > 0 OR
```

```
37:     CHARINDEX('6.50', @@version) > 0 )
38:
39:    CREATE DATABASE pubs ON DEFAULT = 3
40: ELSE
41:    CREATE DATABASE pubs
42: GO
```

Der Kommentar ab Zeile 31 weist darauf hin, daß bei den Versionen 4.2, 6.0 oder 6.5 von SQL Server eine Datenbank der Größe 3 Mbyte erzeugt wird, während ab der Version 7.0 eine Standardgröße mit automatischem Wachstum zur Anwendung kommt.

Die Versionsprüfung findet in den Zeilen 35 bis 37 statt. Die Zeichenfolgenfunktion CHARINDEX liefert die Anfangsposition der ersten Zeichenfolge in der zweiten Zeichenfolge (hier dem Rückgabewert der Funktion @@version) zurück.

In den Versionen vor SQL Server 7.0 bezeichneten die beiden At-Zeichen globale Variablen, die vom System vordeklariert und bereitgestellt wurden. In SQL Server 7.0 stehen die beiden At-Zeichen für Systemfunktionen (auch wenn sie die Online-Dokumentation selbst gelegentlich als globale Variablen bezeichnet). Die Funktion @@version gibt Datum, Versionsnummer und Prozessortyp der aktuellen SQL-Server-Installation zurück.

Bei einer älteren Version von SQL Server erzeugt Zeile 39 die Datenbank pubs mit der Größe 3 Mbyte. Andernfalls erzeugt Zeile 41 die Datenbank mit der Standardgröße von SQL Server 7.0 (oder später).

```
43:
44: CHECKPOINT
45:
46: GO
```

Zeile 44 erzeugt wieder einen Prüfpunkt, um die Änderungen bis zu dieser Stelle festzuschreiben.

```
47:
48: USE pubs
49:
50: GO
```

Die neue Datenbank pubs wird zur aktuellen Datenbank gemacht.

```
51:
52: if db_name() <> 'pubs'
53:    raiserror('Error in InstPubs.SQL, ''USE pubs'' failed!
```

```
       Killing the SPID now.'
54:             ,22,127) with log
55:
56: GO
```

Die Metadatenfunktion db_name gibt den Namen der Datenbank zu einer angegebenen ID zurück. Fehlt die ID wie im Beispiel, liefert die Funktion den Namen der aktuellen Datenbank. Sollte beim Erstellen von pubs ein Fehler aufgetreten sein, gibt es keine aktuelle Datenbank pubs. Der Befehl USE pubs schlägt also fehl, und Zeile 53 bringt eine entsprechende Fehlermeldung.

```
57:
58: execute sp_dboption 'pubs' ,'trunc. log on chkpt.' ,'true'
59:
60: execute sp_addtype id      ,'varchar(11)' ,'NOT NULL'
61: execute sp_addtype tid     ,'varchar(6)'  ,'NOT NULL'
62: execute sp_addtype empid   ,'char(9)'     ,'NOT NULL'
63:
64: raiserror('Now at the create table section ....',0,1)
65:
66: GO
```

Zeile 58 ruft die gespeicherte Prozedur sp_dboption auf, um die Option *trunc. log on chkpt.* für die Datenbank pubs auf true zu setzen. Das bedeutet, daß ein Prüfpunkt den inaktiven Teil des Protokolls abschneidet, wenn sich die Datenbank im Protokollkürzungsmodus befindet (siehe dazu Kapitel 7).

Die Anweisungen in den Zeilen 60 bis 62 erstellen benutzerdefinierte Datentypen.

Zeile 64 weist darauf hin, daß sich der nächste Abschnitt dieses Skripts mit dem Erstellen der Tabellen beschäftigt.

```
67:
68: CREATE TABLE authors
69: (
70:    au_id          id
71:
72:          CHECK (au_id like '[0-9][0-9][0-9]-[0-9][0-9]-
             [0-9][0-9][0-9][0-9]')
73:
74:          CONSTRAINT UPKCL_auidind PRIMARY KEY CLUSTERED,
75:
76:    au_lname       varchar(40)       NOT NULL,
77:    au_fname       varchar(20)       NOT NULL,
78:
```

```
79:    phone          char(12)          NOT NULL
80:
81:          DEFAULT ('UNKNOWN'),
82:
83:    address        varchar(40)           NULL,
84:    city           varchar(20)           NULL,
85:    state          char(2)               NULL,
86:
87:    zip            char(5)               NULL
88:
89:          CHECK (zip like '[0-9][0-9][0-9][0-9][0-9]'),
90:
91:    contract       bit               NOT NULL
92: )
93:
94: GO
```

Als erstes wird die Tabelle authors (Autoren) mit den Spalten au_id, au_lname, au_fname, phone, address, city, state, zip und contract erstellt. Die CHECK-Einschränkung für die Spalte au_id bedeutet, daß nur Werte der Art 123-45-6789, also drei durch Bindestrich getrennte Zahlengruppen mit je drei, zwei und vier Ziffern, zulässig sind. Eine ähnliche Einschränkung ist für die Spalte zip (d.h. die Postleitzahl) definiert.

Die Klausel CONSTRAINT definiert eine Einschränkung für die Spalte au_id, und zwar einen eindeutigen Index, der gruppiert ist und auf den Namen UPKCL_auidind hört.

Die DEFAULT-Klausel definiert einen Standardwert für die Spalte phone.

```
95:
96: CREATE TABLE publishers
97: (
98:    pub_id         char(4)           NOT NULL
99:
100:         CONSTRAINT UPKCL_pubind PRIMARY KEY CLUSTERED
101:
102:         CHECK (pub_id in ('1389', '0736', '0877', '1622',
             '1756')
103:            OR pub_id like '99[0-9][0-9]'),
104:
105:   pub_name       varchar(40)           NULL,
106:   city           varchar(20)           NULL,
107:   state          char(2)               NULL,
108:
109:   country        varchar(30)           NULL
```

```
110:
111:          DEFAULT('USA')
112:)
113:
114:GO
115:
```

Die Zeilen 96 bis 112 erzeugen die Tabelle publishers - ebenfalls mit Einschränkungen und Standardwerten.

```
116:CREATE TABLE titles
117:(
118:   title_id       tid
119:
120:          CONSTRAINT UPKCL_titleidind PRIMARY KEY CLUSTERED,
121:
122:   title          varchar(80)       NOT NULL,
123:
124:   type           char(12)          NOT NULL
125:
126:          DEFAULT ('UNDECIDED'),
127:
128:   pub_id         char(4)               NULL
129:
130:          REFERENCES publishers(pub_id),
131:
132:   price          money                 NULL,
133:   advance        money                 NULL,
134:   royalty        int                   NULL,
135:   ytd_sales      int                   NULL,
136:   notes          varchar(200)          NULL,
137:
138:   pubdate        datetime          NOT NULL
139:
140:          DEFAULT (getdate())
141:)
142:
143:GO
144:
```

Die Zeilen 116 bis 143 legen die Tabelle titles an.

```
145:CREATE TABLE titleauthor
146:(
147:   au_id          id
148:
149:         REFERENCES authors(au_id),
150:
151:   title_id       tid
152:
153:         REFERENCES titles(title_id),
154:
155:   au_ord         tinyint              NULL,
156:   royaltyper     int                  NULL,
157:
158:
159:   CONSTRAINT UPKCL_taind PRIMARY KEY CLUSTERED(au_id,
        title_id)
160:)
161:
162:GO
163:
```

Die Zeilen 145 bis 162 erstellen die Tabelle titleauthor.

```
164:CREATE TABLE stores
165:(
166:   stor_id        char(4)           NOT NULL
167:
168:         CONSTRAINT UPK_storeid PRIMARY KEY CLUSTERED,
169:
170:   stor_name      varchar(40)          NULL,
171:   stor_address   varchar(40)          NULL,
172:   city           varchar(20)          NULL,
173:   state          char(2)              NULL,
174:   zip            char(5)              NULL
175:)
176:
177:GO
178:
```

In den Zeilen 164 bis 177 wird die Tabelle stores erzeugt.

```
179:CREATE TABLE sales
```

```
180:(
181:   stor_id        char(4)           NOT NULL
182:
183:         REFERENCES stores(stor_id),
184:
185:   ord_num         varchar(20)       NOT NULL,
186:   ord_date        datetime          NOT NULL,
187:   qty             smallint          NOT NULL,
188:   payterms        varchar(12)       NOT NULL,
189:
190:   title_id        tid
191:
192:         REFERENCES titles(title_id),
193:
194:
195:   CONSTRAINT UPKCL_sales PRIMARY KEY CLUSTERED (stor_id,
      ord_num, title_id)
196:)
197:
198:GO
199:
```

Die Zeilen 179 bis 198 erzeugen die Tabelle sales.

```
200:CREATE TABLE roysched
201:(
202:   title_id        tid
203:
204:         REFERENCES titles(title_id),
205:
206:   lorange         int                    NULL,
207:   hirange         int                    NULL,
208:   royalty         int                    NULL
209:)
210:
211:GO
212:
```

Die Zeilen 200 bis 211 erzeugen die Tabelle roysched. Die Bezeichnung roysched ist aus royalty schedule - etwa Planung der Tantiemen - zusammengesetzt.

```
213:CREATE TABLE discounts
```

```
214:(
215:   discounttype   varchar(40)       NOT NULL,
216:
217:   stor_id        char(4) NULL
218:
219:         REFERENCES stores(stor_id),
220:
221:   lowqty         smallint             NULL,
222:   highqty        smallint             NULL,
223:   discount       dec(4,2)          NOT NULL
224:)
225:
226:GO
227:
```

Die Zeilen 213 bis 224 erzeugen die Tabelle discounts.

```
228:CREATE TABLE jobs
229:(
230:   job_id         smallint          IDENTITY(1,1)
231:
232:         PRIMARY KEY CLUSTERED,
233:
234:   job_desc       varchar(50)       NOT NULL
235:
236:         DEFAULT 'New Position - title not formalized yet',
237:
238:   min_lvl        tinyint           NOT NULL
239:
240:         CHECK (min_lvl >= 10),
241:
242:   max_lvl        tinyint           NOT NULL
243:
244:         CHECK (max_lvl <= 250)
245:)
246:
247:GO
248:
```

In den Zeilen 228 bis 245 wird die Tabelle jobs erstellt.

```
249:CREATE TABLE pub_info
```

```
250:(
251:   pub_id         char(4)           NOT NULL
252:
253:         REFERENCES publishers(pub_id)
254:
255:         CONSTRAINT UPKCL_pubinfo PRIMARY KEY CLUSTERED,
256:
257:   logo           image                NULL,
258:   pr_info        text                 NULL
259:)
260:
261:GO
262:
```

Die Zeilen 249 bis 259 erstellen die Tabelle pub_info.

```
263:CREATE TABLE employee
264:(
265:   emp_id         empid
266:
267:         CONSTRAINT PK_emp_id PRIMARY KEY NONCLUSTERED
268:
269:         CONSTRAINT CK_emp_id CHECK (emp_id LIKE
270:            '[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]'
                or
271:            emp_id LIKE '[A-Z]-[A-Z][1-9][0-9][0-9][0-9]
               [0-9][FM]'),
272:
273:   fname          varchar(20)       NOT NULL,
274:   minit          char(1)               NULL,
275:   lname          varchar(30)       NOT NULL,
276:
277:   job_id         smallint          NOT NULL
278:
279:         DEFAULT 1
280:
281:         REFERENCES jobs(job_id),
282:
283:   job_lvl        tinyint
284:
285:         DEFAULT 10,
286:
287:   pub_id         char(4)           NOT NULL
288:
```

```
289:         DEFAULT ('9952')
290:
291:         REFERENCES publishers(pub_id),
292:
293:   hire_date      datetime          NOT NULL
294:
295:         DEFAULT (getdate())
296:)
297:
298:GO
299:
```

Die Zeilen 263 bis 296 erstellen die Tabelle employee. Damit ist das Erstellen der Datenbanktabellen für die Datenbank pubs abgeschlossen.

```
300:raiserror('Now at the create trigger section ...',0,1)
301:
302:GO
303:
```

Die Meldung in Zeile 300 weist darauf hin, daß im nächsten Abschnitt des Skripts ein Trigger erstellt wird.

