

DB2 Administration

IBM Software Press

In Zusammenarbeit mit der IBM Deutschland GmbH verlegt Addison-Wesley diese Reihe professioneller Bücher. Die Reihe deckt die Software WebSphere, DB2, Tivoli und Lotus ab. Die Funktionalität, Tools, Anwendungen und Entwicklungsumgebungen der jeweiligen Software werden von Autoren beschrieben, die sich beruflich mit der Software befassen.



Erstellen Sie effiziente J2EE-Anwendungen auf Basis der IBM WebSphere-Produktfamilie. Die Autoren führen Sie zunächst in die J2EE-Architektur ein. Darauf aufbauend werden Grundlagen von Servlets, JSP, JDBC und JavaBeans behandelt. Ein ausführlicher Praxisteil zeigt Ihnen, wie Sie die Theorie in die Praxis umsetzen.



Das Buch stellt den WebSphere-Applikationsserver und den WebSphere Studio Application Server aus Sicht des Administrators vor. Nach einer Übersicht über die WebSphere-Anwendungen erläutert der Autor Installation und Konfiguration sowie verschiedene Tools. An einer Beispielenwicklung wird die WebSphere-Praxis verdeutlicht.

**Jens Orhanovic
Ivo Grodtke
Michael Tiefenbacher**

DB2 Administration

Einführung, Handbuch und Referenz

 **ADDISON-WESLEY**

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <<http://dnb.ddb.de>> abrufbar.

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

Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

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

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

Umwelthinweis:

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

10 9 8 7 6 5 4 3 2 1

07 06 05 04

ISBN 3-8273-2080-1

© 2004 by Addison-Wesley Verlag, ein Imprint der
Pearson Education Deutschland GmbH
Martin-Kollar-Straße 10–12, D-81829 München/Germany
Alle Rechte vorbehalten

Einbandgestaltung:	Barbara Thoben, Köln
Lektorat:	Rainer Fuchs, rfuchs@pearson.de
Korrektorat:	mediaService, Siegen
Herstellung:	Monika Weiher, mweiher@pearson.de
Satz und Layout:	mediaService, Siegen
Druck und Verarbeitung:	Kösel, Kempten (www.KoeselBuch.de)

Printed in Germany

Inhaltsverzeichnis

V	Vorwort	15
V.1	Inhalt	17
V.2	Ziel und Aufbau des Buchs	19
V.3	Die Autoren	19
V.4	Die Datenmodelle	20
V.4.1	Die Sample-Datenbank von DB2 UDB	20
V.4.2	Die Datenbank BUCHDB	21
V.5	Danksagungen	22
1	Einführung in Datenbanken und Datenbanksysteme	23
1.1	Datenbankgrundlagen	23
1.2	Historie der relationalen Datenbanksysteme	23
1.2.1	Die Codd'schen Regeln	23
1.2.2	Historie der DB2	25
1.3	Nutzung von relationalen Datenbankmanagementsystemen	25
1.3.1	Aufgaben des Modellierers	26
1.3.2	Aufgaben des Programmierers	27
1.3.3	Aufgaben des Administrators	28
1.3.4	Aufgaben der Endbenutzer	30
1.4	Was ist eigentlich eine Datenbank?	31
1.4.1	Definition einer Datenbank	31
1.4.2	Ausführliche Beschreibung einer Datenbank	31
1.5	Unterschiedliche Einsatzgebiete	37
1.5.1	DB2 im Einzelplatz-Einsatz auf einem Standalone-Rechner	37
1.5.2	DB2 im homogenen Netzwerkbetrieb	38
1.5.3	DB2 im heterogenen Netzwerkbetrieb	39
1.5.4	DB2 im Internetbetrieb	40
1.5.5	Produktkomponenten der unterschiedlichen Einsatzgebiete – Gesamtübersicht	41
2	Die DB2 UDB-Produktpalette	45
2.1	Übersicht über die DB2-Datenbanksystem-Produkte	45
2.1.1	DB2 UDB Everyplace Edition	47
2.1.2	DB2 UDB Personal Edition (DB2 PE)	49
2.1.3	DB2 UDB Express Edition	50
2.1.4	DB2 UDB Workgroup Server Edition (DB2 WSE bzw. WSUE)	51
2.1.5	DB2 UDB Enterprise Server Edition (DB2 ESE)	53
2.1.6	Zusammenfassung	54
2.2	Weitere DB2-Produktkomponenten	54
2.2.1	DB2 Connect	54
2.2.2	Entwicklerlizenzen	56
2.2.3	Relationale Extender	56
2.2.4	DB2 Data Links Manager	59

2.3	Zusätzliche DB2-Information-Management-Produkte	60
2.3.1	DB2 Warehouse Manager	60
2.3.2	DB2 OLAP Server	61
2.3.3	Die Produktfamilie DB2 Information Integrator	62
2.3.4	DB2 Intelligent Miner for Data	63
2.3.5	DB2 Intelligent Miner Modelling	63
2.3.6	DB2 Intelligent Miner Scoring	64
2.3.7	DB2 Intelligent Miner Visualization	64
2.3.8	Multiplattform-Tools	64
2.4	Die unterschiedlichen DB2-Produkte und -Produktkomponenten	65
3	Installation der DB2 UDB.....	67
3.1	Was ist ein DB2-Exemplar (instance)?	67
3.2	Was ist der DB2-Verwaltungsserver (DAS)?	69
3.3	Was ist der Exemplar-Eigner bzw. Fenced-Benutzer?.....	70
3.4	Installationsmöglichkeiten der DB2 UDB V8	70
3.5	Installation der DB2 UDB über die grafische Installationsroutine am Beispiel von Windows.....	73
3.6	Installation der DB2 UDB mit der Response-File-Methode unter dem Betriebssystem Linux	88
3.6.1	Grundvoraussetzungen für die Installation	89
3.6.2	Vorüberlegungen.....	90
3.6.3	Die eigentliche Installation	92
3.7	Schritte nach der Installation	93
3.8	In welchen Verzeichnissen finden Sie die installierten DB2 UDB-Komponenten?.....	94
3.9	Welche Befehle gibt es, um ein Exemplar zu verwalten?	95
3.9.1	Allgemeines zum Anlegen eines Exemplars	95
3.9.2	Anlegen eines Exemplars unter Linux	96
3.9.3	Anlegen eines Exemplars unter Windows	98
3.9.4	Der Befehl DB2ICRT	99
3.9.5	Starten und Stoppen eines Exemplars	101
3.10	Benutzer für DB2 im Unix-Umfeld einrichten	102
3.11	Die DB2-Befehlszeile (CLP)	102
3.11.1	Interaktiv oder nicht?.....	103
3.11.2	Optionen der DB2-Befehlszeile	107
3.11.3	DB2-Scripts.....	109
3.12	Die Exemplar-Konfiguration	110
3.12.1	Abfrage der Exemplar-Konfiguration	110
3.12.2	Ändern der Exemplar-Konfiguration.....	111
3.13	DAS und DB2-Toolskatalog manuell erstellen	112
3.13.1	Erstellen eines DB2-Verwaltungsservers (DAS)	112
3.13.2	Erstellen eines DB2-Toolskatalogs.....	113

3.14	DB2-Umgebungsvariablen anpassen	115
3.15	Tipps aus der Praxis	118
3.15.1	DB2LEVEL	118
3.15.2	Multifixpack-Installation	118
4	Grafische Werkzeuge	123
4.1	Überblick	123
4.2	Steuerzentrale	125
4.2.1	Das Kontextmenü	128
4.2.2	Assistenten	128
4.2.3	Ansprechpartner	130
4.2.4	Tools-Einstellungen	130
4.2.5	Tipps	132
4.3	Befehlszentrale	134
4.3.1	Interaktiv	134
4.3.2	Prozedur	136
4.3.3	Zugriffsplan	137
4.4	Taskzentrale	137
4.5	Journal	142
4.5.1	Taskprotokoll	142
4.5.2	Datenbankprotokoll	143
4.5.3	Nachrichten	144
4.5.4	Benachrichtigungsprotokoll	144
4.6	Diagnosezentrale	146
4.7	Replikationszentrale	146
4.8	Satellitenverwaltungszentrale	149
4.9	Data Warehouse-Zentrale	149
4.10	Informationskatalogzentrale	153
4.11	Entwicklungszentrale	153
4.12	Lizenzzentrale	155
4.13	Information – Unterstützung	155
4.14	Konfigurationsassistent	158
4.15	Webtools	160
5	Eine Datenbank erstellen	161
5.1	Überlegungen zur Anzahl der Datenbanken	161
5.2	Komponenten einer Datenbank	162
5.2.1	Datenbankkatalog	162
5.2.2	Datenbankpartitionsgruppe	163
5.2.3	Tabellenbereiche	163
5.2.4	Pufferpool	163
5.2.5	Protokolldateien	164

5.3	Wie erstelle ich eine Datenbank?	164
5.3.1	Vorüberlegungen.....	164
5.3.2	Wie erstelle ich eine Datenbank mit dem GUI?.....	165
5.3.3	Das Anlegen einer Datenbank per DB2-Befehl.....	169
5.4	Interne Abläufe beim Anlegen einer Datenbank.....	169
5.5	Aktivieren und Deaktivieren einer Datenbank	172
5.5.1	Verbindung zu einer Datenbank herstellen.....	172
5.5.2	Aktive Datenbanken und Verbindungen anzeigen.....	173
5.5.3	Datenbankverbindungen beenden.....	174
5.6	Die Datenbankkonfiguration	176
5.6.1	Anzeige der Datenbankkonfiguration	176
5.6.2	Ändern der Datenbankkonfiguration	178
5.6.3	Der Konfigurationsadvisor.....	180
5.7	Partitionsgruppen.....	182
5.8	Die Tabellenbereiche.....	183
5.8.1	Der Aufbau von Tabellenbereichen	183
5.8.2	Vom Betriebssystem verwaltete Tabellenbereiche – SMS	189
5.8.3	Vom Datenbankmanager verwaltete Tabellenbereiche – DMS	190
5.8.4	Wie erstelle ich einen Tabellenbereich?	192
5.8.5	Anzeige von Tabellenbereichen und Behältern.....	199
5.8.6	Administration von Tabellenbereichen	203
5.9	Tipps aus der Praxis.....	206
5.9.1	Welche Art Tabellenbereich soll ich nun verwenden?	206
5.9.2	Der Standard-Tabellenbereich.....	207
5.9.3	Der Datenbank-Wartungsmodus.....	209
5.9.4	Unterstützung von RAID-Systemen	209
5.9.5	Das Tool db2empfa	211
5.9.6	Der Befehl db2relocate	212
6	Aufbau verschiedener Datenbankverbindungen	213
6.1	Konzept: DRDA	213
6.2	Die lokale Datenbankverbindung	214
6.2.1	Das Systemdatenbankverzeichnis.....	215
6.2.2	Das lokale Datenbankverzeichnis	217
6.2.3	Schaubild einer lokalen Datenbankverbindung	218
6.3	Authentifizierung	219
6.3.1	Server-Authentifizierung	220
6.3.2	Client-Authentifizierung	221
6.3.3	Kerberos-Authentifizierung	223
6.3.4	Authentifizierungskonfiguration auf dem Client	224

6.4	Datenbankverbindung zu einem entfernten DB2-Datenbank-Server	224
6.4.1	Vorbereitung eines DB2-Servers	228
6.4.2	Konfiguration des Clients mit der grafischen Oberfläche	229
6.4.3	Konfiguration des Clients durch Befehle	240
6.4.4	Schaubild einer Verbindung zu einer entfernten Datenbank.....	245
6.4.5	Konfiguration des Clients über Profile	246
6.5	Verbindungen über DB2 Connect	255
6.6	Zweiphasige Festschreibung (two-phase commit)	256
6.6.1	Aktivierung der zweiphasigen Festschreibung	258
6.7	Verbindungen zu anderen Datenbank-Servern	259
6.7.1	Das „Federated“-Konzept	259
6.7.2	Zugriff auf eine andere DB2-Datenbank	260
6.8	Tipps aus der Praxis	264
6.8.1	Fehlerdiagnose für den Verbindungsaufbau	264
6.8.2	Passwortänderung vom Client	265
6.8.3	LDAP – Lightweight Directory Access Protocol	266
7	Datenbankobjekte	267
7.1	Schemata (Schema)	267
7.2	Tabellen	269
7.2.1	Anlegen von Tabellen	270
7.2.2	Temporäre Tabellen (global temporary tables).....	276
7.2.3	Gespeicherte Abfragetabelle (Materialized Query Table – MQT)	277
7.2.4	Anzeige der Tabellen	280
7.3	Sichten (Views)	282
7.4	Aliasnamen (Alias)	284
7.5	Indizes	284
7.5.1	Eindeutiger Index (unique index).....	289
7.5.2	Clustered Index	289
7.5.3	Informationen zu den Indizes anzeigen lassen.....	292
7.6	Prüfung auf Integritätsbedingung (Check Constraint).....	293
7.7	Referenzielle Integrität (Referential Integrity – RI).....	295
7.7.1	Kochbuch RI	299
7.8	Auslöser (Trigger)	300
7.9	Benutzerdefinierter Datentyp (UDT)	305
7.9.1	Benutzerdefinierter einzigartiger Datentyp (User Defined Distinct Type)	305
7.9.2	Benutzerdefinierter Strukturtyp (User Defined Structure Type).....	306
7.10	Benutzerdefinierte Funktion (User Defined Function – UDF)	307
7.11	Gespeicherte Prozeduren (Stored Procedures)	309

7.12	Mehrdimensionale Cluster-Tabelle (MDC)	311
7.13	Pufferpools (bufferpool)	315
7.13.1	Wie viele Pufferpools sollen angelegt werden?	317
7.13.2	Wie groß soll ein Pufferpool werden?	317
7.14	Pakete (packages)	318
7.15	Die Beispiel-Datenbank SAMPLE	319
7.16	Tipps aus der Praxis	319
7.16.1	Das Tool db2look	319
7.16.2	Der Designadvisor	321
8	Datenbewegungen	325
8.1	Datenformate	325
8.2	Export-Tool	327
8.2.1	Export über die Steuerzentrale	328
8.2.2	Exportieren per DB2-Befehl	331
8.3	Import-Tool	332
8.3.1	Import per DB2-Befehl	333
8.3.2	Grafische Unterstützung des Imports	337
8.4	Lade-Tool	340
8.4.1	Phasen des Ladevorgangs	341
8.4.2	Vorbereitung des Ladeprozesses	342
8.4.3	Grafische Unterstützung mit dem Assistent: Laden	344
8.4.4	Der LOAD-Befehl	352
8.4.5	Aktivitäten während des Ladevorgangs	353
8.4.6	Aktivitäten nach dem Laden	357
8.4.7	Aktivitäten nach einem Fehler im Ladeprozess	363
8.5	Tipps aus der Praxis	365
8.5.1	Übersicht und Vergleich von Import und Load	365
8.5.2	DB2MOVE-Tool	365
9	Sicherung und Wiederherstellung einer Datenbank	369
9.1	Protokollierung in einer Datenbank	369
9.1.1	Arten von Protokolldateien	371
9.1.2	Umlaufprotokollierung (circular logging)	372
9.1.3	Beibehalten von Protokolldateien (Logretain=YES)	373
9.1.4	Wie stellen Sie von Umlaufprotokollierung auf Archivprotokollierung um?	374
9.1.5	Welche Protokollierungsparameter gibt es?	380
9.1.6	Wie und wo werden Protokolldateien angelegt?	384
9.2	Sicherungs- und Wiederherstellungskonzepte	385
9.2.1	Sicherungskonzepte	385
9.2.2	Wiederherstellungskonzepte	388

9.3	Wie sichern Sie eine Datenbank?	389
9.3.1	Dateistruktur eines Sicherungsimages	390
9.3.2	Online- und Offline-Modus	392
9.3.3	Datensicherung mit der Steuerzentrale	393
9.3.4	Sicherung mit DB2-Befehlen	401
9.3.5	Was wird gesichert?	402
9.3.6	Liegt eine Online- oder eine Offline-Sicherung vor?	403
9.3.7	Tabellenbereichssicherung	405
9.4	Wie führt man eine Wiederherstellung durch?	407
9.4.1	Was ist ein Datenbank-Ursprungspunkt (database seed)?	408
9.4.2	Welche Daten werden zur Wiederherstellung benötigt?	409
9.4.3	Wiederherstellen einer Datenbank mit Hilfe des GUI	413
9.4.4	Wiederherstellen einer Datenbank mit DB2-Befehlen	417
9.4.5	Aktualisierende Wiederherstellung (rollforward)	421
9.4.6	Wann werden Protokolldateien überschrieben?	426
9.4.7	Umgeleitete Wiederherstellung (redirected restore)	427
9.5	Tipps aus der Praxis	430
9.5.1	Der Datenbankparameter blk_log_dsk_ful	430
9.5.2	DB2-Tool db2ckbkp	431
9.5.3	Benutzerexit (Userexit)	431
9.5.4	Protokollierung ausschalten	434
9.5.5	Verändern von Kontrollinformationen (db2relocatedb)	435
10	Zugriffsverwaltung in DB2	437
10.1	Allgemeines zu DB2-Berechtigungen	437
10.2	Zugriffsrechte innerhalb einer Datenbank	438
10.2.1	Zugriffsrechte für Datenbanken	438
10.2.2	Zugriffsrechte für Datenbankobjekte	440
10.2.3	Erteilen und Widerrufen von Zugriffsrechten	441
10.3	Berechtigungsstufen in DB2	448
11	Programmierschnittstellen	453
11.1	Aufgabengebiete der Programmierung	453
11.1.1	Programme mit SQL-Anweisungen	454
11.1.2	Programme zur Wartung der Datenbank und des Datenbanksystems	454
11.2	Programmiermethoden für SQL-Anweisungen	455
11.2.1	Embedded-SQL	456
11.2.2	Nicht-Embedded-SQL	457
11.3	Eigenheiten der Abfragesprache SQL	457

11.4	Statisches SQL.....	459
11.4.1	Wann ist eine SQL-Anweisung statisch?	459
11.4.2	Wie programmieren Sie statisches SQL?	460
11.4.3	Statisches SQL am Beispiel der Wirtssprache C	461
11.4.4	Wie kommen Sie zum ausführbaren C-Programm?	467
11.4.5	Statisches SQL am Beispiel der Wirtssprache Java	473
11.4.6	Wie kommen Sie zum ausführbaren Java-Programm?	480
11.4.7	Zusammenfassung der Abläufe	485
11.4.8	Auswirkungen der Programmierung mit statischem SQL.....	485
11.5	Dynamisches SQL.....	489
11.5.1	Wann muss eine SQL-Anweisung dynamisch sein?.....	489
11.5.2	Welche Arten von dynamischen SQL-Anweisungen gibt es?.....	489
11.5.3	Der Grund für die Unterteilung in unterschiedliche Arten.....	490
11.5.4	Embedded-dynamisches SQL am Beispiel der Wirtssprache C	491
11.5.5	Wie kommen Sie zum ausführbaren C-Programm?	494
11.5.6	Auswirkung der Programmiermethode Embedded-dynamisches SQL.....	495
11.6	Die weiteren Nicht-Embedded-SQL-Schnittstellen	496
11.6.1	Die ODBC-Schnittstelle	496
11.6.2	Die CLI-Schnittstelle.....	497
11.6.3	Die OLE DB-Schnittstelle	497
11.6.4	Die ADO-Schnittstelle	497
11.6.5	Die JDBC-Schnittstelle	498
11.6.6	Zusammenfassung.....	498
11.7	Sicherheit.....	498
11.7.1	Zugriffspakete der DB2 UDB	502
12	Konkurrierende Verarbeitung	505
12.1	Welche Problemfälle unterscheidet DB2?.....	506
12.1.1	Was sind Phantom Reads (PR)?	506
12.1.2	Was sind nicht wiederholbare Lesevorgänge (NWL)?	507
12.1.3	Was ist ein Lost Update (LU)?.....	508
12.1.4	Was ist ein Zugriff auf nicht bestätigte Daten (ZNBD)?	508
12.2	Welche Lösungen werden angeboten?	509
12.2.1	Repeatable Read (RR).....	509
12.2.2	Read Stability (RS).....	509
12.2.3	Cursor Stability (CS)	510
12.2.4	Uncommitted Read (UR).....	510
12.3	Detailbeschreibung	510
12.3.1	Was wird bei der Verwendung von RR gesperrt?	511
12.3.2	Was wird bei der Verwendung von RS gesperrt?	512
12.3.3	Was wird bei der Verwendung von CS gesperrt?.....	512
12.3.4	Was wird bei der Verwendung von UR gesperrt?.....	512

12.4	Welche Datenbankobjekte werden gesperrt?	512
12.5	Wie werden die Datenbankobjekte gesperrt?	512
12.5.1	Sperrflags für die Datenbankobjekte Tabelle und Tabellenbereiche	513
12.5.2	Sperrflags für Datenbankobjekte und Tabellenzeilen	515
12.6	Lock-Mode-Kompatibilität	516
12.7	Verschiedene Sperrsituationen	516
12.7.1	Deadlock	516
12.7.2	Lock-Timeout	519
12.7.3	Verhinderung von Schreibkonflikten	521
12.8	Konfiguration der Lock-Parameter	522
12.8.1	LOCK_LIST	522
12.8.2	MAXLOCKS	523
13	Pflege und Wartung von Datenbanken	525
13.1	Pflege von Datenbanken	525
13.1.1	Runstats	525
13.1.2	Reorgchk	531
13.1.3	Reorganisation von Tabellendaten	534
13.1.4	Reorganisation von Indizes	537
13.1.5	Zusammenspiel und Abhängigkeiten	539
13.2	Wartung und Problemanalyse	541
13.2.1	Benachrichtigungsprotokoll für die Systemverwaltung (notification log)	541
13.2.2	Diagnoseprotokoll db2diag.log	543
13.2.3	Integritätsprüfung mit dem INSPECT-Befehl	545
13.2.4	Problemmeldungen und Hilfestellungen	548
14	Überwachung von DB2	553
14.1	Warum ist Überwachung (Monitoring) notwendig?	553
14.2	Überwachungs-Tools in DB2	553
14.3	Der DB2-Systemmonitor	555
14.4	Der Snapshot-Monitor	556
14.4.1	Welche Daten sollen gesammelt werden?	556
14.4.2	Wohin sollen die Monitordaten geschrieben werden?	559
14.4.3	Wie führe ich den Snapshot-Monitor per Befehl aus?	559
14.4.4	Wie führen Sie den Snapshot-Monitor über Tabellen-Funktionen aus?	563
14.5	Der Ereignismonitor	566
14.5.1	Welche Daten sollen gesammelt werden?	567
14.5.2	Wohin sollen die Monitordaten geschrieben werden?	568
14.5.3	Wie aktiviere ich einen Ereignismonitor?	571
14.5.4	Wie stoppe ich einen Ereignismonitor?	572
14.5.5	Wie analysieren Sie die angefallenen Daten?	572

14.6	Diagnosemonitor und Diagnosezentrale (Health Center)	580
14.7	Memory Visualizer	590
14.8	Fehlermonitor (nur in Unix-Umgebungen)	593
14.9	Tipps aus der Praxis	595
14.9.1	db2batch	595
15	Performance	599
15.1	Der DB2-Optimizer	599
15.1.1	Der SQL-Compiler	599
15.1.2	Optimierungsklassen (QUERYOPT)	602
15.1.3	Einstellen der Optimierungsklassen	602
15.2	Visual Explain	603
15.2.1	Wie kommen Sie an die Informationen des Tools Visual Explain?	605
15.2.2	Was ist ein Timeron?	609
15.3	Prozessmodell	612
15.3.1	Windows-Prozesse	615
15.3.2	Prozesse beim Sichern einer Datenbank	616
15.4	Speichermodell	618
15.5	E/A-Performance und Pufferpool	619
15.6	SQL-Performance	621
15.7	Tipps	622
A	Migration auf DB2 V8	623
A.1	Vorbereitung der Migration	623
A.1.1	Kompatibilität zwischen den Versionen	623
A.2	Durchführung der Migration	625
A.3	Erste Schritte nach der Migration	627
A.4	Migration bzw. Update nach Fixpack-Installation	628
A.4.1	Spezifische Aufgaben nach der Installation von Fixpacks unter Unix	628
A.4.2	Aufgaben nach der Installation des Fixpacks 2 (DB2 UDB V8.1.2)	629
B	Beschreibung des CD-Inhalts	631
B.1	DB2 UDB Express Edition V8.1 – Trial-Version	632
B.2	DB2DEMO-Programm	632
B.3	BuchDB-Scripts	633
B.3.1	Nutzung der Beispiel-Datenbankprogramme	634
B.4	ToyStore-Beispiel-Datenbank und Datenbankanwendung	635
C	Literaturverzeichnis	637
D	Glossar	639
	Stichwortverzeichnis	645

V Vorwort

In der heutigen Zeit haben die Datenhaltung, -verarbeitung und -auswertung einen Stellenwert erreicht, der vor kurzem noch undenkbar war.

Selbst ein großes Unternehmen kam vor ca. 30 Jahren noch problemlos damit aus, Daten in Papierform zu erfassen, abzulegen, zu verwalten und auszuwerten. Dies ist heute in den Industrienationen kaum einmal mehr für eine normale Privatperson realisierbar.

Die Datenflut – verursacht durch Bilder, digitale Videofilme, Adressen, Telefonnummern, Geldanlagen, Steuererklärungen, Preisauskünfte über Internet und nicht zuletzt die E-Mails – ist nur der Beginn der Auswirkungen der „Elektronisierung“ unserer Umwelt.

Somit ist es nicht verwunderlich, dass für den Erfolg eines Unternehmens die optimale Verarbeitung von Daten eine Grundvoraussetzung ist. Denn selbst das beste Management ist nur so gut wie die Informationen und die Möglichkeiten zu deren Auswertung, die ihm zur Verfügung stehen. Nicht zu vergessen ist, dass durch die Globalisierung und Öffnung der Märkte die Anforderungen an diese Aufgaben um ein Vielfaches gewachsen sind.

Betrachtet man aber, wie selbst heute noch die immer größer werdenden Datenmengen verwaltet und ausgewertet werden, stellt man fest, dass genau bei dieser Aufgabe nicht selten noch Pionierarbeit geleistet werden muss.

Viele große Unternehmen sind deshalb in den letzten Jahren dazu übergegangen, relationale Datenbanksysteme einzusetzen. Doch es nützt nichts, sich einen Jumbojet zu kaufen, wenn man ihn dazu einsetzt, kurz um die Ecke zu fliegen, um Brötchen zu kaufen. Also: Allein der Einsatz eines relationalen Datenbanksystems führt nicht zwangsweise zu dem gewünschten Erfolg.

Je komplexer und mächtiger ein System ist, umso höher werden die Anforderungen an den Nutzer dieses Systems.

Um auf den Jumbo zurückzukommen: Wer traut sich zu, einfach per „Learning by Doing“ einen Jumbojet zu fliegen, oder wer möchte bei einem solchen Piloten Passagier sein? Und wer weiß das nicht – auch der Absturz eines DV-Systems kann verheerende Folgen haben!



Abbildung V.1: Es schlägt ein – welches EDV-System überlebt das schon!

OLAP (Online Analytical Processing), Data Mining, ERP (Enterprise Reporting Planning, CRM (Customer Relationship Management) und nicht zuletzt die neuen Internetsysteme (z.B. Internetportale, Shop-Systeme) – sie alle benötigen eine korrekte und meist sogar eine ständig verfügbare Datenbasis.

Gerade durch das Internet ergeben sich neue Herausforderungen für ein Unternehmen. Konzentrierten sich die Unternehmen bei ihrer Datenhaltung bisher vorwiegend auf Systeme wie z.B. Auftragserfassung, Lagerhaltung und Abrechnungswesen, so treten diese durch das Internet jetzt in eine neue Dimension. Dies bringt unter anderem Folgendes mit sich:

- ▶ 24-Stunden-Betrieb an sieben Tagen in der Woche
- ▶ ein höheres Datenvolumen
- ▶ globale Absatzchancen
- ▶ globale Konkurrenz
- ▶ die Möglichkeit, Lieferanten und Kunden an firmeninterne Anwendungen und Daten anzubinden

Durch diese Herausforderungen an die Datenhaltung und -verwaltung hat sich der neue Begriff *Content Management* geprägt. Hierunter versteht man das Management einer Vielfalt von verschiedensten Informationen, wie z.B.:

- ▶ Bilder, Audio- und Videodateien
- ▶ HTML- und XML-Dateien
- ▶ herkömmliche strukturierte Daten aus den verschiedensten Anwendungen (z.B. Adressdaten, Auftragsdaten)
- ▶ verschiedene Dokumententypen (.txt, .doc, .pdf, .ppt)
- ▶ E-Mails
- ▶ zusätzliche spezifische Informationen, wie biometrische Daten (z.B. Fingerabdrücke)

Dazu kommt, dass selbst Informationen derselben Kategorie in unterschiedlichen Formaten gespeichert werden.

Um diese Anforderungen erfüllen zu können, werden die meisten ursprünglichen relationalen Datenbanksysteme ständig um neue Funktionalität erweitert. Somit spricht man heute nicht mehr von relationalen Datenbanksystemen, sondern von *objektrelationalen Datenbanksystemen*.

Objektrelationale Datenbanksysteme ermöglichen es durch zusätzliche Funktionen (DB2 UDB Extender), Daten wie z.B. Bilder, XML-Dateien oder Musik nicht nur zu speichern, sondern auch sinnvoll zu verarbeiten und zu verwalten.

Wir möchten Ihnen in diesem Buch unsere Erfahrungen aus Theorie und Praxis so vermitteln, dass Ihnen innerhalb kürzester Zeit unser Wissen, das wir im Team in sämtlichen Gebieten der Datenhaltung erworben haben, zu Gute kommt. Ihnen bleiben dadurch hoffentlich manche negativen Erfahrungen erspart, die wir selbst machen mussten.

V.1 Inhalt

Die Aufteilung der Kapitel wurde so gestaltet, dass sich das Buch sowohl für den Anfänger als auch für den fortgeschrittenen Datenbanknutzer eignet.

Kapitel 1 ist eine Einführung in Datenbanken und Datenbanksysteme. Nach einem kurzen Überblick über die Historie von Datenbankmanagementsystemen werden die Aufgaben der verschiedenen Benutzergruppen erläutert. Für das Grundverständnis werden im Weiteren die Begriffe und Aufgaben einer relationalen Datenbank (RDB) und eines relationalen Datenbankmanagementsystems (RDBMS) genauer beschrieben. Da wir uns bei den Begriffserklärungen nicht auf die herkömmliche Literatur beziehen, sondern uns vielmehr auf unsere aus der Praxis gewonnene Sicht stützen, eignet sich dieses Kapitel sowohl für den Anfänger als auch für den fortgeschrittenen Datenbankbenutzer.

Die verschiedenen Produktkomponenten und Produktpakete der DB2 UDB sowie Hinweise zu deren Lizenzierung werden in **Kapitel 2** beschrieben.

Wie man die DB2 UDB V8 auf Windows- und Unix-Plattformen installiert und was dabei zu beachten ist, behandeln wir in **Kapitel 3**. Zuvor werden der Begriff des Exemplars (Instance) und seine Bedeutung im DB2 UDB-Umfeld geklärt.

Welche Möglichkeiten und Erleichterungen die grafischen Tools zu Verfügung stellen, beschäftigt uns in **Kapitel 4**.

Nachdem Sie genug Grundlagenwissen erworben haben, werden wir uns in **Kapitel 5** damit befassen, was beim Anlegen einer Datenbank passiert und welche Dinge Sie dabei beachten müssen. Dabei spielen die physische Verteilung der Daten und das Konzept der Tabellenbereiche (Tablesaces) eine wichtige Rolle.

Nach der Installation und der Erstellung einer Datenbank sind die verschiedenen Möglichkeiten der Datenbankverbindungen von Bedeutung. Wie man einen Client

an einen Datenbank-Server anbindet und welche Konfigurationen bei Verbindungsproblemen eine Rolle spielen, wird in **Kapitel 6** vermittelt.

Die DB2 UDB-Datenbankobjekte wie z.B. Tabellen, Schemata, Sichten (Views), Indizes, benutzerdefinierte Datentypen und Auslöser (Trigger) werden in **Kapitel 7** erläutert.

Eine wichtige Frage im Bereich der RDBMS lautet: Wie bekomme ich Daten in meine und aus meiner Datenbank, und welche Formate und Tools stehen mir zur Verfügung? **Kapitel 8** hilft Ihnen, diese Fragen zu klären.

Gute Sicherungs- und Wiederherstellungskonzepte bilden die Grundlage, um Datenverluste zu vermeiden und jederzeit an einem konsistenten Punkt wieder neu beginnen zu können. Die unterschiedlichen Verfahren und Vorgehensweisen mit praxisbezogenen Beispielen werden Ihnen in **Kapitel 9** aufgezeigt.

Die Verwaltung der Benutzerrechte innerhalb der DB2 UDB findet sowohl innerhalb des Exemplars (Instance) als auch auf der Datenbankebene statt. Wie das DB2-System dieses Aufgabengebiet meistert und welche Aufgaben auf den Administrator zukommen, werden wir in **Kapitel 10** behandeln.

Die Programmierung von Datenbank Anwendungen steht für einen Datenbank-administrator zwar nicht im Mittelpunkt, dennoch sollte er ein Grundverständnis für die unterschiedlichen Möglichkeiten der Datenbankprogrammierung besitzen. Welche verschiedenen administrativen Aufgaben dadurch entstehen und wie sich dies auf das Thema Datenbanksicherheit auswirkt, wird in **Kapitel 11** besprochen.

Daten innerhalb einer relationalen Datenbank werden häufig von mehreren Benutzern gleichzeitig gelesen, hinzugefügt und geändert. Die dabei auftretenden Zugriffskonflikte und Sperrmechanismen und die Verfahren, wie diese in DB2 UDB behandelt werden, sind Inhalt von **Kapitel 12**.

Um dauerhaft hohe Leistung zu bringen, muss eine relationale Datenbank (RDB) gewartet werden. **Kapitel 13** wird Ihnen helfen, die benötigten Aufgaben zu verstehen und in der Praxis auszuführen.

Wer kennt es nicht, das Thema Performance? **Kapitel 14** zeigt Ihnen die Möglichkeiten, die DB2 UDB V8 im Bereich Überwachung (Monitoring) bietet. Die dabei gewonnenen Informationen sind eine wichtige Voraussetzung für geeignete, leistungssteigernde Maßnahmen. Welche Aufgaben und Hilfsmittel Sie hier erwarten, werden wir am Ende unseres Buchs in **Kapitel 15** beschreiben.

Weitere wichtige Informationen finden Sie im **Anhang**, der in vier Abschnitte aufgeteilt ist:

- ▶ Nützliche Informationen zur Migration auf die DB2 UDB V8 finden Sie in **Anhang A**.
- ▶ Der Inhalt der Buch-CD wird in **Anhang B** beschrieben.
- ▶ **Anhang C** enthält ein Literaturverzeichnis.
- ▶ Die wesentlichen im Buch verwendeten Begriffe werden im **Glossar** in **Anhang D** kurz erläutert.

V.2 Ziel und Aufbau des Buchs

Unsere Absicht war es, ein Buch zu schreiben, das praxisorientiert ist. Somit verzichteten wir oftmals darauf, alle Optionen und Möglichkeiten darzustellen, die DB2 UDB bietet. Vielmehr konzentrieren wir uns darauf, die in der Praxis wirklich *wichtigen* Dinge hervorzuheben und mit Hintergründen zu erläutern. Dies führt dazu, dass wir in den meisten aufgeführten Syntaxdiagrammen nicht alle Parameter darstellen, sondern nur die, die unserer Meinung nach am wichtigsten sind. Wir gehen dabei auf die Dinge ein, die ein Datenbankadministrator am häufigsten im alltäglichen Leben braucht, und erläutern diese mit Tipps und Hinweisen aus der Praxis.

Statt einer reinen Funktionsbeschreibung wollen wir das *Wieso*, *Wie* und *Warum* darstellen.

An dieser Stelle möchten wir darauf hinweisen, dass man eine genaue und exakte Beschreibung der Syntax nur in der aktuellen Online-Dokumentation der DB2 UDB findet. Die Praxis hat uns gelehrt, dass sich selbst bei einer Software-Aktualisierung (Service-Pack) die Syntax eines Befehls um einige Parameter erweitern kann.

Wir betrachten unser Buch nicht als Ersatz, sondern als hilfreiche Erweiterung der DB2 UDB-Dokumentation.

V.3 Die Autoren

Alle drei Autoren beschäftigen sich seit der ersten Verfügbarkeit dieser Software mit der DB2 UDB bzw. mit DB2 auf dezentralen Plattformen und mit deren unterschiedlichen Einsatzgebieten.

► Ivo Grodtke

- Studienabschlüsse als Betriebswirt (VWA) und Anwendungsinformatiker (VWA) sowie eine Ausbildung als Erwachsenenbildner an der PH Weingarten
- 19 Jahre Berufserfahrung:
 - Projektleiter für IT-Projekte im In- und Ausland
 - seit 1991 DB2-Dozent und DB2-Dozenten-Ausbilder bei der Firma IBM, mit internationalem Einsatzgebiet
 - Entwickler von internationalen IBM DB2 UDB-Kursen unter anderem zu DB2 UDB-Administration und DB2 UDB-Performance und Tuning.
- Erste Kontakte mit DB2 1983
- Seit 2002 Mitarbeiter der Firma LIS.TEC GmbH

► Jens Orhanovic

- Studienabschluss als Diplomingenieur für technische Informatik
- 19 Jahre Berufserfahrung:
 - Anwendungsentwickler
 - Unternehmensberater
 - Projektleiter vieler unterschiedlicher Projekte

- Dozententätigkeit für verschiedene Schulungsanbieter auf den Gebieten DB2 und Objektorientierte Programmierung
- Entwickler von DB2 UDB-Kursmaterialien
- Erste Kontakte mit DB2 1988
- Selbstständig seit 1988
- Gründer und Geschäftsführer der Firma LIS.TEC GmbH

► **Michael Tiefenbacher**

- Studienabschluss als Diplominformatiker sowie eine Ausbildung als Erwachsenenbildner an der PH Weingarten
- Acht Jahre Berufserfahrung:
 - Seit 1996 DB2-Dozent bei der Firma IBM mit internationalem Einsatzgebiet
 - Entwickler von internationalen IBM DB2 UDB-Kursen unter anderem zu DB2 UDB-Administration, DB2 UDB-Performance und Tuning und Business Intelligence bzw. Data Warehouse
 - Berater in Datenbank- und Data-Warehouse-Projekten
- Seit 2002 Mitarbeiter der Firma LIS.TEC GmbH

V.4 Die Datenmodelle

In diesem Buch verwenden wir folgende Datenbanken:

- die Sample Database, die mit DB2 UDB mitgeliefert wird
- die Datenbank BUCHDB

V.4.1 Die Sample-Datenbank von DB2 UDB

Da diese Datenbank höchstwahrscheinlich fast jedem bekannt ist, der DB2 installiert hat, wollen wir sie nutzen, um das eine oder andere Beispiel in der Theorie und Praxis zu verdeutlichen. Die Beschreibung des Datenmodells finden Sie in Abbildung V.2.

Das Datenmodell

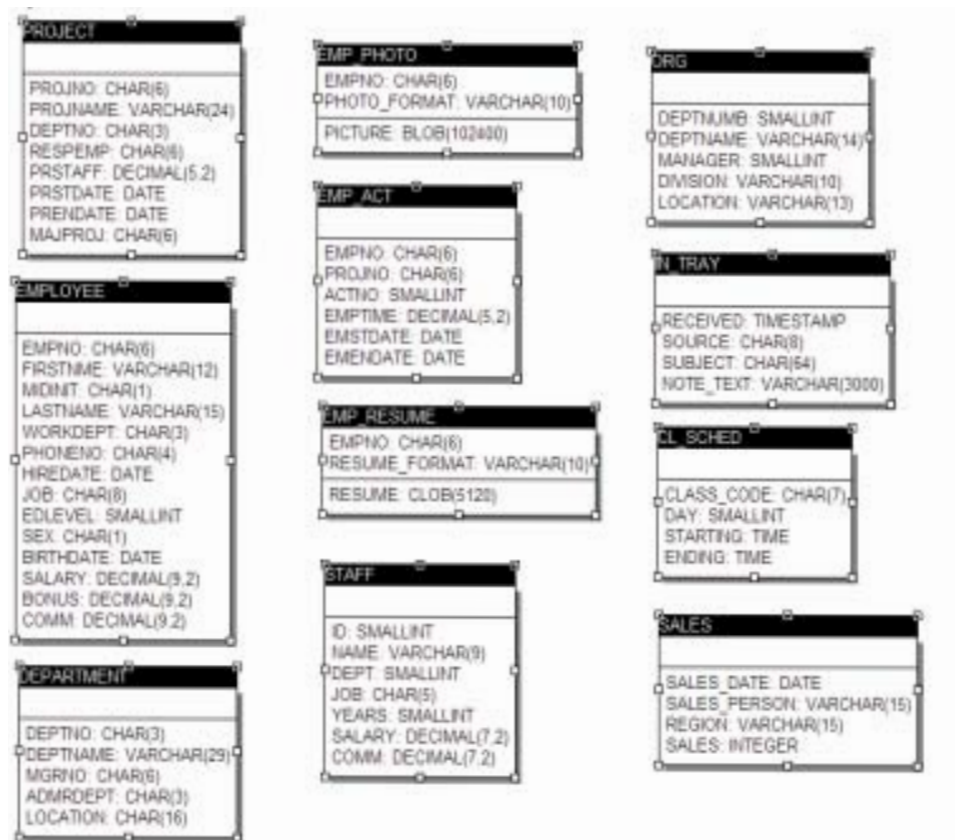


Abbildung V.2: Bild des Datenmodells der Datenbank Sample

V.4.2 Die Datenbank BUCHDB

Da einige Beispiele ein etwas erweitertes Datenbankmodell voraussetzen, haben wir diese Anforderungen in einem eigenen kleinen Datenmodell zusammengefasst.

Das Modell ist nicht so vollständig, dass es für eine produktive Anwendung ausreicht. Ebenso stellen wir nicht den Anspruch, dass es aus Sicht der Datenmodellierung hundertprozentig korrekt ist. Es bietet aber eine geeignete Diskussionsgrundlage für ausgewählte Themen.

Es beinhaltet außer den Tabellen referenzielle Integrität, Auslöser (Trigger) sowie eine N:M-Beziehung. Anhand dieses Modells können die Wichtigkeit der Datenintegrität und deren Realisierung in der Datenbank sehr gut aufgezeigt werden. Es soll uns in einem späteren Kapitel verdeutlichen, was es bedeutet, Programmlogik in die Datenbank zu legen.

Das Datenmodell

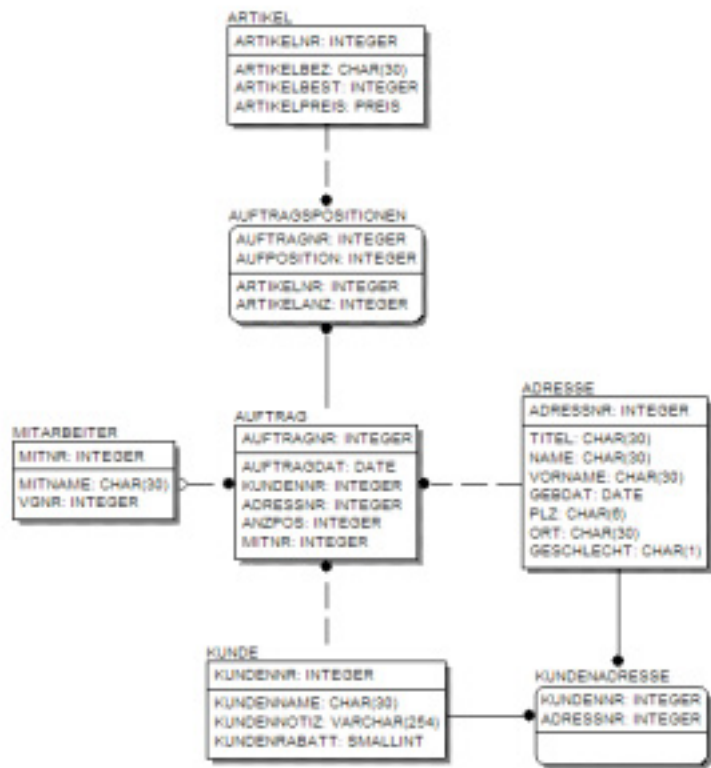


Abbildung V.3: Das Datenmodell der Datenbank BUCHDB

V.5 Danksagungen

Trotz unserer Erfahrungen im Bereich DB2 UDB und unserer Dozententätigkeit, die uns schon oft vor die Aufgabe gestellt hat, Schulungsunterlagen zu erstellen sowie Personen auszubilden, war die Zeit, in der das Buch entstand, eine große Herausforderung. Dies gilt für uns und all die Personen, die sowohl privat als auch geschäftlich das Leben mit uns teilen bzw. teilen mussten und müssen.

Somit möchten wir uns vor allem bei unseren Familien bedanken, die uns so stark unterstützt und so viel Verständnis aufgebracht haben. Während wir dieses Buch schrieben, hatten wir so gut wie keine Zeit für sie, und sogar im Urlaub, am Wochenende und an vielen Abenden standen wir für sie nicht zur Verfügung.

Ebenso vielen Dank an alle LIS.TEC-Mitarbeiter, die uns tatkräftig geholfen bzw. viele Tätigkeiten, die wir sonst in unserem Berufsleben ausführen, übernommen haben! Bei den Mitarbeitern des Verlages Addison-Wesley möchten wir uns besonders bei unseren Lektoren Rolf Pakendorf und Rainer Fuchs für die gute Zusammenarbeit und Unterstützung bedanken.

Wenn Sie Anregungen oder Kritik haben, mailen Sie uns unter DB2Buch@listec.de!

Jens Orhanovic, Ivo Grodtke, Michael Tiefenbacher

1 Einführung in Datenbanken und Datenbanksysteme

1.1 Datenbankgrundlagen

Nicht erst seit der Einführung der elektronischen Datenverarbeitung beschäftigt man sich mit der Haltung, Verwaltung, Verarbeitung und Auswertung von Daten.

Da wir zum Teil selbst die Veränderungen der EDV von der Lochkarte bis heute miterleben durften, denken wir, dass ein Blick auf die Historie und Grundlagen dieser Systeme sehr hilfreich ist, um ein gewisses Grundverständnis für die Aufgaben eines Datenbankmanagementsystems zu erlangen.

Es gibt auch heute noch genug Anwendungen, die auf Vorgänger der relationalen Datenbanksysteme zugreifen, um mit „IMS“ (Information Management System, eine hierarchische Datenbank) nur ein Beispiel zu nennen.

1.2 Historie der relationalen Datenbanksysteme

Den Anfang machten in den 70er Jahren Systeme wie hierarchische bzw. Netzwerkdatenbanken. Diese hatten den Nachteil, dass die gesamte Anwendungslogik im Programmcode vorausgesetzt wurde. Einfache Abfragemöglichkeiten bestanden nicht.

Mitte der 70er Jahre legte dann Edgar Codd in einem Forschungsprojekt bei IBM die theoretischen Grundlagen für relationale Datenbanken und Datenbanksysteme. Zusammen mit Chris Date wurden diese dann weiterentwickelt und vervollständigt.

Die Grundlage für die Aufgaben eines relationalen Datenbanksystems bildeten hierbei die bis heute gültigen *Codd'schen Regeln*, die wir im Folgenden in unseren eigenen Worten aufführen.

1.2.1 Die Codd'schen Regeln

- Zugriff entsprechend dem Drei-Schichten-Modell

Der Zugriff auf die Daten über Sichten (Views), Schemata und direkt auf die Daten muss gewährleistet sein. Dies ermöglicht die Trennung der Benutzer-, der logischen und der physikalischen Schicht.

- Mehrbenutzerzugriff (multi user access)

Der gleichzeitige und konkurrierende Zugriff mehrerer Benutzer auf ein und dieselben Daten für Lese- und Schreiboperationen muss möglich sein.

► Transaktionsorientierung

Die Anforderungen an Serialisierbarkeit, Protokollierbarkeit, Rücknehmbarkeit und Abgeschlossenheit der Daten müssen erfüllt sein. Das bedeutet: Mehrere nacheinander durchgeführte Änderungen müssen als Einheit zusammengefasst werden können und entweder *alle* mit einer Bestätigung (COMMIT) durchgeführt werden, oder es werden durch eine Rücknahme (ROLLBACK) *alle* Änderungen zurückgesetzt. Hierdurch wird die Zusammenfassung mehrerer einzelner Aktionen zu einer logischen Einheit ermöglicht.

► Online-Verarbeitung

Bei einem Datenzugriff aus Programmen heraus müssen die Transaktionseigenschaften gewährleistet sein.

► Anforderungen an Sicherung (backup) und Wiederherstellbarkeit (recovery)

Um einen Datenverlust durch logische Fehler oder Hardware-Fehler zu verhindern, muss ein Wiederherstellen der gesamten Datenbank gewährleistet sein. Hierzu muss das RDBMS geeignete Sicherungs- und Wiederherstellungsverfahren zur Verfügung stellen. Ebenso muss die Möglichkeit bestehen, den Zustand der Daten zu einem bestimmten Zeitpunkt (point in time recovery) bzw. zum letzten konsistenten Zeitpunkt (rollforward bzw. crash recovery) wiederherzustellen.

► Zugriffsrealisierung

Die Datenkonsistenz muss auch dann gewährleistet sein, wenn mehrere Benutzer ein und denselben Datensatz lesen bzw. ändern möchten. So genannte Anomalien („lost updates“) und Inkonsistenzen müssen verhindert werden.

► SQL (Structured Query Language)

Das RDBMS muss eine nicht prozedurale, mengenorientierte Sprache zum Zugriff auf die Daten und Objekte für Lese- und Schreiboperationen zur Verfügung stellen.

► Ad-hoc-Verarbeitung

Die konventionell geplante Verarbeitung der Daten mit SQL oder anderen Schnittstellen muss sowohl über Programme als auch über interaktive Schnittstellen möglich sein. Es gilt ein Unterwanderungsverbot. Das bedeutet, dass der SQL-Sprachumfang – auch durch zusätzlich zur Verfügung gestellte Schnittstellen – nicht erweitert werden darf.

► Integrität des Wertebereichs (domain integrity)

Die zulässige Wertemenge eines Attributs muss über das Datenbanksystem definierbar und ihre Einhaltung muss gewährleistet sein.

► Referenzielle Integrität (referential integrity)

Innerhalb einer relationalen Datenbank können Entitäten statische Beziehungen untereinander besitzen. Diese werden durch das Vorhandensein eines bestimmten Wertes in einem Attribut eindeutig ausgedrückt und müssen vom RDBMS überwacht werden können.

► Entitätenintegrität (entity integrity)

Es muss gewährleistet sein, dass der Datentyp und somit die Dimensionen eines jeden Attributs einer Entität eingehalten werden. So darf zum Beispiel ein numerisches (Integer-)Feld keine Buchstaben (Characters) enthalten.

► Zugriffsberechtigung

Die Regelung der Rechte zum Zugriff auf Daten und Objekte der Datenbank und des Datenbankmanagementsystems muss im RDBMS integriert sein.

1.2.2 Historie der DB2

Auf Basis der zuvor erwähnten Codd'schen Regeln entstand 1975 und 1976 aus einem IBM-Forschungsprojekt mit *System R* die erste Implementierung des relationalen Modells. Diese wurde 1977 erstmalig als Prototyp bei IBM-Kunden installiert.

Auf der Grundlage dieses Systems stellte IBM 1982 mit dem Produkt *SQL/DS* erstmals ein relationales DBMS für die Betriebssysteme VM und VSE zur Verfügung.

DB2 hieß ursprünglich *Database 2* und war zum ersten Mal 1983 unter dem Betriebssystem MVS verfügbar. Als Ausgangssystem diente ebenfalls *System R*. Es handelt sich hierbei also um eine parallele Entwicklung zu *SQL/DS*.

Mit dem Betriebssystem OS/400 wird seit 1988 das integrierte *SQL/400* ausgeliefert, das heute *DB2* für AS/400 heißt.

1987 kündigte IBM mit dem Betriebssystem OS/2 V1 und der Erweiterung *OS/2 Extended Edition* die ersten relationalen Datenbankfähigkeiten auf einem dezentralen System an. Hierbei handelt es sich um den Vorläufer der heutigen *DB2 UDB*. Als eigenständiges Datenbankprodukt erschien 1991 der Vorgänger von *DB2* unter dem Namen *OS/2 Database Manager*. 1993 wurde die *DB2*-Datenbank unter dem Namen *DB2 V1* für die Betriebssysteme OS/2 und AIX vermarktet.

Seit der Version *DB2 Common Server V2*, der ersten objektrelationalen und webfähigen Datenbank, die 1995 veröffentlicht wurde, wird auch das Betriebssystem Windows unterstützt.

Inzwischen läuft die *DB2 UDB V8* auf allen gängigen Server-Betriebssystemplattformen inklusive Linux.

1.3 Nutzung von relationalen Datenbankmanagementsystemen

Um den Umgang mit der Komplexität relationaler DBMS zu erleichtern, teilt man die Aufgaben eines RDBMS wie folgt ein:

► Datenmodellierung

► Programmierung

► Administration

► Anwender (Endbenutzer)

1.3.1 Aufgaben des Modellierers

Das Datenmodell ist die wichtigste Grundlage einer konsistenten und performanten Datenbank. Der Modellierer erstellt aufgrund der Geschäftsanforderung das entsprechende Datenmodell. Die Arbeitsschritte der Modellierung sind in drei konzeptionelle Ebenen unterteilt:

► **Geschäftsebene (Business-Ebene)**

Auf dieser Ebene geht es darum, die Geschäftsprozesse und die Zusammenhänge zu erkennen und zu formalisieren.