```
304:CREATE TRIGGER employee_insupd
305:ON employee
306:FOR insert, UPDATE
307:AS
308:--Get the range of level for this job type from the jobs
      table.
309:declare @min_lvl tinyint,
310:   @max_lvl tinyint,
311:   @emp_lvl tinyint,
312:   @job_id smallint
313:select @min_lvl = min_lvl,
314:   @max_lvl = max_lvl,
315:   @emp_lvl = i.job_lvl,
316:   @job_id = i.job_id
317:from employee e, jobs j, inserted i
318:where e.emp_id = i.emp_id AND i.job_id = j.job_id
319:IF (@job_id = 1) and (@emp_lvl <> 10)
320:begin
321:   raiserror ('Job id 1 expects the default level of
```

```
       10.',16,1)
322:   ROLLBACK TRANSACTION
323:end
324:ELSE
325:IF NOT (@emp_lvl BETWEEN @min_lvl AND @max_lvl)
326:begin
327:   raiserror ('The level for job_id:%d should be between %d
       and %d.',
328:      16, 1, @job_id, @min_lvl, @max_lvl)
329:   ROLLBACK TRANSACTION
330:end
331:
332:GO
333:
```

Der Trigger prüft, ob beim Einfügen oder Aktualisieren von Zeilen die Werte für Job_Id und Job_Lvl in den zulässigen Bereichen liegen. Wenn Sie zum Beispiel mit der Anweisung

```
insert into employee (emp_id, fname, lname, job_id, job_lvl)
              values ('S-W12345F','Susi','Wong',1,2)
select * from employee
```

die Mitarbeiterin Susi Wong mit Job_Id = 1 und Job_Lvl = 2 in die Tabelle employee einfügen wollen, löst der Trigger aufgrund der Bedingung in Zeile 319 die folgende Fehlermeldung aus:

```
Server: Nachr.-Nr. 50000, Schweregrad 16, Status 1,  Prozedur
employee_insupd, Zeile 19
Job id 1 expects the default level of 10.
```

Diese Meldung geht auf die raiserror-Anweisung in Zeile 321 zurück. Der Trigger führt bei derartigen Fehlern einen Rollback aus, so daß die Tabelle nicht mit teilweise richtigen bzw. falschen Daten gefüllt wird.

Der umfangreichste Abschnitt der Datei InstPubs.sql enthält die Daten, mit denen die Tabellen gefüllt werden.

```
334:raiserror('Now at the inserts to authors ....',0,1)
335:
336:GO
337:
```

```
338:insert authors
339:   values('409-56-7008', 'Bennet', 'Abraham',
       '415 658-9932',
340:   '6223 Bateman St.', 'Berkeley', 'CA', '94705', 1)
341:insert authors
342:   values('213-46-8915', 'Green', 'Marjorie',
       '415 986-7020',
343:   '309 63rd St. #411', 'Oakland', 'CA', '94618', 1)
344:insert authors
345:   values('238-95-7766', 'Carson', 'Cheryl',
       '415 548-7723',
346:   '589 Darwin Ln.', 'Berkeley', 'CA', '94705', 1)
347:insert authors
348:   values('998-72-3567', 'Ringer', 'Albert',
       '801 826-0752',
349:   '67 Seventh Av.', 'Salt Lake City', 'UT', '84152', 1)
350:insert authors
351:   values('899-46-2035', 'Ringer', 'Anne', '801 826-0752',
352:   '67 Seventh Av.', 'Salt Lake City', 'UT', '84152', 1)
353:insert authors
354:   values('722-51-5454', 'DeFrance', 'Michel',
       '219 547-9982',
355:   '3 Balding Pl.', 'Gary', 'IN', '46403', 1)
356:insert authors
357:   values('807-91-6654', 'Panteley', 'Sylvia',
       '301 946-8853',
358:   '1956 Arlington Pl.', 'Rockville', 'MD', '20853', 1)
359:insert authors
360:   values('893-72-1158', 'McBadden', 'Heather',
361:   '707 448-4982', '301 Putnam', 'Vacaville', 'CA',
       '95688', 0)
362:insert authors
363:   values('724-08-9931', 'Stringer', 'Dirk',
       '415 843-2991',
364:   '5420 Telegraph Av.', 'Oakland', 'CA', '94609', 0)
365:insert authors
366:   values('274-80-9391', 'Straight', 'Dean',
       '415 834-2919',
367:   '5420 College Av.', 'Oakland', 'CA', '94609', 1)
368:insert authors
369:   values('756-30-7391', 'Karsen', 'Livia', '415 534-9219',
370:   '5720 McAuley St.', 'Oakland', 'CA', '94609', 1)
371:insert authors
372:   values('724-80-9391', 'MacFeather', 'Stearns',
       '415 354-7128',
373:   '44 Upland Hts.', 'Oakland', 'CA', '94612', 1)
```

```
374:insert authors
375:   values('427-17-2319', 'Dull', 'Ann', '415 836-7128',
376:   '3410 Blonde St.', 'Palo Alto', 'CA', '94301', 1)
377:insert authors
378:   values('672-71-3249', 'Yokomoto', 'Akiko',
       '415 935-4228',
379:   '3 Silver Ct.', 'Walnut Creek', 'CA', '94595', 1)
380:insert authors
381:   values('267-41-2394', 'O''Leary', 'Michael',
       '408 286-2428',
382:   '22 Cleveland Av. #14', 'San Jose', 'CA', '95128', 1)
383:insert authors
384:   values('472-27-2349', 'Gringlesby', 'Burt',
       '707 938-6445',
385:   'PO Box 792', 'Covelo', 'CA', '95428', 3)
386:insert authors
387:   values('527-72-3246', 'Greene', 'Morningstar',
      '615 297-2723',
388:   '22 Graybar House Rd.', 'Nashville', 'TN', '37215', 0)
389:insert authors
390:   values('172-32-1176', 'White', 'Johnson',
       '408 496-7223',
391:   '10932 Bigge Rd.', 'Menlo Park', 'CA', '94025', 1)
392:insert authors
393:   values('712-45-1867', 'del Castillo', 'Innes',
       '615 996-8275',
394:   '2286 Cram Pl. #86', 'Ann Arbor', 'MI', '48105', 1)
395:insert authors
396:   values('846-92-7186', 'Hunter', 'Sheryl',
       '415 836-7128',
397:   '3410 Blonde St.', 'Palo Alto', 'CA', '94301', 1)
398:insert authors
399:   values('486-29-1786', 'Locksley', 'Charlene',
       '415 585-4620',
400:   '18 Broadway Av.', 'San Francisco', 'CA', '94130', 1)
401:insert authors
402:   values('648-92-1872', 'Blotchet-Halls', 'Reginald',
       '503 745-6402',
403:   '55 Hillsdale Bl.', 'Corvallis', 'OR', '97330', 1)
404:insert authors
405:   values('341-22-1782', 'Smith', 'Meander',
       '913 843-0462',
406:   '10 Mississippi Dr.', 'Lawrence', 'KS', '66044', 0)
407:
408:GO
409:
```

Das Skript verwendet hier wieder die eigentlich für benutzerdefinierte Fehlermeldungen vorgesehene Anweisung raiserror, um über den Verlauf der Ausführung zu unterrichten. Die INSERT-Anweisungen fügen je einen vollständigen Datensatz in die angegebene Tabelle ein.

```
410:raiserror('Now at the inserts to publishers ....',0,1)
411:
412:GO
413:
414:insert publishers values('0736', 'New Moon Books',
    'Boston', 'MA', 'USA')
415:insert publishers values('0877', 'Binnet & Hardley',
    'Washington', 'DC', 'USA')
416:insert publishers values('1389', 'Algodata Infosystems',
    'Berkeley', 'CA', 'USA')
417:insert publishers values('9952', 'Scootney Books',
    'New York', 'NY', 'USA')
418:insert publishers values('1622', 'Five Lakes Publishing',
    'Chicago', 'IL', 'USA')
419:insert publishers values('1756', 'Ramona Publishers',
    'Dallas', 'TX', 'USA')
420:insert publishers values('9901', 'GGG&G', 'M_nchen', NULL,
    'Germany')
421:insert publishers values('9999', 'Lucerne Publishing',
    'Paris', NULL, 'France')
422:
423:GO
424:
425:raiserror('Now at the inserts to pub_info ....',0,1)
426:
427:GO
428:
429:insert pub_info values('0736', 0xFFFFFFFF, 'None yet')
430:insert pub_info values('0877', 0xFFFFFFFF, 'None yet')
431:insert pub_info values('1389', 0xFFFFFFFF, 'None yet')
432:insert pub_info values('9952', 0xFFFFFFFF, 'None yet')
433:insert pub_info values('1622', 0xFFFFFFFF, 'None yet')
434:insert pub_info values('1756', 0xFFFFFFFF, 'None yet')
435:insert pub_info values('9901', 0xFFFFFFFF, 'None yet')
436:insert pub_info values('9999', 0xFFFFFFFF, 'None yet')
437:GO
438:
439:
```

Die Anweisungen in den Zeilen 429 bis 436 füllen die Tabelle pub_info mit Vorgabewerten (nicht zu verwechseln mit Standardwerten, die SQL Server automatisch einfügt, wenn sie der Benutzer nicht bereitstellt). Normalerweise ist diese Tabelle für ein Logo des jeweiligen Verlegers vorgesehen, das aber im Rahmen dieses Installationsskripts noch nicht eingefügt wird. Deshalb erscheint auch in der Spalte pr_info der Hinweis 'None yet' (noch kein Logo vorhanden). Die Logos und die zugehörigen Texte können Sie über die Stapeldatei Pubimage.bat bzw. Pubtext.bat einfügen. Diese Dateien mitsamt den Bild- und Textdateien installiert das Setup-Programm normalerweise im Verzeichnis \MSSQL7\Install. Weitere Hinweise zum Importieren von Daten und speziell zu den genannten Stapeldateien finden Sie in Kapitel 9.

Das Einfügen der Daten in die einzelnen Tabellen vollzieht sich noch bis zur Skriptzeile 807. Das Skript wird hier ungekürzt wiedergegeben, denn vielleicht finden Sie noch das eine oder andere interessante Detail, wenn Sie selbst INSERT-Anweisungen schreiben und dabei mit Fehlern zu kämpfen haben.

```
440:raiserror('Now at the inserts to titles ....',0,1)
441:
442:GO
443:
444:insert titles values ('PC8888', 'Secrets of Silicon
    Valley', 'popular_comp', '1389',
445:$20.00, $8000.00, 10, 4095, 'Muckraking reporting on the
    world''s largest computer
446:hardware and software manufacturers.', '06/12/94')
447:
448:insert titles values ('BU1032', 'The Busy Executive''s
    Database Guide', 'business',
449:'1389', $19.99, $5000.00, 10, 4095, 'An overview of
    available database systems with
450:emphasis on common business applications. Illustrated.',
   '06/12/91')
451:
452:insert titles values ('PS7777', 'Emotional Security: A New
    Algorithm', 'psychology',
453:'0736', $7.99, $4000.00, 10, 3336, 'Protecting yourself and
    your loved ones from undue
454:emotional stress in the modern world. Use of computer and
    nutritional aids emphasized.',
455:'06/12/91')
456:
457:insert titles values ('PS3333', 'Prolonged Data
    Deprivation: Four Case Studies',
458:'psychology', '0736', $19.99, $2000.00, 10, 4072, 'What
    happens when the data runs dry?
459:Searching evaluations of information-shortage effects.',
```

```
   '06/12/91')
460:
461:insert titles values ('BU1111', 'Cooking with Computers:
    Surreptitious Balance Sheets',
462:'business', '1389', $11.95, $5000.00, 10, 3876, 'Helpful
    hints on how to use your
463:electronic resources to the best advantage.', '06/09/91')
464:
465:insert titles values ('MC2222', 'Silicon Valley Gastronomic
    Treats', 'mod_cook', '0877',
466:$19.99, $0.00, 12, 2032, 'Favorite recipes for quick, easy,
    and elegant meals.',
467:'06/09/91')
468:
469:insert titles values ('TC7777', 'Sushi, Anyone?',
    'trad_cook', '0877', $14.99, $8000.00,
470:10, 4095, 'Detailed instructions on how to make authentic
    Japanese sushi in your spare
471:time.', '06/12/91')
472:
473:insert titles values ('TC4203', 'Fifty Years in Buckingham
    Palace Kitchens', 'trad_cook',
474:'0877', $11.95, $4000.00, 14, 15096, 'More anecdotes from
    the Queen''s favorite cook
475:describing life among English royalty. Recipes, techniques,
    tender vignettes.',
476:'06/12/91')
477:
478:insert titles values ('PC1035', 'But Is It User Friendly?',
    'popular_comp', '1389',
479:$22.95, $7000.00, 16, 8780, 'A survey of software for the
    naive user, focusing on the
480:''friendliness'' of each.', '06/30/91')
481:
482:insert titles values('BU2075', 'You Can Combat Computer
    Stress!', 'business', '0736',
483:$2.99, $10125.00, 24, 18722, 'The latest medical and
    psychological techniques for living
484:with the electronic office. Easy-to-understand
    explanations.',
485:'06/30/91')
486:
487:insert titles values('PS2091', 'Is Anger the Enemy?',
   'psychology', '0736', $10.95,
488:$2275.00, 12, 2045, 'Carefully researched study of the
    effects of strong emotions on the
```

```
489:body. Metabolic charts included.', '06/15/91')
490:
491:insert titles values('PS2106', 'Life Without Fear',
   'psychology', '0736', $7.00, $6000.00,
492:10, 111, 'New exercise, meditation, and nutritional
    techniques that can reduce the shock
493:of daily interactions. Popular audience. Sample menus
    included, exercise video available
494:separately.', '10/05/91')
495:
496:insert titles values('MC3021', 'The Gourmet Microwave',
    'mod_cook', '0877', $2.99,
497:$15000.00, 24, 22246, 'Traditional French gourmet recipes
    adapted for modern microwave
498:cooking.', '06/18/91')
499:
500:insert titles values('TC3218', 'Onions, Leeks, and Garlic:
    Cooking Secrets of the Mediterranean',
501:'trad_cook', '0877', $20.95, $7000.00, 10, 375, 'Profusely
    illustrated in
502:color, this makes a wonderful gift book for a cuisine-
    oriented friend.', '10/21/91')
503:
504:insert titles (title_id, title, pub_id) values('MC3026',
    'The Psychology of Computer Cooking',
505:'0877')
506:
507:insert titles values ('BU7832', 'Straight Talk About
    Computers', 'business', '1389',
508:$19.99, $5000.00, 10, 4095, 'Annotated analysis of what
    computers can do for you: a no-
509:hype guide for the critical user.', '06/22/91')
510:
511:insert titles values('PS1372', 'Computer Phobic AND Non-
    Phobic Individuals: Behavior Variations',
512:'psychology', '0877', $21.59, $7000.00, 10, 375, 'A must
    for the specialist,
513:this book examines the difference between those who hate
    and fear computers and those who
514:don''t.', '10/21/91')
515:
516:insert titles (title_id, title, type, pub_id, notes)
    values('PC9999', 'Net Etiquette',
517:'popular_comp', '1389', 'A must-read for computer
    conferencing.')
518:
```

```
519:GO
520:
521:raiserror('Now at the inserts to titleauthor ....',0,1)
522:
523:GO
524:
525:insert titleauthor values('409-56-7008', 'BU1032', 1, 60)
526:insert titleauthor values('486-29-1786', 'PS7777', 1, 100)
527:insert titleauthor values('486-29-1786', 'PC9999', 1, 100)
528:insert titleauthor values('712-45-1867', 'MC2222', 1, 100)
529:insert titleauthor values('172-32-1176', 'PS3333', 1, 100)
530:insert titleauthor values('213-46-8915', 'BU1032', 2, 40)
531:insert titleauthor values('238-95-7766', 'PC1035', 1, 100)
532:insert titleauthor values('213-46-8915', 'BU2075', 1, 100)
533:insert titleauthor values('998-72-3567', 'PS2091', 1, 50)
534:insert titleauthor values('899-46-2035', 'PS2091', 2, 50)
535:insert titleauthor values('998-72-3567', 'PS2106', 1, 100)
536:insert titleauthor values('722-51-5454', 'MC3021', 1, 75)
537:insert titleauthor values('899-46-2035', 'MC3021', 2, 25)
538:insert titleauthor values('807-91-6654', 'TC3218', 1, 100)
539:insert titleauthor values('274-80-9391', 'BU7832', 1, 100)
540:insert titleauthor values('427-17-2319', 'PC8888', 1, 50)
541:insert titleauthor values('846-92-7186', 'PC8888', 2, 50)
542:insert titleauthor values('756-30-7391', 'PS1372', 1, 75)
543:insert titleauthor values('724-80-9391', 'PS1372', 2, 25)
544:insert titleauthor values('724-80-9391', 'BU1111', 1, 60)
545:insert titleauthor values('267-41-2394', 'BU1111', 2, 40)
546:insert titleauthor values('672-71-3249', 'TC7777', 1, 40)
547:insert titleauthor values('267-41-2394', 'TC7777', 2, 30)
548:insert titleauthor values('472-27-2349', 'TC7777', 3, 30)
549:insert titleauthor values('648-92-1872', 'TC4203', 1, 100)
550:
551:GO
552:
553:raiserror('Now at the inserts to stores ....',0,1)
554:
555:GO
556:
557:insert stores values('7066','Barnum''s','567 Pasadena
    Ave.','Tustin','CA','92789')
558:insert stores values('7067','News & Brews','577 First
    St.','Los Gatos','CA','96745')
559:insert stores values('7131','Doc-U-Mat: Quality Laundry and
    Books',
560:      '24-A Avogadro Way','Remulade','WA','98014')
561:insert stores values('8042','Bookbeat','679 Carson
```

```
    St.','Portland','OR','89076')
562:insert stores values('6380','Eric the Read Books','788
    Catamaugus Ave.',
563:       'Seattle','WA','98056')
564:insert stores values('7896','Fricative Bookshop','89
    Madison St.','Fremont','CA','90019')
565:
566:GO
567:
568:raiserror('Now at the inserts to sales ....',0,1)
569:
570:GO
571:
572:insert sales values('7066', 'QA7442.3', '09/13/94', 75, 'ON
    invoice','PS2091')
573:insert sales values('7067', 'D4482', '09/14/94', 10, 'Net
    60','PS2091')
574:insert sales values('7131', 'N914008', '09/14/94', 20, 'Net
    30','PS2091')
575:insert sales values('7131', 'N914014', '09/14/94', 25, 'Net
    30','MC3021')
576:insert sales values('8042', '423LL922', '09/14/94', 15, 'ON
    invoice','MC3021')
577:insert sales values('8042', '423LL930', '09/14/94', 10, 'ON
    invoice','BU1032')
578:insert sales values('6380', '722a', '09/13/94', 3, 'Net
    60','PS2091')
579:insert sales values('6380', '6871', '09/14/94', 5, 'Net
    60','BU1032')
580:insert sales values('8042','P723', '03/11/93', 25, 'Net
    30', 'BU1111')
581:insert sales values('7896','X999', '02/21/93', 35, 'ON
    invoice', 'BU2075')
582:insert sales values('7896','QQ2299', '10/28/93', 15, 'Net
    60', 'BU7832')
583:insert sales values('7896','TQ456', '12/12/93', 10, 'Net
    60', 'MC2222')
584:insert sales values('8042','QA879.1', '5/22/93', 30, 'Net
    30', 'PC1035')
585:insert sales values('7066','A2976', '5/24/93', 50, 'Net
    30', 'PC8888')
586:insert sales values('7131','P3087a', '5/29/93', 20, 'Net
    60', 'PS1372')
587:insert sales values('7131','P3087a', '5/29/93', 25, 'Net
    60', 'PS2106')
588:insert sales values('7131','P3087a', '5/29/93', 15, 'Net
```

```
    60', 'PS3333')
589:insert sales values('7131','P3087a', '5/29/93', 25, 'Net
    60', 'PS7777')
590:insert sales values('7067','P2121', '6/15/92', 40, 'Net
    30', 'TC3218')
591:insert sales values('7067','P2121', '6/15/92', 20, 'Net
    30', 'TC4203')
592:insert sales values('7067','P2121', '6/15/92', 20, 'Net
    30', 'TC7777')
593:
594:GO
595:
596:raiserror('Now at the inserts to roysched ....',0,1)
597:
598:GO
599:
600:insert roysched values('BU1032', 0, 5000, 10)
601:insert roysched values('BU1032', 5001, 50000, 12)
602:insert roysched values('PC1035', 0, 2000, 10)
603:insert roysched values('PC1035', 2001, 3000, 12)
604:insert roysched values('PC1035', 3001, 4000, 14)
605:insert roysched values('PC1035', 4001, 10000, 16)
606:insert roysched values('PC1035', 10001, 50000, 18)
607:insert roysched values('BU2075', 0, 1000, 10)
608:insert roysched values('BU2075', 1001, 3000, 12)
609:insert roysched values('BU2075', 3001, 5000, 14)
610:
611:GO
612:
613:insert roysched values('BU2075', 5001, 7000, 16)
614:insert roysched values('BU2075', 7001, 10000, 18)
615:insert roysched values('BU2075', 10001, 12000, 20)
616:insert roysched values('BU2075', 12001, 14000, 22)
617:insert roysched values('BU2075', 14001, 50000, 24)
618:insert roysched values('PS2091', 0, 1000, 10)
619:insert roysched values('PS2091', 1001, 5000, 12)
620:insert roysched values('PS2091', 5001, 10000, 14)
621:insert roysched values('PS2091', 10001, 50000, 16)
622:insert roysched values('PS2106', 0, 2000, 10)
623:
624:GO
625:
626:insert roysched values('PS2106', 2001, 5000, 12)
627:insert roysched values('PS2106', 5001, 10000, 14)
628:insert roysched values('PS2106', 10001, 50000, 16)
629:insert roysched values('MC3021', 0, 1000, 10)
```

```
630:insert roysched values('MC3021', 1001, 2000, 12)
631:insert roysched values('MC3021', 2001, 4000, 14)
632:insert roysched values('MC3021', 4001, 6000, 16)
633:insert roysched values('MC3021', 6001, 8000, 18)
634:insert roysched values('MC3021', 8001, 10000, 20)
635:insert roysched values('MC3021', 10001, 12000, 22)
636:
637:GO
638:
639:insert roysched values('MC3021', 12001, 50000, 24)
640:insert roysched values('TC3218', 0, 2000, 10)
641:insert roysched values('TC3218', 2001, 4000, 12)
642:insert roysched values('TC3218', 4001, 6000, 14)
643:insert roysched values('TC3218', 6001, 8000, 16)
644:insert roysched values('TC3218', 8001, 10000, 18)
645:insert roysched values('TC3218', 10001, 12000, 20)
646:insert roysched values('TC3218', 12001, 14000, 22)
647:insert roysched values('TC3218', 14001, 50000, 24)
648:insert roysched values('PC8888', 0, 5000, 10)
649:insert roysched values('PC8888', 5001, 10000, 12)
650:
651:GO
652:
653:insert roysched values('PC8888', 10001, 15000, 14)
654:insert roysched values('PC8888', 15001, 50000, 16)
655:insert roysched values('PS7777', 0, 5000, 10)
656:insert roysched values('PS7777', 5001, 50000, 12)
657:insert roysched values('PS3333', 0, 5000, 10)
658:insert roysched values('PS3333', 5001, 10000, 12)
659:insert roysched values('PS3333', 10001, 15000, 14)
660:insert roysched values('PS3333', 15001, 50000, 16)
661:insert roysched values('BU1111', 0, 4000, 10)
662:insert roysched values('BU1111', 4001, 8000, 12)
663:insert roysched values('BU1111', 8001, 10000, 14)
664:
665:GO
666:
667:insert roysched values('BU1111', 12001, 16000, 16)
668:insert roysched values('BU1111', 16001, 20000, 18)
669:insert roysched values('BU1111', 20001, 24000, 20)
670:insert roysched values('BU1111', 24001, 28000, 22)
671:insert roysched values('BU1111', 28001, 50000, 24)
672:insert roysched values('MC2222', 0, 2000, 10)
673:insert roysched values('MC2222', 2001, 4000, 12)
674:insert roysched values('MC2222', 4001, 8000, 14)
675:insert roysched values('MC2222', 8001, 12000, 16)
```

```
676:
677:GO
678:
679:insert roysched values('MC2222', 12001, 20000, 18)
680:insert roysched values('MC2222', 20001, 50000, 20)
681:insert roysched values('TC7777', 0, 5000, 10)
682:insert roysched values('TC7777', 5001, 15000, 12)
683:insert roysched values('TC7777', 15001, 50000, 14)
684:insert roysched values('TC4203', 0, 2000, 10)
685:insert roysched values('TC4203', 2001, 8000, 12)
686:insert roysched values('TC4203', 8001, 16000, 14)
687:insert roysched values('TC4203', 16001, 24000, 16)
688:insert roysched values('TC4203', 24001, 32000, 18)
689:
690:GO
691:
692:insert roysched values('TC4203', 32001, 40000, 20)
693:insert roysched values('TC4203', 40001, 50000, 22)
694:insert roysched values('BU7832', 0, 5000, 10)
695:insert roysched values('BU7832', 5001, 10000, 12)
696:insert roysched values('BU7832', 10001, 15000, 14)
697:insert roysched values('BU7832', 15001, 20000, 16)
698:insert roysched values('BU7832', 20001, 25000, 18)
699:insert roysched values('BU7832', 25001, 30000, 20)
700:insert roysched values('BU7832', 30001, 35000, 22)
701:insert roysched values('BU7832', 35001, 50000, 24)
702:
703:GO
704:
705:insert roysched values('PS1372', 0, 10000, 10)
706:insert roysched values('PS1372', 10001, 20000, 12)
707:insert roysched values('PS1372', 20001, 30000, 14)
708:insert roysched values('PS1372', 30001, 40000, 16)
709:insert roysched values('PS1372', 40001, 50000, 18)
710:
711:GO
712:
713:raiserror('Now at the inserts to discounts ....',0,1)
714:
715:GO
716:
717:insert discounts values('Initial Customer', NULL, NULL,
    NULL, 10.5)
718:insert discounts values('Volume Discount', NULL, 100, 1000,
    6.7)
719:insert discounts values('Customer Discount', '8042', NULL,
```