► **Logische Ebene**

Hier geht es darum, die formalisierten Geschäftsprozesse in ein Entity-Relationship-Modell (ER-Modell) umzusetzen und dabei die Konsistenzregeln (Normalformen) dieser Modellierungstechnik zu berücksichtigen.

► **Physikalische Ebene**

Das erstellte ER-Modell wird in diesem Schritt um physikalische Aspekte erweitert und dann in der Datenbank implementiert, indem zugehörige Datenbankobjekte erstellt werden.

Warum werden die Schritte der Modellierung auf drei Ebenen unterteilt?

Wenn Sie vor der Aufgabe stehen, ein Datenmodell einer komplexeren Geschäftslogik zu erstellen, merken Sie sehr schnell, dass es sich dabei um eine nicht zu verachtende Herausforderung handelt.

Zum einen ist da der Endbenutzer, der Sie mit sämtlichen Informationen der Geschäftslogik überhäuft, zum anderen sind bestimmte Vorgänge und Verantwortungen in den Firmen häufig nicht definiert. Außerdem müssen Sie die Normalformen des relationalen DB-Designs einhalten, um ganz zum Schluss die Speicherform der Daten für die zum Einsatz kommende Datenbank in optimaler Form bestimmen zu können.

Bei der Verwaltung der eigenen Videofilme mag das ja noch ganz gut aus dem Bauch heraus möglich sein. Bei komplexeren Geschäftsverfahren helfen hier nur noch ein akkurates Vorgehen und die getrennte Betrachtung der einzelnen Aufgaben nach den zuvor erwähnten Ebenen.

Da die Komplexität der Datenmodellierung genug Stoff für ein eigenes Buch liefert, werden wir in diesem Buch nicht weiter darauf eingehen.

Dennoch möchten wir darauf hinweisen, dass bei den meisten Performance- und Tuning-Betrachtungen, die wir in der Praxis durchführen, die Ursachen für Probleme in unzureichendem Datenbankdesign zu suchen sind. Und nicht zuletzt macht häufig eine fehlende Dokumentation die Untersuchung sowie die Nutzung des bestehenden Datenmodells nur schwer möglich.

Hinweis:

Für eine saubere und dokumentierte Datenmodellierung empfiehlt es sich, ein entsprechend mächtiges Werkzeug einzusetzen.

Der Markt bietet eine Reihe von Produkten. Wir haben in der Praxis die besten Erfahrungen mit dem Produkt *AllFusion ERwin Data Modeler* der Firma Computer Associates gemacht, das auch als „Erwin“ bekannt ist.

1.3.2 Aufgaben des Programmierers

Der Programmierer hat im Wesentlichen zwei Hauptaufgaben:

- ▶ Er erstellt die SQL-Anweisungen aufgrund des vorgegebenen Datenmodells.
- ▶ Er erstellt die Verarbeitungsprogramme und die Benutzerschnittstellen in der entsprechenden Programmiersprache.

An dieser Stelle weisen wir darauf hin, dass die Trennung zwischen der eigentlichen Programmierung in einer Programmiersprache und der Erstellung der SQL-Befehle ganz strikt gehandhabt werden sollte.

Warum ist das so notwendig?

Wir selbst hatten bis zum ersten Einsatz einer relationalen Datenbank mehrere Jahre Programmiererfahrung. Somit gingen wir unsere ersten Projekte zur Entwicklung von Datenbankanwendungen mit der Einstellung an: Was man nicht selbst programmiert, kann nicht sehr performant sein.

Somit ist es für uns nicht verwunderlich, dass wir genau diese Einstellung bei den meisten Programmierern wiederfinden.

SQL ist aber im Gegensatz zu einer prozeduralen oder einer objektorientierten Programmiersprache eine mengenorientierte Abfragesprache. Das bedeutet: Hier müssen sich die Gedanken nicht auf das *Wie*, sondern auf das *Was* konzentrieren. Dies sind zwei prinzipiell unterschiedliche Denkansätze, und sie sollten nicht vermischt werden.

Versucht man dennoch, diese Ansätze zu vermischen, führt das in der Praxis dazu, dass meist nur sehr einfache SQL-Statements erstellt werden und die eigentliche Verarbeitungslogik im Programm steckt. Da die Möglichkeiten des relationalen Datenbanksystems nicht genutzt werden, hat dies in den meisten Fällen den gefürchteten Performance-Verlust zur Folge.

Tipp:

Somit lautet unsere Empfehlung: Bevor überhaupt eine einzige Zeile Programmcode in Bezug auf die Datenbank geschrieben wird, sollten erst alle SQL-Anweisungen geschrieben und getestet werden. Beim Testen muss nicht nur auf die Richtigkeit, sondern auch schon auf die Performance geachtet werden. Hierzu stellt DB2 verschiedene ausgezeichnete Tools zur Verfügung (wie z.B. Explain).

Wählt man diese getrennte Vorgehensweise, kann man sehr schnell und noch vor der eigentlichen Programmierung folgende Fragen klären:

- ▶ Ist das Datenbankdesign aus mengenorientierter Sicht in Ordnung?
- ▶ Werden alle benötigten logischen Konsistenzanforderungen wie Auslöser (Trigger) und referenzielle Integrität erfüllt?
- ▶ Werden zusätzliche Datenbankobjekte wie z.B. benutzerdefinierte Funktionen (User Defined Functions – UDFs), benutzerdefinierte Datentypen (User Defined Types – UDTs) und gespeicherte Prozeduren (Stored Procedures) benötigt?
- ▶ Müssen aus Performance-Gründen Änderungen am logischen oder physischen Datenbankdesign vorgenommen werden?
- ▶ Können künstliche Redundanzen, wie z.B. gespeicherte Abfragetabellen (Summary Tables bzw. Materialized Query Tables – MQTs) oder zusätzliche spezielle Indizes leistungssteigernd wirken?

Wir denken, die Vorteile der drei aufeinander folgenden Arbeitsschritte liegen klar auf der Hand. Denn missachtet man diese Trennung und Reihenfolge und ist erst einmal Programmcode geschrieben, sind die Änderungen sehr viel aufwändiger bzw. fast unmöglich.

In Kapitel 11, „Schnittstellen zur Programmierung“, werden wir auf die verschiedenen Möglichkeiten eingehen, die DB2 dem Programmierer bietet.

1.3.3 Aufgaben des Administrators

Die Administration kann aus Sicht eines RDBMS in zwei Gebiete unterteilt werden:

- ▶ Administration des Datenbankmanagementsystems (DB-Systemadministration)
- ▶ Administration der Datenbank

Diese Aufteilung macht eine Trennung der plattformabhängigen und der plattform-unabhängigen Aufgaben möglich. Während die Administration des DBMS stark vom Betriebssystem beeinflusst wird, ist die Administration der Datenbanken nahezu plattformunabhängig.

Je nach Größe und Komplexität des Betriebs ist es sogar sinnvoll, diese Aufgaben-gebiete unterschiedlichen Personen zuzuordnen.

Aufgaben der Datenbanksystemadministration

Das Aufgabengebiet der Datenbanksystemadministration umfasst im Wesentlichen alle betriebssystemnahen Aufgaben.

Dazu gehören:

- ▶ Installation und Pflege des DBMS
- ▶ Erstellen und Wartung von Exemplaren (Instanzen)
- ▶ Administration der Datenbankbenutzer

- ▶ Verwaltung und Überwachung der benötigten Systemressourcen
- ▶ Anlegen von Datenbanken

Aufgaben der Datenbankadministration

Die Aufgabe des Datenbankadministrators besteht in der eigentlichen Verwaltung der Datenbanken.

Hierunter fallen folgende Hauptaufgaben:

- ▶ Anlegen und Verwalten von Datenbankobjekten
- ▶ Performance-Überwachung und gegebenenfalls Tuning-Maßnahmen
- ▶ Erstellen und Verwalten von Sicherungs-(backup-) und Wiederherstellungs-(recovery-)Verfahren
- ▶ Exportieren und Laden von Daten
- ▶ Erstellen, Implementieren und Überwachen von Datenbank- und DBMS-Sicherheitskonzepten inklusive
 - Verwalten von Berechtigungen der Benutzer für die verschiedenen SQL-Befehle (wie Select, Insert, Update) auf die Datenbankobjekte
 - Verwalten von Berechtigungen der Benutzer für die verschiedenen Programme, die auf die Datenbank zugreifen
- ▶ Pflege- und Wartungsarbeiten
 - Binden von Programmpaketen (bind-files) zur Erstellung von Zugriffsplänen
 - Pflegen der Statistiken
 - Reorganisieren von Daten

Der Datenbankadministrator ist derjenige, der das eingesetzte DBMS und dessen Eigenschaften sehr gut kennen muss. Er ist der Hauptansprechpartner für alle weiteren Beteiligten (Benutzer, Programmierer, Modellierer und Systemadministratoren). Je nach Größe des Unternehmens und Anzahl der eingesetzten DBMS bzw. Datenbanken handelt es sich bei dieser Aufgabe (Datenbankadministrator) nicht selten um einen Vollzeit-Job.

Dennoch stellen wir in der Praxis sehr häufig fest, dass genau dieser Job unterschätzt wird und einfach von einem Mitarbeiter nebenbei gemacht werden soll.

Sicherlich wirbt heute jeder Datenbankhersteller mit einem nahezu wartungsfreien System. Jedoch gilt auch hier: Je wartungsfreier ein System ist, desto komplexer ist es.

Um wieder auf das Beispiel des Jumbojets zu kommen: Auch er besitzt eine hochkomplexe Technik, die den Piloten und den Ingenieuren die Arbeit weitgehend abnimmt. Dennoch glauben wir nicht, dass man auf die Idee kommen würde, die Ausbildung und die Arbeit des Piloten zu unterschätzen.

Das soll nicht heißen, dass jedes Unternehmen, das ein RDBMS einsetzt, einen Vollzeit-Administrator benötigt. Hier hängt es vielmehr davon ab, welche Ansprüche an das gesamte DV-System gestellt werden und welche Automatismen und Wartungsverfahren vorhanden und vorgesehen sind.

Wir wollen uns in diesem Buch genau mit diesem Aufgabengebiet der Administration befassen. Sowohl der Anfänger als auch der erfahrene Administrator wird hier wertvolle Informationen sowie Tipps und Tricks aus der Praxis erhalten.

1.3.4 Aufgaben der Endbenutzer

Der Endbenutzer ist derjenige, der die Daten und Objekte, die von der Datenbank zur Verfügung gestellt werden, über die verschiedensten Schnittstellen nutzen kann.

Als Schnittstellen stehen dem Benutzer folgende Möglichkeiten zur Verfügung:

- ▶ SQL-Standardabfrage-Tools, wie z.B.:
 - QMF (Query Management Facility) BO (Business Objects)
 - Crystal Reports
- ▶ OLAP-Werkzeuge, wie z.B.:
 - IBM OLAP-Server
 - Cognos-Werkzeuge
 - MicroStrategy-Werkzeuge
- ▶ Mining-Werkzeuge
- ▶ Speziell entwickelte Anwendungen
- ▶ SQL-Schnittstellen von DB2

Im Prinzip gibt es zwei Gruppen von Endbenutzern:

- ▶ Endbenutzer, die nur fertige Lösungen und vordefinierte Wege nutzen
- ▶ Endbenutzer mit SQL-Kenntnissen, die ihre eigenen Abfragen gestalten

Der Endbenutzer, der mit vorgefertigten Wegen und Lösungen arbeitet, erkennt in der Regel keinen Unterschied zu Systemen, die kein RDBMS als Datenbankgrundlage nutzen. Im Gegensatz dazu wird der Nutzer mit SQL-Kenntnissen die Vorzüge eines RDBMS sehr schnell zu schätzen lernen. Die einfache Nutzung der SQL-Schnittstelle ermöglicht es dem Benutzer, jederzeit die gespeicherten Daten, die zur Verfügung gestellten Objekte und SQL-Erweiterungen, wie z.B. UDFs (User Defined Functions), UDTF (User Defined Table Functions), gespeicherte Prozeduren (Stored Procedures), gespeicherte Abfragetabellen (MQTs, also materialisierte Abfragetabellen) und Sichten (Views) für eigene Ad-hoc-Auswertungen zu nutzen.

Dieser enorme Vorteil einer relationalen Datenbank versetzt ein Unternehmen in die Lage, seine Auswertungen und somit seine benötigten Geschäftsinformationen schnell an die sich ändernden Bedingungen des Markts oder an eine bestimmte Situation anzupassen. Dies führt dazu, dass der eigentlich technische Vorteil sofort als betriebswirtschaftlicher Vorteil genutzt werden kann.

Die Grundvoraussetzungen für eine sinnvolle und performante Nutzung der Daten sind jedoch die SQL-Kenntnisse, das richtige Datenbankdesign und dessen übersichtliche Dokumentation.

1.4 Was ist eigentlich eine Datenbank?

Nachdem wir uns im vorigen Abschnitt mit der Historie, der Nutzung und den verschiedenen Benutzergruppen ganz allgemein befasst haben, wollen wir uns ab jetzt speziell mit der DB2 UDB Version 8 beschäftigen.

In der Literatur finden wir die unterschiedlichsten – und teilweise hoch wissenschaftlichen – Definitionen. Die Anforderungen, die eine relationale Datenbank erfüllen muss, werden ja in den Codd'schen Regeln beschrieben. Alle Anforderungen, die an ein RDBMS gestellt werden, sind uns zwar dann bekannt, doch es bleibt immer noch folgende Frage offen:

Was ist nun tatsächlich eine relationale Datenbank (RDB), und was ist der Unterschied zu dem relationalen Datenbankmanagementsystem (RDBMS)?

1.4.1 Definition einer Datenbank

Wenn wir im Brockhaus nachschlagen, finden wir folgende Beschreibung:

Eine Datenbank ist ein zentral verwaltetes System zur widerspruchsfreien und permanenten Speicherung von umfangreichen Datenmengen, auf die nach unterschiedlichen Anwendungskriterien zugegriffen werden kann. Es besteht aus der Datenbasis, in der die Daten abgelegt werden, und dem Datenbankmanagementsystem, einem Softwarepaket, das insbesondere die Datenbestände und Zugriffsrechte verwaltet. [BROCK]

Reicht uns diese Erklärung aus, um zu verstehen, was eine relationale Datenbank bzw. ein relationales Datenbankmanagementsystem ist?

Wir sind der Meinung, dass sich hinter diesen beiden Begriffen viel mehr verbirgt, als in dieser kurzen Beschreibung ausgedrückt werden kann.

Hinweis:

Wir bezeichnen eine *Datenbank* als ein logisches Objekt, das unter anderem Daten, Programmlogik und viele weitere logische Objekte beinhaltet.

Das *Datenbankmanagementsystem* ist das Programmsystem, das das logische Objekt „die Datenbank“ betriebssystemunabhängig verwaltet und dem Datenbankbenutzer auf einer logischen Ebene zur Verfügung stellt. Ebenso übernimmt das RDBMS die Umsetzung von der logischen auf die physische Ebene.

1.4.2 Ausführliche Beschreibung einer Datenbank

An dieser Stelle möchten wir unsere detaillierte Sicht etwas genauer vorstellen und diese anhand eines einfachen Schaubilds erklären. Dieses Schaubild zeigt verschiedene Komponenten und soll deren Aufgaben sowie deren Zusammenspiel im Gesamtsystem verdeutlichen. Diese Darstellung hat uns selbst und vielen Teilnehmern unserer Kurse geholfen, ein gewisses Grundverständnis zu erwerben. Somit

schaffen wir hiermit eine wichtige Grundlage für das Arbeiten mit Datenbanken und Datenbankmanagementsystemen.

Diese Sicht passt zwar nicht hundertprozentig auf jedes RDBMS (Oracle, DB2 für z/OS, Informix, SQL Server usw.), kann aber bis auf kleine Unterschiede auch hier angewendet werden.

Komponenten des Schaubilds

- ▶ Benutzer
- ▶ CLP (Command Line Processor)
- ▶ DBMS (Datenbankmanagementsystem)
- ▶ Datenbank (logisches Objekt)
- ▶ Objekte der Datenbank
- ▶ Betriebssystem
- ▶ API-Schnittstellen
- ▶ Plattensysteme bzw. Dateien

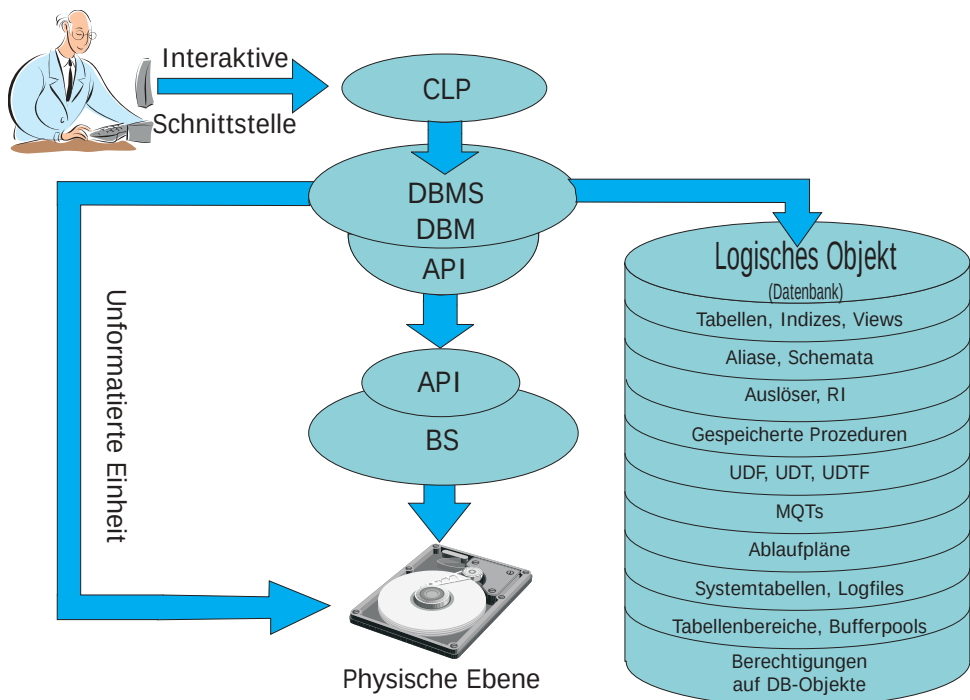


Abbildung 1.1: Was ist eine Datenbank? – Aufgaben der Komponenten

Benutzer

In diesem Schaubild möchte der Benutzer über die interaktive Schnittstelle SQL bzw. Datenbankbefehle eingeben. Da Benutzerzugriffe über Programme und Tools nicht über den CLP (Command Line Processor) an das DBMS gesendet werden, betrachten wir hier nur die interaktive Schnittstelle über den CLP.

CLP

Der **Command Line Prozessor** wird benötigt, da das jeweilige Betriebssystem (Windows, Unix usw.) weder die SQL-Syntax noch die Syntax der Datenbankbefehle verarbeiten kann. Der CLP ist also ein Programm, dem die entsprechenden SQL- oder Datenbankbefehle übergeben werden und das diese dann nach einer einfachen Syntaxprüfung an das DBMS weiterleitet.

DBMS

Das **Datenbankmanagementsystem** interpretiert die eingehenden Befehle (SQL- oder Datenbankbefehle) und führt diese aus, falls sie fehlerfrei sind.

Da in SQL-Befehlen nur logische und keinerlei physische Objekte vorkommen, ist es die Aufgabe des DBMS zu erkennen, welche physischen Objekte angesprochen werden müssen, um den jeweiligen Befehl auszuführen. Somit bildet das DBMS die Schnittstelle zwischen der logischen Ebene (Datenbank) und der physischen Ebene (Betriebssystem). Da die logische Sicht auf die Datenbankobjekte völlig hardware-unabhängig ist, kann dieser Teil der Verarbeitungslogik gekapselt werden und bildet den Kern (Kernel) des DBMS. Hierunter fallen Aufgaben, wie z.B. die Prüfung der SQL-Befehle, SQL-Optimierung sowie die Verwaltung und Zuordnung von logischen und physischen Objekten. Ebenso ist der hardware-abhängige Teil der Verarbeitungslogik, wie Festplattenzugriffe, Zugriffe auf den Hauptspeicher usw., gekapselt. Diese Technik ermöglicht es, den eigentlichen und weitaus größeren Teil, der ein RDBMS ausmacht, völlig unabhängig von der Hardware zu betrachten. Dies war bei der DB2 UDB (**U**niversal **D**ata **B**ase) für die dezentralen Plattformen nicht immer so. Erst ab der Version DB2 UDB V2.x bzw. DB2 UDB V5.x wurde diese Trennung durchgeführt. Dieser Schritt machte es IBM jedoch möglich, die bis dahin verschiedenen DB2-Produkte, die es für die dezentralen Plattformen gab (DB2 für OS/2, DB2 für AIX usw.), zu einem Produkt, der DB2 UDB, zu vereinen.

Heute ist ein Zustand erreicht, in dem sich die DB2 UDB auf allen dezentralen Plattformen aus reiner SQL-Sicht (betrifft hauptsächlich Programmierer) identisch und zu großen Teilen aus Sicht der Datenbankbefehle (betrifft hauptsächlich DB-Administratoren) bis auf ein paar Nuancen gleich verhält.

Dies gilt jedoch nicht für die Systeme AS/400 und z/OS (Mainframe). Diese Systeme besitzen bisher einen völlig anderen Datenbankkern (Kernel) für die logische Verarbeitung. Bei der AS/400 ist dieser Datenbankkern sogar völlig in das Betriebssystem OS/400 integriert.

IBM verwendet inzwischen den Begriff der DB2 UDB für alle Plattformen. Es ist IBM sogar gelungen, für die dezentralen Plattformen dieselbe Versionsnummerierung wie für z/OS zu verwenden und für beide Welten nahezu dieselben logischen Datenbank-

objekte (z.B. Trigger, Stored Procedures) zur Verfügung zu stellen. Dennoch ist es an dieser Stelle sehr leicht zu verstehen, dass sich diese verschiedenen DB2 UDB-Systeme nicht gleich verhalten können, solange der Datenbankkern nicht identisch ist.

Somit müssen sowohl der Programmierer und noch viel mehr der Administrator die Unterschiede zwischen den beiden Welten z/OS, AS/400, VM/VSE und den dezentralen Plattformen (Unix, Windows) beachten.

Datenbank

Die Datenbank kann als logisches Objekt betrachtet werden, das wiederum weitere logische Objekte wie benutzerdefinierte Tabellen, Indizes, Systemkatalogtabellen, Auslöser (Trigger), benutzerdefinierte Funktionen (User Defined Functions – UDF), gespeicherte Prozeduren (Stored Procedures – SPs), Berechtigungen usw. enthält.

Da die Anzahl und das Vorhandensein der logischen Objekte innerhalb einer Datenbank vom Benutzer (Administrator) frei definiert werden kann, muss diese Beschreibung der Objekte dem DBMS zu Verfügung gestellt werden. Dies geschieht in den Systemkatalogtabellen. Somit ist es verständlich, dass es pro möglichem Datenbankobjekt-Typ meistens mindestens eine Systemtabelle gibt.

Beispiele:

- ▶ **SYSIBM.SYSTABLES**
Hier finden Sie die Beschreibung aller Tabellen dieser Datenbank. Pro vorhandener Tabelle gibt es einen Eintrag.
- ▶ **SYSIBM.SYSCOLUMNS**
Hier finden Sie die Beschreibung (Name, Datentyp usw.) jeder Spalte der Tabellen dieser Datenbank. Pro Spalte gibt es ein Eintrag.

Dass für manche Datenbankobjekt-Typen unterschiedliche Systemtabellen zu Verfügung stehen, liegt einfach daran, dass diese Datenbankobjekte weitere Eigenschaften besitzen können, die dann in weiteren Systemtabellen beschrieben werden.

Hinweis:

Die DB2 UDB für dezentrale Plattformen besitzt also pro Datenbank einen umfangreichen Satz von Systemtabellen, die die logische und teilweise die physische Struktur der Objekte beschreiben.

Diese werden automatisch vom DBMS beim Anlegen einer Datenbank erzeugt und während des weiteren Betriebs automatisch vom DBMS bei entsprechenden Benutzeraktionen (z.B. beim Anlegen einer neuen Tabelle und Runstats) aktualisiert und gepflegt.

Sie können vom Benutzer weder explizit erzeugt noch geändert oder gelöscht werden. Eine Ausnahme bilden die Systemtabellen auf den Statistiken. Hier können Änderungen über die vorhandenen Sichten vorgenommen werden. Reine Lesezugriffe auf die Systemtabellen sind über die entsprechenden Sichten (SYSCAT.xxxxxxx) jederzeit möglich.

Vorsicht:

Die obere Aussage über die Systemtabellen gilt nur für die DB2 UDB für dezentrale Plattformen (Windows und Unix).

Ein Beispiel für eine Ausnahme sehen wir bei DB2 für z/OS. Hier gilt, dass die Systemtabellen pro Subsystem erstellt und verwaltet werden. Datenbanken werden hier nur durch Einträge in die entsprechenden Systemtabellen des Subsystems verwaltet.

Wie Sie in Abbildung 1.1 sehen, kann eine RDB nicht nur strukturierte Daten in Form von Tabellen enthalten.

Wir finden hier Objekte wie RI, UDTs, UDF, UDTF, Trigger, Stored Procedures, Views, MQTs usw. (Diese Begriffe werden unten noch erklärt).

Diese Objekte ermöglichen es, Teile der Anwendungslogik, die für die Datenintegrität zuständig sind, sowie weitere Programmmodule in der Datenbank zu implementieren. Somit kann eine RDB nicht nur Daten, sondern auch Logik enthalten.

Hinweis:

In der objektorientierten Welt würde man sagen, dass die Daten durch die Verlagerung der Datenintegritätslogik in die Datenbank mit Leben gefüllt werden. Auch wenn es ein bisschen lustig erscheinen mag: Eine „lebendige“ Datenbank ist sicherlich wesentlich performanter als eine „nicht lebendige“ Datenbank.

Wichtig:

Nur wenn sich die gesamte Datenintegritätslogik in der Datenbank befindet, kann sichergestellt werden, dass die Integrität der Daten zu 100% gewährleistet ist.

Sobald sich ein Teil dieser Logik in Programmen befindet, kann diese Datenintegrität nicht mehr garantiert werden. Dies rührt daher, dass ein RDBMS jederzeit direkte Zugriffe, also ohne zusätzliche Programmlogik, auf die Daten ermöglicht. Dabei kann nicht gewährleistet werden, dass genau diese externen Programmenteile, die die Logik zur Einhaltung der Datenintegrität beinhalten, aufgerufen werden. Somit *müssen*, um die Datenintegrität sicherzustellen, diese Programmenteile in die Datenbank integriert werden.

Leider müssen wir in der Praxis immer wieder feststellen, dass genau diese Forderung viel zu oft verletzt wird. Die Auswirkungen machen sich dann bei Reporting-, OLAP- und Mining-Projekten nicht selten in solch einem Ausmaß bemerkbar, dass die Ergebnisse dieser Projekte für das eigentliche „Business“ unbrauchbar sind.

Nur eine 100%ig richtige Datenbasis gibt einem Unternehmen die so wichtigen Reporting- und Analyse-Möglichkeiten.

Objekte der Datenbank

Das logische Objekt Datenbank kann weitere logische Objekte beinhalten.

In der folgenden Liste führen wir ein paar Beispielobjekte auf und erläutern sie kurz (Näheres siehe Kapitel 7, „Datenbankobjekte“):

- ▶ **Tabellen:** In einer Tabelle sind die Daten logisch nach Spalten (Attribute der Tabelle) und Zeilen (Entitäten einer Tabelle bzw. Datensätze) angeordnet.
- ▶ **Systemtabellen:** Das sind spezielle Tabellen, die nur vom DBMS verwaltet werden und unter anderem die Beschreibung der Datenbank enthalten.
- ▶ **Index:** Eine Menge von Attributen, die nach Benutzervorgaben physisch sortiert und redundant vom RDBMS abgelegt werden. Ein Index wird häufig für den Zugriff auf die eigentlichen Datentabellen verwendet.
- ▶ **Benutzerberechtigungen:** Hiermit sind Benutzerberechtigungen auf Datenbankobjekte gemeint.
- ▶ **Tabellenbereiche (Tablespaces):** Logische Platzhalter für Tabellen.
- ▶ **RI:** Eine referenzielle Integrität definiert die Abhängigkeiten der einzelnen Tabellen voneinander.
- ▶ **Auslöser (Trigger):** Dies sind Aktionen, die beim Insert, Update oder Delete eines Datensatzes ausgelöst werden.
- ▶ **Stored Procedures:** Gespeicherte Prozeduren sind Programmmodule, die sich auf dem Datenbank-Server befinden und über SQL-Befehle aufgerufen werden können.
- ▶ **UDT:** Benutzerdefinierte Datentypen. (User Defined Types).
- ▶ **UDF:** Benutzerdefinierte Funktionen (User Defined Functions) können in einer SQL-Anweisung in Kombination mit oder anstelle von Spalten verwendet werden und erweitern den SQL-Sprachumfang.
- ▶ **UDTF:** Benutzerdefinierte Tabellenfunktionen (User Defined Table Functions) können in einer SQL-Anweisung anstelle von Tabellen stehen.
- ▶ **Ablaufpläne (Packages; dt. Pakete):** Pakete werden benötigt, um in der Datenbank entsprechende Zugriffspläne bekannt zu machen.
- ▶ **Bufferpools:** (Puffer-Pools) Speicherbereiche der Datenbank.
- ▶ **Views:** Logische Sichten auf Tabellen.
- ▶ **MQTs:** Materialisierte Views oder so genannte materialisierte Abfrage-Tabellen (Materialized Query Tables).
- ▶ **Aliase:** Aliasnamen für Datenbankobjekte.
- ▶ **Schemata:** Kennungen, wie z.B. Benutzer-IDs, dienen unter anderem zur Gruppierung von Datenbankobjekten.
- ▶ **Logfiles:** Protokolldateien der Datenbank.

API (Application Programming Interface)

Allgemein: standardisierte Programmierschnittstellen. Diese Interfaces stellen die Funktionalität des jeweiligen Programms für externe Programme zur Verfügung.

Betriebssystem

Alle Hardware-Einheiten wie Platten, Hauptspeicher, Dateien usw. werden von der DB2 UDB nur über das Betriebssystem angesprochen. Eine Ausnahme bildet der Zugriff auf Daten über unformatierte Einheiten (raw devices). Das Betriebssystem stellt seine Funktionalität nach außen über eine standardisierte API-Schnittstelle zur Verfügung.

Platteneinheiten und Dateien

Sie dienen zur physischen Speicherung aller logischen Objekte.

1.5 Unterschiedliche Einsatzgebiete

Ein DBMS kann in unterschiedlichen Systemumgebungen bzw. Einsatzgebieten verwendet werden:

- ▶ Einzelplatz-Einsatz auf einem Einzelrechner (Standalone)
- ▶ im homogenen Netzwerkbetrieb
- ▶ im heterogenen Netzwerkbetrieb
- ▶ im Internetbetrieb

An dieser Stelle möchten wir noch etwas näher erläutern, was wir unter den unterschiedlichen Einsatzgebieten verstehen.

1.5.1 DB2 im Einzelplatz-Einsatz auf einem Standalone-Rechner

Im Einzelplatz-Einsatz läuft das DBMS auf demselben Rechner wie alle Programme, die Anforderungen an das DBMS senden. Das bedeutet, die Hardwareressourcen wie CPU, Hauptspeicher, Platteneinheiten usw. werden sowohl von dem DBMS als auch von den Datenbankprogrammen geteilt. Somit kann der Datenaustausch zwischen dem anfordernden Programm und dem DBMS über einen gemeinsamen Hauptspeicher erfolgen.

Folgende Betriebssysteme werden als Einzelplatz-Betriebssysteme unterstützt:

- ▶ Windows-Derivate wie Windows 2000, Windows NT, Windows XP und Windows .NET
- ▶ OS/2 (weitere Unterstützung ist nur über entsprechende Fixpacks von DB2 UDB Version 7.x verfügbar)
- ▶ Linux

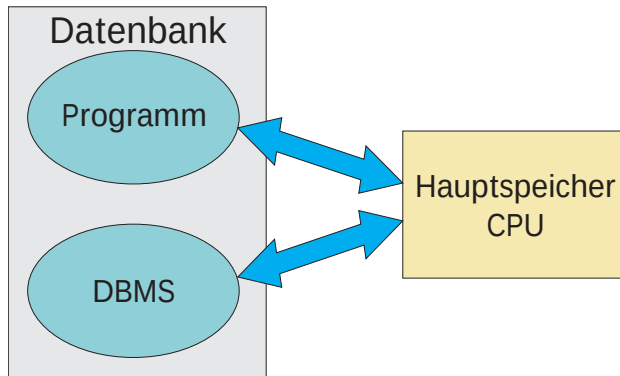


Abbildung 1.2: DBMS im Einzelplatz-Einsatz (siehe Abbildung 1.6 Punkt ①)

1.5.2 DB2 im homogenen Netzwerkbetrieb

Als *homogenen Netzwerkbetrieb* bezeichnen wir ein Netzwerk, in dem sowohl die als Server verwendeten Betriebssysteme als auch die auf dem Client verwendeten Betriebssysteme aus Sicht des DBMS zueinander homogen sind. *Homogen* bedeutet: Das DBMS verhält sich für alle diese verwendeten Betriebssysteme gleich und benötigt somit identische Produktkomponenten.

Folgende Betriebssysteme können als Betriebssystem für den DB2-Datenbankserver verwendet werden:

- ▶ Windows-Server-Derivate wie Windows 2000, Windows NT und Windows .NET
- ▶ die verschiedenen Unix-Derivate wie AIX, Linux, Sun Solaris, HP-UX
- ▶ OS/2 (Unterstützung ist nur bis DB2 UDB Version 7.x verfügbar)

Bei diesen Server-Betriebssystemen handelt es sich um genau die Betriebssysteme, die denselben Datenbankkern besitzen.

Folgende Client-Betriebssysteme können für das DBMS als homogene Client-Betriebssysteme bezeichnet werden:

- ▶ Windows-Derivate wie Windows 3.1, Windows XP, Windows 95, Windows 98, Windows 2000, Windows NT und Windows .NET
- ▶ die verschiedenen Unix-Derivate wie AIX, Linux, Sun Solaris, HP-UX
- ▶ OS/2
- ▶ Mac OS

Im homogenen Netzwerkbetrieb können die Programme, die Anforderungen an das DBMS senden, sowohl auf derselben Maschine als auch auf einem separaten, über das Netz verbundenen Rechner (Client) laufen.

Befindet sich das anfordernde Programm auf derselben Maschine wie das DBMS, spricht man bei dem anfordernden Programm von einer Server-Anwendung **bzw. Server-Applikation**.

Sollte das Programm, das die Anforderungen an das DBMS schickt, von einem über das Netz verbundenen Rechner ausgeführt werden, spricht man bei dem anfordernden Programm von einer *Client-Applikation*.

Dabei spielt es keine Rolle, ob es sich bei dem Rechnersystem, auf dem die DB2-Client-Applikation läuft, um einen Server (z.B. Applikationsserver oder Datei-Server) oder um einen tatsächlichen Client handelt. Für das DBMS ist diese Applikation immer eine DB2-Client-Applikation.

Da sich die DB2-Client-Applikation nicht auf demselben Rechnersystem wie das DBMS befindet, können sich diese beiden Programme auch keine Hardwareressourcen teilen. Der Datenaustausch über einen gemeinsamen Hauptspeicherbereich (shared memory) ist also nicht möglich.

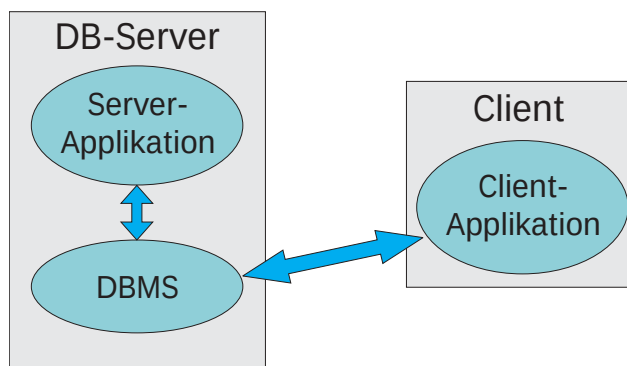


Abbildung 1.3: DBMS im homogenen Netzwerkeinsatz (siehe Abbildung 1.6 Punkt ②)

1.5.3 DB2 im heterogenen Netzwerkbetrieb

Als *heterogenen Netzwerkbetrieb* bezeichnen wir ein Netzwerk, in dem sowohl die als Server verwendeten Betriebssysteme als auch die vom Client verwendeten Betriebssysteme aus Sicht des DBMS zueinander *nicht homogen* sind. Das bedeutet, das DBMS verhält sich für die im Netzwerk verwendeten Betriebssysteme nicht gleich (nicht homogen). Dies macht den Einsatz weiterer Produktkomponenten notwendig.

Wann ist eine Betriebssystemumgebung nicht mehr homogen?

Dies ist der Fall, wenn sich in dem Netzwerk zusätzlich zu den als homogen bezeichneten Betriebssystemen folgende Systeme befinden:

- OS/400
- z/OS, OS/390, VM und VSE

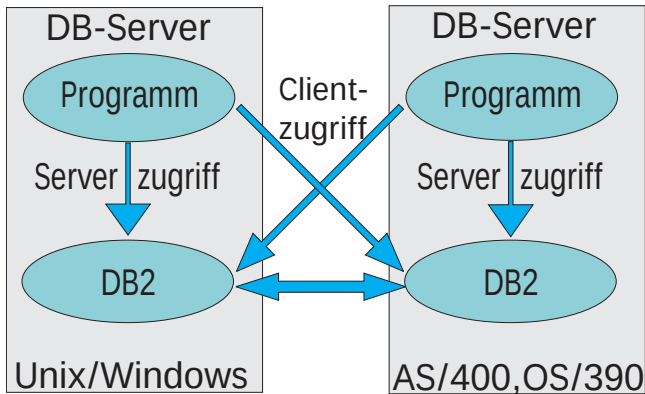


Abbildung 1.4: DB2 im heterogenen Umfeld (siehe Abbildung 1.6 Punkte ③ und ④)

1.5.4 DB2 im Internetbetrieb

Für den Internetbetrieb können alle der oben genannten Systemumgebungen (Einzelplatz, homogene Netzwerkumgebung, heterogene Netzwerkumgebung) eingesetzt werden. Rein technisch sind keine weiteren DB2-Komponenten erforderlich.

Vorsicht:

Falls der Applikationsserver und das DBMS auf ein und demselben Rechner laufen, wäre rein technisch eine Einzelnutzer-Lizenz einsetzbar. Lizenztechnisch ist das aber nicht korrekt. Welche Lizenzen zu verwenden sind, werden wir in Kapitel 2 behandeln.

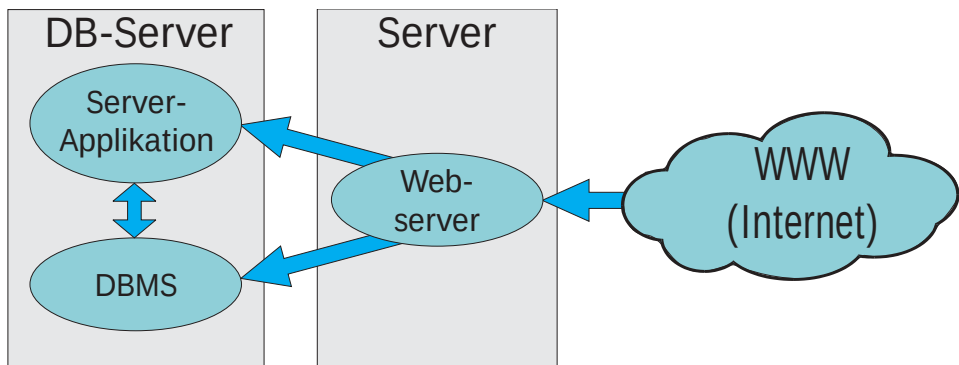


Abbildung 1.5: DB2 im Internet-Umfeld

1.5.5 Produktkomponenten der unterschiedlichen Einsatzgebiete – Gesamtübersicht

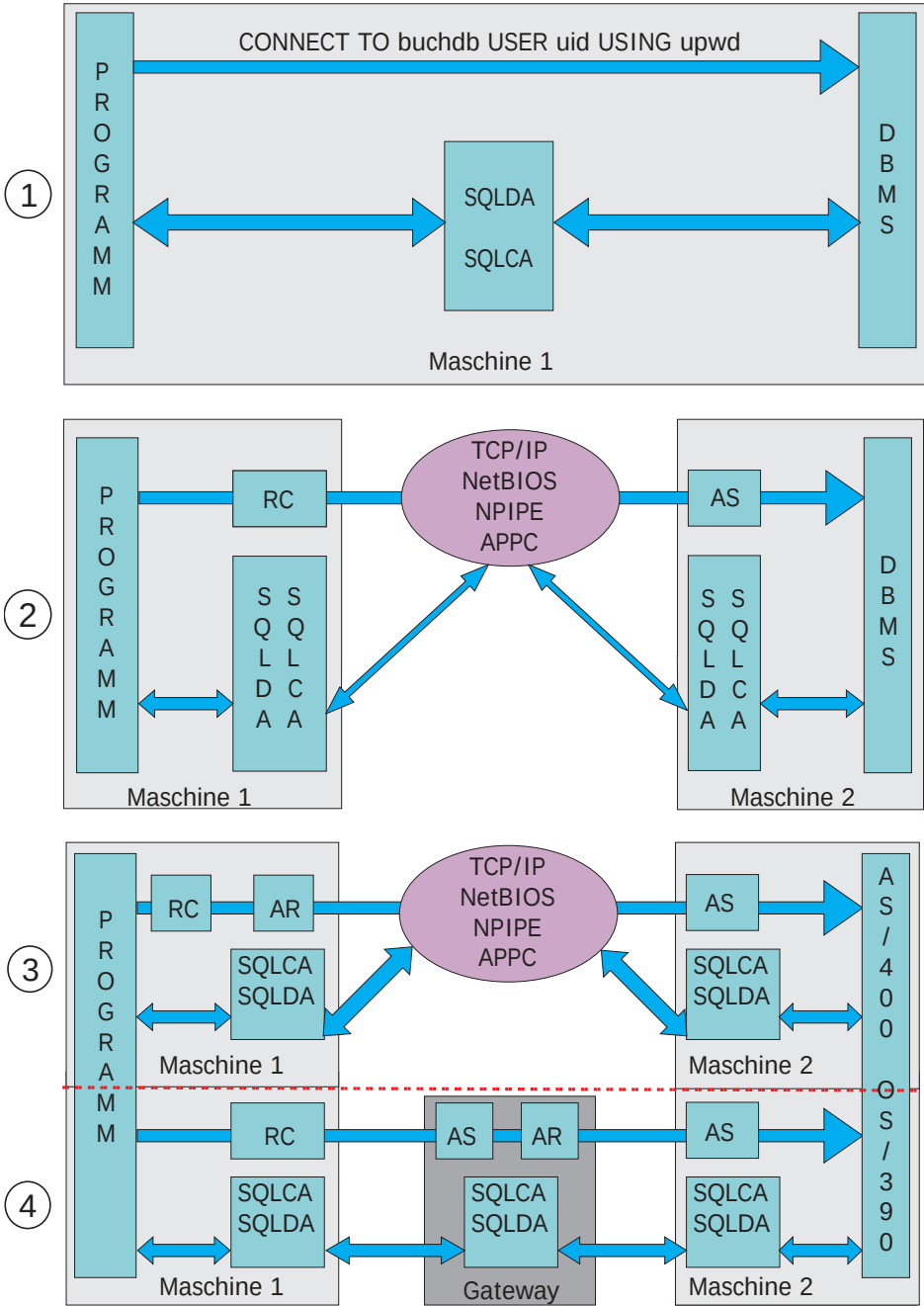


Abbildung 1.6: Produktkomponenten der unterschiedlichen Einsatzgebiete

SQLDA

Die *SQL Data Area*, kurz SQLDA genannt, beinhaltet sowohl die Beschreibung der Daten (Datentyp, Länge, Pointer auf den Dateninhalt) als auch den eigentlichen Dateninhalt.

SQLCA

Die *SQL Communication Area*, kurz SQLCA, dient zur Übermittlung der Kommunikationsdaten, z.B. von Fehlermeldungen, Warnhinweisen und weiteren Informationen.

RC

Der *Run-Time Client* simuliert sozusagen das auf dem Client nicht vorhandene DBMS. Er nimmt den *CONNECT*-Befehl entgegen und schaut in den entsprechenden Katalogeinträgen nach, auf welchem Server sich die Datenbank befindet (Näheres siehe Kapitel 6, „Aufbau verschiedener Datenbankverbindungen“). Über das konfigurierte Übertragungsprotokoll nimmt er dann mit dem entsprechenden Agenten (AS – Applikationsserver) auf dem Datenbank-Server Kontakt auf und stellt über diesen die Verbindung zu dem Ziel-DBMS her.

Das Anwendungsprogramm baut diesem Fall den Speicherbereich für die SQLCA und die SQLDA mit dem RC auf. Der RC wiederum kopiert diese Daten dann über den AS zum entsprechenden Datenbank-Server.

AS

Der Applikationsserver ist sozusagen der Server-Agent für alle RCs (Run-Time Clients), die sich mit dem Datenbank-Server verbinden. Er besitzt mit dem DBMS den Server-Speicherbereich für die SQLDA und die SQLCA und tauscht Daten über das verwendete Protokoll (TCP/IP, NetBios, SNA, NPIPE) mit dem Speicherbereich des RC auf dem Datenbank-Client aus.

Der AS ist die Komponente, die für Remote-Zugriffe auf das DBMS notwendig ist. Er hat Zugriff auf dieselben Hardwareressourcen wie das DBMS. Dadurch wird der Datenaustausch über shared memory möglich.

Vorsicht:

Im Gegensatz zum Einzelplatz-System sind hier jetzt zwei Speicherbereiche notwendig, um die Daten zwischen dem DBMS und dem anfordernden Programm auszutauschen.

Dieser Zusatzaufwand darf in Performance-Betrachtungen nicht außer Acht gelassen werden.

DRDA AR (Application Requestor)

Wie schon erwähnt, besitzen in heterogenen Umgebungen die Datenbank-Server verschiedene Datenbankkerne und verhalten sich nicht nur für den Administrator, sondern auch für den Benutzer und den Programmierer unterschiedlich.

Ebenso sind weitere Unterschiede wie eventuell verschiedene Zeichendarstellungen (EBCDIC-Code oder ASCII-Code), Sicherheitsprüfungen usw. vorhanden.

Um Datenbanken unterschiedlicher Systeme miteinander zu verbinden und diese Unterschiede der Systeme nicht für den Anwender (hier sind alle Nutzergruppen wie Administratoren, Programmierer usw. gemeint) transparent zu machen, wurde das DRDA-Protokoll (Distributed Relational Database Architecture) entwickelt. Diese Architektur wird auf dem eigentlichen Übertragungsprotokoll (z.B. TCP/IP) eingesetzt und besitzt fünf verschiedene Transformationsschichten.

An Abbildung 1.7 sollen die Transformationsschichten des DRDA-Protokolls kurz erläutert werden.

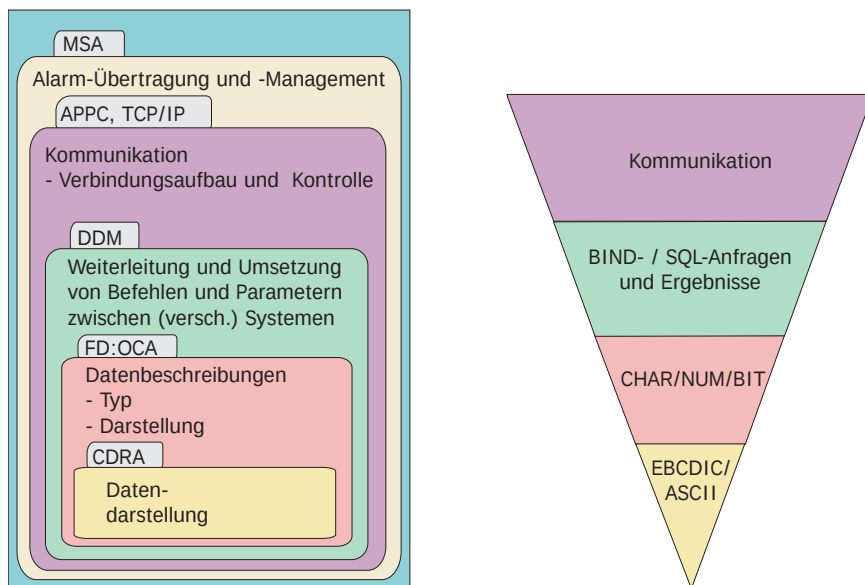


Abbildung 1.7: DRDA-5-Schichten-Architektur

Die Trennung der unterschiedlichen Aufgaben in verschiedene Transformationsschichten ermöglicht eine größtmögliche Flexibilität und eine Anbindung von Anwendungen an die unterschiedlichsten Datenbanksysteme.

MSA (Management Services Architecture)

Diese Schicht übernimmt den Transfer und das Management von Fehlern.

Protokollschicht

Die Protokollschicht übernimmt die Unterstützung und das Management der folgenden beiden Übertragungsprotokolle:

- ▶ SNA (System Network Architecture) bzw. APPC (Advanced Program to Program Communication)
- ▶ TCP/IP (Transmission Control Protocol/Internet Protocol)

DDM (Distributed Data Management Architecture)

Diese Schicht ermöglicht einen Datenaustausch zwischen bekannten und unbekannten Systemen. Unterschiede zwischen den unterschiedlichen Befehlen, Parametern, Datenobjekten und Meldungen der ungleichen Systeme werden in dieser Schicht ausgeglichen.

FD:OCA (Formatted Data: Object Content Architecture)

Unterschiedliche RDBMS besitzen Unterschiede in der Formatierung und der Beschreibung der Attribute. Diese Unterschiede sollen durch diese Schicht eliminiert werden.

CRDA (Character Data Representation Architecture)

Die Nutzung verschiedener Code-Pages und Zeichensätze für grafische und Character-Daten kann durch diese Schicht ausgeglichen werden.

Hinweis:

Ab der Version 8 setzt DB2 UDB für den Datenaustausch zwischen Datenbank-Clients und Datenbank-Servern immer auf DRDA auf. Der DRDA AR übernimmt also auf der Client-Seite die Aufgabe der DRDA-Umsetzung.

Bis zur DB2 UDB V7 wurde die DRDA-Funktionalität im normalen RC (Run-Time Client) für die Anbindung im homogenen Datenbanknetz nicht benötigt, und die Komponente DRDA AR (**D**istribute **R**elational **D**atabase **A**rchitecture **A**pplication **R**equestor) musste separat erworben werden.

Obwohl diese Komponente (DRDA AR) inzwischen im Produkt integriert ist, ist die Lizenzfrage weiterhin separat zu betrachten.

DRDA AS

Der DRDA AS (Distributed Relational Database Architecture Application Server) bildet auf Seiten des Datenbank-Servers quasi das Gegenstück zum DRDA AR. Er ist auf dem anzusprechenden Datenbank-Server installiert.

Diese Produktkomponente ist eine Standardkomponente von DB2 UDB und Bestandteil aller Server-DBMS-Produktpakete. Dies bedeutet, dass diese Komponente lizenztechnisch *nicht* gesondert zu betrachten ist.

Hinweis:

Um in heterogenen Datenbanknetzen eine Datenbankverbindung von z.B. DB2 für z/OS zu DB2 UDB-Servern auf Windows- oder Unix-Plattformen herzustellen, ist keine zusätzliche DB2 UDB-Komponente oder Lizenz erforderlich.

2 Die DB2 UDB-Produktpalette

IBM bietet Ihnen in Verbindung mit der DB2 UDB eine breite Produktpalette an. Diese Produktpalette kann in drei unterschiedliche Kategorien aufgeteilt werden:

- ▶ verschiedene Produktpakete der DB2 UDB
- ▶ weitere DB2-Produktkomponenten
- ▶ zusätzliche DB2 Information-Management-Produkte

Im Folgenden erhalten Sie einen Überblick über die Unterschiede und Möglichkeiten, die diese Produkte bieten, sowie einen Einblick in das Lizenzmodell der verschiedenen Produktpakete. Dieses Kapitel soll Ihnen die Entscheidung für ein bestimmtes Produkt erleichtern. Beachten Sie, dass IBM gelegentlich seine Lizenzpolitik der entsprechenden Marktsituation anpasst. Somit kann es sich bei unserem Überblick über die Lizenzierung der einzelnen Produkte nur um einen Einblick und nicht um eine rechtlich verbindliche Aussage handeln.