```
    NULL, 5.0)
720:
721:GO
722:
723:raiserror('Now at the inserts to jobs ....',0,1)
724:
725:GO
726:
727:insert jobs values ('New Hire - Job not specified', 10, 10)
728:insert jobs values ('Chief Executive Officer', 200, 250)
729:insert jobs values ('Business Operations Manager', 175,
    225)
730:insert jobs values ('Chief Financial Officier', 175, 250)
731:insert jobs values ('Publisher', 150, 250)
732:insert jobs values ('Managing Editor', 140, 225)
733:insert jobs values ('Marketing Manager', 120, 200)
734:insert jobs values ('Public Relations Manager', 100, 175)
735:insert jobs values ('Acquisitions Manager', 75, 175)
736:insert jobs values ('Productions Manager', 75, 165)
737:insert jobs values ('Operations Manager', 75, 150)
738:insert jobs values ('Editor', 25, 100)
739:insert jobs values ('Sales Representative', 25, 100)
740:insert jobs values ('Designer', 25, 100)
741:
742:GO
743:
744:raiserror('Now at the inserts to employee ....',0,1)
745:
746:GO
747:
748:insert employee values ('PTC11962M', 'Philip', 'T',
    'Cramer', 2, 215, '9952', '11/11/89')
749:insert employee values ('AMD15433F', 'Ann', 'M', 'Devon',
    3, 200, '9952', '07/16/91')
750:insert employee values ('F-C16315M', 'Francisco', '',
    'Chang', 4, 227, '9952', '11/03/90')
751:insert employee values ('LAL21447M', 'Laurence', 'A',
   'Lebihan', 5, 175, '0736',
752:'06/03/90')
753:insert employee values ('PXH22250M', 'Paul', 'X',
    'Henriot', 5, 159, '0877', '08/19/93')
754:insert employee values ('SKO22412M', 'Sven', 'K',
    'Ottlieb', 5, 150, '1389', '04/05/91')
755:insert employee values ('RBM23061F', 'Rita', 'B', 'Muller',
    5, 198, '1622', '10/09/93')
756:insert employee values ('MJP25939M', 'Maria', 'J',
```

```
    'Pontes', 5, 246, '1756', '03/01/89')
757:insert employee values ('JYL26161F', 'Janine', 'Y',
    'Labrune', 5, 172, '9901', '05/26/91')
758:insert employee values ('CFH28514M', 'Carlos', 'F',
    'Hernadez', 5, 211, '9999',
759:'04/21/89')
760:insert employee values ('VPA30890F', 'Victoria', 'P',
    'Ashworth', 6, 140, '0877',
761:'09/13/90')
762:insert employee values ('L-B31947F', 'Lesley', '', 'Brown',
    7, 120, '0877', '02/13/91')
763:insert employee values ('ARD36773F', 'Anabela', 'R',
    'Domingues', 8, 100, '0877',
764:'01/27/93')
765:insert employee values ('M-R38834F', 'Martine', '',
    'Rance', 9, 75, '0877', '02/05/92')
766:insert employee values ('PHF38899M', 'Peter', 'H',
    'Franken', 10, 75, '0877', '05/17/92')
767:insert employee values ('DBT39435M', 'Daniel', 'B',
    'Tonini', 11, 75, '0877', '01/01/90')
768:insert employee values ('H-B39728F', 'Helen', '',
    'Bennett', 12, 35, '0877', '09/21/89')
769:insert employee values ('PMA42628M', 'Paolo', 'M',
    'Accorti', 13, 35, '0877', '08/27/92')
770:insert employee values ('ENL44273F', 'Elizabeth', 'N',
    'Lincoln', 14, 35, '0877',
771:'07/24/90')
772:
773:GO
774:
775:insert employee values ('MGK44605M', 'Matti', 'G',
    'Karttunen', 6, 220, '0736',
776:'05/01/94')
777:insert employee values ('PDI47470M', 'Palle', 'D', 'Ibsen',
    7, 195, '0736', '05/09/93')
778:insert employee values ('MMS49649F', 'Mary', 'M',
    'Saveley', 8, 175, '0736', '06/29/93')
779:insert employee values ('GHT50241M', 'Gary', 'H', 'Thomas',
    9, 170, '0736', '08/09/88')
780:insert employee values ('MFS52347M', 'Martin', 'F',
    'Sommer', 10, 165, '0736', '04/13/90')
781:insert employee values ('R-M53550M', 'Roland', '',
    'Mendel', 11, 150, '0736', '09/05/91')
782:insert employee values ('HAS54740M', 'Howard', 'A',
    'Snyder', 12, 100, '0736', '11/19/88')
783:insert employee values ('TPO55093M', 'Timothy', 'P',
```

```
    'O''Rourke', 13, 100, '0736',
784:'06/19/88')
785:insert employee values ('KFJ64308F', 'Karin', 'F',
    'Josephs', 14, 100, '0736', '10/17/92')
786:insert employee values ('DWR65030M', 'Diego', 'W', 'Roel',
    6, 192, '1389', '12/16/91')
787:insert employee values ('M-L67958F', 'Maria', '',
    'Larsson', 7, 135, '1389', '03/27/92')
788:insert employee values ('PSP68661F', 'Paula', 'S',
    'Parente', 8, 125, '1389', '01/19/94')
789:insert employee values ('MAS70474F', 'Margaret', 'A',
    'Smith', 9, 78, '1389', '09/29/88')
790:insert employee values ('A-C71970F', 'Aria', '', 'Cruz',
    10, 87, '1389', '10/26/91')
791:insert employee values ('MAP77183M', 'Miguel', 'A',
    'Paolino', 11, 112, '1389',
792:'12/07/92')
793:insert employee values ('Y-L77953M', 'Yoshi', '',
    'Latimer', 12, 32, '1389', '06/11/89')
794:insert employee values ('CGS88322F', 'Carine', 'G',
    'Schmitt', 13, 64, '1389', '07/07/92')
795:insert employee values ('PSA89086M', 'Pedro', 'S',
    'Afonso', 14, 89, '1389', '12/24/90')
796:insert employee values ('A-R89858F', 'Annette', '',
    'Roulet', 6, 152, '9999', '02/21/90')
797:insert employee values ('HAN90777M', 'Helvetius', 'A',
    'Nagy', 7, 120, '9999', '03/19/93')
798:insert employee values ('M-P91209M', 'Manuel', '',
    'Pereira', 8, 101, '9999', '01/09/89')
799:insert employee values ('KJJ92907F', 'Karla', 'J',
    'Jablonski', 9, 170, '9999',
800:'03/11/94')
801:insert employee values ('POK93028M', 'Pirkko', 'O',
    'Koskitalo', 10, 80, '9999',
802:'11/29/93')
803:insert employee values ('PCM98509F', 'Patricia', 'C',
    'McKenna', 11, 150, '9999',
804:'08/01/89')
805:
806:GO
807:
```

Die Tabellen der Datenbank pubs sind nun vollständig mit Daten gefüllt. Die nächsten Stapelanweisungen erzeugen die Indizes für die einzelnen Tabellen.

```
808:raiserror('Now at the create index section ....',0,1)
    with nowait
809:
810:GO
811:
812:CREATE CLUSTERED INDEX employee_ind ON employee(lname,
    fname, minit)
813:
814:GO
815:
```

Die Zeile 812 erstellt einen gruppierten Index namens employee_ind mit den Spalten lname, fname und minit der Tabelle employee.

```
816:CREATE NONCLUSTERED INDEX aunmind ON authors (au_lname,
    au_fname)
817:GO
818:CREATE NONCLUSTERED INDEX titleidind ON sales (title_id)
819:GO
820:CREATE NONCLUSTERED INDEX titleind ON titles (title)
821:GO
822:CREATE NONCLUSTERED INDEX auidind ON titleauthor (au_id)
823:GO
824:CREATE NONCLUSTERED INDEX titleidind ON titleauthor
    (title_id)
825:GO
826:CREATE NONCLUSTERED INDEX titleidind ON roysched (title_id)
827:GO
828:
```

Die in den Zeilen 816 bis 828 für verschiedene Tabellen erstellten Indizes sind nicht gruppiert. (Pro Tabelle ist nur ein gruppierter Index möglich.)

```
829:raiserror('Now at the create view section ....',0,1)
830:
831:GO
832:
```

Auch an eine Sicht wurde gedacht. Die Zeilen 833 bis 841 erstellen die Sicht titleview, die Informationen

aus den drei Tabellen authors, titles und titleauthor zusammenfaßt.

```
833:CREATE VIEW titleview
834:AS
835:select title, au_ord, au_lname, price, ytd_sales, pub_id
836:from authors, titles, titleauthor
837:where authors.au_id = titleauthor.au_id
838:   AND titles.title_id = titleauthor.title_id
839:
840:GO
841:
```

In dieser Sichtdefinition sind vor allem die Zeilen 837 und 838 interessant. Führen Sie einmal versuchsweise die Anweisungszeilen 835 und 836 aus. Die Abfrage läuft eine kleine Ewigkeit und liefert dann 10350 Zeilen. Hier haben Sie es mit dem sogenannten kartesischen Produkt zu tun, das sämtliche Kombinationen aller Zeilen der drei Tabellen bildet. Die drei Originaltabellen authors, titles und titleauthor der Datenbank pubs enthalten 23, 18 und 25 Zeilen, und 23 * 18 * 25 ergibt 10350. Mit der Bedingung in den Zeilen 837 und 838 umfaßt die Ergebnismenge lediglich 25 Zeilen.

```
842:raiserror('Now at the create procedure section ....',0,1)
843:
844:GO
845:
```

Die Meldung in Zeile 842 weist darauf hin, daß das Skript jetzt gespeicherte Prozeduren erstellt.

```
846:CREATE PROCEDURE byroyalty @percentage int
847:AS
848:select au_id from titleauthor
849:where titleauthor.royaltyper = @percentage
850:
851:GO
852:
```

Die Zeilen 846 bis 849 erzeugen die gespeicherte Prozedur byroyalty. Diese Prozedur liefert aus der Tabelle titleauthor die Autorenkennungen (au_id) für alle Autoren, die einen bestimmten Prozentsatz der Tantiemen erhalten. Der Prozentsatz wird als ganzzahliger Parameter (@percentage mit der Typendeklaration int) an die Prozedur übergeben.

Wollen Sie zum Beispiel wissen, welche Autoren nur 25 Prozent der Tantiemen erhalten, rufen Sie die

Prozedur wie folgt auf:

```
exec byroyalty 25
```

Als Ergebnis erhalten Sie:

```
au_id
-----------
724-80-9391
899-46-2035

(2 row(s) affected)
```

Die nächste Anweisung zeigt den Hinweis an, daß es sich um die erste GRANT-Anweisung handelt, die in den Prozedurabschnitt eingebettet ist.

```
853:raiserror('(Next is the first Grant embedded in the proc
    section.)',0,1)
854:
855:GRANT execute ON byroyalty TO public
856:
857:GO
858:
```

Die Anweisung GRANT execute ON gewährt der gespeicherten Prozedur byroyalty Ausführungsrechte für die public-Rolle - d.h., jedermann darf diese Prozedur ausführen.

```
859:CREATE PROCEDURE reptq1 AS
860:select pub_id, title_id, price, pubdate
861:from titles
862:where price is NOT NULL
863:order by pub_id
864:COMPUTE avg(price) BY pub_id
865:COMPUTE avg(price)
866:
867:GO
868:
```

Die Zeilen 859 bis 865 erzeugen die gespeicherte Prozedur reptq1. Diese Prozedur wählt aus der Tabelle titles die Spalten pub_id, title_id, price und pubdate aus, wenn der Preis ungleich NULL ist. Die Ergebnismenge wird nach der Spalte pub_id gruppiert. Für die einzelnen Gruppen berechnet die Prozedur den Mittelwert der Preise und gibt außerdem den Mittelwert der Preise für die gesamte Tabelle als letzte Zeile aus.

```
869:GRANT execute ON reptq1 TO public
870:
871:GO
872:
```

Zeile 869 weist allen Benutzern der Datenbank (d.h. der public-Rolle) die EXECUTE-Berechtigung (eine Anweisungsberechtigung) für die gespeicherte Prozedur reptq1 zu.

```
873:CREATE PROCEDURE reptq2 AS
874:select type, pub_id, titles.title_id, au_ord,
875:   Name = substring (au_lname, 1,15), ytd_sales
876:from titles, authors, titleauthor
877:where titles.title_id = titleauthor.title_id AND
    authors.au_id = titleauthor.au_id
878:   AND pub_id is NOT NULL
879:order by pub_id, type
880:COMPUTE avg(ytd_sales) BY pub_id, type
881:COMPUTE avg(ytd_sales) BY pub_id
882:
883:GO
884:
```

Die Zeilen 873 bis 881 erzeugen die gespeicherte Prozedur reptq2 ...

```
885:GRANT execute ON reptq2 TO public
886:
887:GO
888:
```

... und weisen ihr wieder die Ausführungsrechte zu.

Das gleiche wiederholt sich für die gespeicherte Prozedur reptq3:

```
889:CREATE PROCEDURE reptq3 @lolimit money, @hilimit money,
```

```
890:@type char(12)
891:AS
892:select pub_id, type, title_id, price
893:from titles
894:where price >@lolimit AND price <@hilimit AND type = @type
    OR type LIKE '%cook%'
895:order by pub_id, type
896:COMPUTE count(title_id) BY pub_id, type
897:
898:GO
899:
900:GRANT execute ON reptq3 TO public
901:
902:GO
903:
```

Damit jeder Benutzer der Datenbank pubs eigene Prozeduren erstellen kann, weist Zeile 904 die Berechtigung zum Erstellen von Prozeduren der public-Rolle zu.

```
904:GRANT CREATE PROCEDURE TO public
905:
906:GO
907:
```

Die folgenden Zeilen beschäftigen sich mit dem Einrichten eines Gastkontos (siehe dazu Kapitel 21).

```
908:raiserror('Now at the GUEST section ....',0,1)
909:
910:GO
911:
912:/* Test to see if guest account needs to be added.
913:   The 7.0 version of the test has to query the
914:   sysusers.hasdbaccess column, which does not
915:   exist in earlier versions of SQL Server. Refering
916:   to this column on earlier versions would generate
917:   a compile error, so the reference is in a string
918:   that is only executed on 7.0 servers.  Could not
919:   use sp_executesql because that is also not
920:   available on earlier versions.
921:*/
```

Der Text in den Zeilen 912 bis 920 sagt folgendes aus:

Prüfen, ob Gastkonto hinzuzufügen ist. Der Test in der Version 7.0 von SQL Server muß die Spalte sysusers.hasdbaccess abfragen. In früheren Versionen von SQL Server gibt es diese Spalte nicht. Ein Bezug auf diese Spalte in früheren Versionen würde zu einem Compiler-Fehler führen. Deshalb ist dieser Verweis in einer Zeichenfolge untergebracht, die nur auf Servern mit der Version 7.0 ausgeführt wird. Die gespeicherte Prozedur sp_executesql läßt sich ebenfalls nicht verwenden, da sie in früheren Versionen von SQL Server nicht vorhanden ist.

```
922:IF CHARINDEX( '7.0', @@VERSION) > 0
923:  EXEC ('IF NOT EXISTS (SELECT *
924:                        FROM sysusers
925:                        WHERE name = ''guest''
926:                        AND hasdbaccess = 1)
927:         EXEC sp_adduser ''guest'' ')
928:ELSE
929:  IF NOT EXISTS (SELECT *
930:                 FROM sysusers
931:                 WHERE name = 'guest')
932:  EXEC sp_adduser 'guest'
933:GO
934:
```

Die Zeilen 922 bis 934 prüfen, ob das Benutzerkonto guest in der Systemtabelle sysusers eingetragen ist. Wenn nicht, wird es mit der gespeicherten Prozedur sp_adduser hinzugefügt. Kapitel 12 geht auf die Ausführung von Zeichenfolgen mit Hilfe von EXECUTE ein.

```
935:GRANT ALL ON publishers TO guest
936:GRANT ALL ON pub_info TO guest
937:GRANT ALL ON employee TO guest
938:GRANT ALL ON jobs TO guest
939:GRANT ALL ON titles TO guest
940:GRANT ALL ON authors TO guest
941:GRANT ALL ON titleauthor TO guest
942:GRANT ALL ON sales TO guest
943:GRANT ALL ON roysched TO guest
944:GRANT ALL ON stores TO guest
945:GRANT ALL ON discounts TO guest
946:GRANT ALL ON titleview TO guest
947:
```

Die Anweisungen GRANT ALL ON *Tabellenname* TO in den Zeilen 935 bis 946 erteilen dem Benutzer

guest alle anwendbaren Objektberechtigungen für die angegebenen Tabellen.

```
948:GRANT execute ON byroyalty TO guest
949:
```

Zeile 948 erteilt dem Benutzer guest die EXECUTE-Berechtigung für die Ausführung der gespeicherten Prozedur byroyalty.

```
950:GRANT CREATE TABLE TO guest
951:GRANT CREATE VIEW TO guest
952:GRANT CREATE RULE TO guest
953:GRANT CREATE DEFAULT TO guest
954:GRANT CREATE PROCEDURE TO guest
955:
956:GO
957:
```

Die Zeilen 950 bis 954 gewähren dem Benutzer guest die Berechtigungen zum Erzeugen von Tabellen, Sichten, Regeln, Standardwerten und gespeicherten Prozeduren.

```
958:UPDATE STATISTICS publishers
959:UPDATE STATISTICS employee
960:UPDATE STATISTICS jobs
961:UPDATE STATISTICS pub_info
962:UPDATE STATISTICS titles
963:UPDATE STATISTICS authors
964:UPDATE STATISTICS titleauthor
965:UPDATE STATISTICS sales
966:UPDATE STATISTICS roysched
967:UPDATE STATISTICS stores
968:UPDATE STATISTICS discounts
969:
970:GO
971:
```

Die Anweisung UPDATE STATISTICS aktualisiert Informationen zur Verteilung von Schlüsselwerten für die jeweilige Tabelle. Da kein Index als Parameter angegeben ist, wird die Verteilungsstatistik für alle Indizes aktualisiert.

```
972:CHECKPOINT
```

```
973:
974:GO
975:
976:USE master
977:
978:GO
979:
980:CHECKPOINT
981:
982:GO
983:
```

Die Prüfpunkte in den Zeilen 972 und 980 schreiben die Änderungen fest, Zeile 976 macht die Datenbank master zur aktuellen Datenbank.