2.1 Übersicht über die DB2-Datenbanksystem-Produkte

Die Anforderungen, die Sie an ein RDBMS stellen, unterscheiden sich je nach Unternehmensgröße, Aufgabengebiet und den verwendeten Hardwareplattformen. So können Sie z.B. einem Mitarbeiter, der eine Inventur des Warenbestands eines großen Einkaufszentrums durchführt, nicht wie einem Außendienstmitarbeiter eines Versicherungsunternehmens einen Laptop als Hilfsmittel zur Verfügung stellen. Auch wenn beide Mitarbeiter nicht online mit ihrem zentralen Datenbanksystem verbunden sind, unterscheiden sich sowohl die Anforderungen als auch die Arbeitsbedingungen stark voneinander. Ebenso spielt es eine Rolle, ob Sie Ihre Daten in einer Online-Verarbeitungssoftware z.B. eines Auftragserfassungssystems oder für verschiedene Online-Analysen Ihres weltweit zusammengeführten Datenpools benötigen.

Die Praxis zeigt, dass eines der wichtigsten Auswahlkriterien für ein DBMS dessen Skalierbarkeit ist. Diese Eigenschaft sorgt dafür, dass Ihre Systeme mit den jeweiligen Anforderungen an das Datenvolumen und der entsprechenden Anzahl von Benutzern, die Ihre Daten nutzen, wachsen können, ohne dass Sie Ihre Applikationen ändern müssen.

Ein weiterer wichtiger Punkt ist, dass Ihre Daten sehr oft nicht nur auf einem System, sondern auf mehreren verteilten Systemen zum Lesen und Ändern zur Verfügung stehen müssen. Dabei ist es häufig der Fall, dass diese unterschiedlichen Datenbanksysteme nicht ständig miteinander verbunden sind. Daher muss die heutige Datenhaltung den asynchronen Austausch der Daten unterschiedlicher Datenbanksysteme ebenfalls bewältigen können (wobei sich durchaus auch das verwendete Betriebssystem unterscheiden kann).

Um dieser Herausforderung gerecht zu werden, wird Ihnen die DB2 UDB für die unterschiedlichsten Betriebssystem- und Hardware-Landschaften angeboten. Damit ist eine Skalierung der Datenverarbeitung vom PDA (Personal Digital Assistant) bis hin zu einer massiv parallelen Umgebung mit Terabytes auf einer geclusterten Unix-Umgebung möglich.

Verschiedene DB2-Produktpakete ermöglichen es, eine passende Lösung sowohl für die entsprechende Anforderung als auch für das jeweilige Preissegment zu finden. Abbildung 2.2 zeigt Ihnen die Produktpakete, die zur Verfügung stehen. Im Anschluss an Abschnitt 2.1 „Übersicht über die DB2-Datenbanksystem-Produkte“ werden wir Ihnen diese Produktpakete näher erläutern. Zuvor möchten wir Sie jedoch mit Abbildung 2.1 noch einmal explizit darauf hinweisen, dass jede Edition der DB2 Universal Database für dezentrale Plattformen – unabhängig von der verwendeten Plattform – auf derselben Code-Basis beruht.

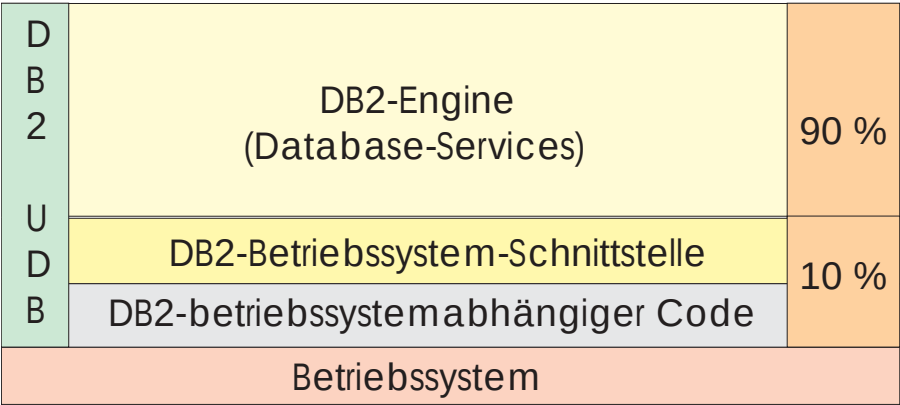


Abbildung 2.1: Code-Basis der DB2 UDB

Diese Technik ermöglicht Ihnen jederzeit die problemlose Portierung Ihres Datenbestands von einem Hardwaresystem bzw. dezentralen Betriebssystem auf ein anderes. Dabei ist sichergestellt, dass die Zugriffe aus Ihrer Anwendungssoftware heraus auf die DB2 UDB nicht verändert bzw. angepasst werden müssen. Ihre Anwendung kann also nach wie vor z.B. auf einem Rechner mit dem Betriebssystem Windows laufen, während sich Ihre Datenbank und Ihr Datenbanksystem auf z.B. einem Rechnersystem von Sun befinden können.

Ebenso ist es jederzeit denkbar, Ihre Datenbank unter dem System z/OS und Ihre Applikation unter Windows zu betreiben. Hier müssen Sie jedoch Folgendes beachten:

- In diesem Fall werden, zumindest lizentechnisch, zusätzliche Produktpakete wie DB2-Connect benötigt.
- Falls nicht schon im Vorfeld auf verschiedene Punkte wie z.B. verwendete SQL-Standards bzw. Performance und Sicherheit (wie z.B. statisches SQL, unnötiger Netzwerkverkehr) geachtet wurde, müssen diese bei der Portierung berücksichtigt werden.

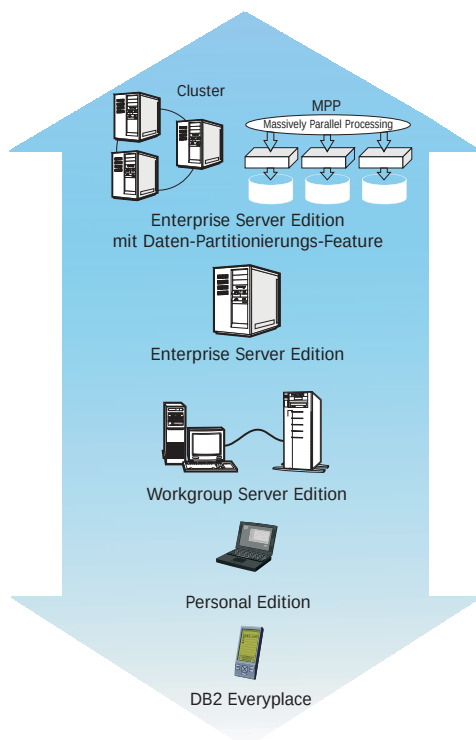


Abbildung 2.2: DB2-Datenbanksystemprodukte

2.1.1 DB2 UDB Everyplace Edition

Die Vorteile des *Mobile Computing* kann ein Unternehmen im Regelfall erst dann richtig nutzen, wenn die auf den mobilen Rechnern befindlichen Anwendungen auf die entsprechenden Unternehmensdaten zugreifen können. Dies setzt entweder

- ▶ eine Synchronisation der Daten zwischen den mobilen Endgeräten und den im Unternehmen befindlichen Datenbankservern voraus oder
- ▶ einen direkten mobilen Online-Zugriff auf die im Unternehmen befindlichen Datenbankserver.

Die DB2 unterstützt mit dem Produktpaket *UDB Everyplace Edition* das Mobile Computing oder „Pervasive Computing“, wie es neudeutsch so schön heißt. Dahinter verbirgt sich – einfach gesprochen – die Unterstützung für PDAs (Personal Digital Assistants) und Pocket-PCs. Somit ermöglicht es DB2 UDB Everyplace, die Vorteile eines relationalen Datenbankmanagementsystems auf kleinstmöglichen mobilen Geräten zur Verfügung zu stellen und die Daten bei Bedarf mit anderen Datenbanksystemen zu synchronisieren bzw. den direkten Online-Zugriff auf dieses System. Sie können somit Ihre Unternehmensdaten sozusagen in Ihre Tasche stecken. Wo auch immer Sie sich befinden, können Sie Ihre Daten verändern und ergänzen, um diese zu einem späteren Zeitpunkt in die zentrale Datenhaltung Ihres Unternehmens zurückzuspielen oder online zu pflegen.

DB2 Everyplace ist eine Untermenge der DB2 UDB. Es stellt Ihnen neben einer etwas verringerten Funktionalität bzw. einem geringeren SQL-Befehlssatz quasi ein Mini-RDBMS mit nur ca. 180 Kbyte Größe und gewissen Sicherheitsfunktionen zur Verfügung. Es unterstützt diverse Programmierschnittstellen (ODBC, JDBC, CLI), und Sie können Ihre Daten nahtlos in Ihr DB2-System replizieren.

Die DB2 Everyplace ist außerdem:

- ▶ sehr einfach zu bedienen – der Anwender benötigt keinerlei Wissen über Datenbankmanagementsysteme.
- ▶ eine komplette Entwicklungsumgebung.

Die DB2 Everyplace unterstützt folgende Systeme:

- ▶ Palm OS
- ▶ Microsoft Windows CE/PocketPC
- ▶ jedes 32-Bit-Betriebssystem von Microsoft
- ▶ Symbian
- ▶ QNX Neutrino
- ▶ JAVA 2 Micro Edition (J2ME)
- ▶ Embedded Linux (z.B. BlueChat Linux)

Hinweis zur Lizenzierung

- ▶ DB2 Everyplace kann als voll synchronisierte Umgebung oder als allein stehendes eingebundenes RDBMS lizenziert werden.
- ▶ DB2 Everyplace ist in zwei Versionen erhältlich:

DB2 Everyplace Database Edition für die mobilen Endgeräte:

- ist pro registriertem Benutzer zu lizenzieren.
- bietet keine Middleware zur Synchronisierung, um die Daten mit Datenbankservern Ihres Unternehmens abzugleichen.
- verfügt über befehlszeilengesteuerte Import- und Export-Möglichkeiten, um Daten auszutauschen.
- wird hauptsächlich genutzt, um in einer Anwendung eingebundene relationale Datenbanken zur Speicherung von Daten zu nutzen.
- Spezielle OEM-Preise für ISV (independent software vendor) sind erhältlich.

DB2 Everyplace Enterprise Edition:

- wird pro Prozessor auf dem jeweils installierten Rechnersystem lizenziert.
- beinhaltet für die Installation auf den mobilen Endgeräten eine unbegrenzte Anzahl der DB2 Everyplace Database Edition-Lizenzen.
- beinhaltet den DB2-Synchronisationsserver zur Synchronisation Ihrer Unternehmensdaten. Dabei werden alle JDBC-konformen Datenbanken wie z.B. DB2, Informix, Oracle und Microsoft SQL Server unterstützt.

Tipp:

Falls Ihre Anwendung keine datenbankspezifischen Funktionen, wie z.B. Trigger, sondern nur einfache SQLs nutzt, können Sie DB2 Everyplace auch auf Laptops verwenden.

2.1.2 DB2 UDB Personal Edition (DB2 PE)

Für die Installation auf Einzelplatzrechnern steht Ihnen die *DB2 UDB Personal Edition* zur Verfügung. Sie ist ein vollwertiges DB2 UDB-Produkt, das alle Tools und Grundfunktionalitäten der DB2 Workgroup Server Edition enthält. Der einzige Unterschied besteht darin, dass die Datenbanken, die sich auf dem Rechnersystem befinden, auf dem die DB2 UDB PE installiert ist, nicht von entfernten Clients angesprochen werden können. Somit können nur lokale Anwendungen auf diese Datenbanken zugreifen.

Da die DB2 UDB PE jedoch über die volle Funktionalität eines DB2-Run-Time-Client verfügt, können Sie von diesem System aus jederzeit Datenbanken auf anderen DB2 UDB-Datenbankserver-Systemen ansprechen und verwalten.

Da die DB2-Engine aller DB2-Datenbanksystemprodukte identisch ist, können Anwendungen, die für eine DB2 PE konzipiert und entwickelt worden sind, problemlos ohne Änderungen auf die übrigen DB2-Server-Versionen, wie z.B. die DB2 Workgroup Edition, übernommen werden.

Als Einsatzgebiet eignen sich Einzelplatzsysteme mit lokalen Anwendungen, wie z.B. eine Anwendung, die auf einem Laptop eines Außendienstmitarbeiters installiert ist.

Die DB2 PE unterstützt folgende Systeme:

- ▶ die Windows-Betriebssysteme:
 - Windows 9x
 - Windows ME
 - Windows NT (SP 6)
 - Windows 2000 (SP 2 empfohlen)
 - Windows XP
 - Windows .NET Server werden unterstützt, sobald Microsoft diese verfügbar macht.
- ▶ Betriebssystem Linux (Intel)

Hinweis zur Lizenzierung der DB2 PE

- ▶ Die DB2 PE ist für eine Produktionsumgebung pro Installation zu lizenzieren.
- ▶ In Verbindung mit der *DB2 Personal Developer Edition* (DB2 PDE) ist die DB2 PE kostenlos per Download erhältlich (unter ibm.com/software/data/db2/udb und dann den Download-Links auf der rechten Seite folgen). Diese Entwicklerlizenz ist in ihrer Nutzung jedoch eingeschränkt für:

- die Entwicklung von ISV-Software
- Demonstrations- und Testzwecke

Sie kann nicht dazu verwendet werden, um für oder im Auftrag eines Unternehmens eine Anwendungsentwicklung auf Basis von DB2 durchzuführen.

Beispiel:

Firma X beauftragt Firma Y, eine Applikationssoftware auf Basis von DB2 für den produktiven Einsatz im Unternehmen X oder Z zu entwickeln.

- ▶ Im Gegensatz zur Version 7.2
 - gehört der XML-Extender zum Lieferumfang.
 - können 64-Bit-Exemplare erstellt werden.
 - wurde die *DB2 Satellite Edition* integriert.
 - kann die DB2 PE in Verbindung mit den DB2-Spatial- und -Net Search-Extendern eingesetzt werden.

Hinweis:

Die DB2 PE kann sowohl auf Einzelplatz-PCs als auch auf PCs eingesetzt werden, die sich im Netz befinden.

Durch die integrierte Funktionalität

- ▶ des Run-Time-Clients kann die DB2 PE jederzeit als Datenbank-Client mit lokalen Datenbanken genutzt werden.
- ▶ des Replikationsmechanismus können über diesen DB2-Daten sowohl mit entfernten Datenbankservern als auch mit Einzelplatzrechnern ausgetauscht werden.
- ▶ des DB2 Warehouse Center kann die DB2 PE als Entwicklungsplattform zur Modellierung von Data-Warehouse-Funktionalitäten genutzt werden.

Die Datenbanken der DB2 UDB können über den DAS (Database Administration Server) remote administriert werden.

2.1.3 DB2 UDB Express Edition

Das neueste Mitglied in der DB2-Produktfamilie soll besonders für kleinere Firmen und den Mittelstand eine attraktive Alternative bieten – sowohl von der Leistungsfähigkeit als auch von der preislichen Gestaltung her.

Hinter der *DB2 UDB Express Edition* verbirgt sich die DB2 WSE in ihrem vollen Funktionsumfang. Einschränkungen zu der folgenden DB2 WSE-Version sind rein lizentechnisch.

Hinweis:

Sie finden die Version DB2 UDB Express Edition als Testlizenz auf der CD-ROM zum Buch.

Die DB2 UDB Express Edition unterstützt folgende Systeme:

- ▶ die Windows-Betriebssysteme:
 - Windows 9x
 - Windows ME
 - Windows NT (SP 6)
 - Windows 2000 (SP 2 empfohlen)
 - Windows XP
 - Windows .NET Server werden unterstützt, sobald Microsoft diese verfügbar macht.
- ▶ Betriebssystem Linux (Intel)

Hinweis zur Lizenzierung der DB2 UDB Express Edition

- ▶ Die Serverversion ist in ihrer Nutzung beschränkt auf:
 - maximal zwei Prozessoren pro installiertem Rechnersystem
 - eine 32-Bit-Version.
- ▶ Die Benutzer können nur als registrierte Benutzer lizenziert werden. Eine Concurrent-User-Lizenzierung steht nicht zur Verfügung. Somit ist eine Nutzung für Internet-Benutzer nicht möglich.

2.1.4 DB2 UDB Workgroup Server Edition (DB2 WSE bzw. WSUE)

Wie der Name bereits andeutet, stellt die DB2 WSE eine DB2-Server-Variante dar. Sie besitzt denselben Funktionsumfang wie die DB2 PE, ist jedoch im Gegensatz zu dieser über entfernte Clients ansprechbar. Dieser Remote-Zugriff wird durch die in allen DB2-Server-Versionen zusätzlich vorhandene technische Komponente ermöglicht: den Applikations-Server, der den einzigen technischen Unterschied zur PE darstellt (siehe Abschnitt 1.5.5). Ab dieser Version sind je nach gewählter Lizenzierung die Zugriffsarten *Internet*, *Intranet* und *normale Netzwerkzugriffe auf den Datenbankserver* lizenziert. Rein technisch sind alle Zugriffsarten unabhängig von der Lizenzierung möglich. Das bedeutet, IBM hat hier keine technische Sperre eingebaut, die sich nach einer erworbenen Lizenz richten könnte.

Die DB2 WSE unterstützt folgende Betriebssysteme:

- ▶ Windows:
 - Windows NT (SP 6)
 - Windows 2000 (SP 2 empfohlen)
 - Windows XP (Professional)
 - Windows .NET Server werden unterstützt, sobald Microsoft diese verfügbar macht.
- ▶ Linux (Intel)
- ▶ AIX 4.3 und 5L

- ▶ Solaris
- ▶ HP-UX

Hinweis zur Lizenzierung der DB2 WSE und DB2 WSUE

Bei der DB2 WSE gibt es zwei unterschiedliche Lizenzierungsarten:

- ▶ *Benutzerbasierende Lizenzierung.* Hier spricht man von der DB2 WSE, und es gilt:
 - Jeder Server, auf dem die DB2 WSE installiert ist, muss lizenziert werden.
 - Die Benutzer können wie folgt lizenziert werden:

- *Als Concurrent-Benutzer*

Die Anzahl der Concurrent-Benutzer ergibt sich aus der Anzahl der maximalen Connections, die am Datenbankserver anliegen.

Besitzt ein Benutzer gleichzeitig mehrere Verbindungen zu einem einzigen Server, zählt nur ein Concurrent-Benutzer.

Liegt der Fall vor, dass verschiedene Verbindungen zu der DB2 WSE von einem Multiplexing-Programm, Anwendungsserver oder anderen Programmen, die ähnlich wie ein Transaktionsserver arbeiten, hergestellt werden, kann diese Lizenzierungsart nicht verwendet werden.

Greift ein Concurrent-Benutzer auf mehrere Datenbankserver zu, muss er pro Server lizenziert werden.

- *Als registrierte Benutzer*

Hier müssen alle Benutzer bekannt sein. Dies sind im Prinzip alle Benutzer, die auf die DB2 WSE zugreifen können.

Da bei Internet-Anwendungen weder die maximale Anzahl der Benutzer noch die Namen der Benutzer vorherbestimmt werden können, ist diese Lizenzierungsart für die Internetnutzung nicht zulässig.

- ▶ *Prozessorbasierende Lizenzierung.* Hier spricht man von der DB2 Workgroup Server Unlimited Edition (WSUE). Es gilt:

- Jeder Server, auf dem die DB2 WSUE installiert ist, muss nach der Anzahl der Prozessoren lizenziert werden.

Diese Lizenzierungsart eignet sich für folgende Fälle:

- Eine Datenbankanwendung stellt ihre Daten über das Internet zur Verfügung.
 - Die Anzahl der Concurrent-Benutzer ist so groß, dass der Break-Even-Point erreicht ist und somit eine Lizenzierung nach Concurrent- oder Named-Benutzern teurer wäre als die Lizenzierungsart nach Prozessoren.

Für beide Lizenzierungsarten gelten folgende weitere Einschränkungen:

- ▶ Es werden maximal bis zu vier Prozessoren unterstützt.
- ▶ Die Erstellung von 64-Bit-Exemplaren wird nicht unterstützt.

2.1.5 DB2 UDB Enterprise Server Edition (DB2 ESE)

Als Erweiterung zur DB2 UDB WSE beinhaltet diese nur nach Prozessoren lizenzierte Servervariante die DB2-Connect-Komponente, um Datenbankverbindungen zum Großsystem (z.B. DB2 für z/OS oder DB2 für AS/400) herzustellen.

Nur in diesem DB2-Server-Produkt ist es möglich, neben der 32-Bit- auch eine 64-Bit-Version zur Verfügung zu stellen.

DB2 UDB ESE mit Partitioning Feature

Mit dem *Partitioning Feature* können Sie eine partitionierte Datenbank über mehrere physikalische Maschinen, auf einer Maschine oder auch in einem Cluster erstellen. Dieses Feature

- ▶ ist automatisch in jedem DB2 ESE-Produkt vorhanden
- ▶ muss lizenztechnisch extra erworben werden.

Die DB2 ESE unterstützt folgende Betriebssysteme:

- ▶ Windows:
 - Windows NT (SP 6)
 - Windows 2000 (SP 2 empfohlen)
 - Windows .NET Server werden unterstützt, sobald Microsoft diese verfügbar macht.
- ▶ Linux (Intel)
- ▶ Linux (zSeries)
- ▶ AIX 4.3 und 5L
- ▶ Solaris
- ▶ HP-UX

Hinweis zur Lizenzierung der DB2 ESE

- ▶ Es ist ausschließlich eine *prozessorbasierende Lizenzierung* möglich.
- ▶ Eine Beschränkung auf die Anzahl der Prozessoren des jeweiligen Datenbankservers ist nicht gegeben.
- ▶ Die Produktkomponente *DB2 Connect Enterprise Edition* ist für bis zu fünf Concurrent-Benutzer mit lizenziert. Weitere Connect-Benutzer müssen z.B. über weitere Concurrent- oder registrierte Benutzer und die entsprechende DB2 Connect EE lizenziert werden.
- ▶ Es können jederzeit 64-Bit-Exemplare erstellt werden.
- ▶ Wird das DPF (Database Partitioning Feature) eingesetzt, muss es pro Prozessor zusätzlich lizenziert werden.

Hinweis:

Die DB2 ESE ermöglicht für die Unix-basierenden Betriebssysteme die gleichzeitige Installation verschiedener Wartungs-Levels ein und derselben DB2-Version (erst ab Version UDB 8 verfügbar).

2.1.6 Zusammenfassung

Abbildung 2.3 stellt die komplette DB2-RDBMS-Produktpalette und deren Eigenschaften in puncto Lizenzierung auf einen Blick dar.

Produktname	Betriebs-system	Bit-breite	Prozessoren	Benannte Benutzer	Aktive Benutzer	Registrierte Benutzer	Installation	sonstiges
Everyplace								
PE	WIN, Linux	32						
PDE	WIN, Linux	32						nur für Entwicklung
Express Edition	WIN, Linux	32	2					max
Express Server Edition	WIN, Linux	32						
WSE	WIN, UNIX, Linux	32	4					
WSUE	WIN, UNIX, Linux	32						
ESE		32/64						
ESE +p		32/64						+ zu ESE

Abbildung 2.3: Übersicht der Produktpakete und ihrer Eigenschaften in puncto Lizenzierung

2.2 Weitere DB2-Produktkomponenten

Je nach Einsatz- und Aufgabengebiet stehen Ihnen zusätzliche Produktkomponenten zur Verfügung. Diese können Sie als einzeln erhältliche Produktpakete erwerben. In den folgenden Abschnitten erläutern wir die Produkte und ihre Funktionalität, so dass Sie die benötigten Produkte auswählen können. Wir geben auch Hinweise zu deren Lizenzierung.

2.2.1 DB2 Connect

DB2 Connect ermöglicht den lizenztechnisch sauberen Zugriff auf DB2-Datenbanken der Systeme iSeries (AS/400) und zSeries (z.B. z/OS) bzw. auf Verbindungen zu Hostressourcen wie CICS, VSAM oder IMS.

Dabei stehen Ihnen lizenztechnisch vier Varianten zur Verfügung:

► *DB2 Connect Personal Edition (CPE)*

Sie wird für den direkten Zugriff auf Datenbanken der Systeme iSeries und zSeries von einem DB2 UDB-Client eingesetzt und muss pro Installation lizenziert werden.

► *DB2 Connect Enterprise Edition (CEE)*

Sie wird als Gateway-Server für mehrere DB2 UDB-Clients verwendet und nach der Anzahl der Concurrent-Benutzer lizenziert.

► *DB2 Connect Unlimited Edition (CUE)*

Sie wird dann genutzt, wenn die Anzahl der Benutzer so hoch ist, dass der Break-Even-Point der Concurrent-User-Lizenzierung zu dieser Lizenzierungsart erreicht wurde.

Sie müssen jeden DB2-Server-Host und die Anzahl der MSU des jeweiligen DB2-Systems lizenzieren.

► *DB2 Connect Application Server Edition (CAE)*

Diese Lizenzierungsmethode eignet sich, wenn die Datenbankzugriffe über einen Applikationsserver gebündelt werden. Dieser Fall liegt zum Beispiel bei Zugriffen über das Internet vor.

Hier muss unabhängig davon, ob das *DB2 Connect Gateway* auf ein und demselben Rechner wie der Applikationsserver installiert ist, die Anzahl der Prozessoren des Applikationsservers lizenziert werden.

Eine Zusammenfassung der zuvor genannten Lizenzierungsarten sehen Sie in Abbildung 2.4.

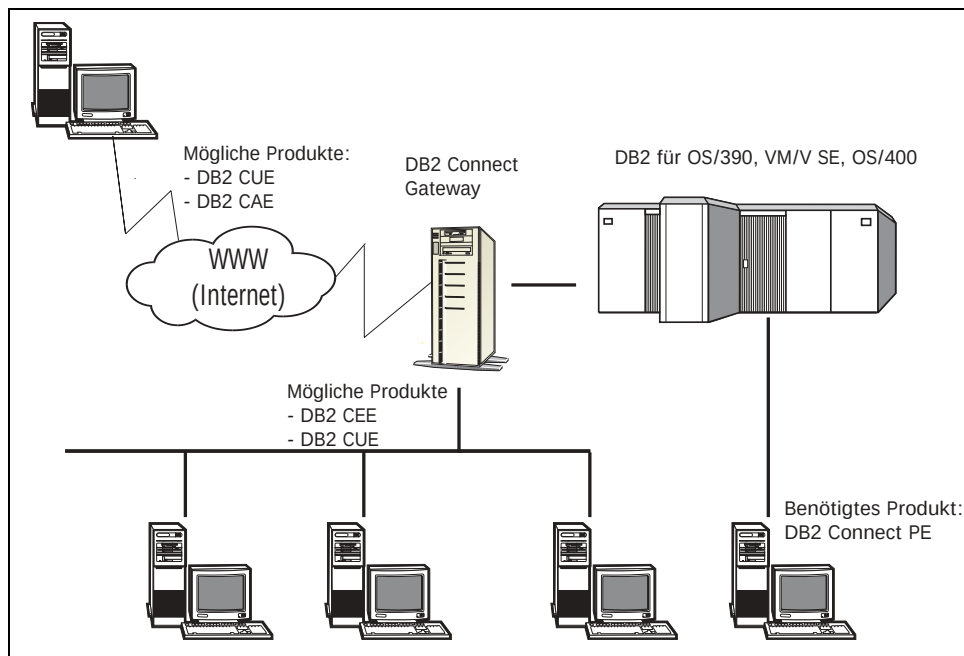


Abbildung 2.4: DB-Connect-Lizenzierungsarten

2.2.2 Entwicklerlizenzen

DB2 Personal Development Edition (PDE)

Dieses Produkt wurde in Abschnitt 2.1.2 „DB2 UDB Personal Edition (DB2 PE)“ unter Abschnitt „Hinweis zur Lizenzierung der DB2 PE“ genau beschrieben.

DB2 Universal Developer Edition (UDE)

Bei diesem Produktpaket handelt es sich um ein stark preisreduziertes Produktpaket für Entwickler. Es beinhaltet bis auf die DB2-ESE-DPF-Komponenten (DPF = Database Partitioning Feature) alle DB2-DBMS-Produktpakete von der DB2 WSE bis hin zur DB2 ESE, alle DB2-Client-Komponenten, die meisten DB2-Extender, DB2 Connect, den *Warehouse Manager* und den *Intelligent Miner*.

Für die Nutzung des Produkts gelten folgende Regeln:

- ▶ Es darf nur für Entwicklungs-, Demonstrations-, Evaluierungs- und Testzwecke eingesetzt werden.
- ▶ Im Gegensatz zur DB2 PDE kann dieses Entwicklerpaket dazu verwendet werden, um für oder im Auftrag eines Unternehmens eine Anwendungsentwicklung auf Basis von DB2 durchzuführen.

Beispiel :

Firma X beauftragt Firma Y, eine Applikationssoftware auf Basis von DB2 für den produktiven Einsatz im Unternehmen X oder Z zu entwickeln.

2.2.3 Relationale Extender

Prinzipiell sind alle Extender-Produkte eine Erweiterung der DB2-Funktionalität und -Datentypen. Diese entsprechende Erweiterung wird unter anderem über entsprechende benutzerdefinierte Funktionen (UDFs), benutzerdefinierte Datentypen (UDTs), gespeicherte Prozeduren (Stored Procedures) und Trigger erreicht.

Somit müssen Sie, um die Erweiterungen zu nutzen, die die jeweiligen DB2 Extender mit sich bringen, nach der Installation der entsprechenden Extender-Produkte im Regelfall gewisse Generierungsscripts zur Erstellung dieser Datenbankobjekte ablaufen lassen.

Net Search Extender

Dieses Produkt ermöglicht eine schnelle Suche über Textdaten, die in DB2 gespeichert sind. Sie können damit nach Volltext, Worten und Sätzen suchen; dabei ist auch die unscharfe Suche möglich. Mit dieser zusätzlichen Produktkomponente können Sie sehr effiziente und komfortable Suchmöglichkeiten in Ihre Web- und E-Commerce-Anwendungen integrieren.

Die Lizenzierung erfolgt pro Prozessor, auf dem der Net Search Extender installiert ist. In der DB2 WSE sind bis zu fünf Benutzerlizenzen kostenlos enthalten. Wird dieses Limit überschritten, erfolgt die Lizenzierung pro Prozessor. Ebenfalls müssen Sie Ihre DB2 WSE-Lizenz auf DB2 WSUE oder DB2 ESE upgraden. Für die Produkt-

pakete DB2 WSUE und DB2 ESE muss der Net Search Extender immer separat lizenziert werden.

Hinweis:

Die zuvor erhältlichen Produktkomponenten *DB2 Net Search Extender* (DB2 NSE) und *DB2 Text Information Extender* (DB2 TIE), die unter der Version DB2 V7 zur Verfügung standen, wurden zu *DB2 Net Search Extender V8* zusammengefasst. Die mächtigeren Suchfunktionen von *DB2 TIE V7* sind nun Teil von *DB2 NSE V8*. Sollten Sie eine linguistische Suche verwenden, müssen Sie weiterhin die *DB2 Text Extender* (DB2 TE) V7 benutzen. IBM hat jedoch vor, die wichtigsten Features der DB2 TE V7 in die DB2 NSE V8 zu integrieren.

Hinweis:

Sollten Sie in der Version DB2 V7 die DB2 TIE oder DB2 TE einsetzen, stellt Ihnen IBM das sonst kostenpflichtige Produkt DB2 NSE V8 kostenlos zur Verfügung.

Für Neukunden ist die DB2 NSE V8-Lizenz kostenpflichtig.

Image-, Audio- und Video-Extender

Diese Extender sind Bestandteil der DB2 UDB und ermöglichen Ihnen den Zugriff auf und die sinnvolle Verarbeitung von Audio-, Video- und Image-Daten Ihrer Datenbank. Sie sind durch diese Extender in der Lage, z.B. folgende Abfragen zu gestalten:

- ▶ „Selektiere alle Hemden, deren Farbe Rot entspricht (Image Extender) oder die einen Rotanteil von mindestens 80 % besitzen.“
- ▶ „Selektiere alle Musiktitel, die sich so ähnlich anhören wie ‚LaLaLa‘ (Audio Extender).“
- ▶ „Selektiere alle Videofilme, deren Spielfilmlänge <120 Minuten ist (Video Extender).“

Die Video-, Audio- und Image-Extender sind Bestandteil der DB2 DBMS-Produktpakete.

DB2 XML Extender

Dieser Extender ermöglicht Ihnen die sinnvolle Speicherung und Verarbeitung von XML-(eXtensible Markup Language)-Dokumenten in Ihrer Datenbank. Dabei stehen Ihnen unter anderem folgende Funktionen zur Verfügung:

- ▶ ein neuer Datentyp zur Speicherung der kompletten XML-Dokumente
- ▶ die Möglichkeit, XML-Dokumente in relationale Bestandteile (Attribute und Tabellen) zu zerlegen (decompose) und in mehreren Tabellen abzuspeichern. Somit können Sie auf die verschiedenen Spalten (Attribute Ihres XML-Dokuments) z.B. Indizes anlegen und weitere SQL-Operationen durchführen. Natür-

lich können Sie das in Spalten und Tabellen zerlegte Dokument wieder als XML-Dokument zusammenführen.

- Ebenso können Sie die erweiterten SQL-Funktionalitäten wie Text- und Sektionssuche auf die eigentliche XML-Spalte (XML-Dokument) oder über DB2 Net Search auf die in Attribute und Tabellen zerlegten XML-Dokumente anwenden.
- Bestehende DB2-Daten können als XML-Dokument ausgegeben werden.

Der XML Extender ist seit der Version 8 Bestandteil der DB2-UDB-Produkte. Abbildung 2.5 gibt Ihnen einen kurzen Überblick darüber, welche Funktionserweiterungen der XML-Extender mit sich bringt.

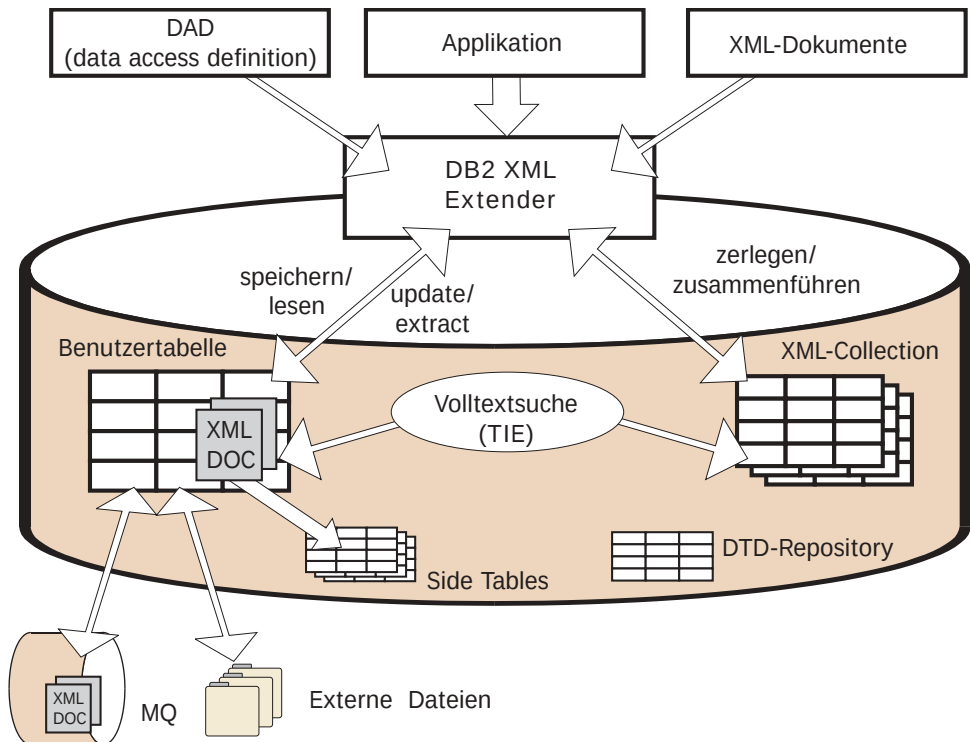


Abbildung 2.5: Übersicht über die Funktionalität des XML-Extender

Hinweis:

Die XML-Extender unterstützen mit dem *Web Service Object Framework (WOF)* eine Reihe von Tools, um die Funktionalität von DB2 Web Services zu ermöglichen.

WebSphere MQ Series-Anwendungen können ebenfalls die Funktionalität der DB2 XML-Extender nutzen.

2.2.4 DB2 Data Links Manager

Wie Abbildung 2.6 zeigt, können Sie mit diesem Zusatzprodukt Daten, die außerhalb des Datenbanksystems in Dateisystemen abgelegt sind, so verwalten, als wären sie ein normaler Bestandteil einer DB2-Datenbank. Der *DB2 Data Links Manager* garantiert referenzielle Integrität auch für diese externen Dateien und bietet eine erweiterte Zugriffskontrolle. Zudem werden eine koordinierte Datensicherung und Wiederherstellung sowie die Replikation ermöglicht.

Die Lizenzierung erfolgt pro Prozessor, auf dem der Data Links Manager installiert ist. In der DB2 WSE sind bis zu fünf Benutzerlizenzen kostenlos enthalten. Wird dieses Limit überschritten, erfolgt die Lizenzierung pro Prozessor. Ebenfalls müssen Sie Ihre DB2 WSE-Lizenz auf DB2 WSUE oder DB2 ESE upgraden. Für die Produktpakete DB2 WSUE und DB2 ESE muss der Data Links Manager immer separat lizenziert werden.

Hinweis:

Dieses Produkt erspart es Ihnen, externe Daten, z. B. von Catia, in Ihre Datenbank kopieren zu müssen, damit diese Bestandteil Ihrer Datenbanken werden.

Somit können diese Daten in Verbindung mit DB2-Funktionalitäten wie z.B. SQL angesprochen werden.

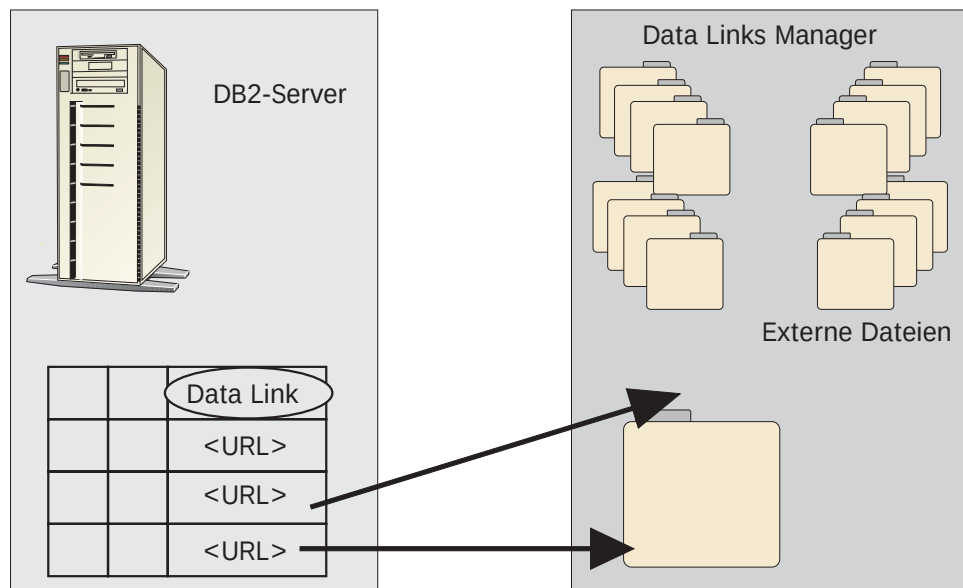


Abbildung 2.6: Data Links Manager

2.3 Zusätzliche DB2-Information-Management-Produkte

In den folgenden Abschnitten geben wir Ihnen einen kurzen Überblick über weitere IBM-Produkte, die sich in die Reihe der Daten verarbeitenden Produkte eingliedern. Dabei wird die Funktion dieser Zusatzprodukte kurz beschrieben.

2.3.1 DB2 Warehouse Manager

Dieses Zusatzprodukt erweitert die Skalierbarkeit, Verwaltungsfähigkeit und Zugriffsmöglichkeit auf ein DB2 Data Warehouse oder Data Mart.

Der DB2 Data Warehouse Manager bietet zusätzliche Funktionalitäten wie:

- ▶ Erweiterung der Funktionalität des Data Warehouse Centers der DB2 UDB z.B. dadurch, dass die Prozesse zur Generierung der Daten aus dem Quell- in das Zielsystem auch auf anderen Rechnern (z.B. AIX) als dem Warehouse Center (z.B. Windows) selbst gesteuert werden können.
- ▶ Folgende Features sind verfügbar:
 - des Query Patroller
 - des Information Catalog
- ▶ Es gibt Konnektoren zu:
 - SAP/R3
Dieser Konnektor ermöglicht die Einbindung der Geschäftsdaten aus einem SAP R3-System (ab Version 4.6C) in das DB2 Data Warehouse. Er steht für die Betriebssysteme Windows NT und Windows 2000 zur Verfügung.
 - dem Web
Dieser Konnektor ermöglicht die Integration von Daten aus einer *WebSphere Site Analyser*-Datenbank (ab Version WSA 3.5) oder einem Web-Mart in ein DB2 Data Warehouse. Er ist für folgende Betriebssysteme verfügbar:
 - Windows NT/2000
 - AIX V4 (32 Bit)
 - Sun Solaris (32 Bit)

Query Patroller

Der *Query Patroller* erlaubt es dem Administrator, Regeln zu bestimmen, nach denen SQL-Abfragen abgearbeitet werden. Dazu gehört z.B., ob eine Abfrage sofort ausgeführt wird oder erst zu einem späteren Zeitpunkt. Diese Abhängigkeiten werden von den geschätzten Kosten, die vom DB2 Optimizer berechnet werden, beeinflusst und sind für die weitere Klassifizierung ausschlaggebend.

Information Catalog Center

Bei diesem Produkt handelt es sich um ein grafisches Tool für den Endbenutzer. Es ermöglicht ihm einen besseren Zugriff, indem er auf relevante Informationen des Data Warehouse einfach zugreifen, sie finden und besser inhaltlich verstehen kann.

Diese Informationen werden dem Benutzer über den Information Catalog zur Verfügung gestellt. Gepflegt wird dieser Information Catalog vom Warehouse-Administrator.

2.3.2 DB2 OLAP Server

Der *IBM DB2 OLAP Server* basiert auf der Essbase OLAP Engine der Firma Hyperion. Er kann sowohl für Enterprise-Management-Informations- und Reporting-Lösungen als auch für Data-Warehouse-Anwendungen und zu Analyse- und Planungszwecken eingesetzt werden.

Der OLAP-Server bietet als einziges Produkt am Markt ein integriertes OLAP-Mining an. OLAP-Mining ist eine Kombination der OLAP-Technologie mit Data-Mining-Algorithmen, über die die Benutzer gezielt auf untypische Datenausprägungen im OLAP-Würfel hingewiesen werden. Des Weiteren ist in der Version 8.1 ein echtes hybrides OLAP realisiert. Das bedeutet, dass Sie Ihre OLAP-Daten sowohl relational als auch multidimensional speichern. Diese Speicherart kann für jede Dimension so flexibel definiert werden können, dass Sie angeben können, für welche Hierarchiestufe die Daten relational gespeichert bzw. ab welcher Aggregationsstufe die Daten multidimensional im OLAP-Würfel gehalten werden. Ein direkter Zugriff (drill-through) aus dem OLAP-Würfel in relationale Detaildaten ist möglich.

DB2 OLAP Integration Server

Er dient zur Definition der relationalen Quelldaten (Star-Schema) und wie sich diese in der Dimension des OLAP-Würfels abbilden. Der OLAP-Metadaten-Katalog im DB2 OLAP-Integrationsserver kann vom DB2 OLAP Server für die Zugriffe auf relational gespeicherte Daten verwendet werden. Eine Verbindung zu mehreren verschiedenen Datenquellen ist möglich.

IBM DB2 OLAP Server Analyzer

Es handelt sich hierbei um ein webbasierendes Frontend-Tool für die Darstellung der Analysen, die aus dem OLAP Server generiert werden.

Dieses Werkzeug ist durch seine intuitive Bedienungsmöglichkeit sehr einfach zu programmieren. Sie können damit Ihre Analysen aus dem OLAP Server über einen Standardbrowser allen Nutzern zugänglich machen.

2.3.3 Die Produktfamilie DB2 Information Integrator

Diese Produktfamilie besteht aus zwei Produkten:

- ▶ *IBM DB2 Information Integrator V8.1* und
- ▶ *IBM DB2 Information Integrator for Content V8.2*.

Beide Produkte ermöglichen es Ihnen, ein gemeinsames Datenmodell über verschiedene verteilte Daten- und Content-Quellen zu legen, auf diese Informationen zuzugreifen und diese zu manipulieren, als wären sie in einem einzigen System gespeichert.

Tipp:

In der heutigen Zeit der Datenflut aus den unterschiedlichsten heterogenen Umgebungen sind beide Produkte sicherlich ein Weg in die richtige Richtung, um ein Unternehmen in die Lage zu versetzen, seine Geschäftsprozesse zu optimieren.

Bei einem richtigen Einsatz des Produkts ist der Return of Investment für ein Unternehmen nahezu garantiert.

IBM DB2 Information Integrator

Dieses Produkt gibt vor allem SQL-Benutzern (Endbenutzern und Anwendungsprogrammierern) die Möglichkeit, über einen *so genannten Federated Data Server auf die unterschiedlichsten Datenquellen* lesend und schreibend zuzugreifen. Dies ermöglicht Ihnen z.B. einen Join innerhalb eines SELECT-Befehls über unterschiedliche Datenbanken unterschiedlicher DBMS hinweg.

Folgende Datenquellen werden unterstützt:

- ▶ relationale Datenquellen wie z.B.:
 - IBM DB2
 - IBM Informix IDS und XPS
 - Microsoft SQL-Server
 - Oracle
 - Sybase
 - Teradata
 - ODBC- und OLE DB-Datenquellen
- ▶ nicht-relationale Datenquellen wie z.B.:
 - Flat Files
 - XML-Dokumente
 - Web Services
 - Microsoft Access
 - Microsoft Excel
 - LDAP-Verzeichnisse
 - WebSphere MQ Message Queues

IBM DB2 Information Integrator for Content

Im Gegensatz zu dem *IBM DB2 Information Integrator* richtet sich dieses Produkt eher an die Benutzer von Content-Management-Systemen. Dieses Produkt ist quasi eine Weiterentwicklung des Produkts *IBM Enterprise Information Portal* und eignet sich insbesondere, wenn Sie:

- ▶ Ihre Informationen in verschiedenen Systemen suchen, wie z.B. Dateisystemen, Lotus Domino-Datenbanken, Microsoft Exchange Server und Web Search Sites. Um dies zu ermöglichen, ist im IBM DB2 Information Integrator for Content der *IBM Lotus Extended Search* integriert.
- ▶ Informationen in verschiedene Workflow-Prozesse integrieren möchten. Es handelt sich bei dieser Funktionalität um eine optional erhältliche Produktkomponente, die unter anderem einen grafischen Workflow-Designer enthält.
- ▶ in Ihren Anwendungen für komplexe Analysen Informationen aus den unterschiedlichsten Quellen zusammenfassen wollen.
- ▶ einen „federated“ Zugriff auf verschiedene Content-Quellen, wie z.B. den IBM Content Manager, Image Plus, Lotus Domino.Doc oder FileNet Panagon Image Services haben möchten.

2.3.4 DB2 Intelligent Miner for Data

Mit diesem Produkt können Sie Daten, die wertvolle geschäftliche Informationen enthalten, in einer großen Menge identifizieren und extrahieren. Das Produkt bietet Ihnen neun verschiedene Mining-Algorithmen, um in Ihren Datenbeständen versteckte Zusammenhänge und Muster zu erkennen.

Vor allem im Bereich Marketing, Cross-Selling, frühzeitige Betrugs- und Missbrauchserkennung, Kundensegmentierung, -bindung und -rückgewinnung findet dieses Produkt ein wichtiges Einsatzgebiet.

Tipp:

Sie werden mit diesem Produkt in die Lage versetzt, aus Vergangenheitswerten Zukunftsprognosen zu erstellen.

2.3.5 DB2 Intelligent Miner Modelling

Bei diesem Produkt handelt es sich um einen DB2-Extender. Er stellt über gespeicherte Prozeduren (Stored Procedures) die Funktionalität zur Erstellung von Mining-Modellen für das demografische Clustering, die Tree-Klassifikation und die Assoziationsanalyse zur Verfügung. Die Modelle werden in einem Format erstellt, das zu dem neuen Industriestandard *Predictive Model Markup Language* (PMML) konform ist.

2.3.6 DB2 Intelligent Miner Scoring

Dieses Werkzeug erweitert die Analysefähigkeiten von DB2 und Oracle und ist wirtschaftlich sowie leicht einsetzbar. Es ermöglicht Ihnen, Data-Mining-Funktionen in Business-Intelligence-, E-Commerce- oder Transaktionsverarbeitungsanwendungen zu nutzen. Ihre Anwendungen können, basierend auf zuvor definierten Data-Mining-Modellen, Daten auswerten, segmentieren, klassifizieren oder einstufen (scoren).

Dieses Produkt steht zur Verfügung für:

- ▶ DB2 UDB ab Version 6 über DB2 Extender
- ▶ Oracle ab Version 8.1.6 als Oracle Cartridge

2.3.7 DB2 Intelligent Miner Visualization

Mit diesem Produkt können Sie Ihre Mining-Analysen über Java-Visualisierer grafisch darstellen.

2.3.8 Multiplattform-Tools

Die folgenden Tools sind für verschiedene Plattformen verfügbar.

DB2 Table Editor

Der *DB2 Table Editor* ermöglicht es Ihnen, Daten in DB2 und auch im IDS (Informix Dynamic Server) einfach zu erstellen, zu aktualisieren und zu löschen.

DB2 Table Editor ist für die Plattformen Windows, Unix, iSeries und zSeries verfügbar.

DB2 Performance Expert

Der *DB2 Performance Expert* meldet, konsolidiert und analysiert DB2-Performance-Daten und macht Vorschläge zur besseren Einstellung der Konfigurationsparameter.

Er ist für die Plattformen Windows, Unix und zSeries verfügbar.

DB2 Recovery Expert

Das *DB2 Recovery Expert* Tool ermöglicht eine einfache und vollständige automatische Wiederherstellung von Daten und bietet umfassende Diagnose- und SMART-Funktionen (Self- Managing And Resource Tuning), um Ausfallzeiten zu minimieren.

Es ist für die Plattformen Windows und Unix verfügbar.

DB2 Web Query Tool

Das *DB2 Web Query* Tool verbindet alle Benutzer direkt und sicher mit den unterschiedlichsten DB2- und Informix-Datenbanken in Ihrem Unternehmen und dies unabhängig von der Größe und den Lokationen der Datenbankserver.

Stellen Sie sich das Tool einfach als großes Portal vor, das auf XML basiert und eine Menge an Schnittstellen besitzt, um unterschiedlichste Abfragesysteme zu unterstützen. Diese Abfragesysteme könnten z.B. WAP-fähige Geräte, wie PDAs und Handys, sein.

Es ist für die Plattformen Windows, Unix, iSeries und zSeries verfügbar.

DB2 High Performance Unload

Sie können dieses leistungsstarke Werkzeug statt des normalen DB2-Export-Kommandos einsetzen, um Ihre Daten möglichst schnell aus der DB2 UDB (ab Version 7) zu entladen bzw. zu extrahieren und diese Daten einem anderen System zur Verfügung zu stellen.

Es ist für die Plattformen Windows, Unix und zSeries verfügbar.

2.4 Die unterschiedlichen DB2-Produkte und -Produktkomponenten

In Abschnitt 2.1 „Übersicht über die DB2-Datenbanksystem-Produkte“ und Abschnitt 2.2 „Weitere DB2-Produktkomponenten“ haben wir Ihnen die verschiedenen DB2-Produkte und deren weitere Komponenten sowie die Lizenzierungsarten erläutert. Mit Abbildung 2.7 geben wir Ihnen ein praxisnahes, selbsterklärendes Beispiel für den Einsatz der DB2-Produkte DB2 PE, DB2 ESE und der zusätzlichen Komponenten DB2 Run-Time Client, DB2 CPE und DB2 CEE.