```
984:declare @dttm varchar(55)
985:select  @dttm=convert(varchar,getdate(),113)
986:raiserror('Ending InstPubs.SQL at %s ....',1,1,@dttm) with
    nowait
987:
988:GO
989:-- -
990:
```

Die Zeilen 984 bis 986 geben eine abschließende Meldung mit aktuellem Datum und Uhrzeit aus. Die hier verwendeten Funktionen wurden bereits zu Beginn für die Zeilen 12 bis 14 beschrieben.

B.3 pubs = Verleger

In SQL Server 6.5 hat Microsoft die Beispieldatenbanken in einer deutschen Version mitgeliefert. SQL Server 7.0 bringt leider nur die englische Version mit. Für Leser, die bereits mit SQL Server 6.5 gearbeitet haben, und zum besseren Verständnis im allgemeinen stellt Tabelle B.1 die englischen und deutschen Tabellen- und Spaltenbezeichnungen der Datenbank pubs vormals Verleger gegenüber, gibt darüber hinaus die Datentypen an und nennt die Zeilenzahl der jeweiligen Tabelle im Anfangszustand nach der Installation von SQL Server.

Tabellen und Spalten englisch (SQL Server 7.0)	Tabellen und Spalten deutsch (SQL Server 6.5)	Datentyp (Größe)/Bemerkungen
titles	Titel	Tabelle mit 18 Zeilen
title_id	Titel_id	tid (varchar(6))

title	Titel	varchar(80)
type	Gebiet	char(12)
pub_id	V_id	char(4)
price	Preis	money(8)
advance	Vorschuß	money(8)
royalty	Tantiemen	int(4)
ytd_sales	Verkäufe	int(4)
notes	Bemerkungen	varchar(200)
pubdate	Erscheinungsdatum	datetime(8)
sales	Auftrag	Tabelle mit 21 Zeilen
stor_id	B_id	char(4)
ord_num	Auftragsnr	varchar(20)
ord_date	Datum	datetime(8)
qty	Menge	smallint(2)
payterms	Zahlungsbed	varchar(12)
title_id	Titel_id	tid (varchar(6))
stores	Buchhandlungen	Tabelle mit 6 Zeilen
stor_id	B_id	char(4)
stor_name	Name	varchar(40)
stor_address	Straße	varchar(40)
city	Ort	varchar(20)
state	Land	char(2)
zip	PLZ	char(5)
discounts	Rabatte	Tabelle mit 3 Zeilen
discounttype	Rabattart	varchar(40)
stor_id	B_id	char(4)
lowqty	Mindestmenge	smallint(2)
highqty	Höchstmenge	smallint(2)
discount	Rabatt	decimal(5)
roysched	Tantiemen_tab	Tabelle mit 86 Zeilen

title_id	Titel_id	tid (varchar(6))
lorange	U_bereich	int(4)
hirange	O_bereich	int(4)
royalty	Tantiemen	int(4)
titleauthor	Titelautor	Tabelle mit 25 Zeilen
au_id	A_id	id (varchar(11))
title_id	Titel_id	tid (varchar(6))
au_ord	A_mit	tinyint(1)
royaltyper	Anteil	int(4)
authors	Autoren	Tabelle mit 23 Zeilen
au_id	A_id	id (varchar(11))
au_lname	A_nname	varchar(40)
au_fname	A_vname	varchar(20)
phone	Telefonnr	char(12)
address	Straße	varchar(40)
city	Ort	varchar(20)
state	Land	char(2)
zip	PLZ	char(5)
contract	Vertrag	bit(1)
publishers	Verleger	Tabelle mit 8 Zeilen
pub_id	V_id	char(4)
pub_name	V_name	varchar(40)
city	Ort	varchar(20)
state	Land	char(2)
country	Staat	varchar(30)
employee	Angestellte	Tabelle mit 43 Zeilen
emp_id	Ang_id	empid (char(9))
fname	Vname	varchar(20)
minit	Initial	char(1)
lname	Nname	varchar(30)

job_id	Job_id	smallint(2)
job_lvl	Job_lvl	tinyint(1)
pub_id	V_id	char(4)
hire_date	Anst_datum	datetime(8)
jobs	Job	Tabelle mit 14 Zeilen
job_id	Job_id	smallint(2)
job_desc	Job_Besch	varchar(50)
min_lvl	Min_lvl	tinyint(1)
max_lvl	Max_lvl	tinyint(1)
pub_info	V_info	Tabelle mit 8 Zeilen
pub_id	V_id	char(4)
logo	Logo	image(16)
pr_info	Pr_info	text(16)

Tabelle B.1: Tabellen und Spalten der Beispieldatenbank pubs

Die angegebenen Zeilenzahlen für die einzelnen Tabellen beziehen sich auf die Installationsversion.

Anhang C Referenzteil

iese Referenz bringt einen Überblick über Transact-SQL-Anweisungen, eine Auswahl von gespeicherten Systemprozeduren, DBCC-Anweisungen und die reservierten Schlüsselwörter von SQL Server.

C.1 Transact-SQL-Anweisungen

Tabelle C.1 gibt einen Überblick über die Transact-SQL-Anweisungen der Datendefinitions-, Datenmanipulations- und Datensteuerungssprache von SQL Server.

Anweisung	Beschreibung
ALL	vergleicht einen Skalarwert mit den Werten einer Spalte
ALTER DATABASE	ändert die Definition einer vorhandenen Datenbank
ALTER PROCEDURE	ändert die Definition einer vorhandenen Prozedur
ALTER TABLE	ändert die Definition einer vorhandenen Tabelle
ALTER TRIGGER	ändert die Definition eines vorhandenen Triggers
ALTER VIEW	ändert die Definition einer vorhandenen Sicht
AND	Verknüpft zwei boolesche Ausdrücke und gibt TRUE zurück, wenn beide Ausdrücke TRUE sind. (Beachten Sie den Sonderfall UNKNOWN.)
ANY	vergleicht einen Skalarwert mit den Werten einer Spalte
BACKUP	sichert eine Datenbank, ein Transaktionsprotokoll oder Dateien bzw. Dateigruppen
BEGIN DISTRIBUTED TRANSACTION	markiert den Beginn einer verteilten Transaktion
BEGIN TRANSACTION	markiert den Beginn einer expliziten lokalen Transaktion
BETWEEN	spezifiziert einen Bereich in einem Testausdruck

BULK INSERT	kopiert Daten aus einer Datei in eine Tabelle oder Sicht
CHECKPOINT	schreibt alle modifizierten Seiten der aktuellen Datenbank auf den Datenträger
CLOSE	schließt einen geöffneten Cursor
COMMIT TRANSACTION	markiert das Ende einer erfolgreichen Transaktion
COMMIT WORK	markiert das Ende einer erfolgreichen Transaktion
CONTAINS	wird bei einer Fuzzy-Suche in Spalten mit Zeichendatentypen verwendet
CREATE DATABASE	erstellt eine neue Datenbank
CREATE DEFAULT	erstellt einen Standardwert (ein Objekt)
CREATE INDEX	erstellt einen Index
CREATE PROCEDURE	erstellt eine gespeicherte Prozedur
CREATE RULE	erstellt eine Regel (ein Objekt)
CREATE SCHEMA	erstellt ein Schema
CREATE STATISTICS	erstellt eine Statistik
CREATE TABLE	erstellt eine neue Tabelle
CREATE TRIGGER	erstellt einen Trigger
CREATE VIEW	erstellt eine Sicht
DEALLOCATE	entfernt einen Cursorverweis
DECLARE CURSOR	definiert einen Servercursor
DELETE	entfernt Zeilen einer Tabelle
DENY	lehnt eine Berechtigung ab
DROP DATABASE	löscht eine Datenbank
DROP DEFAULT	entfernt benutzerdefinierte Standardwerte
DROP INDEX	entfernt Indizes
DROP PROCEDURE	entfernt gespeicherte Prozeduren
DROP RULE	entfernt benutzerdefinierte Regeln
DROP STATISTICS	löscht Statistiken
DROP TABLE	entfernt eine Tabelle

DROP TRIGGER	entfernt Trigger
DROP VIEW	entfernt Sichten
EXECUTE	führt eine gespeicherte Prozedur oder eine Zeichenfolge aus
EXISTS	prüft, ob eine Unterabfrage Zeilen zurückgibt
FETCH	ruft eine Zeile aus einem Cursor ab
FORMATMESSAGE	gibt eine formatierte Meldung aus der Systemtabelle sysmessages zurück
FREETEXT	wird bei einer Fuzzy-Suche in Spalten mit Zeichendatentypen verwendet
FROM	spezifiziert Tabellen, Sichten, abgeleitete und verknüpfte Tabellen in DELETE-, SELECT- und UPDATE-Anweisungen
GRANT	gewährt eine Berechtigung
GROUP BY	gruppiert eine Tabelle
HAVING	spezifiziert eine Suchbedingung für eine Gruppe oder ein Aggregat
IN	testet, ob ein Wert in einer Liste enthalten ist
INSERT	fügt eine Zeile in eine Tabelle oder Sicht ein
IS [NOT] NULL	bestimmt, ob der angegebene Ausdruck NULL oder nicht NULL ist
KILL	bricht einen Prozeß ab
LIKE	führt einen Mustervergleich durch
LOAD	lädt eine Sicherungskopie
NOT	negiert einen booleschen Wert
OPEN	öffnet einen Server-Cursor und füllt ihn
OPENQUERY	führt eine Pass-Through-Abfrage aus
OPENROWSET	führt eine Pass-Through-Abfrage aus
OR	verknüpft zwei boolesche Ausdrücke und gibt TRUE zurück, wenn einer der beiden Ausdrücke TRUE ist (beachten Sie den Sonderfall UNKNOWN)
ORDER BY	sortiert die Ergebnisse einer SELECT-Anweisung

READTEXT	liest Text- oder Bildwerte aus einer text- oder image-Spalte
RECONFIGURE	aktualisiert einen geänderten Konfigurationswert
REPLACE	ersetzt eine Zeichenfolge durch eine andere
REPLICATE	wiederholt einen Zeichenfolge
RESTORE	stellt eine Datenbank, ein Transaktionsprotokoll oder bestimmte Dateien und Dateigruppen wieder her
RESTORE FILELISTONLY	gibt eine Liste der Datenbank- und Protokolldateien zurück, die im Sicherungssatz enthalten sind
RESTORE HEADERONLY	ruft Header-Informationen für Sicherungssätze auf einem Sicherungsmedium ab
RESTORE LABELONLY	liefert Informationen über Sicherungsmedien zurück
RESTORE VERIFYONLY	verifiziert eine Sicherung, stellt sie aber nicht wieder her
RETURN	beendet eine Abfrage oder führt einen Rücksprung aus einer Prozedur aus
REVERSE	gibt die Zeichenfolge in umgekehrter Reihenfolge zurück
REVOKE	entfernt eine Berechtigung
ROLLBACK TRANSACTION	macht Transaktionen rückgängig
ROLLBACK WORK	macht Transaktionen rückgängig
SAVE TRANSACTION	markiert einen Sicherungspunkt in einer Transaktion
SELECT	ruft Zeilen aus einer Datenbank ab
SET	setzt Werte für lokale Variablen oder legt Systemeinstellungen fest
SETUSER	ermöglicht das Übertragen von Benutzeridentitäten
SHUTDOWN	beendet SQL Server sofort
SOME	vergleicht einen Skalarwert mit den Werten einer Spalte

TRUNCATE TABLE	löscht den Inhalt einer Tabelle, nicht die Tabelle selbst
UNION	kombiniert die Ergebnisse mehrerer Abfragen
UPDATE	aktualisiert vorhandene Daten in einer Tabelle
UPDATE STATISTICS	aktualisiert Statistiken
UPDATETEXT	aktualisiert ein vorhandenes Feld des Datentyps text, ntext oder image (teilweise Änderung möglich)
USE	legt die aktuelle Datenbank fest
WHERE	legt die Bedingungen für eine Abfrage fest
WRITETEXT	aktualisiert ein vorhandenes Feld des Datentyps text, ntext oder image (vollständiges Überschreiben)

Tabelle C.1: Überblick über die Transact-SQL-Anweisungen

C.1.1 Ablaufsteuerung

Zum Sprachumfang von Transact-SQL gehören die in Tabelle C.2 genannten Anweisungen zur Ablaufsteuerung und Anweisungen, die Sie in diesem Zusammenhang einsetzen können.

Anweisung	Beschreibung
BEGIN...END	definiert einen Anweisungsblock
BREAK	bewirkt das Verlassen der innersten WHILE-Schleife
CASE	wertet eine Liste von Bedingungen aus und gibt den zutreffenden Ergebnisausdruck zurück
CONTINUE	setzt eine WHILE-Schleife am Schleifenanfang fort
DECLARE *@LokaleVariable*	deklariert eine lokale Variable
EXECUTE	führt eine gespeicherte Prozedur oder eine Zeichenfolge aus
GOTO *Sprungmarke*	setzt die Bearbeitung mit der auf *Sprungmarke* folgenden Anweisung fort
IF...ELSE	führt Anweisungen in Abhängigkeit von einem Testausdruck aus
PRINT	gibt eine benutzerdefinierte Meldung an den Client zurück
RAISERROR	gibt eine benutzerdefinierte Fehlermeldung aus

RETURN	unbedingter Abbruch bzw. Rücksprung aus einer gespeicherten Prozedur
WAITFOR	setzt eine Verzögerungsdauer oder einen Zeitpunkt für die Ausführung einer Anweisung
WHILE	wiederholt den Anweisungsblock im Rumpf der Schleifenkonstruktion, solange die Testbedingung TRUE ist
/* ... */ --	Kommentare (mehrzeilige und einzeilige)

Tabelle C.2: Transact-SQL-Anweisungen zur Ablaufsteuerung

C.2 Gespeicherte Systemprozeduren

Zur SQL-Server-Installation gehört eine umfangreiche Sammlung von gespeicherten Systemprozeduren. Da die Aufgaben einer Prozedur nicht immer ohne weiteres aus dem Namen ersichtlich sind, können Sie sich anhand der Kurzbeschreibungen in dieser Referenz orientieren, welche Prozedur für einen bestimmten Zweck in Frage kommt. Die folgenden Tabellen geben einen nach Kategorien geordneten Überblick über gespeicherte Systemprozeduren von SQL Server. Falls die Prozedur im Buch näher behandelt wird, ist auch ein Hinweis auf die Kapitelnummer enthalten.

Prozeduren, die SQL Server nur noch aus Gründen der Abwärtskompatibilität bereitstellt, wurden nicht berücksichtigt.

Katalogprozeduren

Katalogprozeduren implementieren ODBC-Datadictionaryfunktionen und schirmen ODBC-Anwendungen gegenüber Änderungen an den zugrundeliegenden Systemtabellen ab.

Ein *Datadictionary* beschreibt die Systemtabellen, die Datenbankobjekte und deren Struktur.

Name	Beschreibung
sp_column_privileges	ruft Spaltenprivilegien für eine Tabelle ab
sp_columns	ruft Spalteninformationen für Tabellen oder Sichten ab Siehe Kapitel 19
sp_databases	ruft eine Liste der verfügbaren Datenbanken ab
sp_fkeys	ruft Fremdschlüsselinformationen ab
sp_pkeys	ruft Primärschlüsselinformationen ab
sp_server_info	ruft Informationen zu SQL Server ab
sp_special_columns	gibt Spalten zurück, die eine Tabelle eindeutig definieren, und Spalten, die automatisch bei einer Wertänderung in einer Zeile aktualisiert werden

sp_sproc_columns	ruft Spalteninformationen für eine gespeicherte Prozedur ab
sp_statistics	ruft eine Liste mit allen Indizes einer Tabelle ab
sp_stored_procedures	ruft eine Liste der gespeicherten Prozeduren ab
sp_table_privileges	ruft eine Liste der Tabellenberechtigungen ab
sp_tables	ruft eine Liste der in einer FROM-Klausel zulässigen Objekte ab

Tabelle C.3: Katalogprozeduren

Cursor-Prozeduren

Cursor-Prozeduren geben Cursor-spezifische Informationen zurück.

Name	Beschreibung
sp_cursor_list	ruft Informationen zu geöffneten Cursorn ab
sp_describe_cursor	ruft Attribute eines Cursors ab
sp_describe_cursor_tables	ruft die Basistabellen ab, auf die ein Cursor verweist
sp_describe_cursor_columns	ruft Spaltenattribute eines Cursors ab

Tabelle C.4: Cursor-Prozeduren

SQL-Server-Agent-Prozeduren

Auf die Prozeduren dieser Kategorie greift der SQL-Server-Agent zurück, um geplante und ereignisgesteuerte Aktivitäten zu verwalten.

Name	Beschreibung
sp_add_alert	erstellt eine Warnung
sp_add_category	fügt dem Server eine Kategorie (Auftrag, Warnung oder Operator) hinzu
sp_add_job	fügt einen neuen Auftrag hinzu
sp_add_jobschedule	erstellt einen Terminplan für einen Auftrag
sp_add_jobserver	weist einem Auftrag einen Zielserver zu
sp_add_jobstep	fügt einem Auftrag einen Schritt hinzu
sp_add_notification	fügt eine Benachrichtigung für eine Warnung hinzu
sp_add_operator	fügt einen Operator als Empfänger für Warnungen und Aufträge hinzu
sp_add_targetservergroup	fügt eine Servergruppe hinzu

sp_delete_alert	löscht eine Warnung
sp_delete_category	löscht die Kategorie (Auftrag, Warnung oder Operator) auf dem aktuellen Server
sp_delete_job	löscht einen Auftrag
sp_delete_jobschedule	löscht einen Terminplan aus einem Auftrag
sp_delete_jobstep	löscht einen Schritt aus einem Auftrag
sp_delete_notification	löscht alle Benachrichtigungen für eine Warnung
sp_delete_operator	löscht einen Operator
sp_help_alert	ruft Informationen zu definierten Warnungen ab
sp_help_category	ruft Informationen zu Kategorien (Aufträge, Warnungen oder Operatoren) ab
sp_help_downloadlist	ruft alle Zeilen der Systemtabelle sysdownloadlist für den angegebenen Auftrag ab
sp_help_job	ruft Informationen zu Aufträgen ab
sp_help_jobhistory	erstellt einen chronologischen Bericht für geplante Aufträge
sp_help_jobschedule	ruft Termininformationen zu Aufträgen ab
sp_help_jobserver	ruft Server-Informationen zu einem Auftrag ab
sp_help_jobstep	ruft Informationen zu Auftragsschritten ab
sp_help_notification	ruft eine Liste der Warnungen für einen bestimmten Operator oder eine Liste der Operatoren für eine bestimmte Warnung ab
sp_help_operator	ruft Informationen zu Operatoren ab
sp_manage_jobs_by_login	löscht Aufträge eines Benutzers oder ordnet sie neu zu
sp_purge_jobhistory	löscht die Chronikdatensätze für einen Auftrag
sp_start_job	startet einen Auftrag sofort
sp_stop_job	beendet einen Auftrag sofort
sp_update_alert	aktualisiert eine Warnung
sp_update_category	ändert den Namen einer Kategorie (Warnungen, Aufträge, Operatoren)
sp_update_job	ändert Attribute eines Auftrags
sp_update_jobschedule	ändert Terminplaneinstellungen für einen Auftrag
sp_update_jobstep	ändert die Einstellung für einen Auftragsschritt

sp_update_notification	aktualisiert die Methode, nach der ein Operator über eine Warnung benachrichtigt wird
sp_update_operator	aktualisiert Informationen zu einem Operator

Tabelle C.5: SQL-Server-Agent-Prozeduren

Sicherheitsprozeduren

Die Prozeduren dieser Kategorie implementieren Funktionalität, die sich auf die Verwaltung der Sicherheit von SQL Server bezieht.

Name	Beschreibung
sp_addapprole	fügt eine neue Anwendungsrolle hinzu Siehe Kapitel 21
sp_addlogin	erstellt einen Benutzernamen für SQL-Server-Authentifizierung Siehe Kapitel 21
sp_addrole	erstellt eine neue SQL-Server-Rolle in der aktuellen Datenbank Siehe Kapitel 21
sp_addrolemember	fügt einer vorhandenen SQL-Server-Datenbankrolle ein Sicherheitskonto hinzu
sp_addsrvrolemember	fügt einer festen Server-Rolle einen Benutzernamen hinzu Siehe Kapitel 21
sp_approlepassword	ändert das Kennwort für eine Anwendungsrolle in der aktuellen Datenbank
sp_change_users_login	verknüpft einen Benutzer in der aktuellen Datenbank mit einem Benutzernamen
sp_changedbowner	ändert den Datenbankbesitzer Siehe Kapitel 21
sp_changeobjectowner	ändert den Objektbesitzer Siehe Kapitel 21
sp_dbfixedrolepermission	zeigt Berechtigungen für feste Datenbankrollen an Siehe Kapitel 21
sp_defaultdb	ändert die Standarddatenbank für einen Benutzernamen
sp_defaultlanguage	ändert die Standardsprache für einen Benutzernamen
sp_denylogin	verweigert Benutzern oder Gruppen von Windows NT den Zugriff auf SQL Server Siehe Kapitel 21

sp_dropapprole	löscht eine Anwendungsrolle aus der aktuellen Datenbank Siehe Kapitel 21
sp_droplogin	löscht einen SQL-Server-Benutzernamen Siehe Kapitel 21
sp_droprole	löscht eine SQL-Server-Rolle aus der aktuellen Datenbank Siehe Kapitel 21
sp_droprolemember	löscht ein Sicherheitskonto aus einer SQL-Server-Rolle in der aktuellen Datenbank
sp_dropserver	Löscht einen Server aus der Liste der Remote- und Verbindungsserver
sp_dropsrvrolemember	Löscht Benutzernamen (SQL Server/Windows NT) oder Gruppen (Windows NT) aus einer festen Server-Rolle Siehe Kapitel 21
sp_grantdbaccess	fügt ein Sicherheitskonto hinzu Siehe Kapitel 21
sp_grantlogin	gewährt einem Benutzer-/Gruppenkonto von Windows NT den Zugriff auf SQL Server per Windows-NT-Authentifizierung Siehe Kapitel 21
sp_helpfixedrole	ruft eine Liste der festen Datenbankrollen ab Kapitel 21
sp_helplogins	liefert Informationen zu Benutzernamen und den zugehörigen Benutzern in den Datenbanken Siehe Kapitel 21
sp_helpntgroup	ruft Informationen zu Windows-NT-Gruppen mit Konten in der aktuellen Datenbank ab
sp_helprole	ruft Informationen zu den Rollen in der aktuellen Datenbank ab Siehe Kapitel 21
sp_helprolemember	ruft Informationen zu den Mitgliedern einer Rolle in der aktuellen Datenbank ab Siehe Kapitel 21
sp_helprotect	ruft einen Bericht mit Berechtigungen für Objekte oder Anweisungen in der aktuellen Datenbank ab
sp_helpsrvrole	ruft eine Liste der festen Server-Rollen von SQL Server ab Siehe Kapitel 21
sp_helpsrvrolemember	ruft Informationen zu den Mitgliedern einer festen Server-Rolle von SQL Server ab Siehe Kapitel 21

sp_helpuser	ruft Informationen zu Benutzern (SQL Server/Windows NT) und Datenbankrollen in der aktuellen Datenbank ab Siehe Kapitel 21
sp_password	fügt einem Benutzernamen ein Kennwort hinzu oder ändert es Siehe Kapitel 21
sp_revokedbaccess	löscht ein Sicherheitskonto aus der aktuellen Datenbank Siehe Kapitel 21
sp_revokelogin	löscht Einträge von Benutzernamen in SQL Server für Gruppen oder Benutzer von Windows NT Siehe Kapitel 21
sp_setapprole	aktiviert Berechtigungen einer Anwendungsrolle Siehe Kapitel 21
sp_srvrolepermission	ruft die Berechtigungen für eine feste Server-Rolle ab Siehe Kapitel 21
sp_validatelogins	ruft eine Liste der verwaisten Benutzer und Gruppen ab, die nur noch in den Systemtabellen von SQL Server verzeichnet sind und nicht mehr in der Windows-NT-Umgebung existieren

Tabelle C.6: Sicherheitsprozeduren

Systemprozeduren

Die gespeicherten Prozeduren dieser Kategorie dienen allgemeinen Wartungsaufgaben in SQL Server.

Name	Beschreibung
sp_addmessage	nimmt in die Systemtabelle sysmessages eine neue Fehlermeldung auf Siehe Kapitel 19
sp_addtype	erstellt einen benutzerdefinierten Datentyp Siehe Kapitel 8
sp_addumpdevice	fügt ein Sicherungsmedium hinzu Siehe Kapitel 7
sp_altermessage	ändert den Status einer Fehlermeldung in der Systemtabelle sysmessages
sp_autostats	ruft die automatische UPDATE STATISTICS-Einstellung für Indizes oder Statistiken einer Tabelle in der aktuellen Datenbank ab oder ändert sie

sp_bindefault	bindet einen Standardwert an eine Spalte oder an einen benutzerdefinierten Datentyp Siehe Kapitel 16
sp_bindrule	bindet eine Regel an eine Spalte oder an einen benutzerdefinierten Datentyp Siehe Kapitel 16
sp_configure	ruft globale Konfigurationseinstellungen für den Server ab oder ändert die Einstellungen Siehe Kapitel 6
sp_createstats	erstellt Statistiken
sp_cycle_errorlog	schließt das aktuelle Fehlerprotokoll und numeriert die zugehörigen Dateien neu
sp_datatype_info	ruft Informationen zu den unterstützten Datentypen ab
sp_dboption	ruft Datenbankoptionen ab oder ändert sie Siehe Kapitel 6
sp_depends	zeigt Abhängigkeiten von Datenbankobjekten (z.B. Sichten und Tabellen) an Siehe Kapitel 13 Siehe Kapitel 17
sp_dropdevice	löscht ein Datenbank- oder Sicherungsmedium aus der Systemtabelle master.dbo.sysdevices
sp_dropextendedproc	löscht eine erweiterte gespeicherte Prozedur
sp_dropmessage	löscht eine Fehlermeldung aus der Systemtabelle sysmessages Siehe Kapitel 19
sp_droptype	löscht einen benutzerdefinierten Datentyp aus der Systemtabelle systypes
sp_executesql	führt eine Transact-SQL-Anweisung oder eine Zeichenfolge aus Siehe Kapitel 12 Siehe Anhang B
sp_fulltext_catalog	erstellt und löscht einen Volltextkatalog, startet und beendet die Indizierung eines Katalogs
sp_fulltext_column	legt für eine bestimmte Spalte einer Tabelle fest, ob sie bei der Volltextindizierung zu berücksichtigen ist
sp_fulltext_database	initialisiert die Volltextindizierung oder löscht Volltextkataloge

sp_fulltext_service	ändert Eigenschaften für die Volltextsuche
sp_fulltext_table	markiert eine Tabelle für die Volltextindizierung oder hebt die Markierung auf
sp_help	ruft Informationen zu Datenbankobjekten und Datentypen ab Siehe Kapitel 10 Siehe Kapitel 17
sp_helpconstraint	ruft Informationen zu Einschränkungstypen ab
sp_helpdb	ruft Informationen zu Datenbanken ab
sp_helpextendedproc	ruft Informationen zu erweiterten gespeicherten Prozeduren ab
sp_helpfile	ruft Informationen zu Dateien der aktuellen Datenbank ab
sp_helpfilegroup	ruft Informationen zu Dateigruppen der aktuellen Datenbank ab
sp_help_fulltext_catalogs	ruft Informationen zu volltextindizierten Tabellen ab
sp_help_fulltext_catalogs_ cursor	ruft Informationen zu volltextindizierten Tabellen mit einem Cursor ab
sp_help_fulltext_columns	ruft die für eine Volltextindizierung vorgesehenen Spalten ab
sp_help_fulltext_ columns_cursor	ruft die für eine Volltextindizierung vorgesehenen Spalten mit einem Cursor ab
sp_help_fulltext_tables	ruft die für eine Volltextindizierung vorgesehenen Tabellen ab
sp_help_fulltext_tables_ cursor	ruft die für eine Volltextindizierung vorgesehenen Tabellen mit einem Cursor ab
sp_helpindex	ruft Informationen zu den Indizes in einer Tabelle ab Siehe Kapitel 14
sp_helplanguage	ruft Informationen zu Sprachen ab
sp_helpserver	ruft Informationen zu einem Remoteserver und/oder Replikationsserver ab
sp_helpsort	ruft die Sortierreihenfolge und den Zeichensatz von SQL Server ab

sp_helptext	ruft den Text einer Regel, eines Standardwertes, einer gespeicherten Prozedur, eines Triggers oder einer Sicht ab Siehe Kapitel 17
sp_helptrigger	ruft die Triggertypen einer Tabelle der aktuellen Datenbank ab
sp_indexoption	setzt Indexoptionen
sp_lock	ruft Informationen zu Sperren ab Siehe Kapitel 18
sp_monitor	ruft Statistiken ab
sp_procoption	ruft Optionen für Prozeduren ab oder legt sie fest