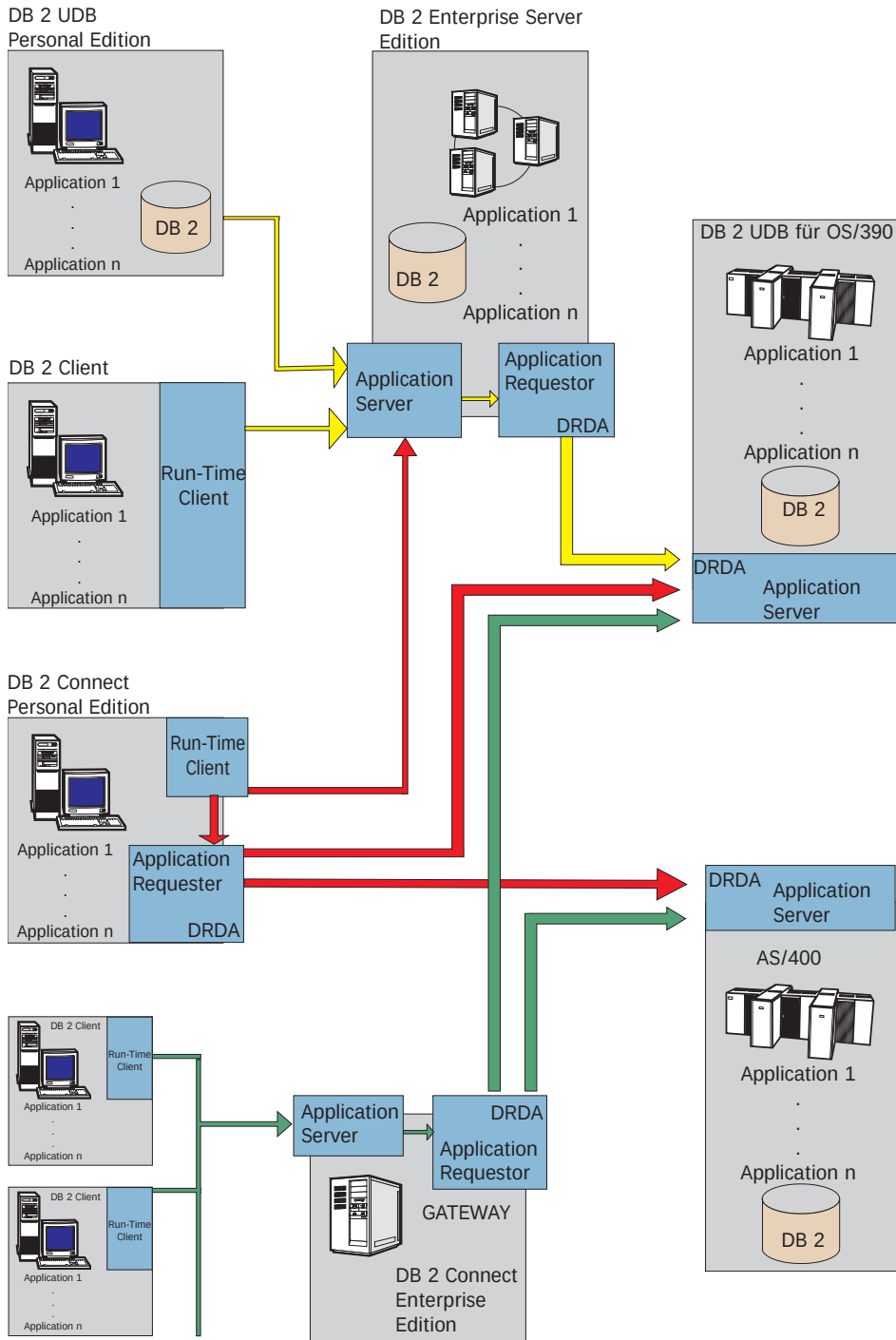


Abbildung 2.7: Die verschiedenen DB2-Produkte und deren zusätzliche Komponenten

3 Installation der DB2 UDB

Bevor wir uns mit der Installation der DB2 UDB V8 befassen, sollten wir noch ein paar Fragen klären. Die Antworten auf diese Fragen schaffen Grundlagen und erleichtern es Ihnen, den Ablauf der Installation besser zu verstehen:

- ▶ Was ist ein DB2-Exemplar (instance)?
- ▶ Was ist ein DB2-Verwaltungsserver (DAS)?
- ▶ Welche Rolle spielen die Benutzer, und was ist ein Exemplar-Eigner (instance owner)?

In den Schritten nach der eigentlichen Installation klären wir dann weitere Punkte, wie z.B.:

- ▶ Wie werden Exemplare verwaltet?
- ▶ Welche Bedeutung haben Umgebungsvariablen, wo und wie können diese gesetzt werden?

3.1 Was ist ein DB2-Exemplar (instance)?

Da der englische Begriff für Exemplar – instance – in anderen RDBMS wie z.B. Oracle und Informix ebenfalls genutzt wird, möchten wir an dieser Stelle ausdrücklich darauf hinweisen, dass dieser Begriff in der DB2 UDB-Welt seine eigene Bedeutung besitzt und nicht mit dem Begriff „Instanz“ aus anderen RDBMS verglichen oder verwechselt werden sollte.

- ▶ Wenden wir uns der Frage zu: Was verbirgt sich hinter dem Begriff „Exemplar“?

Um das besser zu verstehen, stellen wir folgende Überlegung an: Das DBMS ist für das jeweilige Betriebssystem nichts anderes als ein normales speicherresidentes Programm.

Ein speicherresidentes Programm besitzt unter anderem folgende Eigenschaften:

- ▶ Es muss vor der Nutzung gestartet werden.
- ▶ Es kann mit unterschiedlichen Eigenschaften und unterschiedlichen Programmversionen mehrfach gestartet werden.
- ▶ Die Programmeigenschaften können über Konfigurationsparameter gesteuert werden.
- ▶ Umgebungsvariablen können die Programmeigenschaften ebenfalls beeinflussen.
- ▶ Das Programm muss explizit gestoppt und aus dem Speicher entfernt werden können.

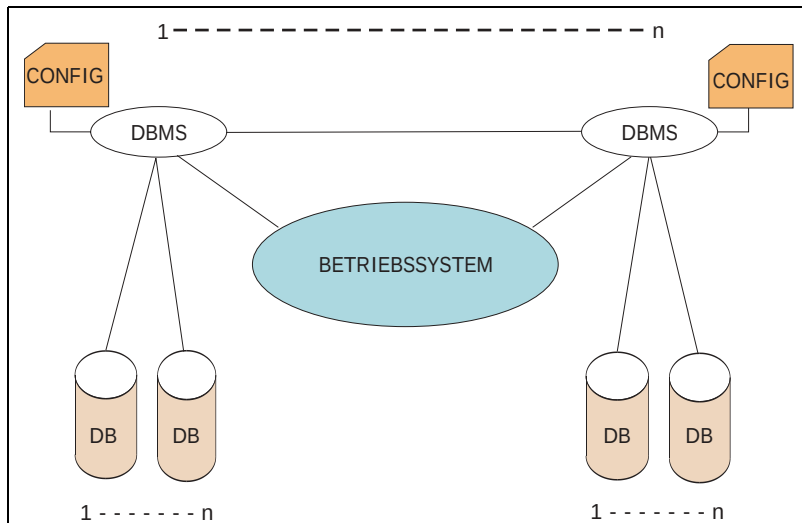


Abbildung 3.1: Mehrere DBMS mit unterschiedlichen Konfigurationen auf einem System

Wenn also das Programm, in unserem Fall das DBMS, mit *unterschiedlichen Eigenschaften* (Konfigurationen) mehrfach gestartet werden kann, müssen auch unterschiedliche Konfigurationsumgebungen vorhanden sein.

Um das Programm in *unterschiedlichen Programmversionen* starten zu können, müssen unterschiedliche Installations- bzw. Programmumgebungen verwaltet werden.

Das bedeutet: Wir müssen vor dem Start des Programms (DBMS) eine Umgebung (environment) schaffen, die es uns ermöglicht, verschiedene Konfigurationen und Programmversionen zu verwalten.

Genau diese Umgebung nennen wir *Exemplar* (instance). Ein Exemplar bildet also eine Laufzeit- und Konfigurationsumgebung für das DBMS. Pro Exemplar kann im nicht partitionierten DB2 UDB-Umfeld nur ein DBMS, und im partitionierten DB2 UDB-Umfeld intern ein DBMS pro Partition gestartet und verwaltet werden. Für den Benutzer sieht das weiterhin wie *ein* DBMS aus.

Wichtig:

Ein *Exemplar* ist eine Umgebung für das Datenbankmanagementsystem, in der die entsprechenden Konfigurationsparameter des DBMS definiert werden können und alle entfernten und lokal verfügbaren Datenbanken katalogisiert sind.

► Wie viele Exemplare kann ich auf einem Server haben?

Abbildung 3.1 zeigt, dass auf einem Server mehrere Exemplare existieren können. Die Anzahl ist dabei nur durch deren Sinn und den Systemressourcen eingeschränkt.

- Wann ist es sinnvoll, mehrere Exemplare zu verwenden?

Auch diese Frage ist im vorangegangenen Text schon angesprochen worden. So ermöglichen die aus DB2-Sicht völlig getrennten Exemplar-Umgebungen eine voneinander völlig unabhängige Konfiguration und Verwaltung. Das umfasst sowohl die Parameter als auch die Administration des Exemplars durch verschiedene Personen bzw. Personengruppen. Auch die Möglichkeit, auf Unix-Systemen Exemplare in unterschiedlichen DB2-Versionen parallel zu betreiben, kann ein weiterer Grund sein. Des Weiteren können Sie durch diese Architektur mehrere Exemplare für Test und Produktion auf ein und demselben Rechner betreiben.

3.2 Was ist der DB2-Verwaltungsserver (DAS)?

Der DAS (Database Administration Server) ist ab der DB2 UDB V8 ein spezieller Prozess auf dem Datenbankserver, der über die feste Portnummer 523 kommuniziert. Er ist der „Server-Agent“ und dient als zentraler Ansprechpartner für alle entfernten und lokalen Verwaltungszugriffe durch die grafischen Tools.

Er unterstützt unter anderem folgende Möglichkeiten:

- Zugriff auf Systemressourcen des Datenbankservers (z.B. Dateisysteme)
- Zugriff auf Prozesse des Datenbankservers (z.B. DBMS)

Diese Eigenschaften ermöglichen zum Beispiel folgende Verwaltungsaufgaben auf dem Datenbankserver:

- Auswahlmöglichkeit der Systemressourcen über Dateialoge (z.B. zur Speicherung von Scripts oder Sicherungsbildern).
- Unterstützung von Überwachungstools (z.B. Ereignismonitor)
- zeitliche Steuerung von Scripts (z.B. Sicherungs- und Pflegeabläufe)
- stellt Informationen zur Verfügung (z.B. auf dem Server befindliche Exemplare und Datenbanken)
- ermöglicht den Einsatz der Benachrichtigungsfunktion (z.B. Versenden von E-Mails bei Alarmen)

Der DAS speichert seine Informationen (z.B. Scripts und Zeitsteuerinformationen) im Gegensatz zu früheren DB2 UDB-Versionen nicht in Dateisystemen, sondern in der speziellen Tools-Datenbank.

Er besitzt eine eigene Konfiguration und eigene Befehle (z.B. zum Starten, Stoppen und Anlegen des DAS).

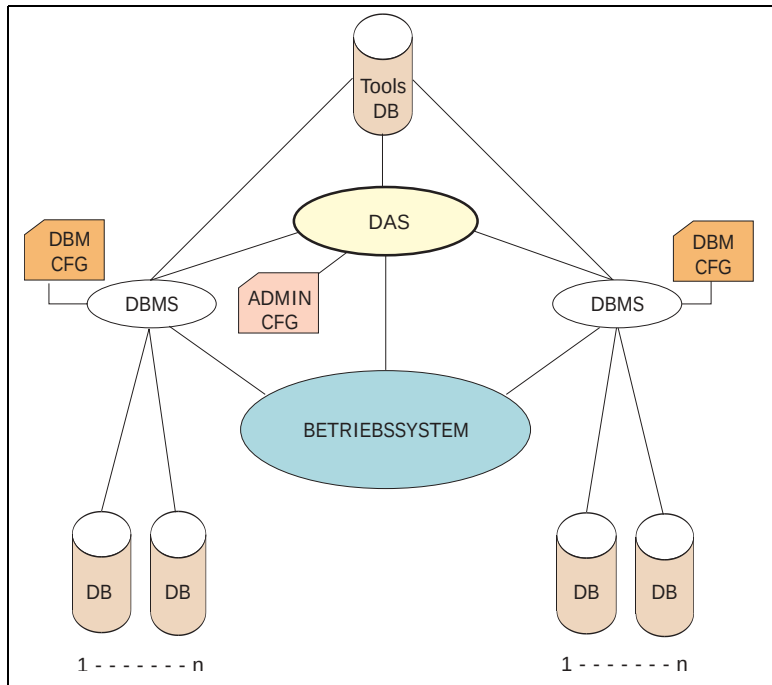


Abbildung 3.2: DBMS-Systemumgebung mit DAS

3.3 Was ist der Exemplar-Eigner bzw. Fenced-Benutzer?

Wie wir später noch sehen werden, spielt der Begriff des Exemplar-Eigners bzw. des Fenced-Benutzers, d.h. des abgeschirmten Benutzers, je nach verwendeter Betriebssystem-Welt (Windows oder Unix) eine unterschiedliche Rolle.

An dieser Stelle reicht es aus zu verstehen, dass der Exemplar-Eigner ein Benutzer (User) des Betriebssystems ist, unter dem das DBMS (bzw. man sagt auch „das Exemplar“) gestartet wird. Der Exemplar-Eigner wiederum kann einer Benutzer-Gruppe zugeordnet sein.

Je nach verwendetem Betriebssystem wird ein zusätzlicher Betriebssystem-Benutzer, der so genannte Fenced-Benutzer, für spezielle DB2 UDB-Prozesse benötigt. Diese Prozesse werden dann unter der entsprechenden Benutzer-ID in einem abgeschirmten (separaten) Adressraum gestartet.

3.4 Installationsmöglichkeiten der DB2 UDB V8

Je nach verwendetem Betriebssystem stehen Ihnen für die Installation der DB2 UDB V8 verschiedene Methoden zur Verfügung. Wir betrachten in unserem Buch die beiden prinzipiell unterschiedlichen Methoden, die in beiden Betriebssystemwelten zur Verfügung stehen:

- ▶ die grafische und vom Benutzer interaktiv gesteuerte Installationsmethode
- ▶ die automatisierte Installationsmethode, die sowohl der in Unix-Welt als auch in der Windows-Welt keine grafische Benutzeroberfläche benötigt

Obwohl wir im Weiteren nur auf die zwei zuvor genannten Arten eingehen möchten, wollen wir Ihnen nachfolgend alle Installationsmethoden vorstellen, auch wenn diese nicht für alle Betriebssysteme zur Verfügung stehen:

- ▶ Nutzung des grafischen Installationsprogramms (**Windows und Unix**)

Diese Methode wird in beiden Betriebssystem-Welten unterstützt. Unter Windows heißt das Programm setup, unter Unix db2setup. Diese Methode setzt jedoch die Unterstützung der grafischen Benutzeroberfläche vonseiten des Betriebssystems voraus. Diese Methode werden wir in Kapitel Abschnitt 3.5 „Installation der DB2 UDB über die grafische Installationsroutine am Beispiel von Windows“ für die Windows-Umgebung näher beschreiben.

- ▶ Installation über ein **Response-File (Windows und Unix)**

Hier können Sie die Installation über ein Response-File (Antwortdatei) exakt nach Ihren Vorgaben steuern. In der Betriebssystem-Welt Unix wird bei dieser Installationsart die Unterstützung der grafischen Benutzeroberfläche nicht benötigt. Standardmäßig stehen Ihnen je nach ausgewähltem Produkt und Betriebssystem vorgefertigte Response-Files zur Verfügung. Diese enthalten sämtliche Konfigurations- und Installationsmöglichkeiten für das entsprechend ausgewählte Produkt. Vor der Installation müssen sie jedoch an Ihre speziellen Bedürfnisse angepasst werden.

Beispiele sind für die Betriebssysteme HP-UX, AIX, Sun Solaris, Linux, Linux64 Linux390 und Windows verfügbar. Sie finden diese auf der Installations-CD im Verzeichnis x:/db2/<Plattform>/samples (x entspricht dem Laufwerksbuchstaben Ihres CD-ROM-Laufwerks):

- db2ese.rsp

Beispiel für das Produkt *DB2 UDB Enterprise Server Edition (ESE)*

- db2adc1.rsp

Dient zur Installation und Vorkonfiguration eines Entwickler-Clients (administration developer client).

- db2admcl.rsp

Mit dem db2admcl.rsp können Sie die Installation eines Administrations-Clients ausführen.

- db2conee.rsp

Beispielvorgabe für die Installation des Produkts *DB2 UDB Connect Enterprise Edition*.

- db2conpe.rsp

Hiermit können Sie das Produkt *DB2 UDB Connect Personal Edition* installieren.

- db2pe.rsp

Dient der Installation der *DB2 UDB Personal Edition (PE)*.

- db2wse.rsp
Response-File für die Installation der DB2 UDB Workgroup Edition (WSE)
- db2rtcl.rsp
Dient der Installation des DB2 UDB Run-Time Client.

Für die Windows-Umgebung stehen Ihnen standardmäßig zwei Möglichkeiten zur Generierung von Response-Files zur Verfügung:

- Über die Installationsmethode setup können Sie ein Response-File erzeugen, anstatt eine Installation auszuführen.
- Mit dem DB2-Kommando db2rspgn -d x:\<pfadangabe> -i exemplarname können Sie nach der Installation ein Response-File für das angegebene Exemplar erzeugen.

Die Installationsmethode über ein Response-File ermöglicht Ihnen eine nahezu vollautomatische Installation. Sollten Sie also die DB2 UDB V8 auf mehreren Servern oder Einzelplatzsystemen bzw. mehreren DB2-Clients installieren müssen, ist die Installation über Response-Files sicherlich die eleganteste und zeitsparendste Möglichkeit.

Voraussetzung ist jedoch, dass Sie über ein korrektes Response-File verfügen. Wir werden diese Installationsart in Abschnitt 3.6 „Installation der DB2 UDB mit der Response-File-Methode unter dem Betriebssystem Linux“ näher beschreiben.

- Manuelle Installation mit dem Installationsscript db2_install (Unix).

Die Script-Datei db2_install ist ein vorgefertigtes Script, das Ihnen die Installation der DB2 UDB V8 in der Unix-Welt über die Kommandozeile ermöglicht.

- Betriebssystemspezifische Kommandos zur Installation von Fremdsoftware (**Unix**).

Als Beispiel steht Ihnen unter dem Betriebssystem AIX das Betriebssystemkommando installp zur Verfügung.

- Spezielle Installations-Tools des Betriebssystems (Unix).

Je nach Betriebssystem stehen Ihnen eventuell weitere Hilfsprogramme zur Installation von Fremdsoftware zur Verfügung. Beispielsweise können Sie unter dem Betriebssystem AIX das *System Management Interface Tool* (SMIT) zur Installation der DB2 UDB V8 einsetzen.

Hinweis:

Im Gegensatz zu der grafischen Installationsmethode db2setup und der Methode über ein Response-File wird bei den letzten drei Methoden nur eine reine Installation des Produkts DB2 UDB V8 vorgenommen. Das bedeutet z.B. :

- Es wird weder ein Standardexemplar (instance) erstellt noch konfiguriert.
- Die Unterstützung des Verwaltungsservers DAS wird nicht eingerichtet.
- Es werden keinerlei Kommunikationsprotokolle konfiguriert.

Tipp:

Sollten Sie die beiden letzten Installationsmethoden nutzen, müssen Sie nach der Installation selbst dafür Sorge tragen, dass der Lizenzschlüssel installiert wird.

Um dies auszuführen, geben Sie folgenden Befehl ein:

```
db2licm -a /<Name des CD-ROM-Laufwerks>/db2/license/<Dateiname>.lic
```

Für die DB2 UDB ESE beispielsweise lautet der Dateiname `db2ese.lic`.

3.5 Installation der DB2 UDB über die grafische Installationsroutine am Beispiel von Windows

Dieser Abschnitt befasst sich mit der grafischen Installationsmethode der DB2 UDB V8 unter der Betriebssystemplattform Windows. Wir werden die Installation anhand von Windows 2000 beispielhaft Schritt für Schritt durchführen und die entsprechenden Bildschirme bzw. Fragestellungen der Installation ausführlich erklären. Die grafische Installation verhält sich für die Betriebssystem-Welt Unix bis auf ein paar Begriffe und Darstellungsweisen im Prinzip identisch. Für die Betriebssystem-Welt Unix sollten Sie allerdings sicherstellen, dass Ihnen die grafische Benutzeroberfläche zur Verfügung steht und die jeweiligen Bedingungen, wie z.B. Betriebssystemversion und Java-Umgebung, vorhanden sind. Diese Informationen entnehmen Sie bitte dem Installationshandbuch der jeweiligen Betriebssystemversion.

Bevor Sie mit der Installation beginnen, sollten Sie sicherstellen, dass der Benutzer, den Sie zur Installation verwenden, folgende Bedingungen erfüllt:

- ▶ Er sollte **lokal** definiert sein.
- ▶ Er sollte über **lokale** Administrationsrechte verfügen.

Sollten Sie z.B. als Domain-Administrator angemeldet sein und nicht lokal, können während bzw. nach der Installation Berechtigungsprobleme auftreten (z.B. beim Anlegen der Sample-Datenbank).

Zur Installation legen Sie die entsprechende DB2-Installations-CD in Ihr Laufwerk. Dabei sollten Sie beachten, dass auf der CD die von Ihnen gewünschte Landessprache enthalten ist. Die Installationsroutine orientiert sich standardmäßig an der Landessprache Ihres Betriebssystems.

Im Regelfall startet die Installationsroutine durch die Auto-Run-Funktion selbstständig. Sollte dies nicht der Fall sein, können Sie durch die Eingabe des Befehls `x:\setup` (x entspricht dem Laufwerksbuchstaben Ihres CD-ROM-Laufwerks) die Installationsroutine ebenfalls starten.



Abbildung 3.3: Installation – Willkommen-Bildschirm

Der Bildschirm aus Abbildung 3.3 zeigt Ihnen das Willkommen-Fenster von Windows. Sie erhalten auf der linken Seite ein Auswahlménü, in dem Sie unter anderem die Installationsvoraussetzungen und andere DB2-Informationen nachlesen können. Über die Auswahlfläche **PRODUKTE INSTALLIEREN** gelangen Sie zu Abbildung 3.4.



Abbildung 3.4: Produktauswahl

Dieser Bildschirm zeigt Ihnen die installierbaren Produkte an, die sich auf der CD befinden. In unserem Fall steht hier nur ein Produkt (DB2 UDB Enterprise Server Edition – ESE) zur Auswahl. Klicken Sie hier auf **WEITER**, um zum Fenster der Abbildung 3.5 zu gelangen.



Abbildung 3.5: Willkommen beim DB2-Installationsassistenten

Hier werden Sie darauf hingewiesen, dass der DB2-Installationsassistent die ausgewählte Produktkomponente ESE installieren wird. Wenn Sie auf **WEITER** klicken, gelangen Sie zum nächsten Fenster (siehe Abbildung 3.6).

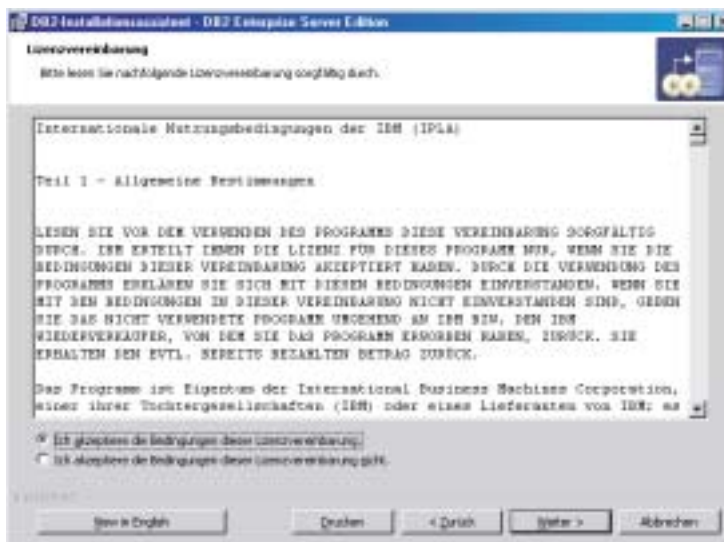


Abbildung 3.6: Lizenz-Vereinbarung

Sie können sich auf dieser Seite über die allgemeinen IBM-Lizenzbestimmungen informieren und diese ausdrucken. Um mit der Installation fortfahren zu können, müssen Sie diese Bestimmungen akzeptieren. Danach klicken Sie auf den **WEITER**-Button.

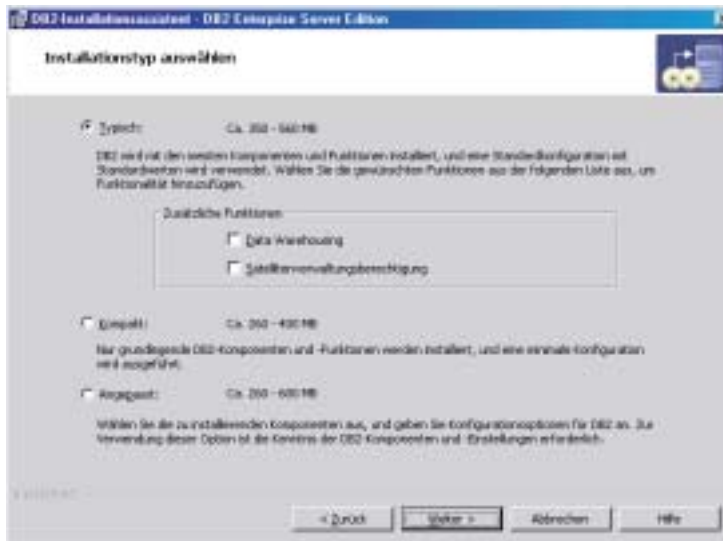


Abbildung 3.7: Auswahl des Installationstyps

Der Bildschirm aus Abbildung 3.7 ermöglicht Ihnen die Auswahl der verschiedenen Installationsarten **TYPISCH**, **KOMPAKT** und **ANGEPASST**. Dem unerfahrenen DB2 UDB-Benutzer empfehlen wir an dieser Stelle die Auswahl **TYPISCH**. Diese Auswahl installiert alle notwendigen DB2-Komponenten sowie die DB2-Tools und lässt Ihnen im Weiteren genug Auswahlmöglichkeiten, um die Installation sinnvoll an Ihr System anzupassen. Mit der Auswahl **DATA WAREHOUSING** veranlassen Sie das Anlegen einer Data Warehouse-Kontrolldatenbank. Diese wird von dem grafischen Tool *Data Warehouse-Zentrale* als Datenbank zur Verwaltung der Metadaten genutzt. Das Anlegen dieser Kontrolldatenbank ist zwar zu einem späteren Zeitpunkt auch möglich, sollten Sie jedoch jetzt schon wissen, dass Sie die Data Warehouse-Zentrale nutzen wollen, empfehlen wir Ihnen, diese Option auszuwählen.

Die Auswahl **SATELLITENVERWALTUNGSBERECHTIGUNG** installiert die notwendigen Komponenten für die Administration der so genannten DB2-Satelliten. DB2-Satelliten sind DB2-Installationen z.B. auf einen Laptop. Durch diese Komponente wird dem Administrator eine asynchrone Administration (z.B. durch Wartungsscripts) beim Anmelden der Laptops an den Datenbankserver ermöglicht. Mehr dazu lesen Sie in Abschnitt 4.8.

Im Gegensatz dazu werden bei dem Installationstyp **KOMPAKT** nur die wichtigsten Datenbankkomponenten (z.B. ohne grafische Tools) und die ODBC-Unterstützung installiert.

Der Installationstyp **ANGEPASST** ermöglicht Ihnen die individuelle Auswahl der entsprechenden DB2-Teilkomponenten.

Wir haben uns in diesem Beispiel für die Auswahl **TYPISCH** entschieden und unser weiteres Vorgehen darauf aufgebaut. Über den **WEITER**-Button kommen Sie zum nächsten Schritt (siehe Abbildung 3.8).

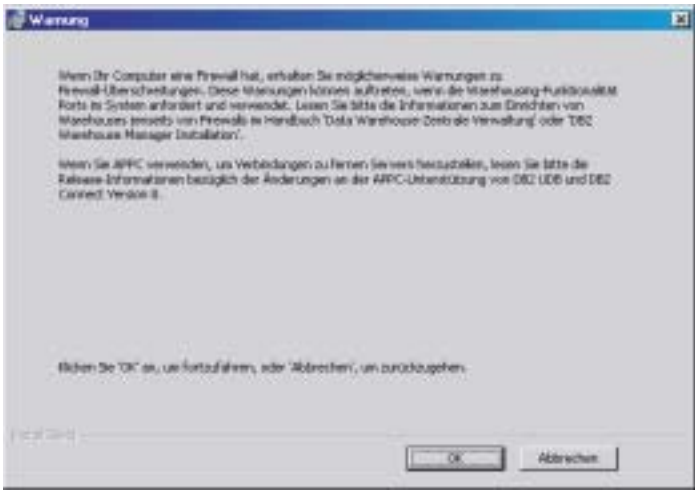


Abbildung 3.8: Warnhinweis beim Einsatz von Firewalls

Da die Data Warehouse-Zentrale zur Kommunikation verschiedene TCP/IP-Ports anfordert, kann es bei der Verwendung einer Firewall zur Warnhinweisen kommen. Bestätigen Sie diesen Bildschirm mit OK.

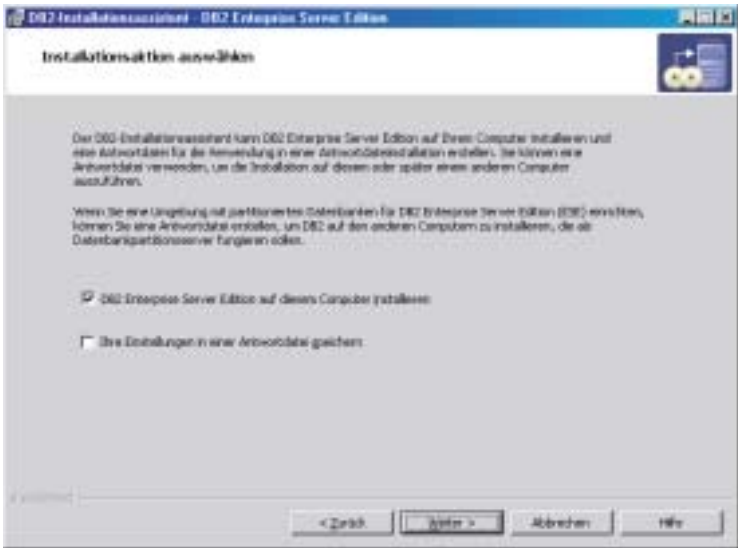


Abbildung 3.9: Auswahl der Installationsaktion

Im Bildschirm aus Abbildung 3.9 haben Sie die Möglichkeit, eine tatsächliche Installation, nur die Generierung einer Antwortdatei (response file) oder beides vorzunehmen. Standardmäßig ist eine tatsächliche Installation voreingestellt. Auf

die Installationsmethode über eine Antwortdatei gehen wir näher in Abschnitt 3.6 „Installation der DB2 UDB mit der Response-File-Methode unter dem Betriebssystem Linux“ ein.

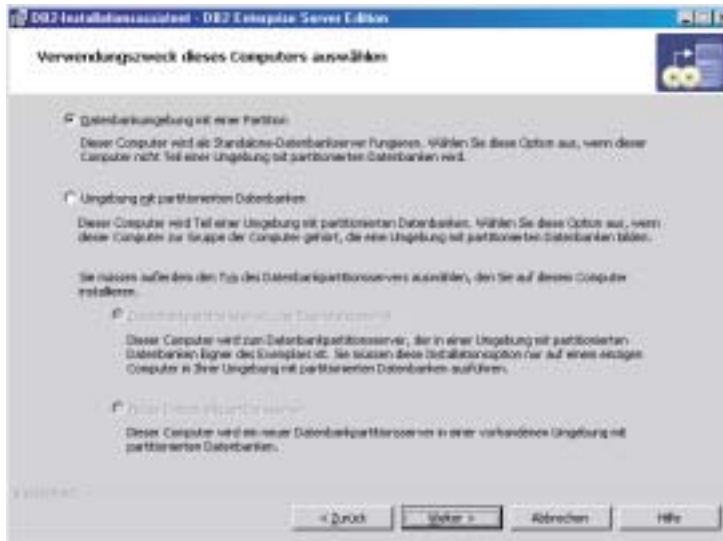


Abbildung 3.10: Auswahl der Partitionierungsart

Die Auswahl aus Abbildung 3.10 steht Ihnen nur bei dem Produkt DB2 UDB ESE zur Verfügung. Bei mehrfach partitionierten Datenbankumgebungen ist pro Server eine DB2-Installation notwendig. Dies hat zur Folge, dass mehrere Datenbankpartitionen koordiniert werden müssen. Somit muss ein Exemplar auf einem Datenbankserver diese Koordinationsaufgabe übernehmen und ist mit der Option DATENBANKPARTITIONSSERVER, DER EXEMPLAREIGNER IST zu installieren. Alle weiteren Installationen werden mit der Option NEUER DATENBANKPARTITIONSSERVER durchgeführt.

In unserem Buch konzentrieren wir uns auf Einzelpartitionsumgebungen bzw. auf nicht-partitionierte Umgebungen. Einzelpartitionsumgebungen entstehen bei der Verwendung des Produkts DB2 UDB ESE mit der Option aus Abbildung 3.10 – DATENBANKUMGEBUNG MIT EINER PARTITION. Die Installation der Produkte DB2 UDB PE oder WSE führt zu einer nicht partitionierten Umgebung.

Wir belassen es bei der Standardauswahl und bestätigen mit WEITER.

Standardmäßig ist das Laufwerk C: voreingestellt. Sie können an dieser Stelle sowohl das Ziel-Laufwerk als auch den Installationspfad der DB2-Installation bestimmen. Anschließend klicken Sie auf Weiter.

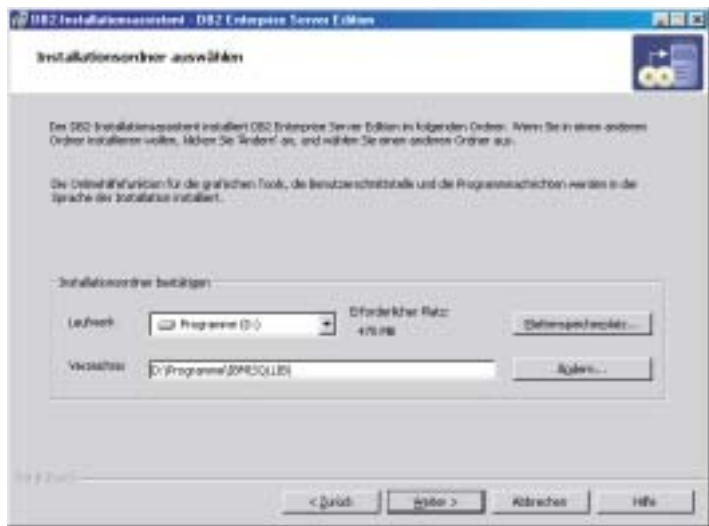


Abbildung 3.11: Auswahl des Installationspfads

Tipp:

Um im Pfadnamen einen Bezug zur DB2 UDB-Installation herstellen zu können, empfehlen wir Ihnen dringend, das eigentliche DB2 UDB-Installationsunterverzeichnis `\SQLLIB` nicht umzubenennen. Der restliche Pfad hingegen ist unkritisch.

Ein Beispiel für eine sinnvolle Pfadangabe wäre: `D:\IBM\SQLLIB`



Abbildung 3.12: Benutzerinformationen des DB2-Verwaltungsservers (DAS)

Wie zuvor erwähnt (siehe Abschnitt 3.2 „Was ist der DB2-Verwaltungsserver (DAS)?“), benötigt die DB2 UDB für die Verwaltung einen DAS-Prozess. Dieser Prozess wird unter der hier angegebenen Benutzer-ID ausgeführt. Standardmäßig wird dieser Benutzer auch für andere DB2-Prozesse verwendet. Damit dieser DAS-Prozess auf dem Server befindliche Exemplare starten und stoppen kann, benötigt die hier angegebene Benutzer-ID Administrationsrechte auf Betriebssystemebene. Wir empfehlen Ihnen, einen noch nicht vorhandenen Benutzer anzugeben, der dann bei der Installation automatisch *korrekt* eingerichtet wird. Nachdem Sie diese Angaben getätigt haben, kommen Sie mit WEITER zum nächsten Fenster.

Hinweis:

Bitte beachten Sie, dass der entsprechende Prozess mit dem hier angegebenen Benutzer und Kennwort in den Windows-Diensteingetragen wird.

Eine Kennwortänderung dieses Benutzers wird *nicht* automatisch in die Windows-Dienste übernommen und muss explizit angepasst werden.

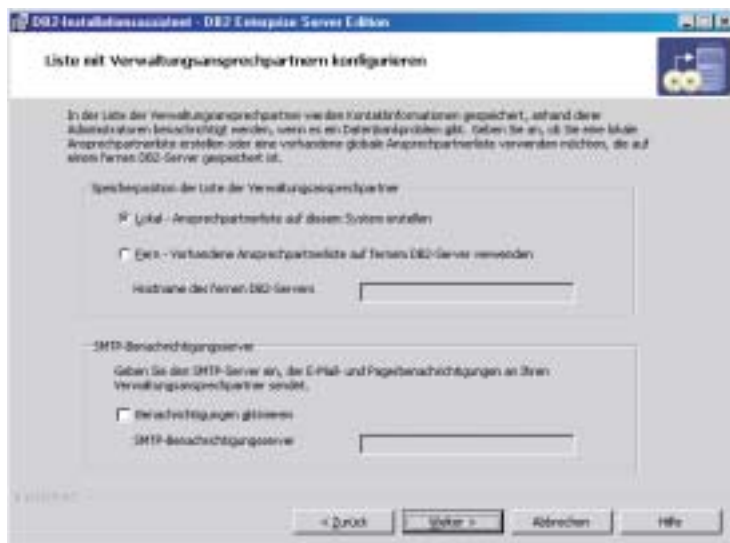


Abbildung 3.13: Definition der Verwaltungsansprechpartner

In der Tools-Datenbank des Verwaltungsservers (DAS) werden unter anderem die Kontaktpersonen eingetragen, die z.B. bei DB2-Systemfehlern benachrichtigt werden sollen.

Damit in einer Umgebung mit mehreren Datenbankservern nicht auf jedem der Server diese Einträge verwaltet und gepflegt werden müssen, können Sie hier (siehe Abbildung 3.13) angeben, ob die Verwaltung lokal oder auf einem zentralen, entfernten Rechner erfolgen soll.

Der im zweiten Teil-Fenster angegebene SMTP-BENACHRICHTIGUNGSSERVER ermöglicht dem DAS eine Benachrichtigung der Ansprechpartner über Mail. Als Abschluss klicken Sie auf die Schaltfläche WEITER.

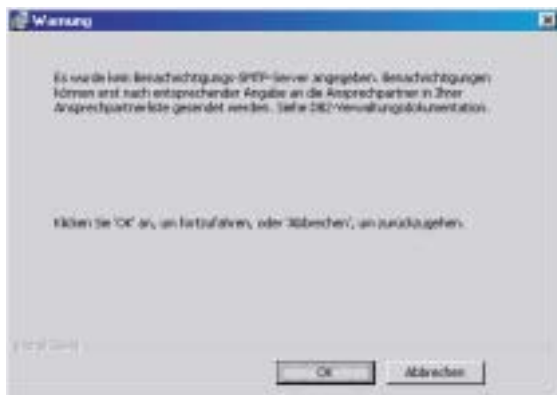


Abbildung 3.14: Warnhinweis

Die Meldung aus Abbildung 3.14 erhalten Sie, wenn Sie in Abbildung 3.13 keinen SMTP-Server eingetragen haben. An dieser Stelle können Sie diese Warnung durchaus ignorieren. Nach der Installation können die den SMTP-Server durch folgenden Befehl konfigurieren:

```
update admin cfg using smtp_server <smtpserver name>
```

Durch OK kommen Sie zu dem Bildschirm aus Abbildung 3.15.

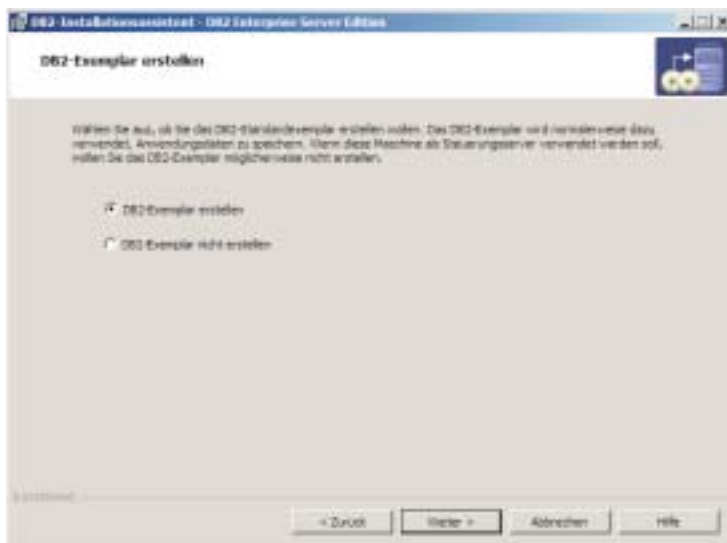


Abbildung 3.15: DB2-Exemplar erstellen

DB2 bietet die Möglichkeit, die beiden Exemplare DB2 (Standardbenutzerexemplar) und DB2CTLSV (Standardexemplar für die Satelliten-Verwaltung) bei der Installation zu erstellen. Durch die Auswahl DB2-EXEMPLAR NICHT ERSTELLEN können Sie diese automatische Erstellung unterdrücken.

Wir empfehlen Ihnen jedoch, die Standardeinstellung DB2-EXEMPLAR ERSTELLEN beizubehalten und hier den WEITER-Button anzuklicken.

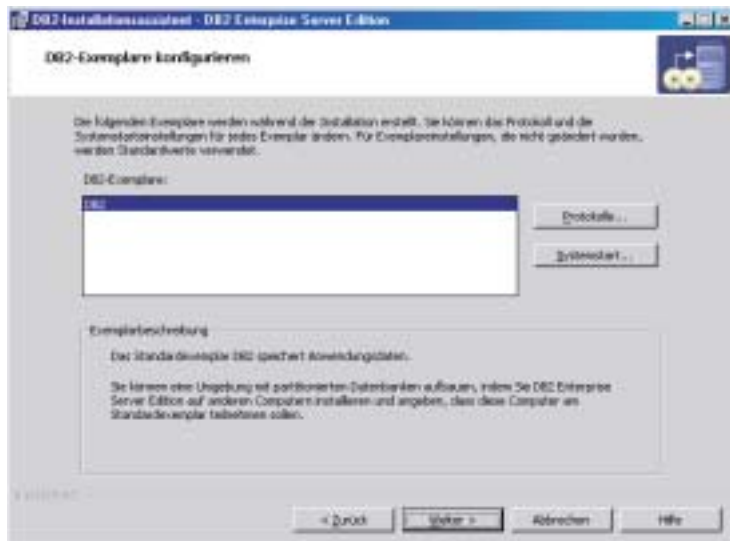


Abbildung 3.16: Konfiguration des Standardexemplars DB2

Bei der Installation mit der grafischen Oberfläche wird standardmäßig das Exemplar DB2 mit angelegt. Wenn Sie in der Abbildung 3.7 die Option SATELLITENVERWALTUNGSBERECHTIGUNG ausgewählt haben, erscheint hier zusätzlich ein Eintrag für das Exemplar DB2CTLSV. Um dieses Exemplar bzw. diese Exemplare in einer Netzwerkumgebung zur Verfügung zu stellen, müssen entsprechende Protokolldefinitionen getätigt werden. Um auf die nächsten Bildschirme zu gelangen, klicken Sie hier auf PROTOKOLLE...



Abbildung 3.17: Konfiguration des Kommunikationsprotokolls TCP/IP

Die DB2-Installationsroutine erkennt automatisch alle unterstützten Netzwerkprotokolle und konfiguriert diese mit Standardwerten. Sie haben jedoch die Möglichkeit, diese Standardwerte anzupassen.

Der hier eingegebene **SERVICENAME** und die entsprechende **PORTNUMMER** werden automatisch in die Datei `...etc\services` Ihres Rechners eingetragen. Ebenso wird das Exemplar DB2 für TCP/IP konfiguriert. Die Details werden in Kapitel 6 erläutert.

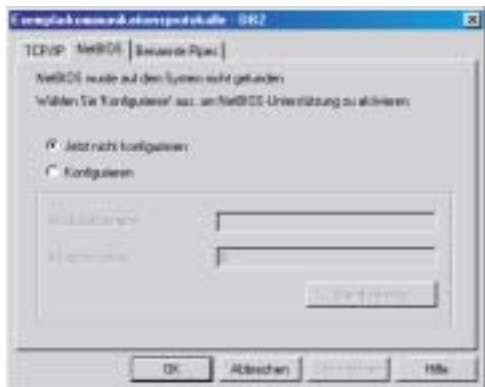


Abbildung 3.18: Konfiguration des Kommunikationsprotokolls NetBIOS

Unter dem Reiter **NETBIOS** (siehe Abbildung 3.18) finden Sie NetBIOS-spezifische Einstellungen. Da auf unserem Rechner NetBIOS nicht konfiguriert ist, ist hier standardmäßig die Option **JETZT NICHT KONFIGURIEREN** aktiv.

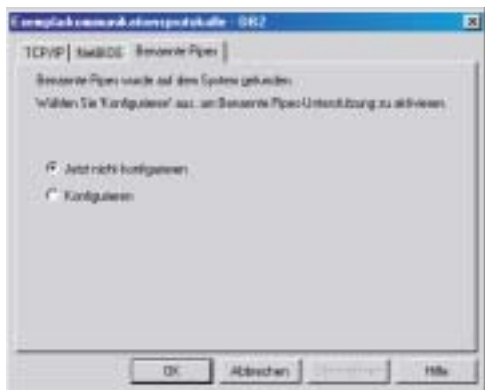


Abbildung 3.19: Auswahl des Kommunikationsprotokolls Benannte Pipes

Auf der Registerkarte aus Abbildung 3.19 können Sie die Unterstützung des Kommunikationsprotokolls **BENANNTE PIPES** (named pipes) aktivieren. Da es sich hierbei um ein Windows-spezifisches Protokoll handelt, ist es nicht standardmäßig als Kommunikationsprotokoll für das DB2-Exemplar vorgesehen.

Nachdem Sie alle von Ihnen gewünschten Kommunikationsprotokolle eingestellt haben, kommen Sie durch den **OK**-Button zur Abbildung 3.16 zurück.

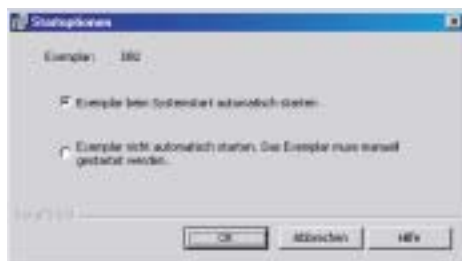


Abbildung 3.20: Exemplar-Startoptionen

Wenn Sie den Auswahl-Button SYSTEMSTART... in dem Bildschirm aus Abbildung 3.16 anklicken, gelangen Sie in das Fenster aus Abbildung 3.20. Hier können Sie entscheiden, ob beim Starten Ihres Rechners das Exemplar automatisch mitgestartet wird. Über die Windows-Dienste besitzen Sie die Möglichkeit, diese Einstellung aufzuzeigen und abzuändern. In den Windows-Diensten finden Sie einen Eintrag pro Exemplar nach dem Schema DB2 - <Exemplarname>.

Mit dem OK-Button kommen Sie zur Abbildung 3.16 zurück. Von dort gehen Sie mit dem WEITER-Button zum nächsten Auswahlfenster (siehe Abbildung 3.21).

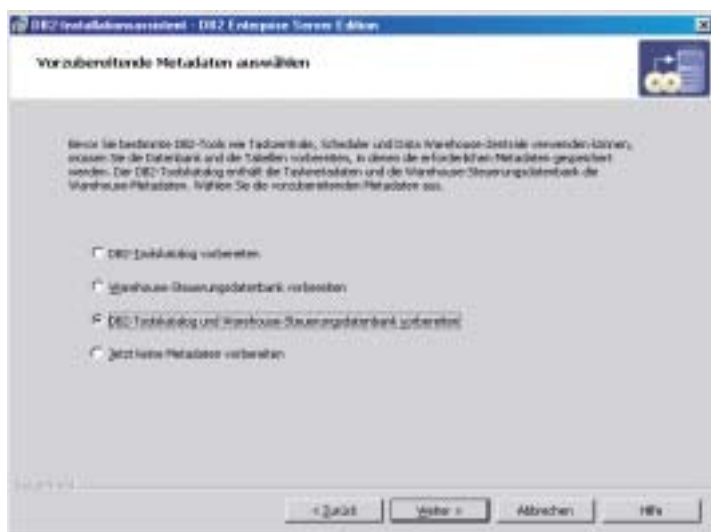


Abbildung 3.21: Erstellung der Verwaltungsdatenbanken

DB2 UDB benötigt für den DAS den Toolskatalog (Tools-Datenbank) und für die Data Warehouse-Zentrale die Warehouse-Steuerungsdatenbank. An dieser Stelle definieren Sie, ob diese Datenbanken bereits bei der Installation automatisch mit angelegt werden. Das Erstellen dieser Datenbanken ist auch jederzeit nach der Installation noch möglich.

Bitte beachten Sie, dass wir hier von der Voreinstellung abweichen und explizit die Auswahl DB2-TOOLS KATALOG UND WAREHOUSE-STEUERUNGSDATENBANK VORBEREITEN gewählt haben. Anschließend bestätigen Sie diese Auswahl mit WEITER.

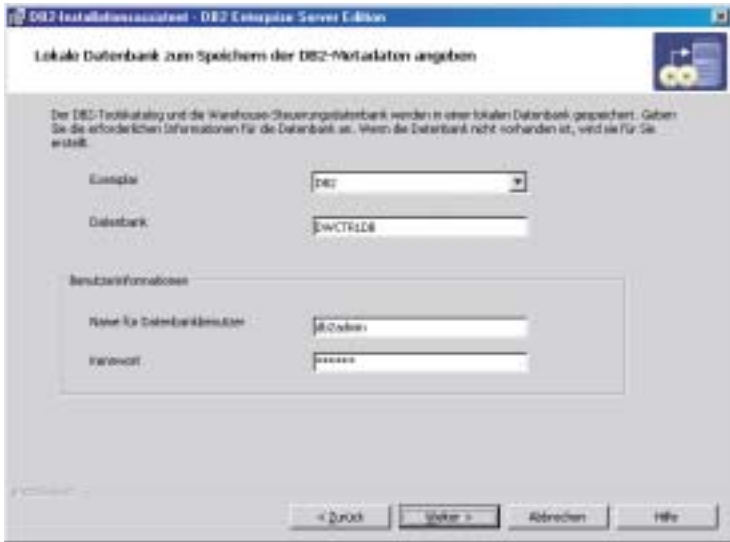


Abbildung 3.22: Einstellungen für die Warehouse-Steuerungsdatenbank

Wie in Abbildung 3.22 gezeigt, geben Sie den Namen der Data Warehouse-Steuerungsdatenbank an und unter welchem Exemplar sie erstellt wird. Zusätzlich sind eine Benutzer-ID und ein Kennwort erforderlich.



Abbildung 3.23: Einstellungen für die Warehouse-Steuerungsdatenbank und den Toolskatalog

Tipp:

Um später diese Datenbank über die Data Warehouse-Zentrale nutzen zu können, sollten Sie sich den hier angegebenen Benutzer und das Kennwort unbedingt merken.

Sie haben auch die Option, den jeweiligen Schema-Namen sowohl für die Objekte der Tools-Datenbank als auch für die Objekte der Warehouse-Steuerungsdatenbank einzugeben (siehe Abbildung 3.22).

Da wir erst in Kapitel 7 auf den Begriff Schema eingehen, empfehlen wir Ihnen, die Voreinstellungen zu akzeptieren und mit WEITER zum nächsten Fenster zu springen.

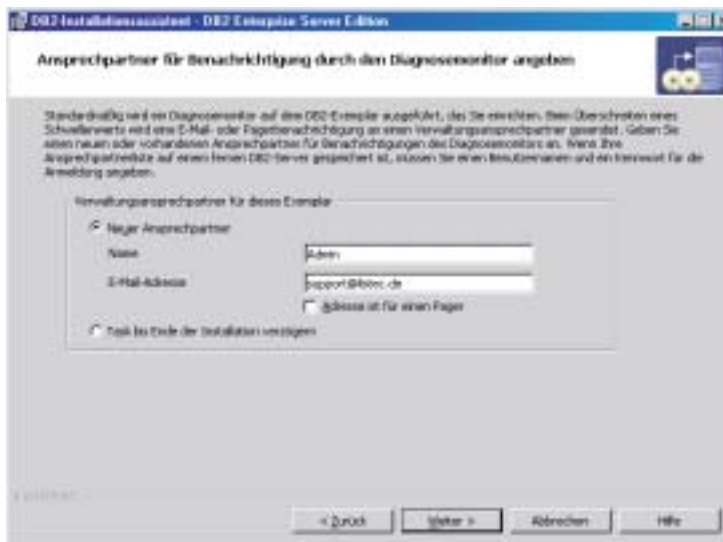


Abbildung 3.24: Ansprechpartner spezifizieren

In Abbildung 3.13 hatten wir die Option **LOKAL – ANSPRECHPARTNERLISTE AUF DIESEM SYSTEM ERSTELLEN** ausgewählt. Im Bildschirm aus Abbildung 3.24 geben Sie nun den konkreten Benutzer inklusive seiner E-Mail-Adresse an, der z.B. bei DB2-Warnmeldungen benachrichtigt werden soll.

Nach der Installation haben Sie die Möglichkeit, weitere Benutzer einzugeben. Näheres dazu finden Sie in Abschnitt 4.2.3.

Mit **WEITER** kommen Sie zu der Anzeige aus Abbildung 3.25.