sp_recompile	veranlaßt, daß gespeicherte Prozeduren und Trigger bei der nächsten Ausführung erneut kompiliert werden
sp_rename	ändert einen Objektnamen Siehe Kapitel 17
sp_renamedb	ändert einen Datenbanknamen Siehe Kapitel 6
sp_spaceused	ruft Informationen zum Platzbedarf für eine Tabelle oder Datenbank ab
sp_tableoption	setzt Optionen für Tabellen
sp_unbindefault	hebt die Bindung eines Standardwertes an eine Spalte oder an einen benutzerdefinierten Datentyp auf
sp_unbindrule	hebt die Bindung einer Regel an eine Spalte oder an einen benutzerdefinierten Datentyp auf Siehe Kapitel 16
sp_updatestats	führt UPDATE STATISTICS für alle Benutzertabellen der aktuellen Datenbank aus
sp_validname	prüft die Gültigkeit eines Bezeichners
sp_who	ruft Informationen zu Benutzern und Prozessen von SQL Server ab

Tabelle C.7: Systemprozeduren

Web-Assistent-Prozeduren

Die Prozeduren dieser Kategorie kommen in Verbindung mit dem Web-Assistenten zum Einsatz.

Name	Beschreibung

sp_dropwebtask	löscht einen Webauftrag
sp_enumcodepages	gibt die von sp_makewebtask unterstützten Codeseiten und Zeichensätze zurück
sp_makewebtask	erstellt einen Webauftrag für ein HTML-Dokument, das die Daten der ausgeführten Abfragen enthält Siehe Kapitel 28
sp_runwebtask	führt einen Webauftrag aus Siehe Kapitel 28

Tabelle C.8: Web-Assistent-Prozeduren

Allgemeine erweiterte Prozeduren

Die gespeicherten Prozeduren dieser Kategorie bieten eine Schnittstelle zwischen SQL Server und externen Programmen zur Wartung.

Name	Beschreibung
xp_cmdshell	führt einen Betriebssystembefehl aus
xp_enumgroups	listet die lokalen Windows-NT-Gruppen oder die globalen Gruppen, die in einer angegebenen Windows-NT-Domäne definiert sind, auf
xp_findnextmsg	übernimmt eine Nachrichten-ID und gibt die Kennung der nächsten Nachricht zurück
xp_logevent	schreibt eine benutzerdefinierte Nachricht in die SQL-Server-Protokolldatei und in die Windows-NT-Ereignisanzeige
xp_loginconfig	liefert die Konfiguration der Anmeldungssicherheit von SQL Server bei Ausführung unter Windows NT
xp_logininfo	liefert Angaben zu einem Benutzerkonto, über das der Zugriff auf SQL Server erfolgt
xp_msver	liefert Informationen zur Version von SQL Server
xp_sqlmaint	ruft das Dienstprogramm sqlmaint mit den (als Zeichenfolge) angegebenen Optionen auf Siehe Kapitel 20
xp_sprintf	formatiert und speichert eine Zeichenfolge in der Art der C-Funktion sprintf mit Werten und Platzhalterzeichen im Ausgabeparameter
xp_sscanf	übernimmt Daten in der Art der C-Funktion sscanf aus Zeichenfolgen

Tabelle C.9: Allgemeine erweiterte Prozeduren

Erweiterte SQL-Mail-Prozeduren

Auf die Prozeduren dieser Kategorie greift SQL Server bei E-Mail-Operationen zurück.

Name	Beschreibung
xp_deletemail	löscht eine eingegangene Nachricht
xp_readmail	liest eine Nachricht
xp_sendmail	Sendet eine Nachricht mit Anlage an die angegebenen Empfänger. Die Anlage enthält die Ergebnismenge einer Abfrage.
xp_startmail	startet eine SQL-Mail-Client-Sitzung
xp_stopmail	beendet eine SQL-Mail-Client-Sitzung
xp_findnextmsg	übernimmt eine Nachrichten-ID und gibt die Kennung der nächsten Nachricht zurück

Tabelle C.10: Erweiterte SQL-Mail-Prozeduren

C.3 DBCC-Anweisungen

Die Transact-SQL-Anweisungen zur Datenbankkonsistenzprüfung lassen sich folgenden Kategorien zuordnen:

- Verwaltungsanweisungen
- Verschiedene Anweisungen
- Statusanweisungen
- Überprüfungsanweisungen

Die folgenden Tabellen geben einen Überblick über diese Anweisungen.

Anweisung	Beschreibung
DBCC DBREINDEX	erstellt Indizes neu
DBCC SHRINKDATABASE	verkleinert Datenbanken
DBCC SHRINKFILE	verkleinert eine Datendatei oder Protokolldatei
DBCC UPDATEUSAGE	meldet und korrigiert Ungenauigkeiten in der Systemtabelle sysindexes

Tabelle C.11: Verwaltungsanweisungen

Anweisung	Beschreibung
DBCC dllname (FREE)	entfernt eine DLL für erweiterte gespeicherte Prozeduren aus dem Speicher
DBCC HELP	ruft die Syntax der angegebenen DBCC-Anweisung ab

DBCC PINTABLE	Markiert eine Tabelle als fixiert. SQL Server löscht die Seiten einer derartigen Tabelle nicht aus dem Speicher.
DBCC TRACEOFF	deaktiviert Ablaufverfolgungsflags
DBCC TRACEON	aktiviert Ablaufverfolgungsflags
DBCC UNPINTABLE	Markiert eine Tabelle als nicht fixiert. SQL Server kann die Seiten einer derartigen Tabelle aus dem Puffercache löschen.

Tabelle C.12: Verschiedene Anweisungen

Anweisung	Beschreibung
DBCC INPUTBUFFER	zeigt die letzte Anweisung vom Client an
DBCC OPENTRAN	zeigt Informationen zu offenen Transaktionen an
DBCC OUTPUTBUFFER	gibt den Inhalt des Ausgabepuffers in hexadezimalem Format und als ASCII-Zeichen aus
DBCC PROCCACHE	liefert Informationen zum Prozedurcache
DBCC SHOWCONTIG	liefert Informationen zur Fragmentierung von Daten und Indizes
DBCC SHOW_STATISTICS	zeigt statistische Informationen für eine Tabelle an
DBCC SQLPERF	liefert Statistiken zur Speicherplatznutzung der Transaktionsprotokolle aller Datenbanken
DBCC TRACESTATUS	ruft den Zustand von Ablaufverfolgungsflags ab
DBCC USEROPTIONS	ruft die aktuellen Einstellungen der SET-Optionen für die aktuelle Sitzung ab

Tabelle C.13: Statusanweisungen

Anweisung	Beschreibung
DBCC CHECKALLOC	überprüft Reservierung und Verwendung aller Seiten einer Datenbank
DBCC CHECKCATALOG	überprüft die Konsistenz von Systemtabellen einer Datenbank
DBCC CHECKDB	überprüft die Reservierung und strukturelle Integrität aller Objekte einer Datenbank
DBCC CHECKFILEGROUP	überprüft die Reservierung und strukturelle Integrität aller Tabellen der aktuellen Datenbank in einer Dateigruppe
DBCC CHECKIDENT	überprüft den aktuellen Identitätswert einer Tabelle und korrigiert ihn gegebenenfalls

DBCC CHECKTABLE	überprüft die Integrität von Daten-, Index, text-, ntext- und image-Seiten für eine Tabelle

Tabelle C.14: Überprüfungsanweisungen

C.4 Reservierte Schlüsselwörter

C.4.1 Transact-SQL

In dem von SQL Server verwendeten SQL-Dialekt Transact-SQL sind die folgenden Schlüsselwörter reserviert. Im allgemeinen wird empfohlen, diese Schlüsselwörter nicht als Bezeichner oder Objektnamen zu verwenden. Andernfalls sind diese Schlüsselwörter in Begrenzer einzuschließen.

ADD	AVG	CASCADE
ALL	BACKUP	CASE
ALTER	BEGIN	CHECK
AND	BETWEEN	CHECKPOINT
ANY	BREAK	CLOSE
AS	BROWSE	CLUSTERED
ASC	BULK	COALESCE
AUTHORIZATION	BY	COLUMN
COMMIT	FESTPLATTE	GOTO
COMMITTED	DISTINCT	GRANT
COMPUTE	DISTRIBUTED	GROUP
CONFIRM	DOUBLE	HAVING
CONSTRAINT	DROP	HOLDLOCK
CONTAINS	DUMMY	IDENTITY
CONTAINSTABLE	DUMP	IDENTITY_INSERT
CONTINUE	ELSE	IDENTITYCOL
CONTROLROW	ENDE	IF
CONVERT	ERRLVL	IN
COUNT	ERROREXIT	INDEX
CREATE	ESCAPE	INNER
CROSS	EXCEPT	INSERT

CURRENT	EXEC	INTERSECT
CURRENT_DATE	EXECUTE	INTO
CURRENT_TIME	EXISTS	IS
CURRENT_TIMESTAMP	EXIT	ISOLATION
CURRENT_USER	FETCH	JOIN
CURSOR	FILE	KEY
DATABASE	FILLFACTOR	KILL
DBCC	FLOPPY	LEFT
DEALLOCATE	FOR	LEVEL
DECLARE	FOREIGN	LIKE
DEFAULT	FREETEXT	LINENO
DELETE	FREETEXTTABLE	LOAD
DENY	FROM	MAX
DESC	FULL	MIN
MIRROREXIT	PREPARE	SESSION_USER
NATIONAL	PRIMARY	SET
NOCHECK	PRINT	SETUSER
NONCLUSTERED	PRIVILEGES	SHUTDOWN
NOT	PROC	SOME
NULL	PROCEDURE	STATISTICS
NULLIF	PROCESSEXIT	SUM
OF	PUBLIC	SYSTEM_USER
OFF	RAISERROR	TABLE
OFFSETS	READ	BAND
ON	READTEXT	TEMP
ONCE	RECONFIGURE	TEMPORARY
ONLY	REFERENCES	TEXTSIZE
OPEN	REPEATABLE	THEN
OPENDATASOURCE	REPLICATION	TO
OPENQUERY	RESTORE	TOP

OPENROWSET	RESTRICT	TRAN
OPTION	RETURN	TRANSACTION
OR	REVOKE	TRIGGER
ORDER	RIGHT	TRUNCATE
OUTER	ROLLBACK	TSEQUAL
OVER	ROWCOUNT	UNCOMMITTED
PERCENT	ROWGUIDCOL	UNION
PERM	RULE	UNIQUE
PERMANENT	SAVE	UPDATE
PIPE	SCHEMA	UPDATETEXT
PLAN	SELECT	USE
PRECISION	SERIALIZABLE	USER
VALUES	WHEN	WORK
VARYING	WHERE	WRITETEXT
VIEW	WHILE	
WAITFOR	WITH	

Tabelle C.15: Reservierte Schlüsselwörter in Transact-SQL

C.4.2 ODBC

ABSOLUTE	BY	CONTINUE
ACTION	CASCADE	CONVERT
ADA	CASCADED	CORRESPONDING
ADD	CASE	COUNT
ALL	CAST	CREATE
ALLOCATE	CATALOG	CROSS
ALTER	CHAR	CURRENT
AND	CHAR_LENGTH	CURRENT_DATE
ANY	CHARACTER	CURRENT_TIME
ARE	CHARACTER_LENGTH	CURRENT_TIMESTAMP
AS	CHECK	CURRENT_USER

ASC	CLOSE	CURSOR
ASSERTION	COALESCE	DATE
AT	COLLATE	DAY
AUTHORIZATION	COLLATION	DEALLOCATE
AVG	COLUMN	DEC
BEGIN	COMMIT	DECIMAL
BETWEEN	CONNECT	DECLARE
BIT	CONNECTION	DEFAULT
BIT_LENGTH	CONSTRAINT	DEFERRABLE
BOTH	CONSTRAINTS	DEFERRED
DELETE	FOUND	ISOLATION
DESC	FROM	JOIN
DESCRIBE	FULL	KEY
DESCRIPTOR	GET	LANGUAGE
DIAGNOSTICS	GLOBAL	LAST
DISCONNECT	GO	LEADING
DISTINCT	GOTO	LEFT
DOMAIN	GRANT	LEVEL
DOUBLE	GROUP	LIKE
DROP	HAVING	LOCAL
ELSE	HOUR	LOWER
ENDE	IDENTITY	MATCH
END-EXEC	IMMEDIATE	MAX
ESCAPE	IN	MIN
EXCEPT	INCLUDE	MINUTE
EXCEPTION	INDEX	MODULE
EXEC	INDICATOR	MONTH
EXECUTE	INITIALLY	NAMES
EXISTS	INNER	NATIONAL
EXTERNAL	INPUT	NATURAL

EXTRACT	INSENSITIVE	NCHAR
FALSE	INSERT	NEXT
FETCH	INT	NO
FIRST	INTEGER	NONE
FLOAT	INTERSECT	NOT
FOR	INTERVAL	NULL
FOREIGN	INTO	NULLIF
FORTRAN	IS	NUMERIC
OCTET_LENGTH	RESTRICT	TEMPORARY
OF	REVOKE	THEN
ON	RIGHT	TIME
ONLY	ROLLBACK	TIMESTAMP
OPEN	ROWS	TIMEZONE_HOUR
OPTION	SCHEMA	TIMEZONE_MINUTE
OR	SCROLL	TO
ORDER	SECOND	TRAILING
OUTER	SECTION	TRANSACTION
OUTPUT	SELECT	TRANSLATE
OVERLAPS	SESSION	TRANSLATION
PAD	SESSION_USER	TRIM
PARTIAL	SET	TRUE
PASCAL	SIZE	UNION
POSITION	SMALLINT	UNIQUE
PRECISION	SOME	UNKNOWN
PREPARE	SPACE	UPDATE
PRESERVE	SQL	UPPER
PRIMARY	SQLCA	USAGE
PRIOR	SQLCODE	USER
PRIVILEGES	SQLERROR	USING
PROCEDURE	SQLSTATE	VALUE

PUBLIC	SQLWARNING	VALUES
READ	SUBSTRING	VARCHAR
REAL	SUM	VARYING
REFERENCES	SYSTEM_USER	VIEW
RELATIVE	TABLE	WHEN
WHENEVER	WORK	ZONE
WHERE	WRITE	
WITH	YEAR	

Tabelle C.16: Reservierte Schlüsselwörter für ODBC

Anhang D Die CD zum Buch

D.1 SQL Server

Die 120-Tage-Testversion von SQL Server 7.0 ist auf der CD im Verzeichnis \MSSQL7 untergebracht. Da sich die Datei Autorun.exe nicht im Stammverzeichnis befindet, startet die Installation nach Einlegen der CD nicht automatisch. Um SQL Server zu installieren, führen Sie die Datei Autorun.exe über Start / Ausführen aus. Klicken Sie im Dialogfeld Ausführen auf Durchsuchen, wechseln Sie in das Verzeichnis \MSSQL7 Ihres CD-Laufwerkes, und öffnen Sie die Datei Autorun.exe. Nachdem Sie im Dialogfeld Ausführen auf OK geklickt haben, erscheint der Startbildschirm für die Installation von SQL Server 7.0. Die weitere Installation läuft nach den gleichen Schritten ab, wie sie Kapitel 2 beschrieben hat. Beachten Sie aber, daß die Testversion nicht den gesamten Leistungsumfang des Vollprodukts bietet.

D.2 Beispieldateien

Das Verzeichnis \SQLKomp\ enthält Skripts für die im Buch angegebenen Transact-SQL-Beispiele. Es wurde bewußt darauf verzichtet, sämtliche Beispiele auf der CD bereitzustellen. Die meisten Beispiele sind ohnehin sehr kurz und einfach gehalten. Vor allem aber werden Sie sich die Befehle besser einprägen können, wenn Sie die Programmzeilen selbst eintippen und von Anfang an wissen, worauf Sie achten müssen. Die Datei Readme.txt im Verzeichnis \SQLKomp enthält eine Liste der ausgewählten Beispiele. Die Skripts zu den einzelnen Kapiteln sind in Unterverzeichnissen von \SQLKomp mit den Namen Kxx zu finden, wobei xx für die zweistellige Kapitelnummer steht.

Für die Anwendung aus Kapitel 27 sind alle erforderlichen Projektdateien im Verzeichnis \SQLKomp\VCPP\ untergebracht.

Die in verschiedenen Kapiteln des Buches verwendete Datenbank Lotto finden Sie zusammen mit der Textdatei für die Quelldaten und einer Reihe von SQL-Skripts im Verzeichnis \SQLKomp\Lotto.

Um die Dateien auf Ihrer Festplatte zu installieren, legen Sie zum Beispiel das Verzeichnis D:\SQLKomp an, wechseln in dieses Verzeichnis und kopieren alle Dateien einschließlich aller Unterverzeichnisse aus dem Verzeichnis \SQLKomp der Begleit-CD in das neue Verzeichnis.

Weitere Informationen entnehmen Sie bitte der Datei Readme.txt.