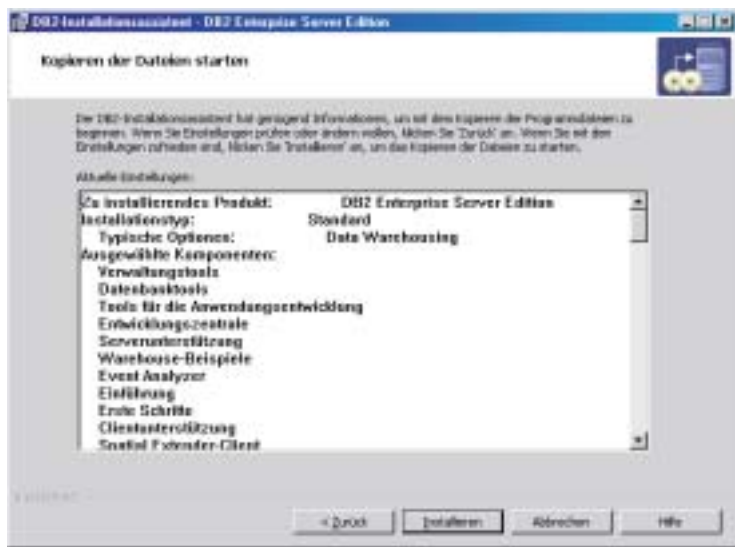


Abbildung 3.25: Zusammenfassung

An dieser Stelle wird Ihnen eine Zusammenfassung der zu installierenden Produkte aufgezeigt. Bestätigen Sie hier mit dem Button **INSTALLIEREN**, um die Installation auszuführen.



Abbildung 3.26: Ende der Installation

Das Fenster aus Abbildung 3.26 erhalten Sie nach erfolgreich ausgeführter Installation.

Hinweis:

Beachten Sie, dass ab der DB2 UDB V8 die Dokumentation von einer weiteren CD separat installiert werden muss.

Der Auswahl-Button **FERTIG STELLEN** beendet den Installationsassistenten.

Beim nächsten Systemstart wird Ihnen mit Abbildung 3.27 zu Ihrer erfolgreichen Installation gratuliert.



Abbildung 3.27: Erste Schritte

Im Fenster **ERSTE SCHRITTE** können Sie aus verschiedenen Optionen wählen. In der Regel werden Sie zuerst die Beispieldatenbank erstellen. In unserem Fall behandeln wir das Erstellen der Beispieldatenbank in Kapitel 5.

3.6 Installation der DB2 UDB mit der Response-File-Methode unter dem Betriebssystem Linux

In diesem Abschnitt werden wir uns mit der *Response-File-Installationsmethode* der DB2 UDB V8 unter der Betriebssystemplattform Linux befassen. Wir werden die Installation anhand von SuSe Linux Version 8.1 beispielhaft durchführen.

Im Gegensatz zu Windows kommt es in der Betriebssystem-Welt Unix schon mal vor, dass ein bestimmter Datenbankserver die grafische Benutzeroberfläche nicht unterstützt und dass eine reine Kommandozeilenversion des Betriebssystems installiert ist. Bis zur DB2 UDBV8 handelte es sich bei der Installationsroutine

db2setup um eine nicht grafische Installationsroutine. Heute hingegen unterstützt diese nur noch die grafische Variante. Somit ist, unserer Meinung nach, die Installationsart über das *Response-File* die einzig wahre Alternative zu *db2setup*. Die Installationsvariante über die Scriptdatei *db2_install* bzw. über die *Betriebssystembefehle* haben einen wesentlichen höheren manuellen Installationsaufwand zur Folge und bergen somit auch wesentlich mehr Gefahren einer nicht korrekten DB2 UDB-Installation.

3.6.1 Grundvoraussetzungen für die Installation

Bevor Sie sich mit der eigentlichen Installation befassen, sollten Sie sicherstellen, dass Ihr System die nachstehenden Voraussetzungen erfüllt:

- Liegen der benötigte Linux-Kern und die entsprechende Library vor?

Für die von uns verwendete SUSE Linux Version 8.1 gilt: *Kernel 2.4.19* und *library glibc 2.2.5*.

- Steht Ihnen für die Installation genug Plattenplatz in dem entsprechenden Verzeichnis zur Verfügung? Denken Sie daran, dass DB2 unter dem Verzeichnis */opt/IBM/db2/V8.1/* installiert wird. Je nach Installationsart wird folgender Speicherplatz benötigt:

- Typical benötigt ca. 400 bis 500 Mbyte Speicherplatz.
- Compact benötigt ca. 350 bis 400 Mbyte Speicherplatz.
- Custom benötigt ca. 300 bis 700 Mbyte Speicherplatz.

- Verfügt Ihr System über genügend Hauptspeicher?

IBM empfiehlt hier ein Minimum von 256 Mbyte RAM für die Installation des Produkts DB2 UDB ESE.

Beachten Sie weiter:

- Der Platz für die Auslagerungsdatei sollte mindestens doppelt so groß wie Ihr Hauptspeicherplatz sein.
- Für weitere Software außer DB2 sollten Sie zusätzlichen Speicher zur Verfügung stellen.
- Der Hauptspeicher, der DB2 zur Verfügung gestellt wird, ist direkt ausschlaggebend für das Performance-Verhalten der DB2 UDB.
- Weitere Datenbankaktivitäten wie z.B. die Unterstützung von DB2 UDB-Clients benötigen zusätzlichen Speicherplatz.

- Unterstützt Ihr System das TCP/IP-Kommunikationsprotokoll?

Die korrekte Unterstützung des TCP/IP-Kommunikationsprotokolls ist die Grundvoraussetzung für einen einwandfreien entfernten (remote) Zugriff auf Ihren Datenbankserver.

- Sind die zusätzlichen Software-Voraussetzungen erfüllt?

Für die Verwendung der grafischen Tools benötigen Sie auf Ihrem System das Java Developer's Kit (JDK) mindestens in der Version 1.3.1 mit dem Service Release 1.

Tipp:

Um eine einwandfreie Funktion der Steuerzentrale von DB2 zu garantieren, empfehlen wir Ihnen, nicht benötigte JDK-Versionen zuvor zu deinstallieren.

Sollte eine andere Anwendung eine zusätzliche Version des JDKs benötigen, müssen Sie sicherstellen, dass die Umgebungsvariablen PATH und CLASSPATH korrekt gesetzt sind.

Weitere sehr wertvolle Informationen erhalten Sie z.B. in [DB2LIN].

3.6.2 Vorüberlegungen

Bei der Installation gehen wir von folgenden Voraussetzungen aus:

- ▶ Wir installieren das Produkt **DB2 UDB ESE**.
- ▶ Die Installationsroutine soll uns alle benötigten **Benutzer**, wie den Exemplar-Eigner, die Benutzer für spezielle geschützte (fenced) DB2-Prozesse und die entsprechenden **Gruppen** automatisch anlegen. Die Namen der Benutzer und Gruppen lauten:
 - **db2ebuch** für den Exemplar-Eigner
 - **db2fbuch** für den Fenced-Benutzer
 - **sgebuch** für die Gruppe des Exemplar-Eigners
 - **fgebuch** für die Gruppe des Fenced-Benutzers
- ▶ Wir wählen den Installationstyp **CUSTOM**.
- ▶ Die zu installierende Landessprache ist **Deutsch**.
- ▶ Nachstehende Produktkomponenten sollen mit installiert werden:
 - DB2-Beispielprogramme
 - Anwendungsentwicklungs-Tools
- ▶ Die Lizenzvereinbarungen müssen **akzeptiert werden**.
- ▶ Das bei der Installation angelegte **Exemplar** soll den Namen **db2ebuch** erhalten.
- ▶ Als Kommunikationsprotokoll soll **TCP/IP** unterstützt werden.
- ▶ Der DB2 UDB **Verwaltungsserver (DAS)** soll bei der Installation mit angelegt werden.
- ▶ Die Toolskatalog-Datenbank soll mit dem Schema SYSTOOLS bei der Installation mit generiert werden.
- ▶ Die Kontaktliste der zu benachrichtigenden Personen soll **lokal** verwaltet werden.
- ▶ Wir geben die Namen der Kontaktpersonen, die bei DB2-Warnmeldungen informiert werden, bei der Installation mit an.

Vor der eigentlichen Installation kopieren wir uns von der Installations-CD das Response-File db2ese.rsp und passen dieses an unsere zuvor erwähnten Bedingungen an.

Die von uns angepassten Zeilen des Response-File sehen Sie in Abbildung 3.28.

** Sample response file for IBM DB2 Universal Database		** Installations-
** -----		** konfiguration
PROD	= ENTERPRISE_SERVER_EDITION	** Zu installierendes Produkt
** Select your install type.		** Installations-Typ
** -----		** -----
INSTALL_TYPE	= CUSTOM	** Installationstyp TYPICAL,COMPACT,CUSTOM
** The following are not part of typical installs		** Zu Installierende Komponenten
** -----		** -----
COMP	= APPLICATION_DEVELOPMENT_TOOLS	** Basis Anwendungsentwicklungs-Tools
COMP	= DB2_SAMPLE_APPLICATIONS	** Beispiel Programme
*COMP	= DEV_CENTER	** Entwicklungszentrale
** The following languages will apply to all translated components		
** -----		
LANG	= DE	** zu unterstützende Landessprache
*LANG	= EN	
* License Agreement Information for this Product		
* -----		
LIC_AGREEMENT	= ACCEPT	** Annahme der Lizenzbedingungen
* Instance Creation Settings		** Exemplar-Konfiguration
* -----		
INSTANCE	= db2ebuch	** Name des Exemplars
db2ebuch.NAME	= db2ebuch	** Name des Exemplar-Eigners
db2ebuch.GROUP_NAME	= sgebuch	** Name der Gruppe des Exemplareigners
db2ebuch.HOME_DIRECTORY	= /home/db2ebuch	** Verzeichnis des Exemplar-Eigners
db2ebuch.PASSWORD	= listec	** Kennwort des Exemplar-Eigners
db2ebuch.AUTHENTICATION	= SERVER	** Authentifizierungsart beim Login
db2ebuch.AUTOSTART	= YES	** Autostart des Exemplars
db2ebuch.SVCENAME	= db2cdb2ebuch	** Name des TCP/IP-Ports des Exemplars
db2ebuch.PORT_NUMBER	= 50002	** Portnummer für db2cdb2ebuch
db2ebuch.MAX_LOGICAL_NODES	= 1	** Anzahl der logischen Knoten
db2ebuch.TYPE	= ESE	
db2ebuch.FEDERATED	= YES	** Aktivierung für zusammengeschl. Systeme
* Fenced User Creation Settings		
* -----		
db2ebuch.FENCED_USERNAME	= db2fbuch	** Benutzer für abgeschirmte UDF/SP
db2ebuch.FENCED_GROUP_NAME	= fgebuch	** Gruppe des Benutzers db2fbuch
db2ebuch.FENCED_HOME_DIRECTORY	= /home/db2fbuch	** Verzeichnis des Benutzers db2fbuch
db2ebuch.FENCED_PASSWORD	= listecfu	** Kennwort des Benutzers db2fbuch
* Instance Profile Registry Settings		
* -----		
db2ebuch.DB2COMM	= TCPIP	** Kommunikationsprotokoll des Exemplars
* Administration Server Creation Settings		** DAS-Konfiguration
* -----		
DAS_USERNAME	= db2da	** Benutzer des Verwaltungsservers (DAS)
DAS_GROUP_NAME	= dasg	** Linux Gruppe des Verwaltungsservers (DAS)
DAS_HOME_DIRECTORY	= /home/db2das	** Verzeichnis des Verwaltungsservers
DAS_PASSWORD	= listecdas	** Benutzerpasswort des Verwaltungsservers
DAS_SMTP_SERVER	= gh.listec.de	** Name des SMTP Servers (Mail-Server)
* Database Settings		
* -----		
DATABASE	= toolsdb	** Schlüsselwort für Setup
toolsdb.DATABASE_NAME	= toolsdb	** Datenbankname
toolsdb.INSTANCE	= db2ebuch	** Exemplar, in dem die DB angelegt wird
toolsdb.LOCATION		= local
		** Datenbank lokal anlegen
TOOLS_CATALOG_DATABASE		= TOOLSDB
TOOLS_CATALOG_SCHEMA		= SYSTOOLS
		** Name der Toolskatalog-Datenbank
		** Schema für Tabellen der Toolskatalog-DB
* Contact List Information		
* -----		
DAS_CONTACT_LIST	= LOCAL	** Wo befindet sich die Liste der Kontakt-
		** Personen (LOKAL oder REMOTE)
* Contact Information for Each Instance		
* -----		
CONTACT	= contact1	** Kontaktperson bei DB2-Warnhinweisen
contact1.CONTACT_NAME	= admin	** Name der Kontaktperson
contact1.EMAIL	= admin@listec.de	** Mail-Adresse der Kontaktperson
contact1.PAGER	= NO	
contact1.INSTANCE	= db2ebuch	

Abbildung 3.28: Angepasstes Linux-Response-File

Da wir einige Dinge (z.B. das Anlegen eines Exemplars bzw. die genaue Bedeutung des Exemplar-Eigners unter Linux) erst in Abschnitt 3.9.2 „Anlegen eines Exemplars unter Linux“ behandeln, haben wir das Response-File mit den entsprechenden Kommentaren versehen.

Tipp:

Sollten Sie für die Benutzer, Gruppen und die Exemplare andere Namen vergeben, achten Sie darauf, diese an allen Stellen entsprechend zu ändern.

Tipp:

Im Beispiel-Response-File der CD sind die verschiedensten Installationsmöglichkeiten aufgeführt. Die Zeilen, die mit „*“ (Stern) anfangen, sind auskommentiert. Durch das Entfernen des Kommentarzeichens (*) aktivieren Sie die jeweilige Zeile. Bitte achten Sie darauf, dass standardmäßig gewisse Zeilen aktiviert sind. Sollten Sie diese nicht benötigen, müssen Sie diese auskommentieren.

3.6.3 Die eigentliche Installation

Nachstehend erklären wir Ihnen Schritt für Schritt die Installation der DB2 UDB V8 mit Hilfe eines Response-File unter Linux.

Die Installation ist so beschrieben, dass durchaus auch ein ungeübter Linux-Benutzer sie ausführen kann:

1. Melden Sie sich als Root an.
2. Stellen Sie den Zugriff auf Ihr CD-ROM-Laufwerk sicher. Sie können Ihr CD-ROM-Laufwerk mit nachstehendem Befehl mounten:

```
mount /dev/cdrom /media/cdrom
```
3. Sie finden das Response-File db2ese.rsp in folgendem Verzeichnis:

```
media/cdrom/db2/linux/samples
```
4. Da Sie das Response-File nicht direkt auf der CD-ROM ändern können, kopieren Sie bitte mit nachstehendem Befehl das Response-File in ein temporäres Verzeichnis:

```
cp /media/cdrom/db2/linux/samples/db2ese.rsp /tmp/
```
5. Entfernen Sie den Schreibschutz des Response-File mit dem Befehl:

```
chmod 777 /tmp/db2ese.rsp
```
6. Ändern Sie das Response-File entsprechend unseres Beispiels in Abbildung 3.28, und speichern Sie die Änderung ab.
7. Nach der Anpassung des Response-File können Sie die Installation wie folgt ausführen:

```
media/cdrom/db2/db2setup -r /tmp/db2ese.rsp -l /tmp/protokolldatei
```

 - -r bedeutet, dass Sie die Installation mit einem Response-File ausführen möchten.

- -l bedeutet, dass die Meldungen, die bei der Installation entstehen, in das File protokolldatei gespeichert werden. Diese können Sie sich bei eventuellen Installationsproblemen mit einem entsprechenden Editor ansehen.

Die allgemeine Befehlssyntax lautet:

```
<cd-rom>/db2setup -r <response file pfad>/<responsefile> -l <name der protokoll-  
datei>
```

Wenn Sie sich bei der Installation exakt an unsere Vorgaben gehalten haben und alle Grundvoraussetzungen für die Installation auf Ihrem System erfüllt waren, sollten Sie jetzt über eine einwandfreie DB2 UDB ESE-Installation verfügen.

Sollte dies nicht der Fall sein, müssen Sie jetzt in der bei der Installation angegebenen Protokolldatei nach den aufgetretenen Fehler- bzw. Warnmeldungen suchen, die entsprechenden Maßnahmen treffen und eventuell die Installation erneut durchführen.

Nachfolgende Abschnitte werden Ihnen helfen, eventuell aufgetretene Fehler zu verstehen.

3.7 Schritte nach der Installation

Je nachdem, mit welcher Installationsmethode Sie die DB2 UDB V8 durchgeführt haben, und abhängig von der Parameterauswahl, die Sie dabei treffen, müssen Sie nach der Installation eventuell verschiedene weitere Schritte vornehmen, um das DB2 UDB an Ihre Anforderungen anzupassen.

Ebenso ist es denkbar, dass Sie die bei der Installation angegebene Konfiguration erweitern oder ändern möchten.

In den folgenden Abschnitten schaffen wir die Grundlagen, um diese Aufgaben in der Praxis ausführen zu können, und gehen dabei auf folgende Punkte ein:

- In welchen Verzeichnissen finden Sie die installierten DB2 UDB-Komponenten?
- Wie legen Sie ein Exemplar an, und was ist dabei zu beachten?
- Wie ändern Sie die Konfiguration eines bestehenden Exemplars?
- Wie können Sie den Verwaltungsserver DAS und den Toolskatalog manuell erstellen?
- Welche Bedeutung haben die Umgebungsvariablen, und wie können Sie diese anpassen?
- Welche Möglichkeiten und Vorteile stellt Ihnen die DB2 UDB-Befehlszeile (CLP) zur Verfügung?
- Tipps aus der Praxis.

3.8 In welchen Verzeichnissen finden Sie die installierten DB2 UDB-Komponenten?

Welche Verzeichnisse bei der Installation der DB2 UDB angelegt werden, möchten wir anhand unseres Installationsbeispiels mit der DB2 UDB ESE in Abbildung 3.29 und Abbildung 3.30 kurz erläutern.

Für das Betriebssystem Linux gehen wir an dieser Stelle nur auf die Verzeichnisse ein, die durch die reine Produktinstallation entstehen. Die weiteren Verzeichnisse, die durch das Anlegen des Standardexemplars DB2 (in unserem Fall DB2EUBCH) und des DAS entstehen, werden hier nicht betrachtet.

Verzeichnisse unter Windows

adsm	- Tivoli Storage Manager Dateien
BIN	- ausführbare Dateien
bnd	- Bind-Dateien für DB2-Tools
cfg	- Standard System-Konfigurationsdateien
conv	- Codepage-Konvertierungstabellen
dasfcu	- DLLs für DAS-Funktionen
db2xml	- Hilfe zum XML Extender
doc	- Dokumentation
FUNCTION	- Standard-Verzeichnis für UDFs
FUNCTION\unfenced	- Standard-Verzeichnis für nicht abgeschirmte UDFs
help	- Hilfedateien
include	- Include-Files
infopop	- Popup-Info
java	- Java-Programme
lib	- Bibliotheken
license	- Lizenzinformationen
map	- Map-Dateien für DB2 Connect
MISC	- Script zum Anlegen der Explain-Tabellen
msg	- Nachrichten-Katalog
Readme	- Readme-Datei
samples	- Beispiel-Programme und -Scripts
tivready	- Tivoliunterstützung
TOOLS	- DB2-Tools
tutorials	- Lernprogramme

Abbildung 3.29: Durch die Installation angelegte Verzeichnisse (Windows)

Verzeichnisse unter Linux

Readme	- Readme-Dateien usw.
adm	- Systemadministrator und ausführbare Dateien
adsm	- Tivoli Storage Manager Dateien
bin	- ausführbare Dateien
bnd	- Bind-Dateien für DB2-Tools
cc	- Dateien der Steuerzentrale
cfg	- Standard System-Konfigurationsdateien
conv	- Codepage-Konvertierungstabellen
doc	- Dokumentation
function	- Standard-Verzeichnis für UDFs
function\unfenced	- Standard-Verzeichnis für nicht abgeschirmte UDFs
install	- Installationsprogramm
instance	- Exemplar-Scripts
java	- Java-Programme
lib	- Bibliotheken
map	- Map-Dateien für DB2 Connect
misc	- Script zum Anlegen der Explain-Tabellen
msg /\$L	- Nachrichten-Katalog
samples	- Beispiel-Programme und -Scripts

Abbildung 3.30: Durch die Installation angelegte Verzeichnisse (Linux)

3.9 Welche Befehle gibt es, um ein Exemplar zu verwalten?

Um Exemplare (instances) zu verwalten, gibt es folgende Befehle:

- ▶ db2icrt – Anlegen eines Exemplars (instance)
- ▶ db2idrop – Löschen eines Exemplars
- ▶ db2ilist – Auflisten aller vorhandenen Exemplare
- ▶ db2iupdt – Dient zur Migration von Exemplaren (siehe Anhang „Migration auf DB2 V8“)
- ▶ db2start – Starten eines Exemplars
- ▶ db2stop – Stoppen eines Exemplars

Zur Konfiguration eines Exemplars stehen Ihnen weitere Befehle über die DB2-Befehlszeile (CLP) zur Verfügung (siehe Abschnitt 3.11 „Die DB2-Befehlszeile (CLP)“).

3.9.1 Allgemeines zum Anlegen eines Exemplars

Beim Anlegen eines Exemplars (instance) müssen wir zwischen den Betriebssystem-Welten Windows und Unix unterscheiden.

Dies ist notwendig, da es auf den Unix-Betriebssystemen möglich ist, mehrere DB2-Versionen mit unterschiedlichen Fix-Packs (ab DB2 UDB V8 und nur bei ESE) parallel zu betreiben. Dies gilt nicht für die Windows-Betriebssysteme.

Somit muss bei Unix-Betriebssystemen darauf geachtet werden, aus welchem Pfad heraus der Befehl zum Anlegen des Exemplars aufgerufen wird. Für Windows-Betriebssysteme spielt dies keine Rolle, da hier nur eine DB2 UDB-Version installiert sein kann und somit der Befehl auf dem System nur einmal vorhanden ist. Dies setzt natürlich eine korrekte Installation von DB2 voraus.

Weiter benötigen wir bei den Unix-Betriebssystemen vor dem Anlegen eines Exemplars einen Exemplar-Eigner. Der Exemplar-Eigner ist der Benutzer (user), unter dem das DBMS gestartet wird. Der Name des Exemplars entspricht exakt dem Namen des Exemplar-Eigners.

Des Weiteren benötigen wir einen Benutzer, unter dessen Berechtigung spezielle DBMS-Prozesse für geschützte (fenced) benutzerdefinierte Funktionen (User Defined Functions – UDF) und gespeicherte Prozeduren (Stored Procedures – SP) ablaufen.

In den folgenden Beispielen werden wir für die Betriebssystem-Welt Unix das Betriebssystem Linux (SUSE) und für die Windows-Welt das Betriebssystem Windows 2000 betrachten.

3.9.2 Anlegen eines Exemplars unter Linux

Wie schon zuvor erwähnt, benötigen wir vor dem Anlegen des Exemplars zwei eigenständige Linux-Benutzer: einen Benutzer als Exemplar-Eigner und einen Benutzer, unter dem die entsprechenden DBMS-Prozesse für geschützte UDFs und SPs ablaufen können.

Da sich die Sicherheitseigenschaften eines Exemplars nicht auf den einzelnen Benutzer, sondern auf Benutzergruppen beziehen, legen wir zusätzlich zu den beiden Benutzern folgende zwei Gruppen an:

- ▶ eine Gruppe für Administratoren des Exemplars, der auch der Exemplar-Eigner zugeordnet ist
- ▶ eine Gruppe für den Fenced-Benutzer, dem die speziellen DBMS-Prozesse (die geschützten UDFs und SPs) zugeordnet sind. Diese Gruppe ist optional und wird hier der Ordnung halber angelegt.

Anlegen der Linux-Gruppen

Wie zu vor erwähnt, benötigen Sie eine Gruppe für die Exemplar-Administration und optional eine zweite Gruppe. Die Administrationsgruppe des Exemplars nennen wir für unser Beispiel SGEBUCH (System Gruppe Exemplar BUCH). Diese wird beim Erstellen des Exemplars automatisch als SYSADM_GROUP (siehe Kapitel 9, „Sicherung und Wiederherstellung einer Datenbank“ in der Exemplar-Konfiguration (DBM CFG) eingetragen.

Die Gruppe, unter der die Datenbankprozesse für die geschützten (fenced) benutzerdefinierten Funktionen (UDF) und die gespeicherten Prozeduren (Stored Procedures – SP) laufen, nennen wir FGEBUCH (Fenced Gruppe Exemplar Buch).

Mit den folgenden Befehlen legen wir die beiden Gruppen an:

```
groupadd -g sgebuch
groupadd -g fgebuch
```

Bei -g handelt es sich um die jeweilige Gruppen-ID, die pro System nur einmal vergeben werden darf. Bitte beachten Sie, dass Gruppen-IDs unter 100 für Systemgruppen reserviert sind.

Anlegen der benötigten Linux-Benutzer

Bevor wir den Benutzer anlegen, sollten wir uns noch einmal darüber klar werden, dass der Name des Benutzers später als der Name des Exemplars (instance) verwendet werden muss.

In unserem Fall möchten wir dem Exemplar den Namen **DB2EBUCH** geben (DB2 Exemplar Buch).

Den Benutzer für die speziellen DBMS-Prozesse nennen wir **DB2FBUCH**.

Um die beiden Benutzer unter Linux anzulegen, geben wir folgenden Befehl ein:

```
useradd -g sgebuch -m -p <kennwort> db2ebuch  
useradd -g fgebuch -m -p <kennwort> db2fbuch
```

Hinweis:

Ersetzen Sie die Variable <kennwort> durch ein entsprechend sicheres Kennwort für den jeweiligen Benutzer!

Anlegen des Exemplars

Als Nächstes legen wir das Exemplar mit dem Namen **db2ebuch** an. Dabei konzentrieren wir uns auf die wesentlichen Bestandteile des Befehls.

Mit dem Befehl

```
./db2icrt -u db2fbuch db2ebuch
```

legen wir das Exemplar db2ebuch an.

Um den Befehl in der Art aufzurufen, müssen wir uns in dem entsprechenden Verzeichnis befinden:

```
AIX:    /usr/opt/db2_08_01/instance  
Unix:   /opt/IBM/db2/V8.1/instance
```

Im Gegensatz zur Windows dauert die Ausführung des Befehls in der Unix-Welt ein wenig länger. Warum das so ist, werden Sie nach dem nächsten Abschnitt nachvollziehen können.

Was passiert bei der Ausführung des Befehls?

Wir konzentrieren uns an dieser Stelle nur auf die automatisch angelegten Verzeichnisse unter dem Hauptverzeichnis des Exemplar-Eigners (instance owner).

Unter dem \$HOME-Verzeichnis des Exemplar-Eigners db2ebuch wird durch das Ausführen des Befehls db2icrt das Unterverzeichnis /sqllib/ mit Dateien und Strukturen automatisch angelegt. Wie Sie in Abbildung 3.31 sehen, befinden sich in diesem Verzeichnis viele Verweise (Links) auf das DB2 UDB-Installationsverzeichnis.

Diese Technik ermöglicht es unter Unix, pro Exemplar auf eine andere DB2 UDB-Installation zu verweisen. Somit ist es möglich, dass auf ein und derselben Maschine mehrere DB2 UDB-Versionen installiert und verwaltet werden können.

```

insgesamt 34
drwxrwsr-t 13 db2ebuch      928 2003-10-15 09:39 .
drwxr-xr-x 12 db2ebuch     1008 2003-10-15 10:16 ..
-r--r--r-- 1 db2ebuch       4 2003-10-01 15:06 .ftok
drwxrwxr-x 2 db2ebuch      48 2003-10-01 15:06 .netls
lrwxrwxrwx 1 root         24 2003-10-01 15:06 Readme -> /opt/IBM/db2/V8.1/Readme
drwxr-xr-x 2 db2ebuch     800 2003-10-01 15:06 adm
lrwxrwxrwx 1 root        22 2003-10-01 15:06 adsm -> /opt/IBM/db2/V8.1/adsm
drwxr-x-- 2 db2ebuch      48 2003-10-01 15:06 backup
lrwxrwxrwx 1 root        21 2003-10-01 15:06 bin -> /opt/IBM/db2/V8.1/bin
lrwxrwxrwx 1 root        21 2003-10-01 15:06 bnd -> /opt/IBM/db2/V8.1/bnd
drwxrwsr-t 2 db2ebuch    312 2003-10-01 15:06 cfg
lrwxrwxrwx 1 root        22 2003-10-01 15:06 conv -> /opt/IBM/db2/V8.1/conv
drwxrwsr-t 2 db2ebuch    256 2003-10-01 15:06 ctrl
-rwxr-xr-x 1 db2ebuch   5150 2003-10-01 15:06 db2cshrc
drwxrwsrwt 2 db2ebuch    112 2003-10-01 15:06 db2dump
-r--r--r-- 1 db2ebuch     11 2003-10-01 15:06 db2nodes.cfg
-rwxr-xr-x 1 db2ebuch   4776 2003-10-01 15:06 db2profile
-rw-rw-r-- 1 db2ebuch   4096 2003-10-15 09:26 db2system
lrwxrwxrwx 1 root        21 2003-10-01 15:06 doc -> /opt/IBM/db2/V8.1/doc
drwxrwsr-t 4 db2ebuch    448 2003-10-01 15:06 function
lrwxrwxrwx 1 root        25 2003-10-01 15:06 include -> /opt/IBM/db2/V8.1/include
lrwxrwxrwx 1 root        25 2003-10-01 15:06 infopop -> /opt/IBM/db2/V8.1/infopop
lrwxrwxrwx 1 root        22 2003-10-01 15:06 java -> /opt/IBM/db2/V8.1/java
lrwxrwxrwx 1 root         4 2003-10-01 15:06 java12 -> java
lrwxrwxrwx 1 root        21 2003-10-01 15:06 lib -> /opt/IBM/db2/V8.1/lib
drwxrwsr-t 2 db2ebuch     48 2003-10-15 09:26 log
lrwxrwxrwx 1 root        21 2003-10-01 15:06 map -> /opt/IBM/db2/V8.1/map
lrwxrwxrwx 1 root        22 2003-10-01 15:06 misc -> /opt/IBM/db2/V8.1/misc
lrwxrwxrwx 1 root        21 2003-10-01 15:06 msg -> /opt/IBM/db2/V8.1/msg
-rw-rw-r-- 1 db2ebuch     35 2003-10-01 15:06 profile.env
lrwxrwxrwx 1 root        25 2003-10-01 15:06 samples -> /opt/IBM/db2/V8.1/samples
drwxr-xr-x 2 db2ebuch    176 2003-10-01 15:06 security
drwxrwxr-x 2 db2ebuch    120 2003-10-15 09:39 sqldbdir
drwxrwsrwx 2 db2ebuch    112 2003-10-15 09:38 tmp
lrwxrwxrwx 1 root        23 2003-10-01 15:06 tools -> /opt/IBM/db2/V8.1/tools
-rwxr-xr-x 1 db2ebuch      0 2003-10-01 15:06 usercshrc
-rwxr-xr-x 1 db2ebuch      0 2003-10-01 15:06 userprofile

```

Abbildung 3.31: Verzeichnisstruktur des Exemplars db2ebuch

3.9.3 Anlegen eines Exemplars unter Windows

In der Windows-Umgebung ist das Anlegen eines Exemplars mit weitaus weniger Aufwand verbunden. Mit dem Befehl `db2icrt db2ebuch` können Sie das Exemplar `db2ebuch` anlegen. Benutzer oder Gruppen müssen hier nicht extra angelegt werden. Allerdings ist darauf zu achten, dass dem Exemplar unter Windows keine Gruppe für die `SYSADM_GROUP` zugewiesen wird. Dies bedeutet, dass Sie zur Administration des Exemplars vorerst die Administrator-Berechtigung auf der Betriebssystemebene benötigen. Sicherlich kann im Nachhinein dem Exemplar über den Parameter `SYSADM_GROUP` eine Gruppe zugewiesen werden (Näheres erfahren Sie in Kapitel 10).

Was passiert bei dem Befehl `db2icrt`?

Anders als unter Unix wird hier ein Verzeichnis `DB2EBUCH` unter dem eigentlichen Hauptverzeichnis `<DB2 UDB-Installationsverzeichnis>\SQLLIB\` angelegt.

Dieses Verzeichnis enthält weitere Unterverzeichnisse und dient dazu, dass die für das Exemplar `DB2EBUCH` notwendigen Konfigurations- und Informationsdateien abgelegt werden können.

Abbildung 3.32 zeigt Ihnen, welche weiteren Unterverzeichnisse unter dem Exemplarverzeichnis `DB2EBUCH` angelegt werden.

```

Datenträger in Laufwerk D: ist Proramme
Volumeseriennummer: 8431-3800

Verzeichnis von D:\IBM\SQLLIB\DB2EBUCH

12.10.2003  19:59    <DIR>          .
12.10.2003  19:59    <DIR>          ..
28.09.2003  19:43    <DIR>          ctrl
28.09.2003  19:43             4.096    db2system
28.09.2003  19:43    <DIR>          log
28.09.2003  19:43    <DIR>          security
28.09.2003  19:43    <DIR>          TMP
                2 Datei(en)             4.096 Bytes
                6 Verzeichnis(se),      551.088.128 Bytes frei

```

Abbildung 3.32: Verzeichnisstruktur des Exemplars DB2EBUCH

3.9.4 Der Befehl DB2ICRT

Um ein Exemplar mit dem Befehl `db2icrt` anzulegen, benötigen Sie unter Windows keine besondere Berechtigung. Unter Unix hingegen benötigen Sie root-Rechte.

Da sich bei diesem Befehl die Syntax der beiden Betriebssystem-Welten Linux und Windows stark unterscheidet, zeigen wir Ihnen die Syntax des Befehls etwas genauer.

Der Befehl DB2ICRT unter Unix

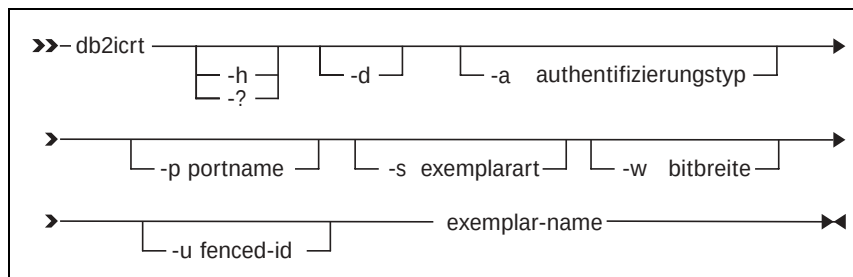


Abbildung 3.33: Befehlssyntax des Befehls DB2ICRT unter Unix

Erklärung der Optionen

- ▶ `-?` oder `-h` ermöglicht den Aufruf der Hilfe.
- ▶ `-d` ermöglicht es, den *Debug*-Modus einzuschalten. Dies macht allerdings nur dann Sinn, wenn Sie mit dem IBM-Support in Kontakt stehen, der diese Debug-Informationen auswerten kann.
- ▶ Die Option `-a` kann genutzt werden, um den Authentifizierungstyp für das Exemplar festzulegen (Details finden Sie in Kapitel 6.3). Mögliche Werte sind hierbei:
 - `SERVER` (ist der Standardwert)
 - `CLIENT`
 - `SERVER_ENCRYPT`

- ▶ Mit dem Parameter `-p` können Sie die Portnummer oder den Portnamen angeben, unter dem der Dienst (DBMS) des Exemplars läuft.
- ▶ Mit der Option `-s` haben Sie die Möglichkeit, die Art des Exemplars (instance) zu bestimmen. Folgende Möglichkeiten stehen Ihnen zur Verfügung:
 - Client kennzeichnet das Exemplar als Client-Exemplar.
 - ESE wird genutzt, um das Exemplar als Personal Edition (PE), Einzel-Partition-Enterprise Server Edition (ESE) oder als Exemplar für DB2 Connect zu kennzeichnen.
 - Der Parameter WSE kennzeichnet das Exemplar als *Workgroup Server Edition*-Exemplar, das nicht-partitioniert ist.
- ▶ Die Option `-w` bitbreite ermöglicht Ihnen bei den Betriebssystemen AIX 5L, HP-UX und Sun Solaris zwischen einem 32- oder 64-Bit-Exemplar zu wählen. Gültige Werte dabei sind 32 oder 64.
- ▶ `-u` steht für die fenced-ID. In dieser Option geben Sie den Benutzer an, unter dem die geschützten (fenced) benutzerdefinierten Funktionen (User Defined Functions) bzw. gespeicherte Prozeduren (Stored Procedures) laufen.

Der Befehl DB2ICRT unter Windows

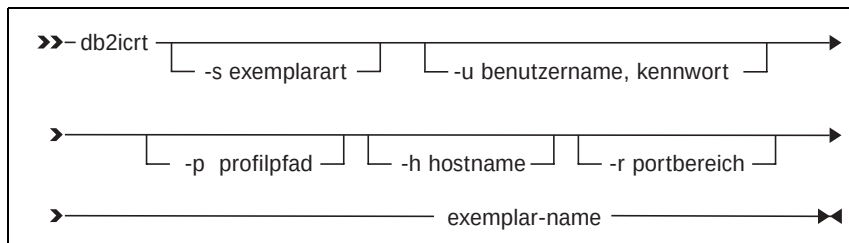


Abbildung 3.34: Befehlssyntax des Befehls DB2ICRT unter Windows

Erklärung der Optionen

- ▶ `-?` oder `-h` ermöglicht den Aufruf der Hilfe.
- ▶ `-s` steht für die Exemplarart. Folgende Parameter sind möglich:
 - Client wird genutzt, um ein Exemplar auf einem Client anzulegen. Diese Option sollten Sie beim Anlegen eines Exemplars mit dem Produkt *DB2 Connect Personal Edition* nutzen.
 - Standalone benutzen Sie für ein Exemplar, das nur lokale Datenbanken verwaltet.
 - ESE wird genutzt, um das Exemplar als *Personal Edition* (PE), Einzel-Partition-Enterprise Server Edition (ESE) oder als Exemplar für *DB2 Connect* zu kennzeichnen.
 - Der Parameter WSE kennzeichnet das Exemplar als *Workgroup Server Edition*-Exemplar.

- ▶ -u Benutzername, Kennwort – Diese Option ermöglicht es Ihnen, die User-ID und das Kennwort des Benutzers anzugeben, unter dessen Berechtigung das Exemplar läuft. Dieser Parameter wird für partitionierte Datenbankexemplare benötigt.
- ▶ -p profilpfad ermöglicht es Ihnen, den Profilpfad zu spezifizieren.
- ▶ -h – Wird nur bei partitionierten Exemplaren benötigt. Mit diesem Parameter können Sie den Standard-Hostnamen der Maschine für das entsprechende Exemplar abändern.
- ▶ -r portbereich – Diese Option ermöglicht es Ihnen, für partitionierte Exemplare eine Range der Portnummer der Exemplare anzugeben. Wenn diese Option genutzt wird, wird die Service-Datei der lokalen Maschine geändert. Nachstehende Einträge werden vorgenommen:
 - DB2_InstName baseport/tcp
 - DB2_InstName_END endport/tcp
- ▶ exemplar-name spezifiziert den Namen des Exemplars.

Resümee

Der Befehl DB2ICRT ist eine der wenigen Ausnahmen, bei denen sich die Syntax zwischen den Betriebssystemen Unix und Windows unterscheidet. Dennoch zeigt Ihnen dieses Beispiel, dass sich selbst bei einem identischen Kern des DBMS das verwendete Betriebssystem auf die Administration der DB2 UDB auswirkt und Sie ganz ohne gewisse Grundlagenkenntnisse über das jeweilige Betriebssystem nicht auskommen werden.

Eine wesentlich größere Rolle spielen diese Kenntnisse bei den Performance- und Tuning-Untersuchungen bzw. -Maßnahmen.

3.9.5 Starten und Stoppen eines Exemplars

Ein Exemplar kann vom SYSADM über den Befehl db2start gestartet werden. Dabei wird der Name des (zu startenden) Exemplars aus der Umgebungsvariable DB2INSTANCE übernommen. Dies wird in Unix durch das db2profile in der Umgebung gesetzt (siehe auch Abschnitt 3.10 „Benutzer für DB2 im Unix-Umfeld einrichten“) und in Windows durch die Systemumgebung.

Das Stoppen eines Exemplars erfolgt ähnlich: durch die Eingabe des Befehls db2stop.

Sollten Datenbankverbindungen noch aktiv sein, wenn Sie ein Exemplar stoppen wollen, bekommen Sie eine entsprechende Meldung. Wollen Sie trotzdem den Stopp durchsetzen, haben Sie die Möglichkeit, mit der Option force Verbindungen aller Benutzer zu entfernen.

3.10 Benutzer für DB2 im Unix-Umfeld einrichten

Im Gegensatz zu Windows-Betriebssystemen müssen Sie im Unix-Umfeld noch einen kleinen Konfigurationsschritt durchführen, um es Benutzern zu ermöglichen, mit einem DB2-Exemplar arbeiten zu können.

In diesem Konfigurationsschritt geht es darum, verschiedene Umgebungsvariablen, wie z.B. den Pfad für DB2, in der Umgebung des Benutzers bekannt zu machen.

Für den Exemplareigner wurde dies bei der Installation automatisch durchgeführt. Für alle weiteren Benutzer müssen Sie diese Anpassung noch vornehmen, indem Sie in der `.profile`-Datei des Benutzers einen Verweis auf das `db2profile` aus dem Exemplarverzeichnis einfügen.

Dazu fügen Sie folgende Zeile in das `.profile` des Benutzers ein:

```
. /<$HOME des Exemplareigners>/sqllib/db2profile
```

`<$HOME des Exemplareigners>` ist das Verzeichnis des Exemplareigners, also z.B. `/home/buchinst` für das Exemplar bzw. für den Exemplareigner `buchinst`.

Durch diesen Eintrag werden bei der nächsten Anmeldung des Nutzers oder der expliziten (Neu-) Ausführung der `.profile`-Datei (durch Eingabe von `. .profile` in der Shell) der DB2-Pfad und andere Variablen in der Umgebung des Benutzers gesetzt.

Tipp:

Einen kleinen Funktionstest können Sie mit der Eingabe von `db2 ?` in der Shell durchführen. Er sollte eine DB2-Rückmeldung liefern.

3.11 Die DB2-Befehlszeile (CLP)

CLP steht für `command line processor` und stellt die Befehlszeile von DB2 dar. Sie haben so die Möglichkeit, DB2- und SQL-Befehle direkt aus dieser Umgebung abzusetzen.

Dabei ist besonders die Möglichkeit interessant, Scripts über diese Schnittstelle auszuführen, um so verschiedene Abläufe zu automatisieren.

Prinzipiell stehen drei Möglichkeiten zu Verfügung, um DB2-Befehle über eine Befehlszeile abzusetzen:

- ▶ das Befehlsfenster
- ▶ der Befehlszeilenprozessor
- ▶ die Befehlszentrale

Während wir auf die Befehlszentrale, die eine grafische Benutzeroberfläche darstellt, in Kapitel 4 eingehen, wollen wir uns hier auf die reine Befehlszeile, also sozusagen die reine Shell, konzentrieren. Diese DB2-Schnittstelle steht Ihnen auf jeder Oberfläche bzw. auch auf allen Client-Installationen zu Verfügung, und zwar auch dann, wenn das GUI nicht installiert ist.

3.11.1 Interaktiv oder nicht?

Das DB2-Befehlsfenster heißt auch *nicht interaktive Befehlszeile*, während der DB2-Befehlszeilenprozessor als *interaktive Befehlszeile* bezeichnet wird.

Beide besitzen den gleichen DB2-Funktionsumfang, unterscheiden sich aber leicht in der Handhabung. Auch das Erscheinungsbild ist ebenfalls leicht unterschiedlich, so erscheint die interaktive Befehlszeile mit einem DB2 =>-Prompt, während die nicht-interaktive Befehlszeile der normalen Shell (Windows-Befehlsfenster) gleicht (siehe Abbildung 3.35).

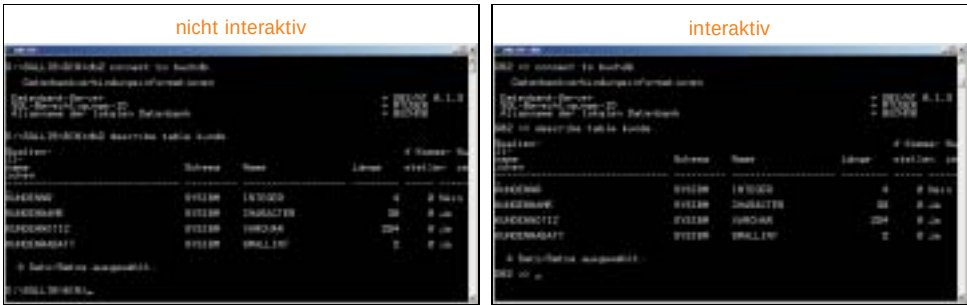


Abbildung 3.35: Vergleich zwischen nicht interaktiver und interaktiver Befehlszeile

Die interaktive Befehlszeile ermöglicht die Eingabe von DB2-Befehlen und SQL-Abfragen ohne das führende db2. Wie Sie in Abbildung 3.35 sehen, wird der Befehl zum Verbindungsaufbau mit der Datenbank in der nicht-interaktiven Umgebung durch ein

db2 connect to buchdb

gegeben, während

connect to buchdb

bei der interaktiven Befehlszeile ausreicht.

Über den Befehl DB2 kann von dem nicht-interaktiven in den interaktiven Modus gewechselt werden. Auf der Befehlszeile wird also das Programm db2 ausgeführt, das zu diesem DB2 =>-Prompt führt. Über den QUIT-Befehl kann dieser Modus wieder verlassen werden.

Letztendlich ist Gewohnheit oder der persönliche Geschmack mit ausschlaggebend dafür, welche Variante Sie einsetzen wollen. Rein DB2-technisch ist die Funktionalität gleich.

Wir empfehlen allerdings die nicht-interaktive Befehlszeile, da diese zusätzlich zum „normalen“ Funktionsumfang einer Befehlszeile auch direkt Betriebssystem-Befehle verarbeiten kann. So kann hier ohne Probleme ein `mkdir`-Befehl zu Erstellung von Verzeichnissen zwischen DB2-Befehlen abgesetzt werden und auch ein `| more` ist oft hilfreich, um eine seitenweise Anzeige zu bekommen. Obwohl dies auch teilweise mit der interaktiven Umgebung möglich ist, bietet die nicht-interaktive Befehlszeile mehr Flexibilität und rechtfertigt die „Mehreingabe“ von `db2` am Anfang jedes Befehls.

Dieses `db2` kennzeichnet DB2- oder SQL-Befehle und grenzt sie so von den Betriebssystembefehlen ab. Um eine Missinterpretation von Sonderzeichen zu vermeiden, sollten Sie sich angewöhnen, SQL-Befehle in doppelte Anführungszeichen zu schreiben, also z.B. so:

```
db2 "select * from kunde where kundenr > 100"
```

Somit wird das `>`-Zeichen als „größer als“ interpretiert und nicht als Umleitungs-symbol für die Ausgabe.

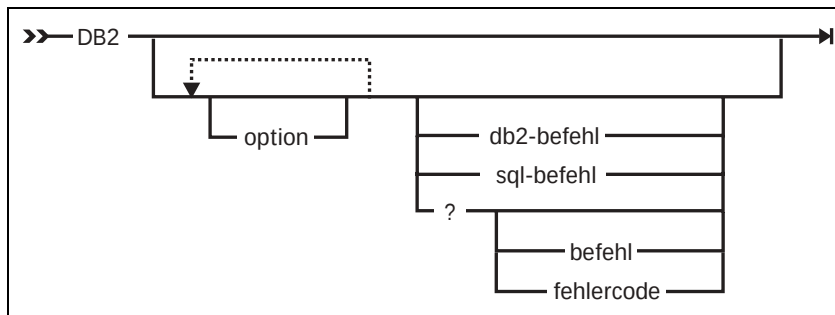


Abbildung 3.36: db2-Syntaxdiagramm

Abbildung 3.36 zeigt die Möglichkeiten mit dem Schlüsselwort `DB2`. Der reine Aufruf ohne Optionen startet den interaktiven Modus.

Die Möglichkeiten für `option` werden im nächsten Teilkapitel separat behandelt.

In der (nicht-interaktiven) Befehlszeile steht das Schlüsselwort `DB2` bzw. `db2` vor jedem DB2-Befehl und jeder SQL-Abfrage. Auch wenn das Syntaxdiagramm dies nicht explizit ausweist, sind Anführungszeichen um dem `sql-befehl` empfehlenswert, auch wenn sie nicht immer benötigt werden.

Über das Fragezeichen bekommen Sie einen Hilfetext mit einer Auswahl an DB2-Befehlen angezeigt. Um einen einzelnen Befehl oder eine Befehlsgruppe auszuwählen, können Sie diesen bzw. diese nach dem Fragezeichen angeben. Sie erhalten so detailliertere Hinweise zu den Optionen und zur Syntax dieses Befehls (siehe Abbildung 3.37).

Ein weitere in der Praxis sehr nützliche Funktionalität ist die Abrufmöglichkeit von Fehler-Hilfeinformationen. Dazu geben Sie nach dem Fragezeichen den Fehlercode ein (siehe Abbildung 3.38).

```
D:\SQLLIB\BIN>db2 ? list command
LIST COMMAND OPTIONS

Anm.: Stellen Sie den Befehlen in der Eingabeaufforderung 'db2' voran.
      Sonderzeichen KÖNNEN eine Escape-Folge (\) erfordern, z. B.:
      db2 \? change database
      db2 ? change database xxx comment with \"text\"

D:\SQLLIB\BIN>db2 ? update command
UPDATE COMMAND OPTIONS USING {optionen ...}

Optionen:
a {ON|OFF}          SQLCA anzeigen
c {ON|OFF}          Automatisches Festschreiben
e {ON {C|S} | OFF} SQLCODE/SQLSTATE anzeigen
l {ON dateiname | OFF} Befehle protokollieren
n {ON|OFF}          Zeilenvorschubzeichen entfernen
o {ON|OFF}          Ausgabe anzeigen
p {ON|OFF}          DB2-Eingabeaufforderung anzeigen
r {ON dateiname | OFF} Ausgabebericht in Datei speichern
s {ON|OFF}          Bei Fehler Befehlsausführung stoppen
v {ON|OFF}          Aktuellen Befehl zurückmelden
w {ON|OFF}          Warnungen für FETCH/SELECT anzeigen
z {ON dateiname | OFF} Gesamte Ausgabe in Datei speichern

Anm.: Stellen Sie den Befehlen in der Eingabeaufforderung 'db2' voran.
      Sonderzeichen KÖNNEN eine Escape-Folge (\) erfordern, z. B.:
      db2 \? change database
      db2 ? change database xxx comment with \"text\"

D:\SQLLIB\BIN>
```

Abbildung 3.37: Beispiel: db2 ? für Befehlshilfe

```
D:\SQLLIB\BIN>db2 ? SQL30082

SQL30082N Die Verbindung konnte aufgrund der
          Sicherheitsbedingung "<ursachencode>"
          ("<ursachenzeichenfolge>") nicht hergestellt werden.

Erklärung:

Die Verbindung zu einem fernen Datenbankserver konnte aufgrund
ungültiger oder falscher Sicherheitsinformationen nicht
hergestellt werden. Die Ursache des Sicherheitsfehlers wird
durch den <fehlercode> und den entsprechenden Wert für die
<ursachenzeichenfolge> beschrieben.

Liste der Ursachencodes und der entsprechenden
Ursachenzeichenfolgen:

0 (NOT SPECIFIED) Der spezifische Sicherheitsfehler wurde nicht
angegeben.

1 (PASSWORD EXPIRED) Das in der Anforderung angegebene Kennwort
ist abgelaufen.

...
```

Abbildung 3.38: Beispiel: db2 ? für Informationen zu Fehlercodes

Hinweis:

Eine nützliche „Schnellreferenz“ der DB2-Befehle – viele werden erst in späteren Kapiteln besprochen – finden Sie unter:

<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0211yip/0211yip2.html>

Tipp:

Sollten Sie in Windows-Umgebungen Probleme mit der Darstellung der deutschen Umlaute haben, stellen Sie in den Eigenschaften des Befehlsfensters die Schrift auf eine TrueType-Schriftart um (siehe Abbildung 3.39). Das hört sich seltsam an – aber es funktioniert!

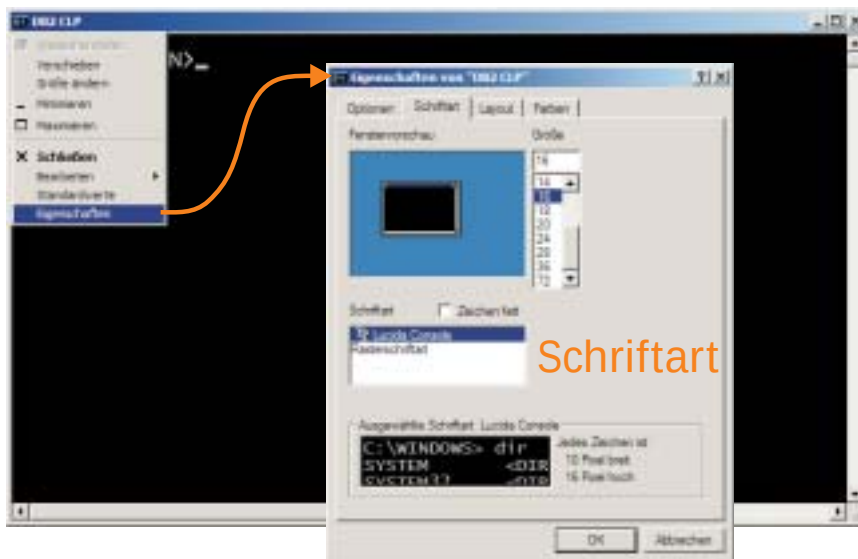


Abbildung 3.39: Befehlsfenster anpassen

Über das Icon in der linken oberen Fenster-Ecke kommen Sie durch einen Klick mit der linken Maustaste zum Fenster-Menü, aus dem Sie dann den EIGENSCHAFTEN-Eintrag auswählen können. Die Registerkarte SCHRIFTART bietet dann eine TrueType-Schriftart zur Auswahl an. Diese ist durch das TT-Zeichen markiert.

Bevor Sie diese Änderung mit dem OK-Button bestätigen, haben Sie noch die Möglichkeit, über den LAYOUT-Reiter die FENSTERGRÖSSE und FENSTERPUFFERGRÖSSE einzustellen, um so z.B. durch einen größeren Zeilenpuffer über einen Scroll-Balken in größeren Ausgaben navigieren zu können.

3.11.2 Optionen der DB2-Befehlszeile

Verschiedene Optionen ermöglichen individuelle Einstellungen und ein komfortableres Arbeiten mit der DB2-Befehlszeile.

Die Anzeige der Optionen und deren Standardbelegung können Sie über folgenden Befehl bekommen:

```
list command options bzw. db2 list command options
```

Hinweis:
Im Verlauf dieses Buchs geben wir die Befehle und Beispiele immer ohne das führende db2 an. Das soll *nicht* bedeuten, dass der interaktive Modus verwendet werden sollte!

D:\SQLLIB\BIN>db2 list command options

Optionseinstellungen für den Befehlszeilenprozessor

Wartezeit für den Nachrechnerprozess (s) (DB2BQTIME) = 1
Wiederholungen f. Verbindung zu Nachrechnerprozess (DB2BQTRY) = 60
Wartezeit für Anforderungswarteschlange (s) (DB2RQTIME) = 5
Wartezeit für Eingabewarteschlange (s) (DB2IQTIME) = 5
Befehlsoptionen (DB2OPTIONS) =

Option	Beschreibung	Aktuelle Einstellung
-a	SQLCA anzeigen	OFF
-c	Automatisches Festschreiben	ON
-e	SQLCODE/SQLSTATE anzeigen	OFF
-f	Eingabedatei lesen	OFF
-l	Befehle protokollieren	OFF
-n	Zeilenvorschubzeichen entfernen	OFF
-o	Ausgabe anzeigen	ON
-p	DB2-Eingabeaufforderung anzeigen	ON
-r	Ausgabebericht in Datei speichern	OFF
-s	Bei Fehler Befehlsausführung stoppen	OFF
-t	Als Abschlusszeichen ';' verwenden	OFF
-v	Aktuellen Befehl zurückmelden	OFF
-w	Warnungen für FETCH/SELECT anzeigen	ON
-x	Drucken von Spaltenüberschr. unterdrücken	OFF
-z	Gesamte Ausgabe in Datei speichern	OFF

Abbildung 3.40: Beispiel: list command options mit Standardwerten

Die wichtigsten Optionen sind:

- -c
Die Autocommit-Option ist standardmäßig aktiviert. Das heißt, jeder Befehl wird automatisch sofort festgeschrieben. Wollen Sie mehrere Befehle zu einer Transaktion zusammenfassen, müssen Sie diese Option explizit ausschalten.
- -f dateiname
Mit dieser Option können Scripts zur Abarbeitung übergeben werden.

► -l dateiname

Ein Befehlsprotokoll wird hier in eine Datei gespeichert. Neben dem Befehl wird der Rückgabewert (RETURN-Code) jeweils mit Datum und Zeitstempel festgehalten.

► -t

Der Strichpunkt wird als Befehlstrennzeichen verwendet.

► -td zeichen

Über -td können andere Befehls-Trennzeichen definiert werden. Das ist besonders dann interessant, wenn Sie andere Konstrukte, wie z.B. bei der SQL Procedural Language, einbinden, die ebenfalls einen Strichpunkt für die Befehlstrennung einsetzen.

► -r dateiname

Eine Reportdatei nur mit den *Ergebnissen* des Scripts wird gespeichert. Das wären z.B. ein erfolgreicher Verbindungsaufbau und die SQL-Abfrageergebnisse ohne die SQL- und DB2-Befehle.

► -s

Stoppt die Script-Ausführung, wenn ein Fehler auftritt. Weitere Befehle werden nicht mehr ausgeführt, um Folgefehler und Inkonsistenzen zu vermeiden.

► -v

Diese Option zeigt in der Ausgabe die abgesetzten Befehle mit an. Dies gilt sowohl für die Bildschirmausgabe als auch für die Speicherung in Dateien (-z-Option).

► -z dateiname

Speicherungsmöglichkeit der kompletten Ausgabe.

Bei den Optionen, die die Eingabe eines dateinamens erfordern, muss der Dateiname direkt hinter der jeweiligen Option folgen. Zwischen Option und Dateiname sind nur Leerzeichen erlaubt.

Ändern der DB2-Befehlsoptionen

Es gibt verschiedene Möglichkeiten, die Befehlsoptionen an Ihre Bedürfnisse anzupassen. Dabei spielt es einerseits eine Rolle, ob Sie den interaktiven oder nicht-interaktiven Modus verwenden, und andererseits, für wie lange diese Änderungen an den Befehlsoptionen wirksam sein sollen:

► Generell gültige Voreinstellungen erfolgen über die DB2-Registriervariable DB2OPTIONS, z.B.:

```
db2set DB2OPTIONS="-vt"
```

► für die Dauer einer Session (nur für den interaktiven Modus), also bis das Befehlszeilen-Fenster geschlossen wird:

```
update command options using <option> [on | off]
```


- für einen einzelnen Befehl:

```
db2 [-option | +option] <weiterer Befehl>
```

Optionen werden mit -option aktiviert und mit +option deaktiviert, also z.B.:

```
db2 -v "select * from kunde"
```

3.11.3 DB2-Scripts

Wie in der Einleitung zu diesem Unterkapitel erwähnt wurde, bietet die Befehlszeile die Möglichkeit, Scripts auszuführen. Ein Script besteht dabei aus einer Reihe einzelner DB2-Befehle oder auch SQL-Abfragen bzw. aus einer Mischung beider.

Aufbau von Scripts

Mit einem Texteditor können Sie sich Befehle zusammenstellen und in einer Datei (Script) abspeichern. Sie haben dabei die Wahl, pro Zeile einen Befehl (SQL-Abfrage oder DB2-Befehl) zu schreiben. Alternativ können Sie einen einzelnen Befehl für eine bessere Strukturierung und Übersichtlichkeit über mehrere Zeilen verteilen und auch gegebenenfalls entsprechend einrücken. Bei der zweiten Methode müssen Sie dann das Ende des tatsächlichen Befehls mit einem Befehlstrennzeichen, standardmäßig einem Strichpunkt, markieren.

```
-- Ein kleines Beispiel-Script
-----
-- Verbindungsaufbau mit der Datenbank
connect to buchdb;

-- SQL-Abfragen
select kundenr, kundenname, kundenrabatt
    from kunde
    where kundenrabatt > 10;

select count(*) from kunde where kundenrabatt > 15;

-- Verbindungsabbau mit der Datenbank
terminate;
```

Abbildung 3.41: Beispiel für ein DB2-Script

Abbildung 3.41 zeigt ein Beispiel für ein DB2-Script. Dabei markieren doppelte Minuszeichen einen Kommentar. Jeder Befehl ist mit einem Strichpunkt abgeschlossen. So konnte die erste SQL-Abfrage auf drei Zeilen verteilt werden. Dass das nicht unbedingt notwendig ist, zeigt der zweite SQL-Befehl.

Diese Datei können Sie mit beliebigem Namen und beliebiger Dateierweiterung in Ihrem System speichern.

Ausführung von Scripts

Zur Ausführung Ihrer Scripts benötigen Sie einige der Optionen, die wir in Abschnitt 3.11.2 „Optionen der DB2-Befehlszeile“ besprochen haben.

Um eine Scriptdatei an die Befehlszeile „übergeben“ zu können, benötigen wir zumindest die `-f`-Option. Da wir aber den Strichpunkt als Befehlstrennzeichen verwendet haben, wird zusätzlich die `-t`-Option notwendig. Im einfachsten Fall genügt also folgender Befehl zur Ausführung:

```
db2 -tf <scriptname>
```

Diesen Aufruf können Sie jetzt noch beliebig mit den anderen Optionen erweitern. Nützlich und häufiger verwendet ist noch folgender Aufruf:

```
db2 -stvf <scriptname>
```

Dabei wird also im Fehlerfall die Scriptausführung gestoppt, und durch `-v` werden auch die Befehle des Scripts in der Ausgabe mit angegeben.

3.12 Die Exemplar-Konfiguration

Wie schon erwähnt wurde, besitzt jedes Exemplar seine eigenen Konfigurationsparameter. Folgende Fragen können über die Parameter der Exemplar-Konfiguration (DBM CFG) beantwortet werden:

- ▶ Welche Kommunikationskonfiguration ist für das Exemplar gewählt?
- ▶ Welche Gruppe besitzt die Administrations- und Verwaltungsrechte für das Exemplar?
- ▶ Welche Monitorparametergruppen sind aktiv?
- ▶ Wie viele Datenbanken werden von dem Exemplar maximal unterstützt?
- ▶ Welchen Diagnoselevel besitzt das Exemplar?
- ▶ Welcher Standardpfad soll zum Anlegen von Datenbanken verwendet werden?

3.12.1 Abfrage der Exemplar-Konfiguration

Den Inhalt der Exemplar-Konfiguration, die oft auch als Datenbankmanager-Konfiguration (DBM CFG) bezeichnet wird, können Sie über folgenden Befehl abfragen:

```
get dbm cfg
```

Die Konfiguration wird wie in Abbildung 3.42 dargestellt ausgegeben, allerdings haben wir die Ausgabe stark gekürzt.

Bereits am Kapitelanfang hatten wir erwähnt, dass es oft sinnvoll ist, mehrere Exemplare auf einem System zu haben.

Welche Exemplar-Konfiguration ist das nun, und wieso muss man bei dem `get dbm cfg`-Befehl keinen Exemplarnamen angeben?

```
D:\SQLLIB\BIN>db2 get dbm cfg

Datenbankmanagerkonfiguration

Knotentyp                                = Enterprise Server
Edition mit lokalen und fernen Clients

Releasestand der Datenbankmanagerkonfiguration = 0x0a00

Max. Anzahl offener Dateien                (MAXTOTFILOP) = 16000
CPU-Geschwindigkeit (ms/Anweisung)        (CPUSPEED)    = 5,510680e-007
Kommunikationsbandbreite (MB/s)            (COMM_BANDWIDTH) = 1,250000e+000

Max. Anzahl gleichzeitig aktiver Datenbanken (NUMDB)    = 8
Data Links-Unterstützung                   (DATALINKS)    = NO
Unterst. f. System zusammengeschloss. Datenbanken (FEDERATED) = YES
...
```

Abbildung 3.42: Beispiel für eine Exemplar-Konfiguration

Der Exemplarname wird aus den Umgebungseinstellungen übernommen. Dort enthält die Variable DB2INSTANCE den aktuellen Exemplarnamen.

Dieser wird in der Systemumgebung von Windows bzw. durch das db2profile in Unix-Umgebungen (vgl. Abschnitt 3.10 „Benutzer für DB2 im Unix-Umfeld einrichten“) gesetzt.

3.12.2 Ändern der Exemplar-Konfiguration

Die Exemplar-Konfiguration kann nur vom SYSADM, also dem Exemplar-Eigner bzw. einem Mitglied der Exemplar-Administratorengruppe, geändert werden.

Der zugehörige Befehl hat folgende Struktur:

```
update dbm cfg using <parameter> <neuer wert> [immediate | deferred]
```

Im Feld <parameter> geben Sie dabei den Parameternamen an, der bei der Auflistung der Exemplar-Konfiguration in Klammern angezeigt wurde (siehe Abbildung 3.42).

Einige dieser Parameteränderungen werden sofort – im laufenden Betrieb – wirksam, andere erst nach einem Neustart des Exemplars. Mit der Option deferred kann eine Einstellung aber bis zum nächsten Start des Exemplars hinausgezögert werden, um eventuell den Aufwand, zusätzlichen Speicher zu reservieren, auf einen späteren Zeitpunkt zu verschieben.

Über die um den Zusatz SHOW DETAIL erweiterte Abfrage der Exemplar-Konfiguration sehen Sie die aktuell aktiven Einstellungen und Ihre Änderungen, die beim nächsten Start des Exemplars wirksam werden:

```
attach to <exemplarname>
```

```
get dbm cfg show detail
```

Der erste Befehl stellt dabei eine explizite Verbindung zum Exemplar her und ist Voraussetzung für diese Variante der Abfrage der Exemplar-Konfiguration.

Das Ergebnis wird dann mehrspaltig ausgegeben und ist hier übersichtshalber umformatiert dargestellt.

D:\SQLLIB\BIN>db2 get dbm cfg show detail

Datenbankmanagerkonfiguration			
Knotentyp		= Datenbank-Server mit lokalen und fernen Clients	
Beschreibung	Parameter	Aktueller Wert	Verzögerter Wert

Releasestand der Datenbankmanagerkonfiguration		= 0x0a00	
Max. Anzahl offener Dateien	(MAXTOTFILOP)	= 16000	16000
CPU-Geschwindigkeit (ms/Anweisung)	(CPUSPEED)	= 5,668131e-007	5,668131e-007
Max. Anzahl gleichzeitig aktiver Datenbanken	(NUMDB)	= 8	8
Data Links-Unterstützung	(DATALINKS)	= NO	NO
Unterstüt. f. System zusammengeschloss. Datenbanken	(FEDERATED)	= YES	YES
Name des TP-Monitors	(TP_MON_NAME)	=	

Standardkonto für Rückerstattung	(DFT_ACCOUNT_STR)	=	
Java Development Kit: Installationspfad	(JDK_PATH)	= D:\SQLLIB\java\jdk	D:\SQLLIB\java\jdk

Max. Anzahl von Agenten	(MAXAGENTS)	= 200	250
Größe des Agentenpools	(NUM_POOLAGENTS)	= 100	125(berechnet)
Ursprüngliche Anzahl Agenten im Pool	(NUM_INITAGENTS)	= 0	0
Max. Anz. koordinierender Agenten	(MAX_COORDAGENTS)	= 200	MAXAGENTS
Max. Anz. gleichzeitig koordinierender Agenten	(MAXCAGENTS)	= 200	MAX_COORDAGENTS
Max. Anzahl von Clientverbindungen	(MAX_CONNECTIONS)	= 200	MAX_COORDAGENTS
...			

Abbildung 3.43: Beispiel einer Exemplar-Konfiguration mit SHOW DETAIL-Option

Dabei sehen Sie ebenfalls in Abbildung 3.43, dass manche Parameterwerte durch Formeln berechnet sein können oder standardmäßig direkt von anderen Parametern abhängen können.

3.13 DAS und DB2-Toolskatalog manuell erstellen

Der Verwaltungsserver (DAS) und der Toolskatalog können, wenn es nicht schon bei der Installation geschehen ist, im Nachhinein angelegt werden.

3.13.1 Erstellen eines DB2-Verwaltungsservers (DAS)

Auf einem Server kann nur ein DAS aktiv sein. Die Vorgehensweise bei der nachträglichen Erstellung ist hierbei vom jeweiligen Betriebssystem abhängig.

In einem Windows-System sieht dies wie folgt aus:

```
db2admin create /user <benutzer-id> /password <passwort>
```

Bei dieser Eingabe sind die benutzer-id und das passwort optional. Diese zusätzlichen Optionen benötigen Sie nur, wenn der Verwaltungsserver mit einer ganz bestimmten Benutzer-ID laufen soll. Als Standard wird die Benutzer-ID verwendet, mit der Sie am System angemeldet sind. Dieser Benutzer sollte lokale Administrationsrechte haben.

Nach der Erstellung des DAS können Sie das Eigentumsrecht für den DAS durch die Angabe eines Benutzerkontos und eines Kennworts im Befehl

```
db2admin stop
db2admin setid <benutzer-id> <passwort>
db2admin start
```

festlegen bzw. ändern.

Über den Befehl `db2admin` können Sie sich den Namen des DAS anzeigen lassen.

Auf Unix-Systemen sollte Folgendes berücksichtigt werden:

- ▶ Stellen Sie sicher, dass Sie über die root-Berechtigung verfügen bzw. als root-Benutzer angemeldet sind.
- ▶ Setzen Sie aus dem DB2-Verzeichnis in

AIX: `/usr/opt/db2_08_01/instance/`

Linux: `/opt/IBM/db2/V8.1/instance/`

folgenden Befehl ab:

```
./dascrt -u <DAS-Benutzer>
```

Hierbei ist darauf zu achten, dass der <DAS-Benutzer> vorher als eigenständiger Benutzer im Unix-System angelegt wurde. Dieser benötigt keine „logon“-Berechtigung.

Über den Befehl `./daslist` können Sie prüfen, ob schon ein DAS im System registriert ist.

3.13.2 Erstellen eines DB2-Toolskatalogs

Die Toolskatalog-Datenbank (Tools Catalog) enthält alle Taskinformationen, die von der Taskzentrale sowie von der Steuerzentrale erstellt werden. Diese Tasks werden von der Zeitplanfunktion (scheduler) des DB2-Verwaltungsservers (DAS) gesteuert und ausgeführt.

Damit dies auch alles funktioniert, müssen der DB2-Verwaltungsserver und die Toolskatalogdatenbank eingerichtet sein.

Bevor Sie den Toolskatalog anlegen, sollten Sie folgende Fragen klären:

- ▶ In welchem Exemplar bzw. in welcher lokalen Datenbank soll der Toolskatalog angelegt werden?
- ▶ Soll der Toolskatalog in einer eigenständigen Datenbank erstellt werden?

Diese Option sollte aus administrativen Gründen bevorzugt werden.

Wollen Sie z.B. den Toolskatalog in einer eigenen Datenbank haben, was wir empfehlen, so ist der folgende Befehl abzusetzen:

```
CREATE TOOLS CATALOG <schema-name> CREATE NEW DATABASE <toolsdb-name> FORCE
```

Wichtig:

Die Option `FORCE` führt zum Stoppen und zum Neustart des Exemplars. Dazu müssen alle Anwendungen beendet werden. Gleichzeitig werden auch die Konfigurationswerte des DB2-Verwaltungsservers (DAS) angepasst und die Zeitplanfunktion (scheduler) wird aktiviert.

Für den Fall, dass hier einmal etwas nicht klappt, zeigen wir Ihnen, wie die Konfiguration im DAS angepasst werden kann.

Mit dem Befehl: `get admin cfg` erhalten Sie alle Konfigurationswerte des DAS. Wir führen hier nur die wichtigsten auf:

► `sched_enable`

gibt an, ob die Zeitplanfunktion (scheduler) vom DB2-Verwaltungsserver (DAS) gestartet wird oder nicht. Dieser Parameter wird automatisch auf ON gesetzt, wenn der Toolskatalog angelegt wird.

► `toolscat_inst`

Name des Exemplars, in dem die Toolskatalog-Datenbank angelegt wurde.

► `toolscat_db`

► Datenbankname, in dem der Toolskatalog angelegt wurde bzw. verwendet werden soll.

► `toolscat_schema`

Schema-Name der Toolskatalog-Tabellen. Wichtig ist dies, wenn die Toolskatalog-Tabellen nicht in einer separaten Datenbank angelegt wurden, sondern in einer bestehenden Datenbank. DB2 muss dann über den Schema-Namen eindeutig die Toolskatalog-Tabellen identifizieren können.

Die bis hierher aufgezählten Werte können über den folgenden Befehl verändert werden:

```
update adm cfg using <parameter> <wert>
```

Für die Definition des Schema-Namens `toolsdb` für den Toolskatalog wäre das:

```
update adm cfg using toolscat_schema toolsdb
```

Der folgende Wert ist zwar auch in der DAS-Konfiguration aufgelistet, muss aber über den `db2admin`-Befehl angepasst werden:

► Benutzerid für Scheduler

Benutzer-ID, die die Zeitplanfunktion (scheduler) verwendet, um eine Verbindung zur Toolskatalog-Datenbank herzustellen. Dieser Parameter ist nur zu setzen, wenn der Toolskatalog auf einem fernen Datenbankserver verwendet werden soll.

Um diesen Wert neu zu setzen oder zu verändern, müssen Sie folgenden Befehl absetzen:

```
db2admin SETSCHEDID sched-user/sched-passwort
```

Im Unix-Umfeld suchen Sie den `db2admin`-Befehl unter der DAS-Benutzer-ID im Verzeichnis `$HOME\das\bin\` und führen ihn aus.

3.14 DB2-Umgebungsvariablen anpassen

Umgebungsvariablen werden benötigt, um Ihre DB2-Systemumgebung anzupassen. Hierunter fallen zum Beispiel Aufgaben wie:

- ▶ Setzen des zu verwendenden Exemplarnamens für die DB2-Befehlszeile bzw. die Shell
- ▶ Bekanntmachung, welches Kommunikationsprotokoll für das jeweilige Exemplar unterstützt werden soll

Die DB2 UDB unterscheidet zwei prinzipielle Typen von Umgebungsvariablen:

- ▶ Umgebungsvariablen, die nur auf der Betriebssystemebene gesetzt werden können. Hierzu gehört für alle Betriebssysteme die Umgebungsvariable `DB2INSTANCE`. Sie bestimmt durch ihren Wert, auf welches Exemplar sich die jeweilige Betriebssystem-Session (Shell bzw. DB2-Befehlszeile) bezieht. Weitere Umgebungsvariablen auf dieser Ebene sind:
 - `DB2TEMPDIR` Temporäres Verzeichnis des DB2-Exemplars
 - `DB2PATH` DB2-Installationsverzeichnis
 - `DB2CLP` Spezielle Variable nur für Windows

Ein Beispiel zum Setzen der Betriebssystemvariable ist:

```
set DB2INSTANCE = db2ebuch
```

Bitte vergessen Sie nicht, wenn Sie sich in der Unix-Welt befinden, die Variable zu exportieren. Weitere Möglichkeiten, die DB2-Umgebungsvariablen auf Betriebssystemebene zu setzen, haben Sie unter:

- Windows
über die Umgebungsvariablen in Ihren Windows-Systemeigenschaften
- Unix
entweder im Benutzer-Profil (z.B. `.profile`) des jeweiligen Benutzers oder im `db2profile` im Verzeichnis `/sqllib/` des jeweiligen Exemplar-Eigners.
- ▶ Umgebungsvariablen, die auf DB2-Ebene gesetzt werden.

Man spricht bei diesen Umgebungsvariablen auch von DB2-Registriervariablen bzw. von der DB2-Profilregistrierdatenbank (DB2-Registry) und unterscheidet dabei folgende vier Ebenen:

- DB2-Profilregistrierdatenbank auf Exemplarebene
Die Profilvariablen sind für ein bestimmtes Exemplar gültig. Der Befehl zum Setzen der DB2-Kommunikationsumgebungsvariablen lautet zum Beispiel:
 - `db2set db2comm=tcPIP`
 - `db2set db2comm=tcPIP -i db2ebuch`

Beide Befehle bewirken im Prinzip dasselbe, wenn die Umgebungsvariable `DB2INSTANCE` den Wert `db2ebuch` enthält. Der Parameter `-i` bewirkt, dass die Umgebungsvariable für das Exemplar `db2ebuch` gesetzt wird. Diese Option wird dann notwendig, wenn Sie sich gerade in einer Betriebssystem-Session

befinden, in der die Umgebungsvariable DB2INSTANCE einen anderen Wert als db2ebuch enthält.

- Globale DB2-Profilregistrierdatenbank

Die Registriervariablen (DB2-Umgebungsvariablen) besitzen für alle Exemplare des Systems Gültigkeit. Nachfolgendes Beispiel zeigt Ihnen, wie der Wert der Umgebungsvariable DB2COMM entsprechend gesetzt wird:

```
db2set db2comm=tcPIP -g
```

Die Option `-g` sorgt dafür, dass sich die Gültigkeit auf alle Exemplare erstreckt.

- DB2-Profilregistrierdatenbank auf Exemplarknotenebene

Die Umgebungsvariablen sind, innerhalb einer partitionierten Datenbankumgebung für einen Konten bzw. eine Partition (node) gültig.

Wie Sie z.B. die Umgebungsvariable DB2COMM für eine bestimmte Datenbankpartition setzen, zeigt Ihnen nachstehendes Beispiel:

```
db2set db2comm=tcPIP -i db2ebuch 3
```

Die Option `-i` bewirkt in diesem Fall, dass die Umgebungsvariable nur für die Partition 3 des Exemplars db2ebuch gültig ist.

- DB2-Exemplarliste

Dieses Register enthält eine Liste aller auf dem System verfügbaren Exemplarnamen.

```
db2ilist
```

Dieser Befehl listet Ihnen die Namen aller Exemplare Ihres Systems auf.

Tipp:

Durch `db2ilist -a` sehen Sie für alle Exemplare auf Ihrem System zusätzlich die Exemplarversion und die Bitbreite sowie den zugehörigen Installationspfad:

```
db2ebuch    32    /opt/IBM/db2/V8.1
inst00      32    /opt/IBM/db2/V8.FP3
```

Der Befehl db2set allgemein

Abbildung 3.44 zeigt Ihnen die komplette Befehlssyntax des Befehls DB2SET. Die wichtigsten Befehle, die Sie sich, außer den schon zuvor erwähnten, noch merken sollten, sind:

- `db2set -lr`

Listet Ihnen alle zur Verfügung stehenden Umgebungsvariablen auf.

- `db2set -?`

Gibt Ihnen die Hilfe für den Befehl aus.

► db2set -all

Listet Ihnen sowohl die DB2-Umgebungsvariablen für das in dieser Session gültige Exemplar als auch die globalen Umgebungsvariablen, die für alle Exemplare gültig sind, auf.

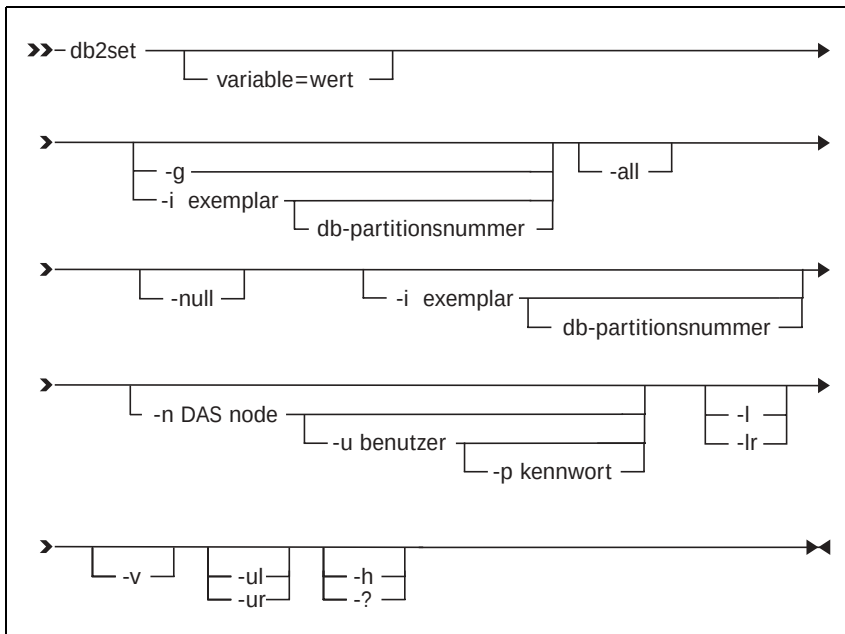


Abbildung 3.44: db2set-Syntaxdiagramm

Tipp:

Um eine gesetzte Variable zu entfernen, verwenden Sie folgenden Befehl:
`db2set <variable>=`

Auflösung der DB2-Umgebungsvariablen

Der Auflösung der DB2-Umgebungsvariablen wird in folgender Reihenfolge durchgeführt:

1. Umgebungsvariablen auf Betriebssystemebene (set / export)
2. Umgebungsvariablen einer bestimmten Partition (`db2set ... -i <exemplarname> <partitionsnummer>`)
3. Umgebungsvariablen eines bestimmten Exemplars (`db2set ... -i <exemplarname>`)
4. für alle im Exemplar gültigen Umgebungsvariablen (`db2set ... -g`)

Tipp:

Wenn Sie verhindern wollen, dass eine DB2-Registriervariable niedriger Priorität dadurch Gültigkeit erlangt, weil sie zuvor nicht gesetzt wurde, können Sie eine Registriervariable explizit auf NULL setzen. Wollen Sie z.B. verhindern, dass ein bestimmtes Exemplar Kommunikationsprotokolle unterstützt, obwohl DB2COMM=TCPIP auf globaler Ebene gesetzt ist, setzen Sie für dieses Exemplar die Variable DB2COMM explizit auf NULL:

```
db2set db2comm=NULL -i <exemplarname>
```

3.15 Tipps aus der Praxis

Die Migration ist eine besondere Art der Installation, da in diesem Fall DB2 schon auf der Maschine installiert ist. Wir haben diesem Thema einen extra Abschnitt im Anhang gegönnt. Dort finden Sie auch die Vorgehensweise mit Tipps und Hinweise, was Sie nach der Installation von Fixpacks beachten sollten.

Zwei weitere nützliche Tools und Techniken sind DB2LEVEL und die Multifixpack-Installation.

3.15.1 DB2LEVEL

Wenn Sie den Befehl `db2level` aus der Befehlszeile ausführen, zeigt er Ihnen schnell die installierte DB2-Version und das verwendete Fixpack an. Einfach, aber sehr nützlich!

3.15.2 Multifixpack-Installation

Mit einer Multifixpack-Installation können Sie mehrere DB2-Installationen, nicht nur mit unterschiedlichen Versionen, sondern sogar mit verschiedenen Fixpack-Stufen (Modifikationsstufen) parallel nebeneinander auf einer Maschine betreiben.

Es gibt jedoch zwei Voraussetzungen, um diese Möglichkeit überhaupt nutzen zu können:

- ▶ Die Multifixpack-Installation ist *nur in Unix-Umgebungen* möglich.
- ▶ Sie wird *nur im Produkt DB2 UDB ESE (Enterprise Server Edition)* unterstützt.

Erfüllen Sie diese Voraussetzungen, können Sie mit DB2 UDB V8 nicht nur mehrere Datenbankversionen parallel laufen lassen, wie es schon immer im Unix-Umfeld und unabhängig vom DB2-Produkt möglich war, sondern eben auch mehrere Fixpack-Stufen.

Weitere wichtige Punkte, die in Abbildung 3.45 grafisch dargestellt wurden, sind dabei:

- ▶ Es gibt zwei Arten von Fixpacks.
 - Einerseits die „normalen“ Fixpacks, um eine bestehende Installation auf die neuste Modifikationsstufe zu bringen, und andererseits die so genannten

„alternativen“ Fixpacks, die zur Installation des kompletten Produkts in einem alternativen Pfad führen. Die alternativen Fixpacks sind für den Aufbau einer Multifixpack-Umgebung gedacht.

- Eine solche alternative Installation kann nicht durch einen normalen Fixpack modifiziert werden.
- Die Installation über eine Response-Datei (Antwortdatei) wird nicht unterstützt.
- Die Exemplare können durch den db2iupdt-Befehl auf die entsprechende Modifikationsstufe versetzt werden. Es erfolgt dabei ein „Umhängen“ der Verweise (Links) auf die jeweiligen neuen Produktpfade.

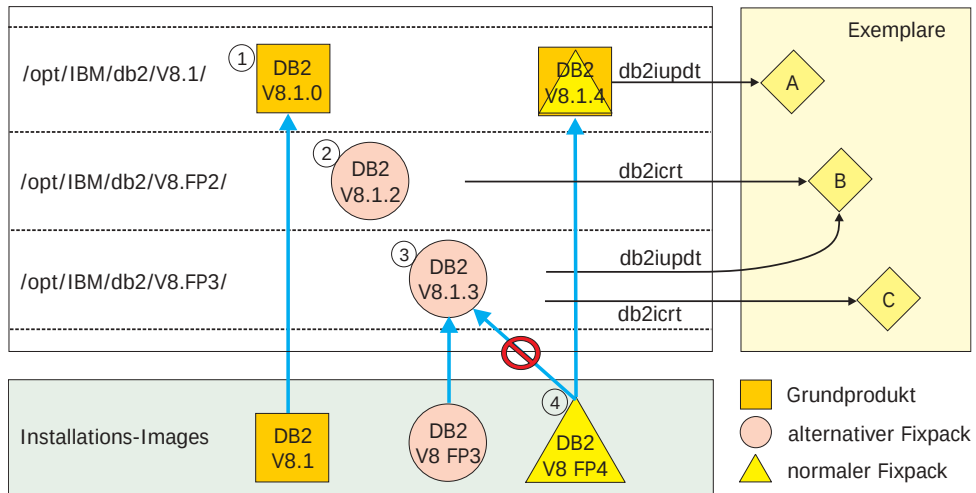


Abbildung 3.45: Multifixpack-Installation

In Abbildung 3.45 wurde bei Punkt ① eine DB2 UDB V8 über die original Produkt-CD-ROM neu installiert. Die Installation des Grundprodukts erfolgt bei Linux in das Verzeichnis /opt/IBM/db2/V8.1/, in AIX nach /usr/opt/db2_08_01/. Nach dieser Installation wurde das Exemplar A erstellt.

Zu einem späteren Zeitpunkt wurde die Installation der alternativen Fixpacks 2 und 3 durchgeführt, deren Zielverzeichnis die Struktur /opt/IBM/db2/V8.FPn/ hat, wobei n die entsprechende Fixpackstufe ist. In AIX hätte das Zielverzeichnis die Struktur /usr/opt/db2_08_FPn/.

Das Exemplar B wurde durch einen db2icrt (siehe Punkt ② aus dem alternativen Installationsverzeichnis (/opt/IBM/db2/V8.FP2/instance/)) erstellt und hat somit die Modifikationsstufe V8.1.2.

Das Gleiche geschah entsprechend mit Fixpack 3 (Punkt ③). Somit hat das Exemplar C die Modifikationsstufe V8.1.3, da es durch den db2icrt aus dem Pfad der Modifikationsstufe 3 erstellt wurde. Zusätzlich wurde durch einen db2iupdt aus demselben alternativen Installationspfad das Exemplar B von der Version 8.1.2 auf die Stufe V8.1.3 gehoben. Der Programmpfad des Exemplars B zeigt nun auf das Installationsverzeichnis /opt/IBM/db2/V8.FP3/.

Hinweis:

Der normale Fixpack bei Punkt ④ kann nicht auf eine alternative Installation angewandt werden! Er ist nur für die Modifikation des Grundprodukts im Standard-Installationspfad gedacht. Auch nach der Installation des Fixpack 4 ist ein explizites db2iupdt auf das bestehende Exemplar A notwendig, um diese neue Modifikationsstufe für dieses Exemplar zu aktivieren.

Vorgehensweise bei der Multifixpack-Installation

1. Lesen Sie zuerst die Readme-Datei des entsprechenden Fixpacks.

Sie finden diese auf dem FTP-Server, von dem Sie auch den alternativen Fixpack herunterladen können:

<http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winoss2unix/support/download.d2w/report>

Beachten Sie dabei im Besonderen spezielle fixpack- und betriebssystembezogene Hinweise!

Hinweis:

Die README-Datei enthält z.B. den Hinweis, dass Sie, wenn Sie DB2 V8 Fixpack 3 unter Linux installiert haben, es nicht mehr deinstallieren und zu der vorhergehenden DB2-Version zurückkehren können!

2. Stellen Sie sicher, dass Sie über die root-Berechtigung verfügen.
3. Stoppen Sie alle Exemplare (als SYSADM des jeweiligen Exemplars) und den Lizenzserver:

```
su - <exemplarbenutzer>  
db2 force applications all  
db2 terminate  
db2stop  
db2licd -end
```
4. Stellen Sie sicher, dass der Fehlermonitor für das jeweilige Exemplar deaktiviert ist:

```
db2fm -i <exemplarname> -f off
```
5. Stoppen Sie den DAS:

```
su - <dasbenutzer>  
db2admin stop  
exit
```
6. Installieren Sie den Fixpack durch Aufruf von

```
./installAltFixPak
```

aus dem Pfad des alternativen Fixpacks.

Damit ist die Installation abgeschlossen.

Sie können nun neue Exemplare mit der neuen Modifikationsstufe anlegen (db2icrt) oder bestehende Exemplare mit db2iupdt auf die neue Modifikationsstufe heben.

Dabei ist es sehr wichtig, dass Sie die Befehle db2iupdt bzw. db2icrt aus dem instance-Unterverzeichnis des Installationsverzeichnisses des alternativen Fixpacks aufrufen, um Exemplare der neuen Modifikationsstufe zu bekommen. Die Aktualisierung des DAS ist durch den Befehl dasupdt möglich.

Diese Installationspfade sind:

AIX: /usr/opt/db2_08_FPn

Unix: /opt/IBM/db2/V8.FPn

n ist dabei die Modifikationsnummer des Fixpacks.

Weitere Schritte nach der Installation sollten sein:

1. Neustart der Exemplare und des DAS
2. Erneutes Binden der Datenbankprogramme durch:

```
CONNECT TO <datenbankname>
BIND <pfad>/bnd/@db2ubind.lst BLOCKING ALL GRANT PUBLIC
BIND <pfad>/bnd/@db2cli.lst BLOCKING ALL GRANT PUBLIC
TERMINATE
```

3. Binden der eigenen Datenbankprogramme (Details siehe Kapitel 11).

4 Grafische Werkzeuge

DB2 bietet eine Reihe leistungsstarker grafischer Tools. Während diese in der Windows-Welt gern verwendet werden, spürt man immer wieder eine gewisse Abneigung aus Richtung der Unix-Gemeinde. Unix-Administratoren lieben typischerweise Befehlszeilenkommandos, da sich diese einfach zu einem Script zusammenbauen lassen.

Wir sind jedoch der Meinung, dass man beide Welten – also die grafische Oberfläche und die Befehlszeile – kennen sollte, um sich dann kompetent einmal für das GUI oder in einem anderen Fall für die Befehlszeile entscheiden zu können.

Obwohl wir als Dozenten die meisten Befehle auswendig kennen und auch die Vorteile von Scripts schätzen, kann die grafische Oberfläche doch oft sehr nützlich sein und einem Administrator das Leben einfacher machen.

Die Vorteile der grafischen Oberfläche sind z.B.:

- ▶ Nach kurzer Einarbeitung ist sie einfach zu bedienen.
- ▶ Sie müssen sich keine Befehlssyntax merken, sondern können diese über die BEFEHL ANZEIGEN-Buttons sogar lernen.
- ▶ So genannte Popup-Fenster bieten eine schnelle Hilfe bei der Orientierung.
- ▶ Die Übersichtlichkeit, z.B. durch die Steuerzentrale, ist gut.

4.1 Überblick

Der Aufruf der grafischen Oberfläche erfolgt auf einem Windows-Rechner über die Startleiste mit START/PROGRAMME/IBM DB2. Auf Linux-Systemen werden Ihnen nach der Installation ein Fenster und ein Starteintrag – IBM DB2 – auf Ihrem (KDE-) Desktop angezeigt, in dem folgende GUI-Tools zu finden sind:

- ▶ Information mit Release Notes
- ▶ Befehlszeilenprozessor
- ▶ Konfigurationsassistent
- ▶ Steuerzentrale
- ▶ Erste Schritte

Die anderen grafischen Tools können Sie unter Unix über die Steuerzentrale (siehe Abbildung 4.2) oder direkt von der Befehlszeile aus aufrufen (siehe Abbildung 4.1).

Abbildung 4.1 zeigt die Einträge im IBM DB2-Ordner im Windows-Startmenü. In Klammern ist aufgeführt, wie Sie die einzelnen Tools per Befehl über die Befehlszeile starten können.

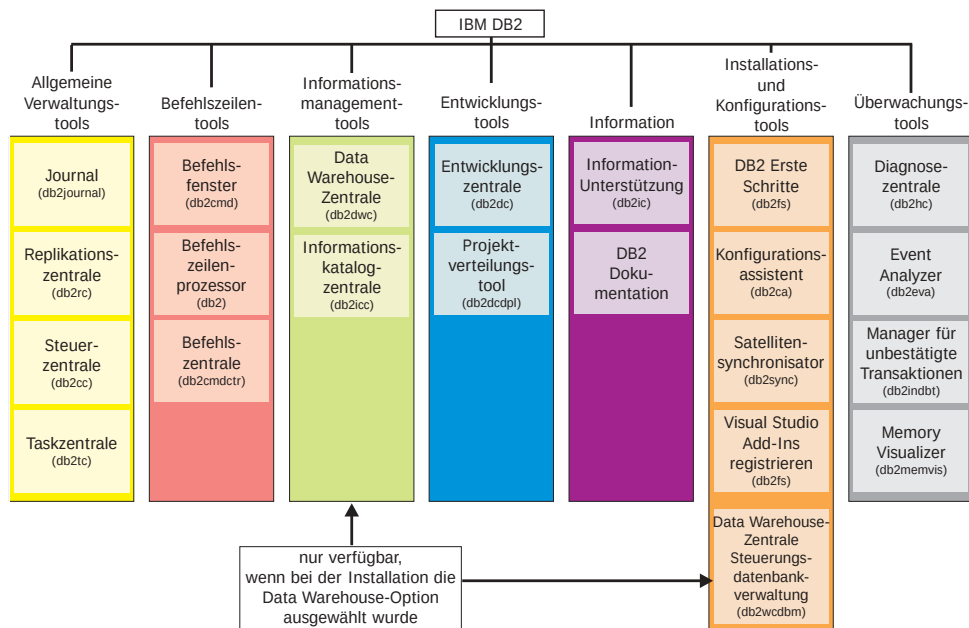


Abbildung 4.1: GUI-Aufruf aus der Windows-Startleiste

Sie haben die Möglichkeit, durch die Funktionsleiste innerhalb der grafischen Tools schnell zu den anderen „Zentralen“ (also GUI-Tools) zu wechseln. Dafür wählen Sie einfach den entsprechenden Tool-Button unterhalb des Menüs aus der Funktionsleiste aus.

Eine Übersicht, welches Icon welches grafische Tool startet, sehen Sie in Abbildung 4.2.

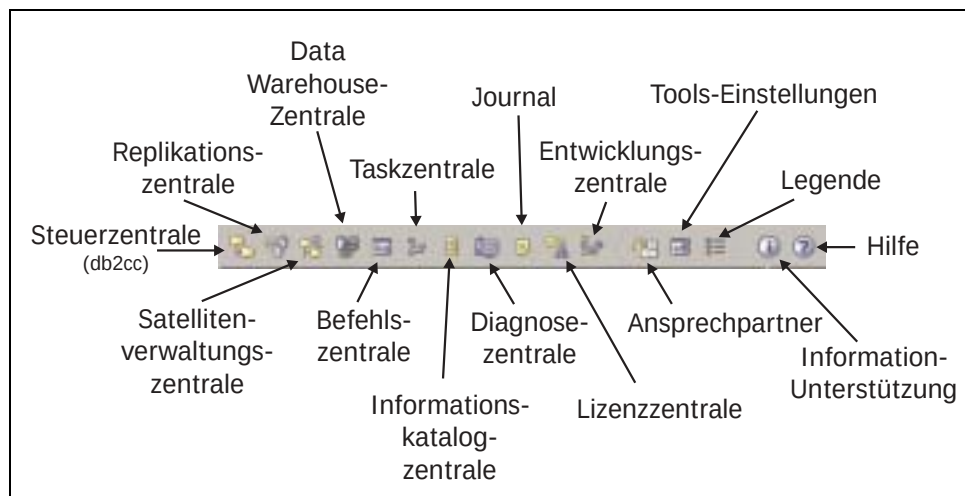


Abbildung 4.2: Funktionsleiste mit den Schnellstart-Icons der Steuerzentrale

Im Folgenden stellen wir Ihnen die einzelnen grafischen Schnittstellen vor und zeigen ihre Einsatzgebiete auf. Dabei liegt der Schwerpunkt auf den zentralen Tools zur Administration eines DB2-Servers. Die Oberflächen für Zusatzsoftware bzw. fortgeschrittene Einsatzgebiete, wie Data Warehousing oder Satellitenverwaltung, werden auch vorgestellt, sind aber im weiteren Verlauf des Buchs nicht mehr Thema.

Weitere Informationen zu den grafischen Oberflächen von DB2 finden Sie auch in [GUITOOLS].

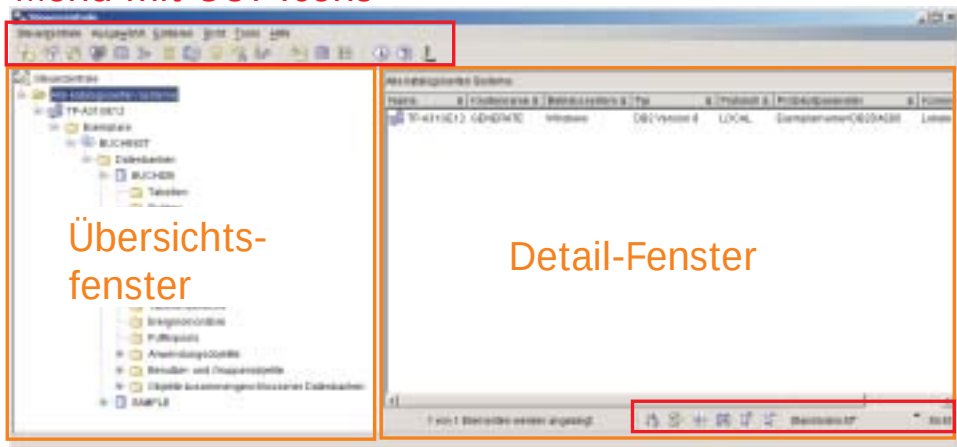
4.2 Steuerzentrale

Die Steuerzentrale ist „das“ GUI für die DB2-Administration. Mit ihr können Sie u.a. folgende Aufgaben durchführen:

- ▶ Hinzufügen neuer Systeme, Exemplare und Datenbanken
- ▶ Konfiguration der Exemplare und Datenbanken – lokal und entfernt
- ▶ Anlegen, Anpassen und Löschen verschiedener Datenbankobjekte
- ▶ Pflege und Wartung des Systems (z.B. Reorganisation)
- ▶ Sicherung und Wiederherstellung
- ▶ Überwachung der Systemkomponenten

Vor allem bekommen Sie durch die Steuerzentrale einen sehr guten Überblick über die verfügbaren Objekte und Strukturen. Der so genannte Objektbaum – den Sie im Übersichtsfenster der Steuerzentrale sehen – zeigt Ihnen die Hierarchie und Zusammengehörigkeit der Objekte an.

Menü mit GUI-Icons



Filter- und Sicht-Einstellungen

Abbildung 4.3: Die Steuerzentrale

Dabei befinden sich die Datenbanken im Objektbaum unterhalb des Exemplars, das wiederum – neben anderen Exemplaren desselben Systems – unter einem System (Maschine) angeordnet ist.

Die Verwaltung von lokalen Datenbank-Servern unterscheidet sich nicht von der Verwaltung entfernter Datenbank-Server. Sie haben also keinerlei Einschränkungen in der Administration entfernter Systeme.

Das Detail-Fenster zeigt Ihnen die enthaltenen Strukturen und Objekte – je nach der im Übersichtsfenster getroffenen Auswahl. Wenn Sie beispielsweise den Eintrag TABELLEN unterhalb einer Datenbank auswählen, werden alle in der Datenbank verfügbaren Tabellen im Detail-Fenster angezeigt.

Da es sich dabei um sehr viele Objekte handeln kann, können Sie über die Filter- und Sicht-Einstellungen der Detail-Fenster-Funktionsleiste die Anzeige anpassen. Dabei haben Sie die Möglichkeit, die Auswahl

- zu sortieren



Abbildung 4.4: Sortierung angeben

- zu filtern

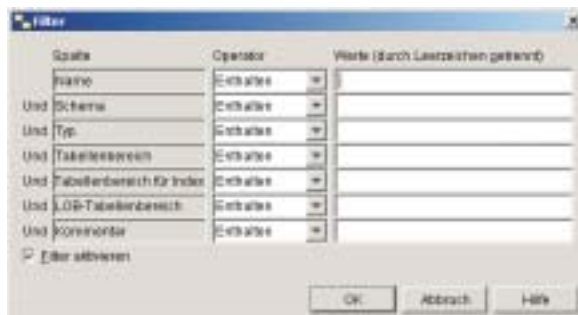


Abbildung 4.5: Filterung vornehmen

- die angezeigten Spalten zu bestimmen
- zu suchen

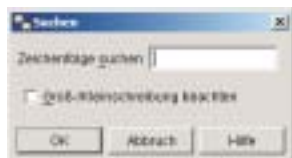


Abbildung 4.6: Suchen-Fenster, um in der Auswahl zu suchen

- ▶ zu selektieren bzw. die Selektierung rückgängig zu machen
- ▶ die Sicht auszuwählen und anzupassen.

Der letzte Punkt ist interessant, da Sie sich eigene „Sichten“ erstellen können. Dabei können Sie eine individuelle Filterung, Sortierung und Anzeige von Spalten und Spaltenbreiten unter einem bestimmten Namen abspeichern und haben dadurch immer die Möglichkeit, auf diese Anzeige-Sicht zu wechseln.

Dazu passen Sie Ihre Spaltenauswahl, Spaltenbreite, Sortierung und eventuell Filterung so an, wie Sie es gerne hätten, und wählen dann über den SICHT-Button der Funktionsleiste des Detail-Fensters SICHERN UNTER... aus (siehe Abbildung 4.7). Den vergebenen Namen für die Sichten können Sie dann bei der Sicht-Auswahl einstellen bzw. dieser wird gleich als neuer Standard übernommen. Die Sicht wird beim nächsten Mal genau so angezeigt, wie Sie es vor dem Schließen der Steuerzentrale eingestellt hatten. Das heißt, der letzte Stand wird automatisch gespeichert.

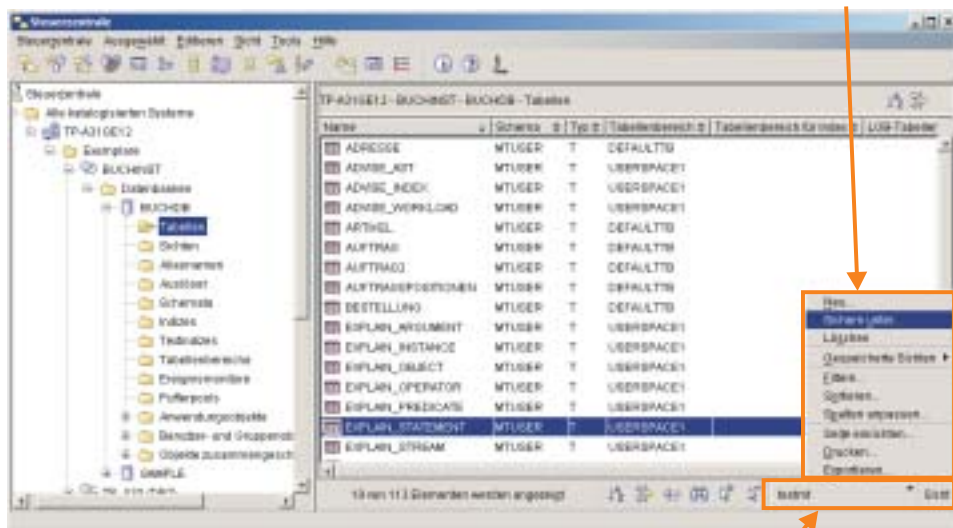


Abbildung 4.7: Sicht-Auswahl und Sicht-Menü

Sie können die Assistenten entweder über das jeweilige Kontextmenü starten, oder Sie wählen den entsprechenden Eintrag aus dem Assistenten-Fenster (siehe Abbildung 4.9) aus, das Sie über den Menüeintrag TOOLS/ASSISTENTEN... erreichen.

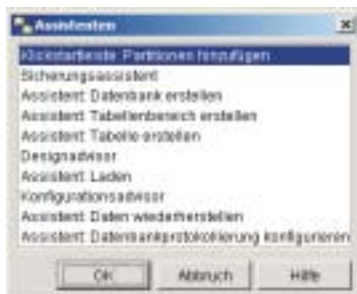


Abbildung 4.9: Assistenten-Fenster

In diesem Fenster finden Sie alle Assistenten, die Ihnen zur Verfügung stehen.

Bei der Auswahl eines Assistenten erscheint ein neues Fenster mit Steuerbuttons am unteren Fensterrand und mit einer Übersicht der Schritte im linken Fensterteil (siehe Abbildung 4.10).

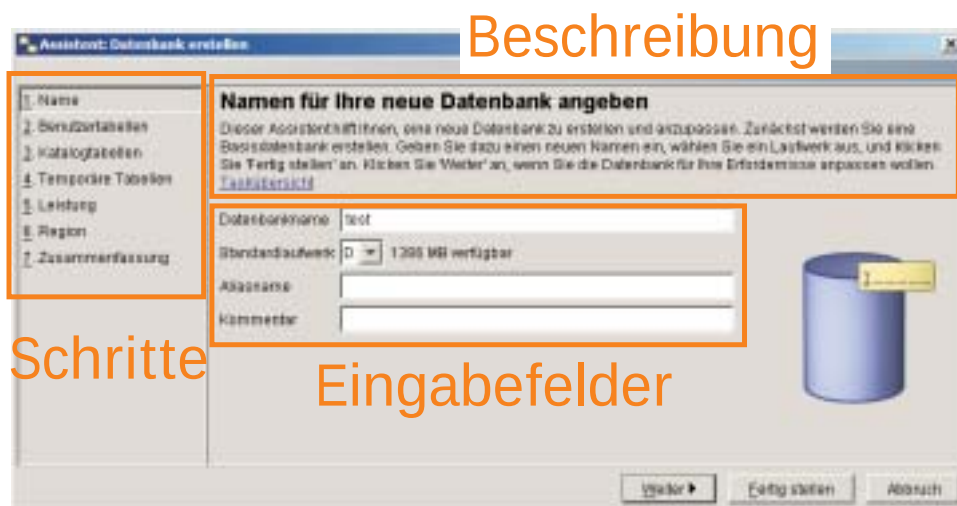


Abbildung 4.10: Bestandteile eines Assistenten-Fensters

Nachdem Sie die Hilfestellung in der Beschreibung gelesen haben, können Sie die nötigen Eingaben vornehmen.

Sobald die minimal erforderlichen Eingaben für die gestellte Aufgabe – in unserem Beispiel die Erstellung einer Datenbank – getätigt sind, wird der FERTIG STELLEN-Button auswählbar.

Der letzte Schritt zeigt immer eine Zusammenfassung der gewählten Einstellungen an und ermöglicht es auch, den generierten DB2-Befehl bzw. die DB2-Befehle anzeigen zu lassen. Dieser Punkt ist hilfreich bei der Einarbeitung und beim Erlernen der DB2-Befehle.

Wir besprechen die Assistenten jeweils bei den zugehörigen Themen in den entsprechenden Kapiteln. Den Assistenten *Datenbank erstellen* finden Sie z.B. in Kapitel 5.3.2, „Wie erstelle ich eine Datenbank im GUI?“

4.2.3 Ansprechpartner

Das Ansprechpartner-Fenster (siehe Abbildung 4.11) erreichen Sie über die Funktionsleiste der Steuerzentrale (vgl. Abbildung 4.2).

Als Ansprechpartner können verschiedene Personen und Personengruppen erfasst werden, die in Ausnahmesituationen, z.B. bei fehlgeschlagenen Aufgaben in der Datenbank, benachrichtigt werden können.



Abbildung 4.11: Ansprechpartner-Fenster

Dazu werden die Ansprechpartner inklusive ihrer E-Mail-Adresse über dieses Fenster erfasst und verwaltet.

4.2.4 Tools-Einstellungen

Die Tools-Einstellungen sind ebenso wie die Ansprechpartner über die Funktionsleiste der grafischen Tools, z.B. der Steuerzentrale zu erreichen (siehe Abbildung 4.2). In fünf Reitern bzw. ihren Registerkarten (siehe Abbildung 4.12) können Sie verschiedene Einstellungen für die grafische Umgebung vornehmen.

Folgende Einstellungen können auf den Registerkarten getroffen werden:

► ALLGEMEIN

Durch *Kurzinfos* und *Kurzhilfe* aktivieren Sie kleine Popup-Hilfen und -Erläuterungen, wenn Sie den Mauszeiger über einem Symbol oder Feld positionieren.

Das *lokale Exemplar* kann automatisch beim Start der Tools gestartet werden.

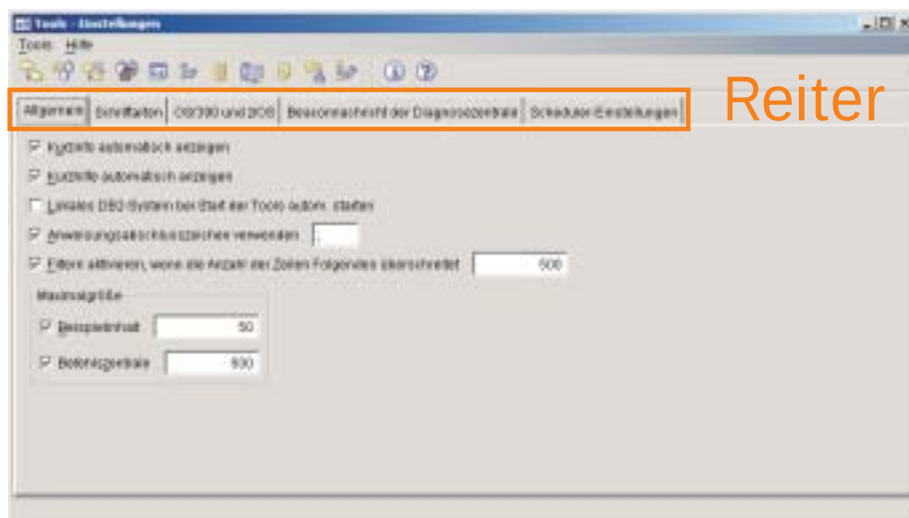


Abbildung 4.12: Tools-Einstellungen – Allgemein

Verschiedene Anweisungen (DB2-Befehle) werden durch das Strichpunktzeichen getrennt. So kann ein einzelner Befehl über mehrere Zeilen gehen und wird – dank des Semikolons – als ein Befehl erkannt. Dies kommt insbesondere bei der Befehlszentrale zum Tragen.

Außerdem können Sie automatisch den *Filter-Dialog* (siehe Abbildung 4.5) aufrufen lassen, wenn mehr als 500 Einträge (z.B. in der Tabellen-Detailsicht) angezeigt würden.

Auch die Größe der Ausgabe der *Beispielinhalte* – im Tabellen-Kontextmenü – ist standardmäßig auf 50 Zeilen begrenzt, kann aber hier angepasst werden.

Als Ergebnis werden in der *Befehlszentrale* nach obiger Einstellung 500 Datensätze angezeigt.

► SCHRIFTARTEN

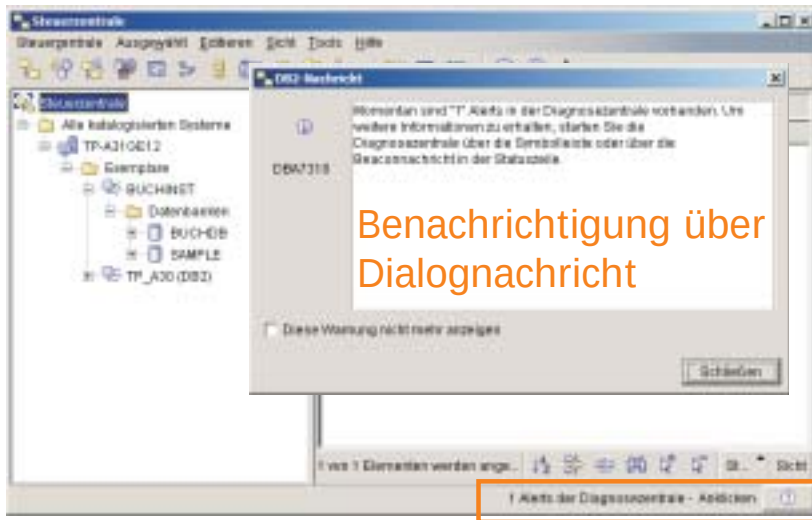
Unter diesem Punkt können Sie die Schriftarten, -größen und -farben auswählen, die im GUI verwendet werden sollen. Das ist nicht nur für die Weitsichtigen unter uns gut, sondern auch bei Präsentationen hilfreich, um die Lesbarkeit zu verbessern. Für die Aktivierung dieser neuen Einstellung müssen alle GUIs vorher beendet werden.

► OS/390 UND Z/OS

In dieser Registerkarte können spezifische Einstellungen vorgenommen werden, wenn z/OS-Systeme mit der Steuerzentrale verwaltet werden sollen. Diese können, ähnlich wie entfernte DB2-Server, in die Steuerzentrale aufgenommen werden und über sie administriert werden. Dabei entsprechen dann die Objektbaum-Einträge der z/OS- (bzw. OS/390-)Systeme der DB2-Architektur mit z.B. Subsystemen etc.

► BEACONNACHRICHT DER DIAGNOSEZENTRALE

„Beacon“ bedeutet – falls Sie das wie wir nicht wussten – im Englischen u.a. „Blinklicht“, „Funkfeuer“ oder „Leuchtfeuer“. Hier wird konfiguriert, wie man bei Nachrichten der Diagnosezentrale „verständigt“ wird: entweder über eine Nachricht in der Statuszeile der GUIs, über ein Nachrichtenfenster oder über beide (bzw. keine der) Möglichkeiten.



Benachrichtigung über Statuszeile

Abbildung 4.13: Benachrichtigungsoptionen der Diagnosezentrale

Hinweis:

Die Option BENACHRICHTIGUNG ÜBER DIALOGNACHRICHT erzeugt ein DB2-Nachrichtenfenster, das andere Aktivitäten im GUI blockiert. Diese Einstellung ist somit eher hinderlich als produktivitätsfördernd.

► SCHEDULER-EINSTELLUNGEN

In diesem Reiter kann eine neue Toolskatalog-Datenbank erstellt und die Zeitplanung definiert werden.

4.2.5 Tipps

Angepassten Ordner erstellen

Ab DB2-Version 8.1.2, das entspricht Fixpack 2, wird in der Steuerzentrale als erste Stufe der Objekthierarchie Alle katalogisierten Systeme angezeigt. Erst in der zweiten Stufe folgen die eigentlichen Systeme, also die Datenbank-Server.

Durch diese zusätzliche Ebene wird es möglich, eigene Gruppierungen Ihrer registrierten Server vorzunehmen. Damit können Sie bei der Verwaltung von vielen entfernten Servern Ordnung wahren bzw. schaffen.

Abbildung 4.14 zeigt den Aufruf dieser Funktion über das Kontextmenü auf der Ebene der Steuerzentrale.

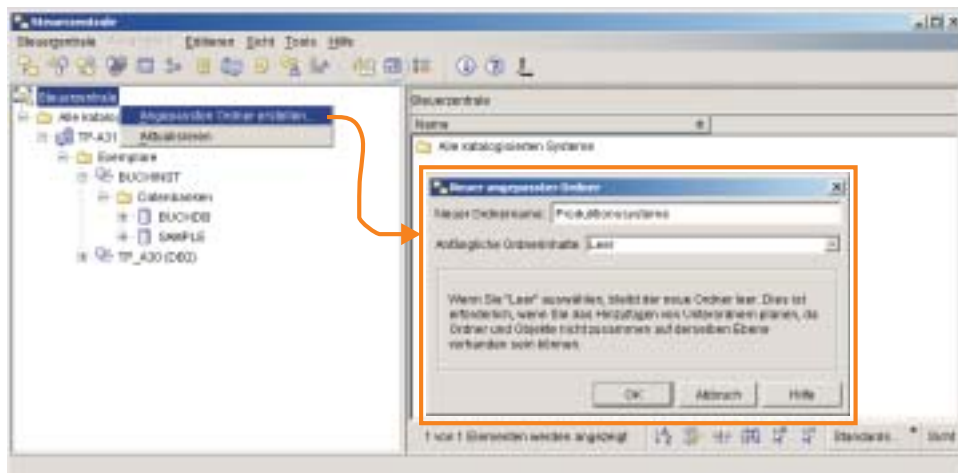


Abbildung 4.14: Angepassten Ordner in der Steuerzentrale erstellen

Um nichts zu verwechseln, können Sie beispielsweise eigene Ordner für Test- und Produktionssysteme bzw. -Datenbanken anlegen. Die „Original“-Anordnung bleibt weiterhin zusätzlich bestehen.

Eine andere Idee wäre die separate Ausweisung Ihrer Tools-Datenbank oder die Trennung nach Data Warehouse-Umgebung und OLTP-Systemen (OnLine Transactional Processing).

Über die Auswahl ANFÄNGLICHE ORDNERINHALTE können Sie bestimmen, ob die neue Gruppierung SYSTEME oder DATENBANKEN beinhalten soll.

Optionen für den Aufruf der Steuerzentrale

Der Aufruf der Steuerzentrale erfolgt über eine Script- bzw. Batch-Datei mit dem Namen `db2cc`. Dieser Datei können verschiedene Startoptionen mitgegeben werden. Die meisten Optionen beziehen sich auf den Inhalt, der gleich nach dem Start angezeigt werden soll. Die Option `-ict` („ict“ steht für „idle connection time“) allerdings ermöglicht es, einen Wert in Sekunden mitzugeben, nach dem von der Steuerzentrale gehaltene (inaktive) Verbindungen zu Datenbanken automatisch getrennt werden.

Folgender Aufruf würde bedeuten, nach zehn Minuten ohne Aktion die Verbindung zu trennen:

```
db2cc -ict 600
```


Das Status-Teilfenster zeigt Ihnen neben Statusinformationen auch alle Rückmeldungen von DB2-Befehlen an. In unserem Beispiel in Abbildung 4.15 ist das die Bestätigung der Datenbankverbindung. Diese wird zusätzlich dauerhaft in der Zeile DATENBANKVERBINDUNG, inklusive System und Exemplar, angezeigt.

Über die Zeile BEFEHLSPROTOKOLL haben Sie eine (Befehls-)Historie der abgesetzten Befehle zur Verfügung und können so diese Befehle immer wieder zurückholen und erneut ausführen.

Wenn Sie hier einen SQL-Befehl ausführen, werden die Ergebnisse – diese liegen immer in tabellarischer Form vor – standardmäßig auf dem separaten Reiter ERGEBNISSE angezeigt (siehe Abbildung 4.16). „Standardmäßig“ deshalb, weil Sie dies über die OPTIONEN...-Auswahl des Menüs BEFEHLSZENTRALE ändern können. Dabei können Sie alternativ die Ausgabe in eine Datei umleiten oder die Anzeige im Statusfenster zurückgeben lassen.

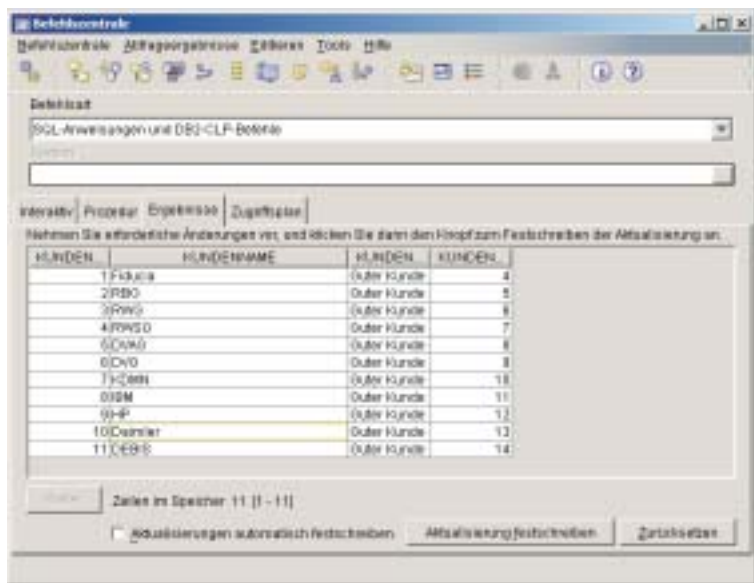


Abbildung 4.16: Befehlszentrale – Reiter ERGEBNISSE

Der Vorteil dieser grafischen Oberfläche im Gegensatz zur Befehlszeile ist die Ausgabetabelle, in der alle Spalten sauber untereinander angezeigt werden. In der Befehlszeile haben Sie bei längeren Zeilen Umbrüche, die die Zuordnung von Daten zu Spaltennamen schwierig und die Ausgabe unübersichtlich machen.

Ein weiterer Vorteil ist, dass Sie hier im Ergebnis gleich die Daten ändern können, falls Sie dazu in der Datenbank berechtigt sind. Der Button AKTUALISIERUNG FESTSCHREIBEN entspricht dabei einem Commit-Befehl, ZURÜCKSETZEN einem Rollback-Befehl. Dies ist sicherlich nicht als „die“ Benutzerschnittstelle schlechthin zu sehen, ist aber für kleine Änderungen sehr nützlich.

Doch zurück zum Reiter INTERAKTIV.

Dort können Sie mit dem Button AN PROZEDUR ANFÜGEN einzelne Befehle im INTERAKTIV-Fenster testen und dann einer Prozedur (einem Script) anfügen. Dazu kopiert dieser Button den aktuellen Befehl in das Prozedurfenster. Dies bringt uns zum zweiten Reiter der Befehlszentrale – dem Reiter PROZEDUR.

4.3.2 Prozedur

In diesem Teil der Befehlszentrale können Sie sich Prozeduren – also Scripts – zusammenstellen.

Abbildung 4.17 zeigt den Script-Fenster-Bereich, in dem Sie die Befehle eingeben können. Dabei gilt: „--“ markiert einen Kommentar, und alle Befehle werden mit dem Strichpunkt abgeschlossen. Dieses Befehlsende-Symbol wird über das Tools-Einstellungen-GUI (siehe Abbildung 4.12) definiert.

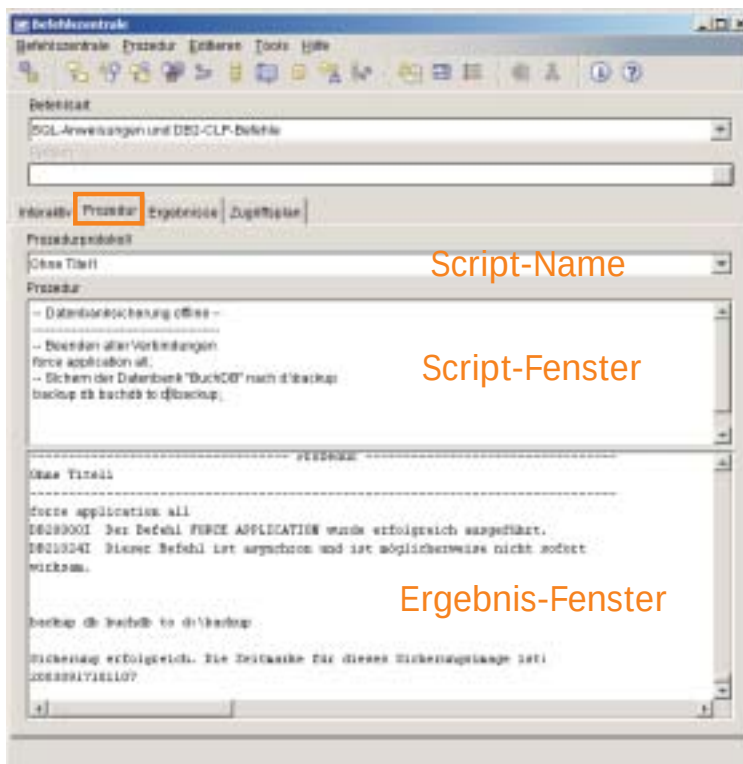


Abbildung 4.17: Befehlszentrale – Reiter Prozedur

Das Ergebnis-Fenster zeigt alle Rückmeldungen des Systems an – inklusive jeglicher SQL-Ergebnisse. Eine Verzweigung auf den ERGEBNIS-Reiter erfolgt hier nicht.

Der Script-Name wird über das Menü PROZEDUR/PROZEDUR SICHERN UNTER... vergeben (siehe Abbildung 4.18). Über die Auswahl IN TASKZENTRALE SICHERN bestimmen

Sie, ob der untere Fensterteil aktiv bleibt, in dem Sie das Script (bzw. die Prozedur) in der Taskzentrale ablegen können, oder ob der obere Fensterteil aktiv wird, um das Script im Dateisystem zu speichern.

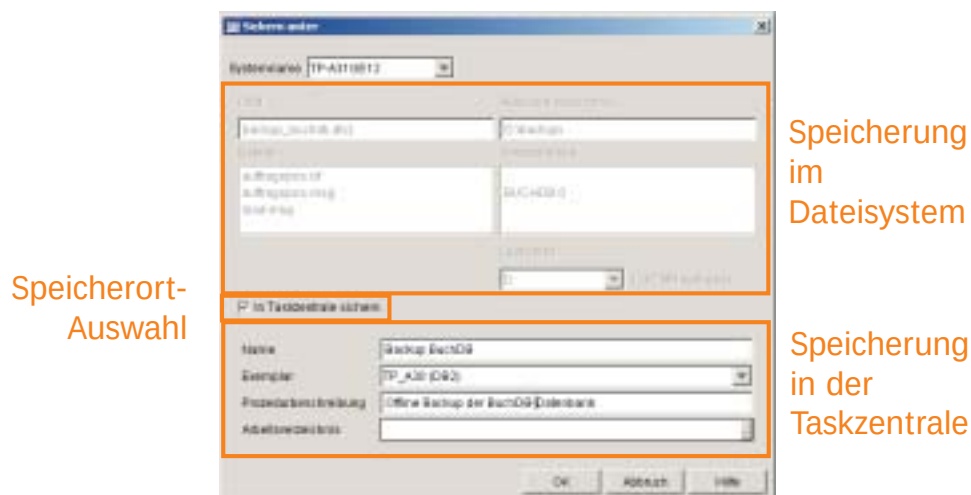


Abbildung 4.18: Befehlszentrale – Reiter Prozedur – Sichern unter-Fenster

Für die Speicherung in der Taskzentrale ist es erforderlich, dass Sie eine Beschreibung der Prozedur mitgeben, die so „sprechend“ sein sollte, dass Sie später darüber feststellen können, für welchen Zweck das Script gedacht war – ohne den Inhalt erneut zu studieren.

Das Eingabefeld ARBEITSVERZEICHNIS ist dann interessant, wenn Sie aus der Prozedur andere Dateien ansprechen und diese entweder ohne Pfad oder mit relativem Pfad angegeben haben. Dann können Sie hier das Verzeichnis angeben, in dem die Prozedur ausgeführt wird, also das Verzeichnis, in dem die Dateien stehen bzw. von dem der relative Pfad zu den Dateien gefunden wird.

4.3.3 Zugriffsplan

Der Reiter ZUGRIFFSPLAN zeigt standardmäßig ein leeres Fenster. Erst wenn Sie über die OPTIONEN...-Auswahl des BEFEHLSZENTRALE-Menüs ZUGRIFFSPLAN AUTOMATISCH GENERIEREN im ZUGRIFFSPLAN-Reiter auswählen, wird bei der nächsten SQL-Abfrage hier der Zugriffsweg grafisch angezeigt. Mehr dazu erfahren Sie im Performance-Kapitel 15, „Performance“.

4.4 Taskzentrale

Mit der Taskzentrale können Sie Ihre Scripts (Prozeduren) verwalten. Sie haben dabei die Möglichkeit, bestehende Prozeduren zeitlich gesteuert ablaufen zu lassen und Benachrichtigungsoptionen festzulegen. Zusätzlich gibt es die Möglichkeit, die Ergebnisse der Script-Aufrufe zu kontrollieren und Statistikinformationen über die Ausführungen zu bekommen.

Ein DAS (DB2 Administration Server) und ein funktionsfähiger Toolskatalog sind Voraussetzung für die Nutzung der Taskzentrale.

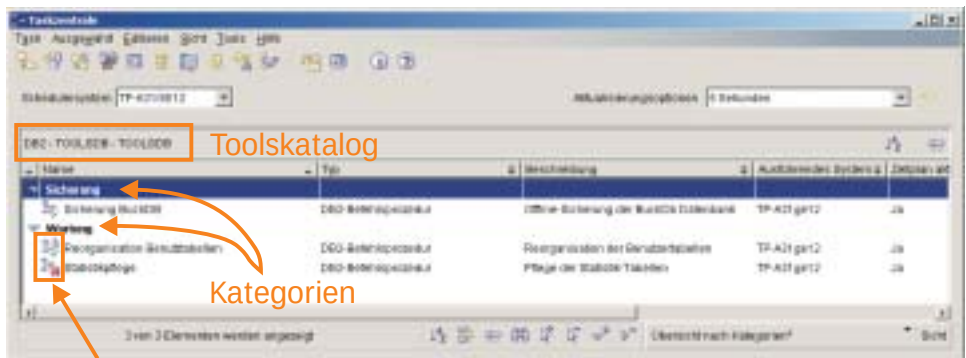


Abbildung 4.19: Taskzentrale

Die Anzeige der Taskzentrale bezieht sich immer auf einen bestehenden Toolskatalog, der die Tasks beinhaltet und sich um die Verwaltungsaufgaben wie das Scheduling – also die zeitliche Steuerung – kümmert. Standardmäßig findet keine automatische Aktualisierung der Anzeige statt, das kann aber über die Auswahl AKTUALISIERUNGSOPTIONEN im rechten oberen Teil des Fensters angepasst werden.

Bestehende Prozeduren lassen sich über die IMPORTIEREN...-Option des TASK-Menüs in die Taskzentrale aufnehmen.

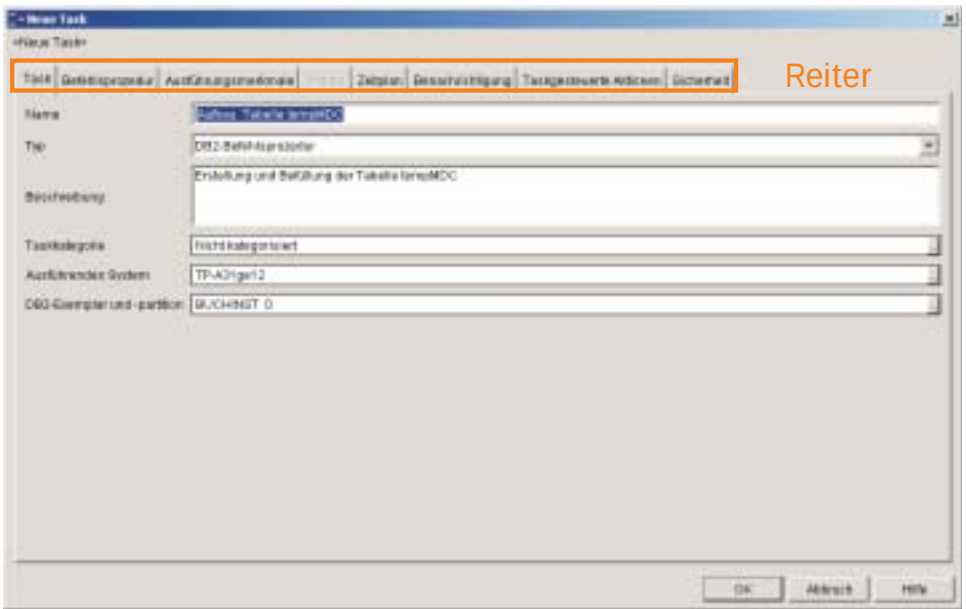


Abbildung 4.20: Taskzentrale – Neue Task aufnehmen

Im Fenster NEUE TASK können Sie auf acht Registerkarten Angaben zu einem Script machen:

► TASK

Wie in Abbildung 4.20 gezeigt, können neben dem Namen eine Beschreibung sowie das Zielsystem eingegeben werden, auf dem die Task ausgeführt werden soll.

► BEFEHLSPROZEDUR

Auf der zweiten Registerkarte sehen Sie die Befehle der Prozedur und können das Anweisungsabschlusszeichen („;“) anpassen. Auch die Angabe des Arbeitsverzeichnisses ist hier möglich.

► AUSFÜHRUNGSMERKMALE

Hier können Rückgabewerte eingegeben werden, die eine erfolgreiche Ausführung bedeuten.

► GRUPPE

Die Gruppierung mehrerer Scripts bzw. Tasks unter einer Task ist möglich. Damit kann die Steuerung über diese einzelne neue Task ausgeführt werden.

► ZEITPLAN

Diese Registerkarte bietet diverse Möglichkeiten, um die zeitliche Steuerung der Ausführung einzustellen.

► BENACHRICHTIGUNG

Aus der Gruppe der Ansprechpartner können hier Personen und Gruppen ausgewählt werden, die bei Erfolg, Misserfolg oder einer bestimmten Bedingung benachrichtigt werden. Zudem hat man die Möglichkeit, den Status dieser Task im Journal eintragen zu lassen. Auch der Nachrichtentext kann angepasst und mit Systemparametern erweitert werden.

► TASKGESTEUERTE AKTIONEN

Ebenso wie bei der Benachrichtigungsfunktionalität können bei Erfolg, Misserfolg oder einer Bedingung folgende weitere Aktionen ausgelöst werden:

- TASK AUSFÜHREN
- ZEITPLAN AKTIVIEREN VON
- ZEITPLAN DEAKTIVIEREN VON
- DIESE TASK LÖSCHEN

Die Zeitplan-Aktionen beziehen sich auf die Aktivierung bzw. Deaktivierung von bestehenden Zeitplänen einer auswählbaren Task.

Sie können mehrere Aktionen zu einer Bedingung definieren. Somit lassen sich Abhängigkeiten zwischen den Tasks herstellen, und die Tasks können dann z.B. automatisch hintereinander ausgeführt werden.

► SICHERHEIT

Lese-, Schreib- und Ausführrechte können im letzten Reiter bestimmten Gruppen oder Personen übertragen werden.

Die Tasks können zur besseren Übersichtlichkeit in Taskkategorien unterteilt werden.

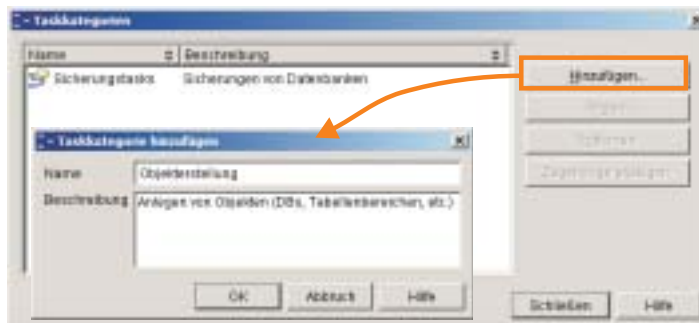


Abbildung 4.21: Taskzentrale – Taskkategorien

Über das Menü TASK – TASKKATEGORIEN können Sie neue Kategorien vorgeben (siehe Abbildung 4.21).

Die Icons vor der Taskbezeichnung (siehe Abbildung 4.19) zeigen Folgendes an:

- ▶ Grüner Haken: Die Task ist – bei der letzten Ausführung – erfolgreich beendet worden.
- ▶ Rotes Kreuz: Die Task ist fehlgeschlagen.
- ▶ Kleine Stoppuhr: Eine Zeitsteuerung für die Task ist aktiv.
- ▶ Schwarzer Läufer: Diese Task wird gerade ausgeführt.

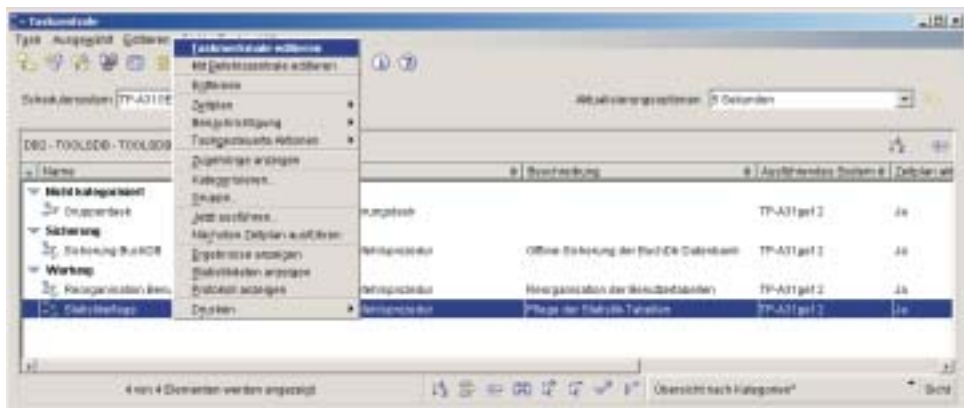


Abbildung 4.22: Taskzentrale – Task-Kontextmenü

Wenn Sie mehr Details über den Task-Status wissen wollen, haben Sie über das Kontextmenü der einzelnen Tasks (siehe Abbildung 4.22) folgende Optionen:

► ERGEBNISSE ANZEIGEN

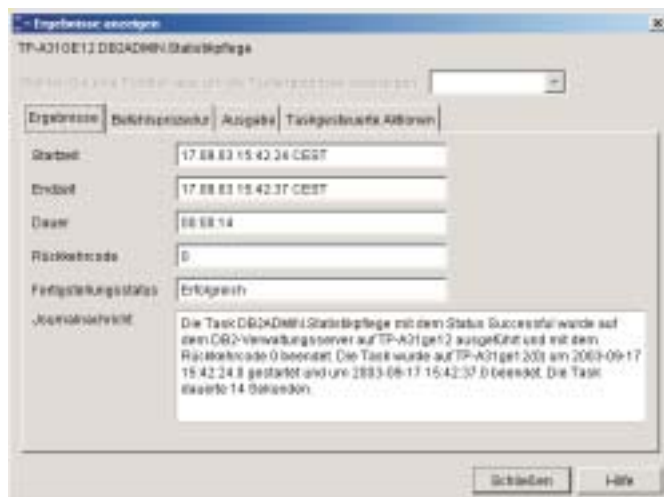


Abbildung 4.23: Taskzentrale – Ergebnisse anzeigen

Neben der Dauer und dem Rückkehrcode auf der ERGEBNIS-Registerkarte haben Sie über die anderen Reiter die Möglichkeit, sich weitere Details, wie den Inhalt und die Ausgabe des ausgeführten Scripts anzuschauen. Außerdem wird gemeldet, ob eine bzw. welche Aktion unter welcher Bedingung angestoßen wurde, z.B. dass die Sicherung der Datenbank ausgeführt wurde, weil die Task erfolgreich beendet worden ist.

► STATISTIKDATEN ANZEIGEN



Abbildung 4.24: Taskzentrale – Statistikdaten anzeigen

Durch die Statistiken erfahren Sie, wann und wie lange die Task gelaufen ist, und sehen die Gesamtanzahl der erfolgreichen Ausführungen bzw. der Fehlschläge. Diese Information ist besonders bei zeitkritischen Abläufen interessant, da Sie hier die Möglichkeit haben, auch die Entwicklung der Laufzeit zu sehen und abzuschätzen. So können Sie z.B. sehen, wie lange diese Task noch im gegebenen Zeitfenster ausgeführt werden kann, wenn die Datenmenge weiter so ansteigt wie bislang.

► PROTOKOLL ANZEIGEN

Dieser Punkt führt Sie direkt in das Journal.

Hinweis:

Die Steuerung komplexer Abläufe von Programmen, SQL und Prozeduren kann auch über die Data Warehouse-Zentrale erfolgen. Diese ermöglicht ebenfalls Verknüpfungen von einzelnen Schritten, deren Steuerung und Überwachung.

Das klassische Einsatzgebiet der Data Warehouse-Zentrale ist die Steuerung von Data Warehouse Populations-Systemen, also jenen Komponenten, die die Daten in das Data Warehouse einstellen.

4.5 Journal

Mit Hilfe des Journal-GUI können Sie durch verschiedene automatisch geführte Protokolle nachvollziehen, was in Ihrem System bzw. bei Ihren Datenbanken vorgefallen ist und ausgeführt wurde.

Beispielsweise ist es immer interessant zu wissen, was vorgefallen ist, während Sie im Urlaub waren und der Kollege stellvertretend für Sie als Datenbankadministrator tätig war. Da stellt sich dann oft die Frage, wann die letzte Datenbanksicherung durchgeführt worden ist, da der Fehler ja immer genau dann passiert, wenn man nicht damit rechnet.

Das GUI ist in vier Registerkarten unterteilt, die wir uns nun etwas genauer ansehen.

4.5.1 Taskprotokoll

Im TASKPROTOKOLL werden die Informationen angezeigt, die durch die Benachrichtigungsfunktion der Taskzentrale (siehe Abbildung 4.20) erstellt wurden. So sehen Sie – je nach Definition in der Taskzentrale –, wann welche Tasks erfolgreich bzw. nicht erfolgreich liefen.

Da dieses Fenster standardmäßig nicht automatisch aktualisiert wird, sollten Sie durch ein manuelles Aktualisieren (z.B. durch den kleinen gelben Pfeil rechts oben) oder die Aktualisierungsoptionen sicherstellen, dass Sie den tatsächlichen Zustand angezeigt bekommen. Diese Anzeige bezieht sich wieder auf ein System bzw. einen Toolskatalog, der über dem Taskprotokoll-Fenster angezeigt wird.

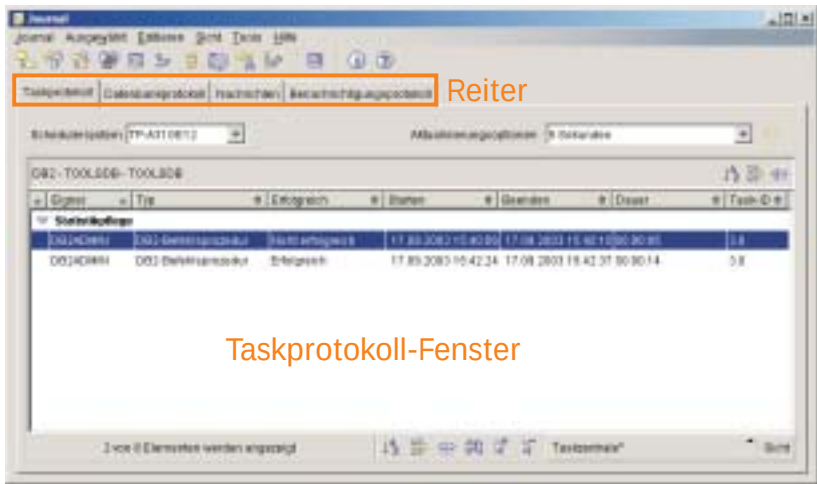


Abbildung 4.25: Journal – Taskprotokoll

4.5.2 Datenbankprotokoll

Das Datenbankprotokoll (recovery history file) umfasst wichtige Operationen einer Datenbank, wie z.B.:

- ▶ Ladevorgänge
- ▶ Sicherungen
- ▶ Wiederherstellungen
- ▶ Reorganisationen
- ▶ Erstellung und Löschung von Objekten (z.B. Tabellenbereichen)

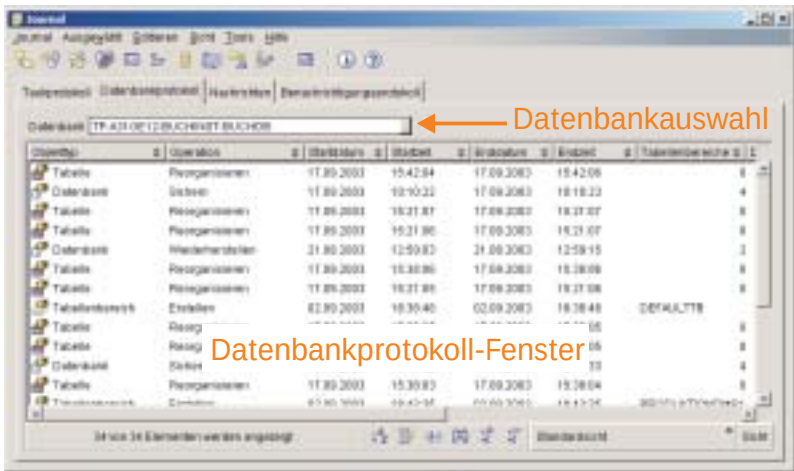


Abbildung 4.26: Journal – Datenbankprotokoll

Bevor Sie eine Anzeige wie in Abbildung 4.26 bekommen, müssen Sie die Datenbank auswählen, von der Sie das Protokoll gezeigt bekommen wollen. Das funktioniert über den Button „...“ oberhalb des Datenbankprotokoll-Fensters. Da das GUI generell auch für die Administration mehrerer und auch entfernter Systeme ausgelegt ist, ist es notwendig – wie Abbildung 4.27 zeigt –, das System, das Exemplar und dann die Datenbank auszuwählen.



Abbildung 4.27: Journal – Datenbankauswahl

Tipp:

In dieser Ansicht können Sie für bestimmte Einträge auch ein Kontextmenü mit weiteren Möglichkeiten öffnen. Die wichtigste Funktionalität dabei ist das Kontextmenü der Operation SICHERN, das Ihnen einerseits erlaubt, die gesicherten Tabellenbereiche aufzulisten, und andererseits – über die WIEDERHERSTELLEN...-Auswahl – eine direkte Schnittstelle zum ASSISTENTEN: DATEN WIEDERHERSTELLEN bietet. Darin ist dann Ihre gewählte Sicherung vorselektiert. So können Sie auf einfache Weise im Fehlerfall die aktuellste Sicherung zurückspielen. Details zur Sicherung und Wiederherstellung – auch mit Hilfe des Assistenten – finden Sie in Kapitel 9, „Sicherung und Wiederherstellung einer Datenbank“.

4.5.3 Nachrichten

Unter dem NACHRICHTEN-Reiter finden Sie Informationen, Warnungen und Fehlermeldungen, die durch interaktive Befehle oder Scripts im GUI verursacht wurden.

Das Textfeld zeigt die Nachricht in Kurzform. Insgesamt ist dieser Teil des Journals weniger interessant, da Sie diese Informationen bei der Befehls- oder Scriptausführung auch direkt gezeigt bekommen haben.

4.5.4 Benachrichtigungsprotokoll

Das Benachrichtigungsprotokoll (notification log) zeigt – je nach Konfiguration im Exemplar (notifylevel) – Fehler, Warnungen und evtl. auch informative Meldungen eines Exemplars an. Dabei handelt es sich im Gegensatz zum Nachrichten-Fenster um DB2-interne Meldungen für den Administrator der Datenbank bzw. des Exemplars.

Der BENACHRICHTIGUNGSPROTOKOLLFILTER bestimmt den Umfang der Anzeige. Hier legen Sie fest, ob Meldungen der Diagnosezentrale mit angezeigt werden sollen.

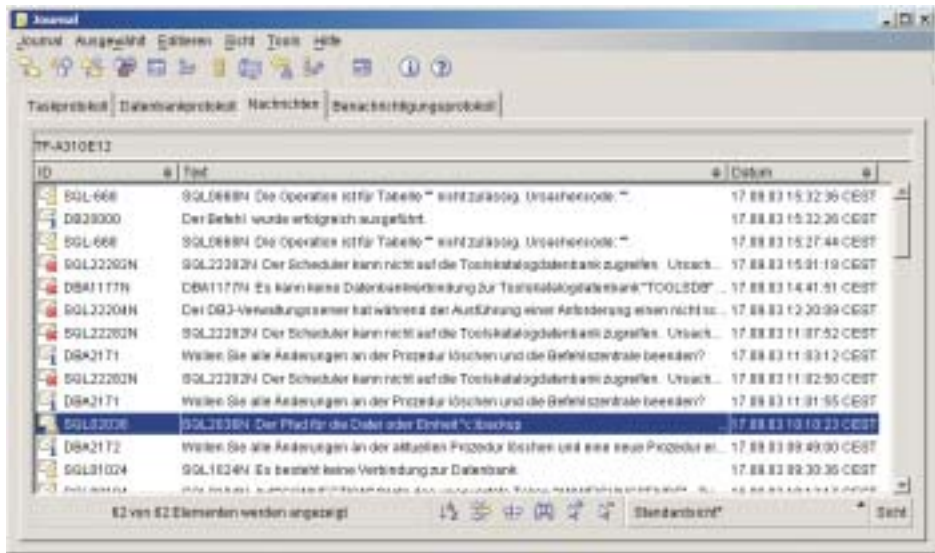


Abbildung 4.28: Journal – Nachrichten

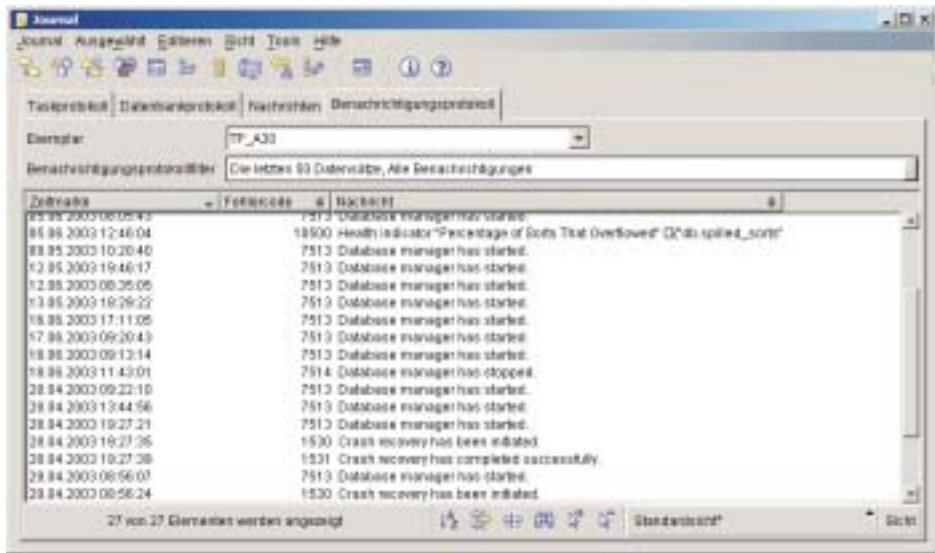


Abbildung 4.29: Journal – Benachrichtigungsprotokoll

Zu jedem Eintrag ist ein kleines Detailfenster über das Kontextmenü verfügbar. Mehr Details zur DB2-Diagnose und -Fehlerbehandlung finden Sie unter im Abschnitt 13.2, „Wartung und Problemanalyse“.

4.6 Diagnosezentrale

In der Diagnosezentrale können auch relativ unerfahrene DB2-Administratoren schnell einen Einblick in den „Gesundheitszustand“ Ihrer Systeme bekommen. Automatische Warnungen und Alarme sollen Missstände schnell ans Licht bringen, und wertvolle Informationen und Hilfestellungen bieten dem Administrator eine schnelle und unkomplizierte Eingriffsmöglichkeit.

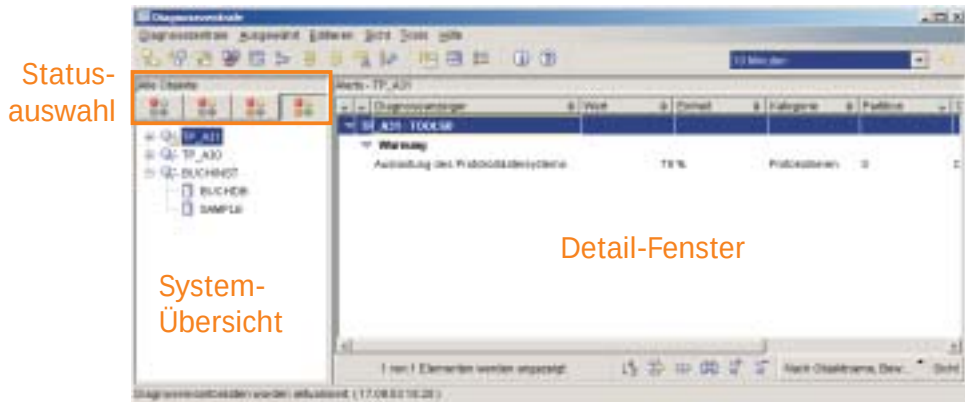


Abbildung 4.30: Diagnosezentrale

In der Statusauswahl können Sie den Umfang der Anzeige bestimmen. Sie haben dabei folgende Auswahl:

- Objekte im Alarmstatus
- Objekte im Alarm- oder Warnungsstatus
- Objekte im beliebigen Alertstatus (hier wurde wohl vergessen, „Alert“ mit „Alarm“ zu übersetzen)
- Alle Objekte

Das Detail-Fenster zeigt dann die genaue Meldung und das System an, das die Meldung verursacht hat.

Dieses sehr hilfreiche Tool behandeln wir in Kapitel 14, „Überwachung“.

4.7 Replikationszentrale

Die Replikationszentrale ist die grafische Oberfläche für die Verwaltung der Replikation. Aber darauf sind Sie sicherlich auch gekommen. Bevor wir ein wenig genauer auf diese Oberfläche eingehen, wollen wir ein paar grundsätzliche Aspekte zur Replikation erläutern.

Die Replikation ist ein *asynchroner* Vorgang für die automatische „Versetzung“ von Daten. Damit ist gemeint, dass Daten aus Tabellen bzw. auf Tabellenebene – das können einzelne Tabellen, Ausschnitte davon oder auch eine Kombination aus

mehreren Tabellen sein – auf einer anderen Datenbank zeitlich versetzt (asynchron) den gleichen Stand haben sollen wie im Ausgangssystem.

Schauen wir uns dieses Szenario anhand der Abbildung 4.31 an.

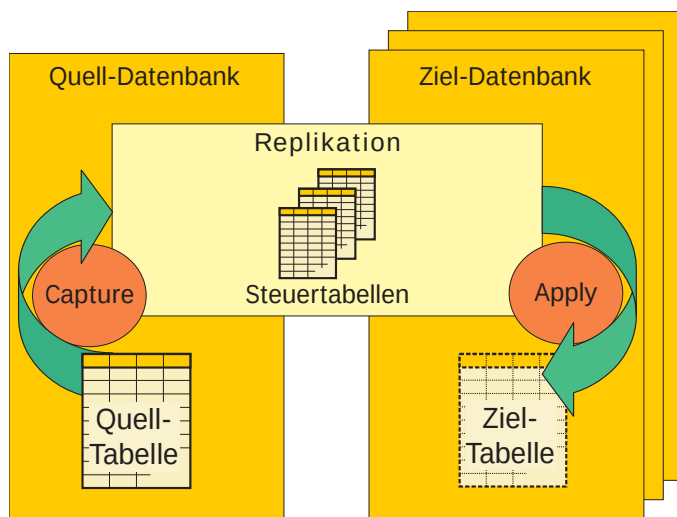


Abbildung 4.31: Replikation – das Konzept

Der Einfachheit halber soll hier in der Abbildung eine Tabelle repliziert werden. Dafür wird die Quell-Tabelle als Replikationsquelle markiert. Das „Herzstück“ der Replikation – neben einigen Steuertabellen – sind die zwei Prozesse *Capture* und *Apply*. Der Capture-Prozess liest die Quell-Tabelle (bzw. deren Veränderungen) regelmäßig aus und schreibt alle Änderungen in eine Änderungstabelle, die Bestandteil der Replikation ist. Diese Änderungstabelle ist dann der Ausgangspunkt des Apply-Prozesses, der dann die Änderungen in das Ziel, also die Ziel-Tabelle, überträgt. Ein replizierter Datenbestand kann auf beliebig viele Ziel-Systeme übertragen werden – dort ist jeweils ein eigener Apply-Prozess notwendig.

Weitere Steuertabellen dienen zur Konfiguration und Verwaltung des Replikationsprozesses. In regelmäßigen konfigurierbaren Abständen werden nun die Änderungen von der Quelle zum Ziel übertragen und so die Inhalte „nachgezogen“.

Bei einer „beidseitigen“ Replikation funktioniert diese Übertragung in beide Richtungen, d.h., die Quelle ist auch Ziel und das Ziel ist auch Quelle.

Das soll an dieser Stelle als Grundlage für das Replikationskonzept reichen.

Die Replikationszentrale ist nun die administrative grafische Schnittstelle, um diese Replikationsdefinitionen durchzuführen und die Prozesse und den Ablauf zu überwachen.

Gleich beim Start dieses Tools erscheint ein Assistent – die Klickstartleiste –, mit dessen Hilfe Sie ein Replikationsszenario aufbauen können (siehe Abbildung 4.32).

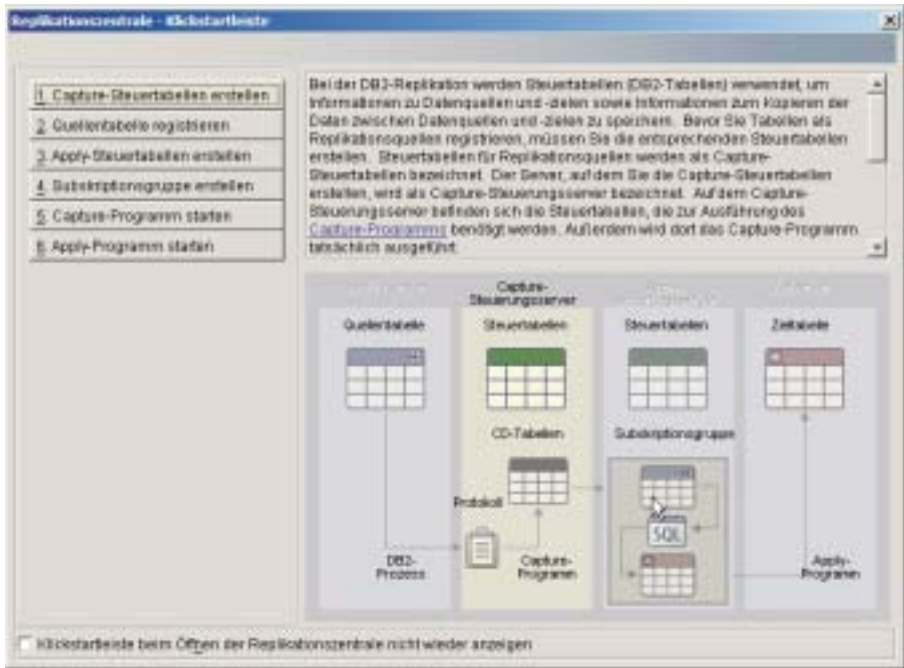


Abbildung 4.32: Replikationszentrale – Klickstartleiste

Anschließend gelangen Sie in die Replikationszentrale (siehe Abbildung 4.33), in der Sie weitere Definitionen vornehmen und den Ablauf überwachen können. Der Aufbau dieser Oberfläche erinnert stark an die Steuerzentrale – mit Überblicks- und Detail-Fenster. Letztendlich ist die Replikationszentrale sozusagen die „Steuerzentrale“ der Replikation.

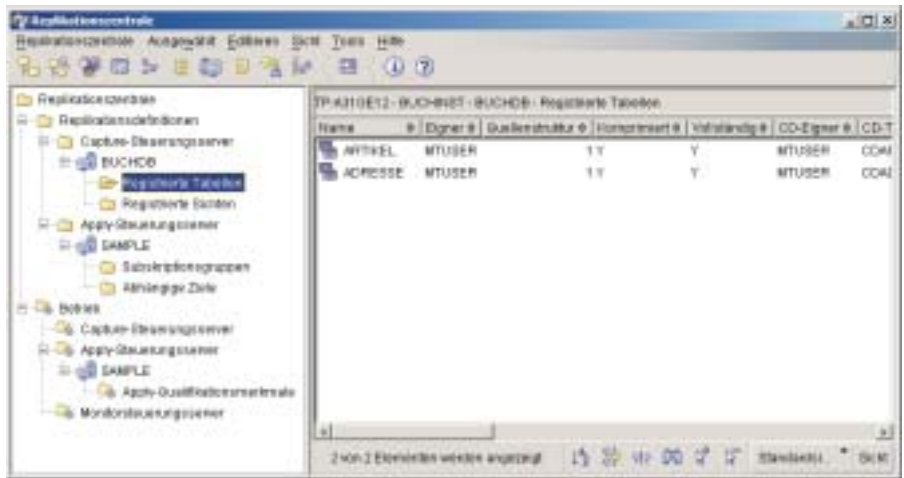


Abbildung 4.33: Replikationszentrale

4.8 Satellitenverwaltungszentrale

Die Satellitenverwaltungszentrale hat einen ähnlichen Aufbau wie die Steuerzentrale. Mit Hilfe dieses Tools können Sie Datenbank-Server (hier Satelliten genannt) zentral verwalten.

Der Unterschied zur Steuerzentrale ist hierbei, dass es sich um Gruppen von DB2-Servern mit gleicher Konfiguration handelt, die alle bestimmte identische Geschäftsanwendungen ausführen und für die kein DB2-Administrator vor Ort sein kann bzw. muss.

Beispielsweise könnte das bei Versicherungsvertretern der Fall sein, die einen lokalen DB2-Server auf ihren Laptops laufen haben, um Kundenverträge zu verwalten. Da diese Vertreter nur wenig IT-Wissen brauchen, kann ihnen die Verwaltung der Datenbank nicht zugemutet werden und muss stattdessen zentral stattfinden.

Mit der Satellitenverwaltungszentrale können Sie Gruppen von Satelliten (also DB2-Servern) aufbauen und verwalten, indem Sie zentral Verwaltungsscripts erstellen. Diese werden neben anderen Konfigurationen in einer DB2-Datenbank (SATCTLDB) gespeichert und dann, sobald eine Verbindung von den entfernten DB2-Servern aufgebaut wird, auf diese verteilt. Die Ergebnismeldungen werden dann wieder zentral gesammelt und können ausgewertet werden.

Diesen Austausch nennt man auch *Synchronisation*.

Diese kann durch das Satellitensynchronisator-GUI gestartet werden, das auch über das Startmenü in Windows verfügbar ist (siehe Abbildung 4.1). Die Satellitenverwaltungszentrale wiederum hat keinen eigenen Eintrag im Windows-Startmenü, sondern wird über das Schnellstart-Icon der anderen GUIs aufgerufen.

4.9 Data Warehouse-Zentrale

Die Data Warehouse-Zentrale dient zur Verwaltung und Steuerung von Data Warehouse-Umgebungen.

Eine Data Warehouse-Umgebung ist eine Software- und Hardware-Architektur. Sie soll Ihre Daten im Unternehmen so aufbereiten und bereitstellen, dass Sie diese Informationen in Ihrem Entscheidungsfindungsprozess nutzen können und so „bessere“ (fundiertere) Entscheidungen treffen können. Bisher werden häufig nur einfache Berichte verwendet, und die Gesamtheit Ihrer gespeicherten Daten und Informationen wird nur sehr eingeschränkt genutzt. Das liegt zum einen an der „gewachsenen“ heterogenen Struktur, die eine Datenhaltung in den verschiedensten Systemen zur Folge hat, die niemand mehr versteht, und zum anderen an einem Mangel an Informationen (z.B. einer fehlenden Historie). Mit einer Data Warehouse-Umgebung soll durch die Zentralisierung und Aufbereitung aller für die Analyse und Entscheidungsfindung wichtigen Daten und durch die Bereitstellung geeigneter Auswertungstools dieser Missstand beseitigt werden.

Bevor Sie mit diesem Tool arbeiten können, müssen Sie folgende vorbereitende Schritte durchführen:

1. Definition einer Steuerungsdatenbank für die Data Warehouse-Zentrale.

Alle Einstellungen, die Sie in der Data Warehouse-Zentrale vornehmen, werden in einer DB2-Datenbank gespeichert. Diese müssen Sie erst definieren bzw. über folgende Schritte in der Oberfläche DATA WAREHOUSE-ZENTRALE/STEUERUNGSDATENBANKVERWALTUNG anlegen lassen:

- Starten Sie die GUI-Oberfläche DATA WAREHOUSE-ZENTRALE/STEUERUNGSDATENBANKVERWALTUNG (siehe Abbildung 4.34) per Direktaufruf mittels `db2wcdm` oder in Windows über das Startmenü: START/PROGRAMME/IBM DB2/INSTALLATIONSTOOLS/DATA WAREHOUSE-ZENTRALE/STEUERUNGSDATENBANKVERWALTUNG.



Abbildung 4.34: Data Warehouse-Zentrale – Steuerungsdatenbankverwaltung

- Geben Sie einen Namen für die Steuerungsdatenbank ein.
 - Geben Sie eine entsprechende Berechtigung (mit BENUTZER-ID und KENN-WORT) für die Erstellung einer Datenbank ein.
 - Durch Bestätigung mit OK wird eine neue Steuerungsdatenbank erstellt bzw. eine bestehende Steuerungsdatenbank aktiviert.
- 2. Aufruf der Data Warehouse-Zentrale**
- über den Befehl `db2wcdm` oder
 - über das Schnellstart-Icon (vgl. Abbildung 4.2) aus einem geöffneten DB2-GUI oder
 - in Windows über das Startmenü
- 3. Anmeldung an der Data Warehouse-Zentrale**
- Dabei müssen Sie zuerst über den ERWEITERT...-Button die Steuerungsdatenbank angeben, wie in Abbildung 4.35 zu sehen ist.
 - Durch Angabe einer gültigen Benutzer-ID und des entsprechenden Kennworts wird das GUI freigegeben.



Abbildung 4.35: Data Warehouse-Zentrale – Anmeldung

Auch hier begrüßt Sie zuerst eine Klickstartleiste, die Ihnen hilft, sich schnell in dieser Oberfläche zurechtzufinden, indem sie eine sinnvolle Reihenfolge vorgibt und eine begleitende Anleitung bei den notwendigen Definitionen zur Verfügung stellt.



Abbildung 4.36: Data Warehouse-Zentrale – Klickstartleiste

Die eigentliche Data Warehouse-Zentrale zeigt dann wieder den typischen „Explorer-Look“ mit Übersichts- und Detail-Fenster (siehe Abbildung 4.37).

Typischerweise definiert man mit diesem Tool Abläufe zum Bereitstellen der Daten für Analysezwecke. Darunter fallen z.B. der sog. ETL- oder Populationsprozess. „ETL“ steht für:

- ▶ Extract – also für die Selektion und Extrahierung der Daten aus den operationalen Datenbankquellen
- ▶ Transform – also für die Aufbereitung der Daten
- ▶ Load – also für die Bereitstellung im Data Warehouse

Dafür können in der Data Warehouse-Zentrale einzelne Prozesse im Ordner THEMENBEREICHE definiert werden. Diese setzen sich dann aus einzelnen sog. Schritten zusammen, die Sie grafisch modellieren können (siehe Abbildung 4.37).

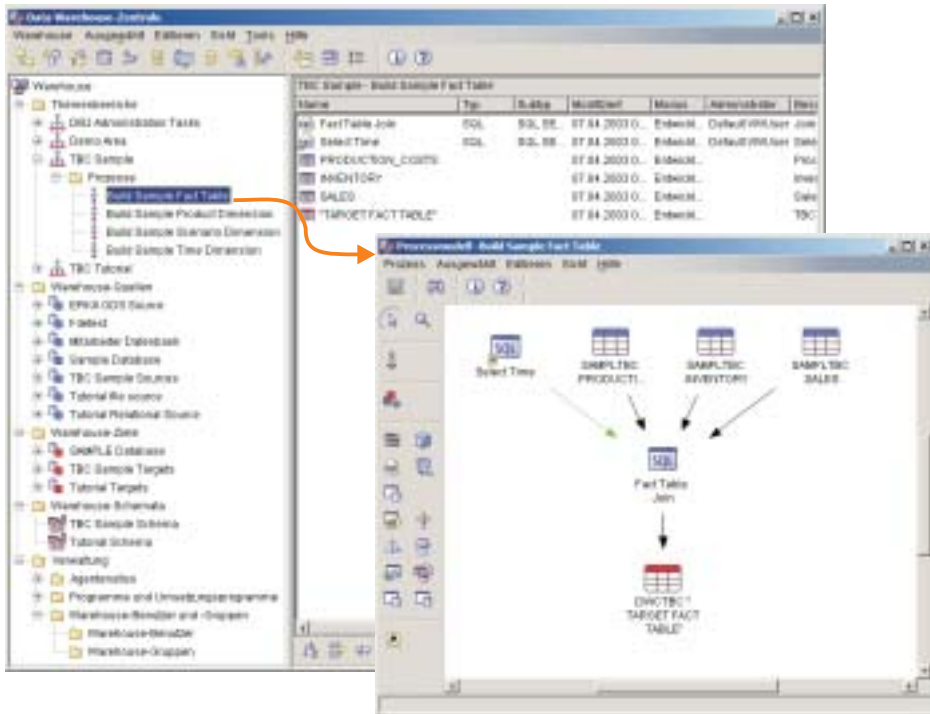


Abbildung 4.37: Data Warehouse-Zentrale mit Prozessmodell

Hinweis:

Um die Funktionalität dieser Oberfläche besser kennen zu lernen und auch mit Testdaten hantieren zu können, empfiehlt es sich, das LERNPROGRAMM FÜR INFORMATIONSMANAGEMENT: DATA WAREHOUSE/EINFÜHRUNG (Data Warehouse Tutorial) durchzugehen.

Zum Starten dieser Anleitung wählen Sie aus dem Menü HILFE – DB2-LERNPROGRAMME und dann LERNPROGRAMM FÜR INFORMATIONSMANAGEMENT: DATA WAREHOUSE/EINFÜHRUNG.

Sie werden dann in mehreren Schritten mit Hilfe der Data Warehouse-Zentrale durch den Aufbau und Test einer Beispielumgebung geführt.

Hier bietet sich dann ebenfalls die Möglichkeit, neben SQL-Befehlen und DB2-Tools externe Programme und selbst geschriebenen Programmcode einzubinden. Sowohl die Definition der Ablaufreihenfolge (zeitlich- oder ereignisorientiert) als auch deren Überwachung ist über diese Oberfläche möglich.

4.10 Informationskatalogzentrale

Die Informationskatalogzentrale ist ein weiteres GUI für die Unterstützung von Data Warehouse-Umgebungen. Dort wird ein sog. Metadatenkatalog benötigt, der alle Informationen über die Objekte (z.B. Tabellen) und deren Inhalte bzw. deren Bedeutungen speichert. Diese Informationen sind für die Endbenutzer wichtig, um die in der Analyse angezeigten Informationen „richtig“ interpretieren zu können. Nur so werden sinnvolle Abfragen und Analysen möglich. Administratoren andererseits benötigen auch sog. technische Metadaten mit Informationen über die ursprüngliche Datenquelle, Berechtigungen und Aufbereitungsschritte.

Die Informationskatalogzentrale bietet die Möglichkeit, so ein zentrales Metadaten-System abzufragen und Endbenutzern zugänglich zu machen.

Die eigentliche Datenhaltung erfolgt in einer DB2-Datenbank durch den Information Catalog Manager, der Bestandteil des DB2 Warehouse Manager-Produkts ist.

4.11 Entwicklungszentrale

Mit der Entwicklungszentrale wurde ein mächtiges Werkzeug bereitgestellt, um die Entwicklung von

- ▶ gespeicherten Prozeduren
- ▶ benutzerdefinierten Funktionen
- ▶ strukturierten Typen

zu unterstützen.

Für die Entwicklung dieser Komponenten werden zuerst Projekte erzeugt, die festlegen, wo die Dateien gespeichert werden sollen und für welche Datenbank die Entwicklung gedacht ist. Ein Projekt kann manuell oder mit dem Assistenten erstellt werden, der gleich beim Start des Tools erscheint (siehe Abbildung 4.38).

Für alle drei Entwicklungstypen (gespeicherte Prozeduren, benutzerdefinierte Funktionen und strukturierte Typen) stehen zusätzlich weitere Assistenten zur Verfügung, die Ihnen die Entwicklung vereinfachen.

Das Hauptfenster der Entwicklungszentrale (siehe Abbildung 4.39) ist in vier Bereiche gegliedert:

- ▶ Projekt-Überblicks-Fenster: Hier finden Sie die unter einem Projekt zusammengefassten Entwicklungen.
- ▶ Projekt-Detail-Fenster: Hier sehen Sie die Detail-Daten zu den im Überblicks-Fenster ausgewählten Objekten.
- ▶ Status-Fenster: Hier bekommen Sie Rückmeldungen über die Erstellung (Kompilierung etc.) Ihrer Entwicklungen.
- ▶ Ergebnis-Fenster: Hier werden die Nachrichten und Ergebnisse z.B. einer gespeicherten Prozedur dargestellt.

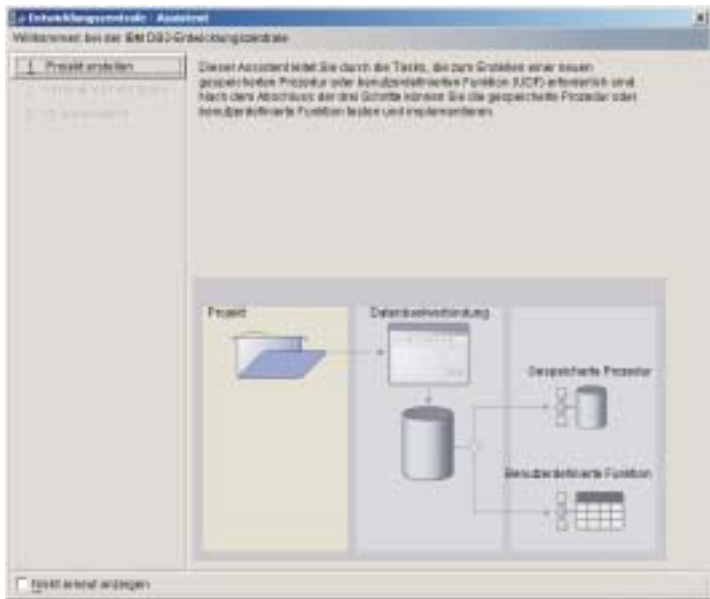


Abbildung 4.38: Entwicklungszentrale – Assistent

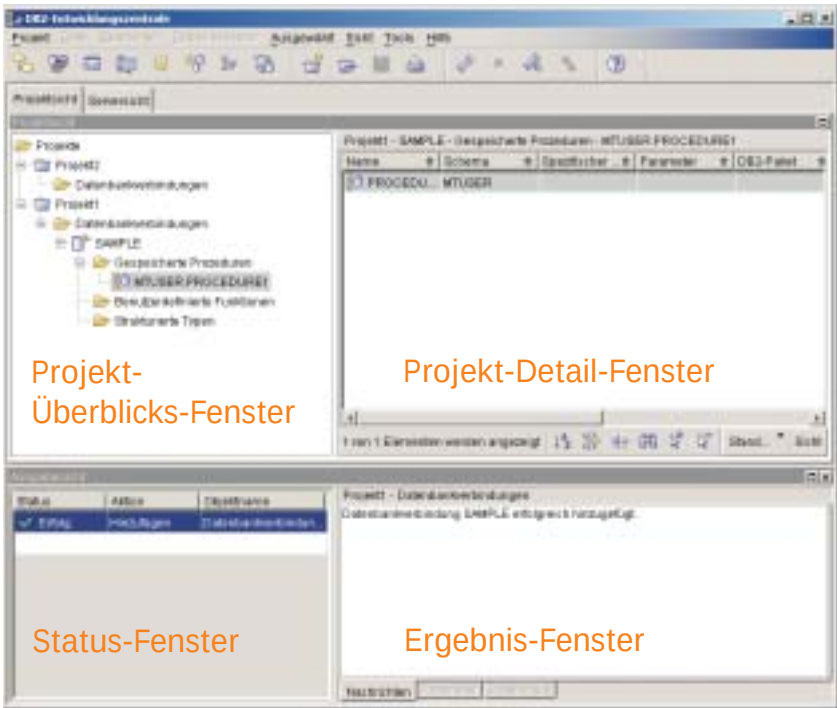


Abbildung 4.39: Entwicklungszentrale

Eine zweite Registerkarte – SERVERSICHT – zeigt erstellte und implementierte Objekte in der Zieldatenbank an.

Dieses Tool hat weitere Fenster für die Entwicklung und Code-Anzeige. Eine Syntaxmarkierung ist ebenso selbstverständlich wie zahlreiche Debug-Möglichkeiten.

Für die Implementierung gibt es zusätzlich das separat aufrufbare Projekt-Verteilungstool.

Insgesamt handelt es sich um eine durchaus komplette Entwicklungsumgebung, die hier mit DB2 ausgeliefert wird. Zusätzlich gibt es einige Add-Ins für Entwicklungstools von Microsoft, um die Entwicklung für DB2-Objekte dort zu unterstützen.

4.12 Lizenzzentrale

Die Lizenzzentrale ist ein kleines grafisches Tool, mit dem Sie Ihre Lizenzen auflisten und so sehen können, welche Lizenzierungsart für welche Installation gültig ist.

Dabei unterscheidet man in DB2 zwei prinzipielle Arten der Lizenz:

- ▶ Prozessorlizenz: Diese ist unabhängig von der Anzahl der Benutzer.
- ▶ Benutzerlizenz: Dabei wird die Anzahl der Benutzer lizenziert.

Abbildung 4.40 zeigt die Oberfläche am Beispiel einer DB2 UDB ESE (Enterprise Server Edition), die für einen Prozessor lizenziert ist.

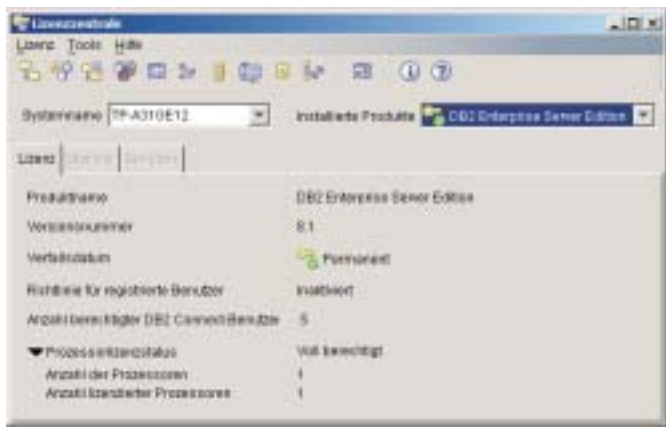


Abbildung 4.40: Lizenzzentrale

Bei einer Benutzerlizenzierung haben Sie hier die Möglichkeit, die Benutzer einzutragen bzw. über eine Statistik auszuwerten, ob Sie genug Benutzer lizenziert haben.

4.13 Information – Unterstützung

Die DB2-Hilfe erreichen Sie über die Oberfläche INFORMATION – UNTERSTÜTZUNG. Sechs Reiter gruppieren die aufgelisteten Informationen nach:

- ▶ TASKS – eine aufgabenorientierte Auflistung der Hilfeinformationen
- ▶ KONZEPTE – eine objekt- und stichwortorientierte Anordnung
- ▶ REFERENZ – befehls- und toolorientierte Zusammenstellung
- ▶ FEHLERBEHEBUNG – Gruppierung nach DB2-Fehlerkategorien
- ▶ BEISPIELE – Verweise auf DB2-Beispiele, vor allem für die Programmierung
- ▶ LERNPROGRAMME – Das sind „Schritt-für-Schritt“-Anleitungen.

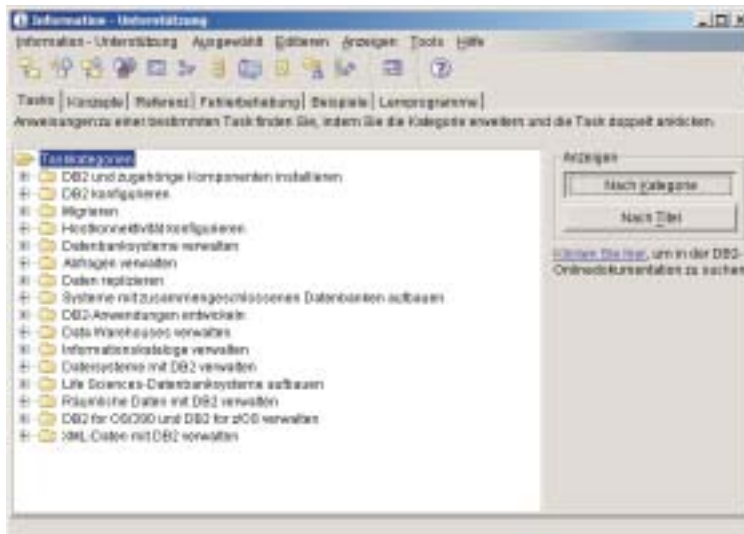


Abbildung 4.41: Information – Unterstützung

Wenn Sie einen Eintrag auswählen, öffnet sich automatisch ein Browserfenster mit der HTML-Dokumentation von DB2 (siehe Abbildung 4.42).

Wenn Sie die angezeigte Hilfe-Information im Hilfe-Fenster durchgeschaut haben, können Sie über das Übersicht-Fenster weitere Themen auswählen und dabei auch in andere Kategorien springen.

Über die Navigations-Icons haben Sie folgende weitere Möglichkeiten (von links nach rechts):

- ▶ SUCHEN – öffnet ein separates Suchfenster (siehe Abbildung 4.43) zur Volltextsuche.
- ▶ VERDECKEN – blendet das Übersicht-Fenster aus bzw. ein.
- ▶ SUCHEN – (leider genau wie das erste Icon übersetzt) Hiermit können Sie das zugehörige Kapitel bzw. den Eintrag der aktuell im Hilfe-Fenster angezeigten Information im Übersicht-Fenster anzeigen lassen.
- ▶ UNTERSTÜTZUNG – Dieses Icon führt Sie zur Supportseite von DB2 im Internet (<http://www.ibm.com/software/data/db2/udb/win02unix/support>).

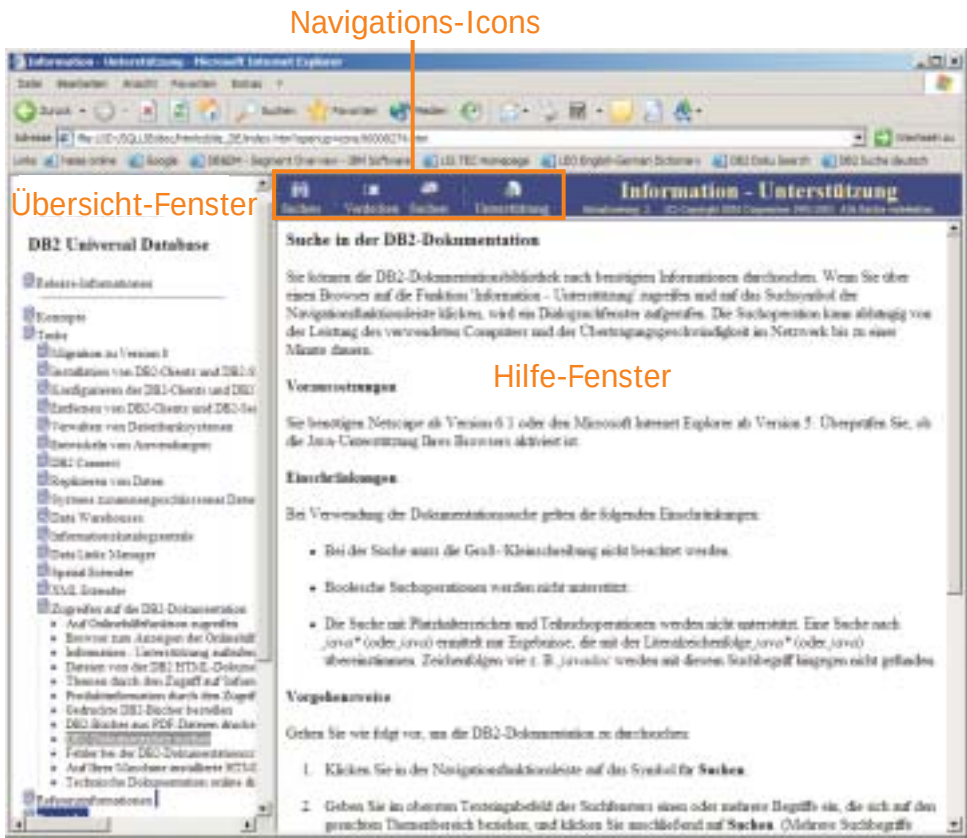


Abbildung 4.42: Information – Unterstützung – Browserfenster



Abbildung 4.43: Information – Unterstützung – Suchfenster



Abbildung 4.45: Konfigurationsassistent

Über das Menü KONFIGURIEREN haben Sie folgende Möglichkeiten:

- ▶ **DBM-KONFIGURATION**
Hier können Sie – wie über die Steuerzentrale – die Exemplarkonfiguration einsehen und anpassen.
- ▶ **DB2-REGISTRIERDATENBANK**
DB2-Registrierwerte können über ein weiteres Fenster angezeigt werden. Es können auch neue hinzugefügt werden, wobei die Liste der verfügbaren Registrierungseinstellungen angezeigt wird, allerdings leider ohne Hilfetext und gültigen Wertebereich.
- ▶ **PROFILE IMPORTIEREN ODER EXPORTIEREN**
Hier können Verbindungsprofile erstellt oder importiert werden (siehe Abschnitt 6.4.5).
- ▶ **ANDERES EXEMPLAR KONFIGURIEREN**
Wie die anderen grafischen Oberflächen ist auch diese für mehrere Exemplare geeignet – über diesen Menüeintrag können Sie auf andere Systeme und Exemplare umschalten.
- ▶ **KONFIGURATION ZURÜCKSETZEN**
Über ein weiteres Fenster können fast alle Einstellungen (System-Datenbank-Verzeichnis, Konfiguration etc.) zurückgesetzt werden.
- ▶ **BEENDEN**
Was wird das wohl machen?

Der Menüeintrag AUSGEWÄHLT hält den Assistenten DATENBANK HINZUFÜGEN bereit, mit dem Verbindungen zu entfernten Datenbanken konfiguriert werden können. Das behandeln wir in Kapitel 6.

Hinweis:

Mit DB2 V8 wird dieses GUI nicht mehr automatisch bei einer Client-Installation mit installiert, sondern ist nur noch auf dem Server oder dem administrativen RTC (Run Time Client) verfügbar.

4.15 Webtools

DB2 ermöglicht es auch, einige Verwaltungsaufgaben über eine Webschnittstelle – und somit wirklich komplett ortsunabhängig – durchzuführen. Dazu stehen die Webbefehlszentrale und die Webdiagnosezentrale zur Verfügung, die beide eine sichere, da SSL-verschlüsselte (SSL: Secure Socket Layer) Verbindung nutzen.

Als Voraussetzung ist ein Web-Anwendungsserver, wie z.B. der WebSphere Application Server, erforderlich.

5 Eine Datenbank erstellen

Bevor Sie Daten verarbeiten können, müssen Sie mit einer Datenbank die geeignete Umgebung erstellen.

Unter einer *Datenbank* verstehen wir ein logisches DB2-Objekt, das für die Datenhaltung zuständig ist und dessen grundsätzliche Aufgaben wir in Kapitel 1 besprochen haben.

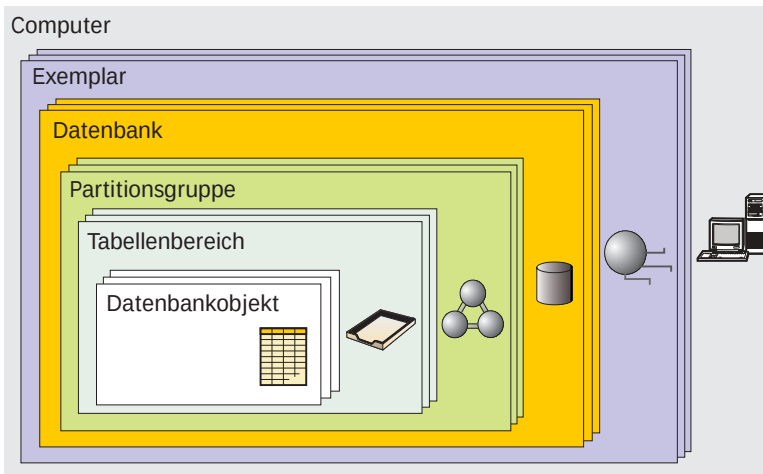


Abbildung 5.1: DB2-Objekthierarchie

In DB2 können Sie mehrere Datenbanken in einem Exemplar anlegen. Die Datenbanken sind untereinander absolut unabhängig, unterliegen aber der übergeordneten Exemplar-Konfiguration.

5.1 Überlegungen zur Anzahl der Datenbanken

Warum sollte man überhaupt mehr als eine Datenbank anlegen?

Die Gründe dafür, mehrere Exemplare auf einem System zu erzeugen, haben wir bereits in Kapitel 3 angesprochen.

Folgende Punkte können ausschlaggebend sein, um mehrere Datenbanken (in einem Exemplar) anzulegen:

► Getrennte Optimierung

Unterschiedliche Anwendungsanforderungen, z.B. OLTP (Online Transactional Processing – also transaktionsorientierte Verarbeitung) im Gegensatz zu OLAP (Online Analytical Processing – also auswertungsorientierte Verarbeitung), erfordern grundsätzlich andere Optimierungsmaßnahmen innerhalb der Datenbank.

- ▶ Ressourcenbeschränkungen
zum Beispiel Speicherlimitierung bei 32-Bit-Systemen (z.B. 2-Gbyte-Limit für DB-Pufferpools) oder E/A-Engpässe bei der Protokollierung
- ▶ Sicherungszyklen
Abhängig von den Datenbankanwendungen können unterschiedliche Anforderungen an das Sicherungskonzept notwendig sein, z.B. tägliche oder wöchentliche Sicherungsläufe (backups).
- ▶ Sicherheitsmanagement
Die verschiedenen Datenbanken können von unterschiedlichen Datenbankadministratoren verwaltet werden.

5.2 Komponenten einer Datenbank

Abbildung 5.2 illustriert die Bestandteile einer Datenbank. Neben den klassischen Datenbankobjekten (z.B. Tabellen) werden Verwaltungskomponenten benötigt.

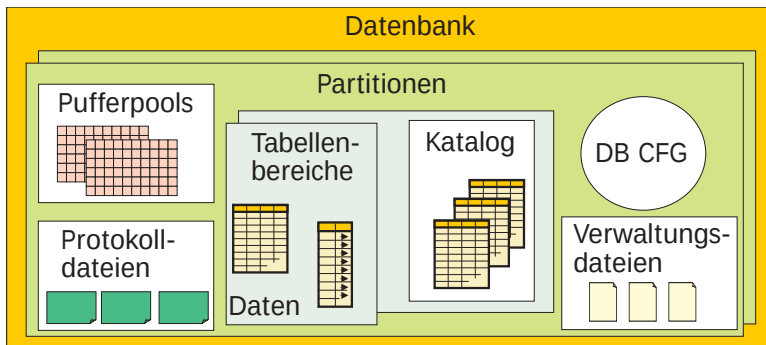


Abbildung 5.2: Datenbankobjekte

5.2.1 Datenbankkatalog

Jede DB2-Datenbank hat einen eigenen Datenbankkatalog, der aus mehreren Tabellen, Daten-Sichten und Indizes besteht und die logische und physische Struktur der Datenbank beschreibt. Diese Beschreibung (auch als Metadaten bezeichnet) enthält u.a. die Informationen über:

- ▶ bestehende Datenbankobjekte, wie z.B. Benutzertabellen
- ▶ Zugriffsberechtigungen auf Objekte, Programme und Werkzeuge
- ▶ Größe des Datenbank-Hauptspeichers (Pufferpools)
- ▶ Datenmengen und deren Verteilung (Statistiken)

5.2.2 Datenbankpartitionsgruppe

Eine Datenbankpartitionsgruppe ist eine logische Einheit innerhalb einer Datenbank. Diese ist nur dann relevant, wenn eine einzelne Datenbank über mehrere so genannte *Partitionen* (in früheren DB2-Versionen *Knoten* bzw. *Nodes* genannt) verteilt werden soll, die auf verschiedenen Maschinen liegen können.

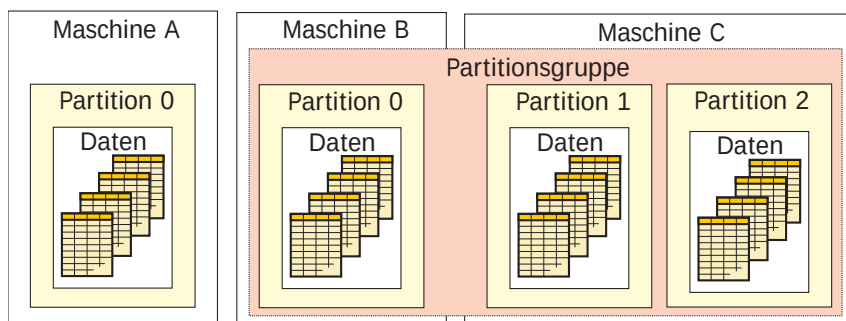


Abbildung 5.3: Datenbankpartition und -partitionsgruppe

In Abbildung 5.3 stellt Maschine A eine Einzelpartitionsdatenbank dar. In dieser Umgebung hat die Partition (Partition 0) keine Funktion, existiert aber für eine spätere Erweiterung auf eine partitionierte Umgebung. Das Gleiche gilt für Maschine B. Die DB2-Codebasis ist in beiden Fällen identisch.

Auf einer weiteren Maschine (Maschine C) können eine oder auch mehrere Partitionen definiert werden (per Assistent oder manuell über die Partitionskonfigurationsdatei `db2nodes.cfg`), die in Partitionsgruppen organisiert werden. Die Partitionsgruppen dienen zur einfacheren Administration sowie der optimierten Datenverteilung in einer partitionierten Datenbankinstallation.

5.2.3 Tabellenbereiche

Eine Datenbank besteht aus mehreren Tabellenbereichen, die ihrerseits Tabellen, Indizes und große Objekte (Large Objects-(LOB-)Daten und LONG-Daten) enthalten.

Der Tabellenbereich stellt die Verbindung zwischen den logischen Objekten (z.B. Tabellen) und der Hardware dar und ermöglicht die gezielte Verteilung der Daten auf die Festplatten.

5.2.4 Pufferpool

Der Pufferpool ist ein Hauptspeicherbereich, der für die Datenbank reserviert ist und der zum Zwischenspeichern (caching) von Daten dient. In diesen Speicherbereich werden die Tabellendaten auf Anforderung einer Anwendung kopiert (E/A-Anforderung), und dort werden sie bei Bedarf verändert. Bei einem erneuten Zugriff auf die gleichen Daten – auch von anderen Anwendungen – können diese nun sehr viel schneller verarbeitet werden, da sie bereits im Hauptspeicher vorliegen.

5.2.5 Protokolldateien

Ein Datenbanksystem benötigt für die Konsistenzerhaltung und für die Wiederherstellbarkeit einer Datenbank Informationen über die Veränderungen der Tabellendaten. Diese Informationen werden in Protokolldateien auf der Festplatte aufgezeichnet.

5.3 Wie erstelle ich eine Datenbank?

Eine Datenbank anzulegen ist sehr einfach, da Sie nur wenige Parameter setzen müssen und alle anderen Werte mit Standardwerten vorbelegt sind.

Für die Datenbankerstellung gibt es mehrere Alternativen:

- ▶ mit dem Datenbank-Assistenten in der grafischen Oberfläche
- ▶ mit einem DB2-Befehl
- ▶ durch das Zurückspielen einer Datenbanksicherung

5.3.1 Vorüberlegungen

Bevor Sie mit dem Aufbau einer Datenbank beginnen, sollten Sie sich über die Datenbankeinstellungen Gedanken machen, die nach der Erstellung nicht mehr verändert werden können.

Die folgenden Punkte können – müssen aber nicht – explizit angegeben werden, da sie aus den Systemeinstellungen automatisch übernommen werden bzw. mit Standardwerten belegt sind:

- ▶ Kodierter Zeichensatz (codeset)

Codeset ist die ISO-Bezeichnung (ISO – International Organization for Standardization) für eine Codepage. Eine Codepage ordnet Zeichen (z.B. Buchstaben, Zahlen, Sonderzeichen) einem Bitmuster zu, das in der Datenbank abgespeichert wird. So enthält u.a. die Codepage 850 die deutschen Umlaute.

- ▶ Datenbankgebiet (Territory) und Datenbankgebietscode (Country)

Die Gebietseinstellung legt u.a. das Datums- und das Uhrzeitformat fest. Das Gebiet (Territory) und der Gebietscode (Country) hängen direkt voneinander ab.

- ▶ Sortierreihenfolge (Collating Sequence)

DB2 bietet mehrere Möglichkeiten der Sortierung an. Der Standard ist die Einstellung SYSTEM. Sie bedeutet, dass die Sortierung durch das Gebiet bzw. den Gebietscode vorgegeben wird.

Beispiel:

Sortierreihenfolge SYSTEM: 123....aAbB...zZ...

Sortierreihenfolge IDENTITY: 123...AB...Zab...z...

Hinweis:

Der für diese Einstellung gewählte Wert kann später nur über das DB2-API abgefragt werden.

Weitere Details finden Sie in [ADMIN1] im Literaturverzeichnis.

5.3.2 Wie erstelle ich eine Datenbank mit dem GUI?

Das Anlegen einer Datenbank über das GUI bringt Ihnen einige Vorteile:

- ▶ schnelle Übersicht über das Zielsystem und dessen Exemplar durch eine Objektbaumstruktur
- ▶ Benutzerführung mit Erklärungen innerhalb des Assistenten
- ▶ Anzeige der von DB2 verwendeten Standardwerte

Mit dem GUI legen Sie eine Datenbank über die Steuerzentrale und unter Verwendung des Datenbank-Assistenten an. Wir zeigen im Folgenden den Aufruf des Assistenten und die wichtigsten Schritte für das Anlegen einer Datenbank.

In der Objektbaumstruktur navigieren Sie bis auf die Datenbanken-Ebene, d.h., Sie wählen Ihr gewünschtes System (Maschine) und ein bestehendes Exemplar. Über das Kontextmenü, das Sie immer mit der rechten Maustaste aktivieren, kommen Sie zur Auswahl in Abbildung 5.4.

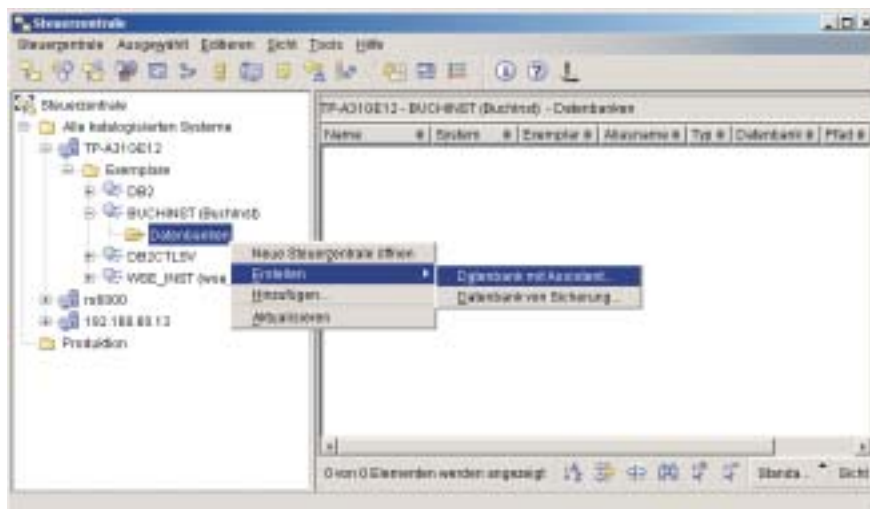


Abbildung 5.4: Aufruf des Datenbank-Assistenten

Der ASSISTENT: DATENBANK ERSTELLEN wird in einem separaten Fenster angezeigt.

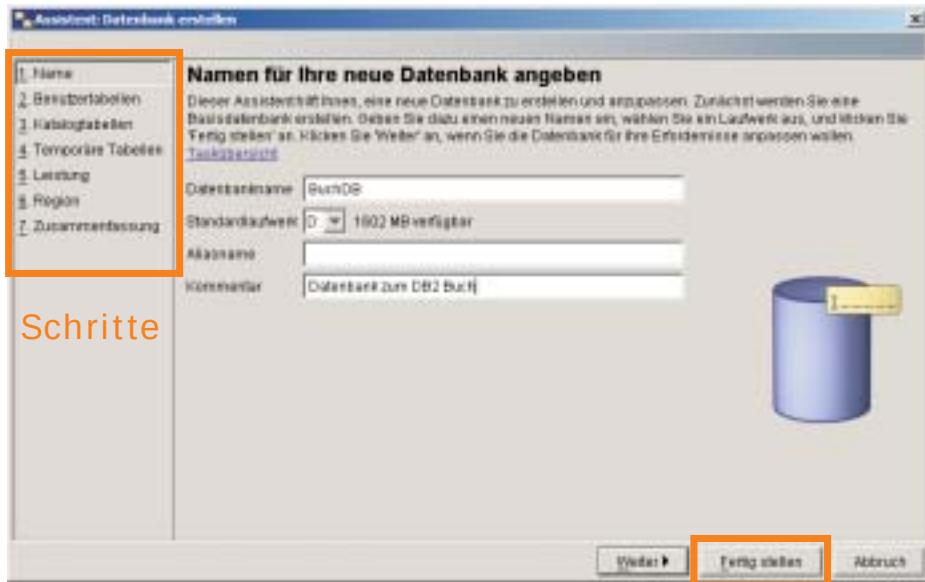


Abbildung 5.5: Assistent: Datenbank erstellen – Schritt 1 – Name

Die Eingabe eines ein- bis achtstelligen Datenbanknamens aktiviert sofort den Button FERTIG STELLEN rechts unten und zeigt an, dass alle weiteren Optionen mit Standardwerten vorbelegt oder optional sind. Das ist der schnellste und einfachste Fall der Datenbankerzeugung.

In Windows-Umgebungen wird das Standardlaufwerk (z.B. C:), in Unix-Systemen der Standardpfad (\$HOME des Exemplar-Besitzers) angezeigt, in dem die physikalische Datenbankstruktur (siehe Abbildung 5.10) erzeugt wird. Diese Vorgabe stammt aus der Exemplar-Konfiguration und kann dort unter dem Namen DFTDBPATH (DeFaulT DataBase PATH) gefunden werden.

Hinweis:

Die Datenbankstruktur wird auf der Festplatte in definierten Verzeichnissen abgelegt, die in Abschnitt 5.4 „Interne Abläufe beim Anlegen einer Datenbank“ aufgezeigt und näher beschrieben werden. Dabei handelt es sich um Kontrollstrukturen, die nur wenig Platz benötigen, und um Standard-Datenverzeichnisse, die später noch geändert werden sollten.

Die Auswahl der physikalischen Platzierung der Datenbankdaten, die die eigentliche Datenmenge darstellen, ist wesentlich entscheidender und wird in Abschnitt 5.8 „Die Tabellenbereiche“ behandelt.

Auf der in Abbildung 5.5 abgebildeten Seite können Sie zudem einen ALIASNAMEN für die Datenbank vergeben. Unter diesem Alias wird die Datenbank später ange-

sprochen; er ist sozusagen der „Rufname“ der Datenbank. Wird er nicht explizit angegeben, entspricht der Alias standardmäßig dem Datenbanknamen.

Ein Kommentar-Text hilft Ihnen dabei, sich schneller zu orientieren, wenn Ihr System viele Datenbanken enthält.

Entweder gelangen Sie über den WEITER-Button oder durch Direktauswahl des Schritts oben links zu den nächsten Schritten.



Abbildung 5.6: Assistent: Datenbank erstellen – Schritt 2 – Benutzertabellen

Schritt 2 bis Schritt 4 ermöglichen eine individuelle Definition der Tabellenspeicherung; in Schritt 5 können Sie optimierende Einstellungen dazu festlegen. Wir begnügen uns an dieser Stelle mit den Standards, da dies in den meisten Fällen ausreichend ist und der Hintergrund dazu erst in Abschnitt 5.7 „Partitionsgruppen“ beschrieben wird.

Die in Abschnitt 5.3.1 „Vorüberlegungen“ angesprochenen regionsspezifischen Einstellungen können dann in Schritt 6 (siehe Abbildung 5.7) vorgenommen werden. Entspricht die Anzeige bzw. die gesetzten Standards nicht Ihren Anforderungen, haben Sie hier die Möglichkeit, Änderungen vorzunehmen.

Der letzte Schritt (siehe Abbildung 5.8) zeigt nur noch die Zusammenfassung der gewählten Einstellungen an und ermöglicht es, den generierten DB2-Befehl (mit dem BEFEHL ANZEIGEN...-Button) einzusehen und als Task in der Taskzentrale abulegen. Näheres zum CREATE DATABASE-Befehl finden Sie im folgendem Abschnitt.

Mit dem Button FERTIG STELLEN wird die Datenbank angelegt und der Assistent beendet.

Alternativ kann die Datenbank auch direkt über einen DB2-Befehl erstellt werden.



Abbildung 5.7: Assistent: Datenbank erstellen – Schritt 6 – Region

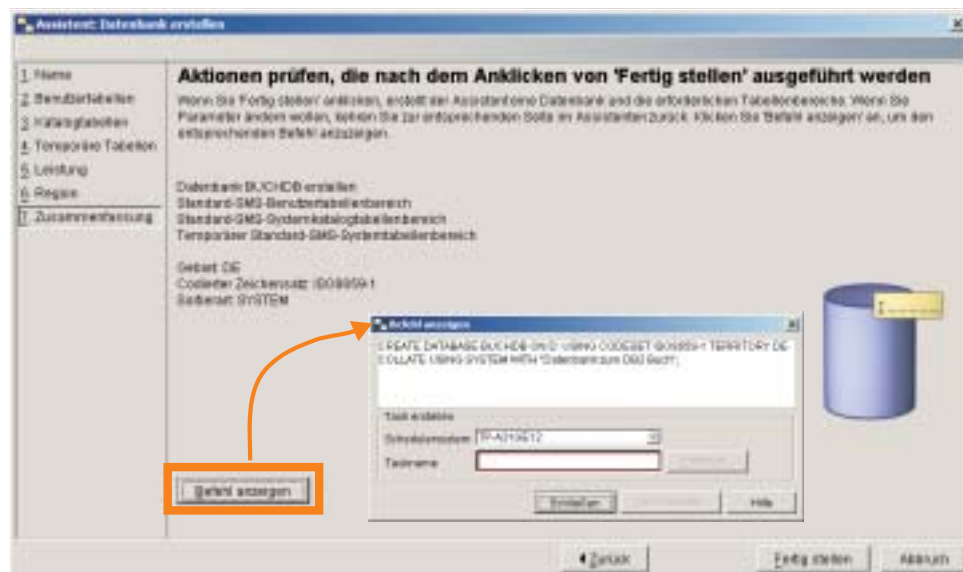


Abbildung 5.8: Assistent: Datenbank erstellen – Schritt 7 – Zusammenfassung

5.3.3 Das Anlegen einer Datenbank per DB2-Befehl

Auch das Anlegen einer Datenbank über die Befehlszeile bringt Vorteile mit sich:

- ▶ schnelle Eingabe
- ▶ Möglichkeit, den Befehl in eine Prozedur einzubinden
- ▶ Automatisationsmöglichkeit

Der einfachste DB2-Befehl zum Anlegen einer Datenbank sieht im gekürzten Syntaxdiagramm folgendermaßen aus:

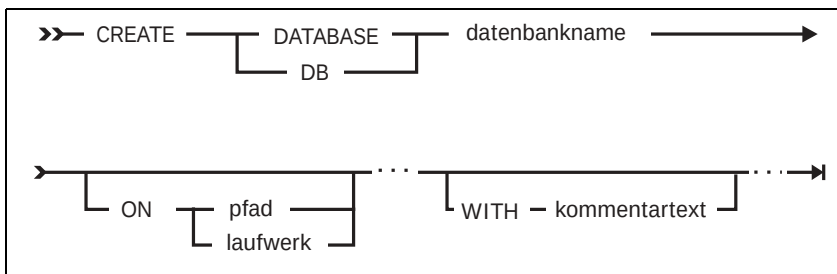


Abbildung 5.9: CREATE DATABASE-Syntaxdiagramm

Im Syntaxdiagramm steht der Pfad für Unix-Umgebungen (z.B. /datenbanken/) und das Laufwerk (z.B. D:) für die Windows-Welt.

Der Befehl zum Erstellen der Datenbank „BuchDB“ könnte folgendermaßen lauten:

```
create db Buchdb on c: with 'Datenbank zum DB2-Buch'
```

5.4 Interne Abläufe beim Anlegen einer Datenbank

Wenn Sie einen CREATE DATABASE-Befehl abgesetzt haben, werden Sie merken, dass einige Zeit vergeht, bis die Datenbank erstellt ist, auch wenn Sie die einfachste Ausprägung des Befehls gewählt hatten. Das liegt daran, dass dieser Befehl einige Aktionen anstößt, die im Folgenden aufgeführt sind:

- ▶ Die benötigte Verzeichnisstruktur wird erstellt.
- ▶ Interne Verwaltungsdateien werden erzeugt.
- ▶ Drei Standardpartitionsgruppen werden generiert:
 - IBMCATGROUP
 - IBMTEMPGROUP
 - IBMDEFAULTGROUP
- ▶ Drei Standardtabellenbereiche werden generiert:
 - SYSCATSPACE
 - TEMPSPACE1
 - USERSPACE1

- ▶ Der Datenbankkatalog wird erstellt, inklusive:
 - der Schemata SYSIBM, SYSCAT, SYSSAT, SYSFUN, SYSPROC und NULLID
 - Systemtabellen und -Sichten
 - Indizes
 - Pakete
 - Funktionen
 - gespeicherter Prozeduren
- ▶ Standardsicherheitsrichtlinien werden umgesetzt.
- ▶ Die Datenbankkonfiguration wird erstellt und mit Standardwerten vorbelegt.

Obwohl momentan noch keine Benutzerdaten gespeichert sind, werden alle Strukturen so erstellt, wie es später für große und größte Benutzerdatenmengen benötigt wird.

Doch bevor wir zu den Datenobjekten in einem der folgenden Kapitel kommen, wollen wir einen genaueren Blick auf die erzeugten Strukturen werfen.

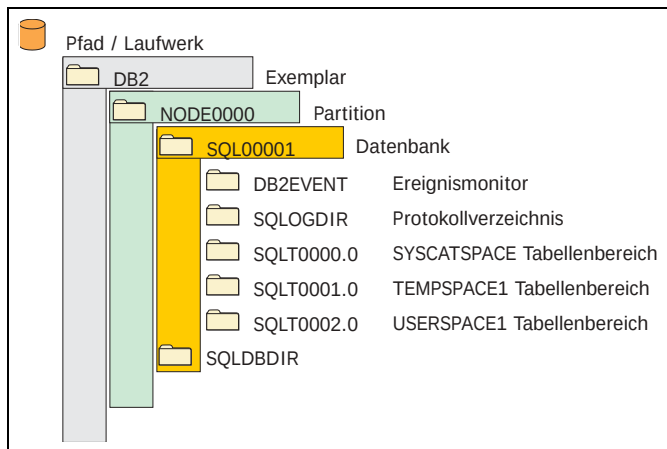


Abbildung 5.10: Verzeichnisstruktur einer Datenbank

Auch auf der Verzeichnisebene wird die Objekthierarchie (vgl. Objektbaumstruktur im GUI – z.B. in Abbildung 5.4) abgebildet. Auf dem beim Befehl `CREATE DATABASE` angegebenen Laufwerk bzw. Pfad wird automatisch ein Verzeichnis mit dem Exemplarnamen erstellt. Darunter findet man das Verzeichnis, das der Partitions-ebene entspricht. In einer Einzelpartitions Umgebung ist dies immer `NODE0000`.

Das Datenbankverzeichnis ist die nächste Ebene und wird, beginnend mit `SQL00001`, mit jeder neuen Datenbank sequenziell durchnummeriert, also `SQL00002` usw. In diesem Verzeichnis befinden sich die internen DB2-Verwaltungsdateien und die Standardverzeichnisse für die Tabellenbereiche und Protokolldateien – doch dazu später mehr.

Die Verwaltungsdateien im Datenbankverzeichnis dürfen nicht verändert oder gelöscht werden; einige Dateien liegen bereits mit Sicherungskopie doppelt vor.

Diese binären Dateien werden nur über entsprechende DB2-Befehle angesprochen oder von DB2 intern verwendet.

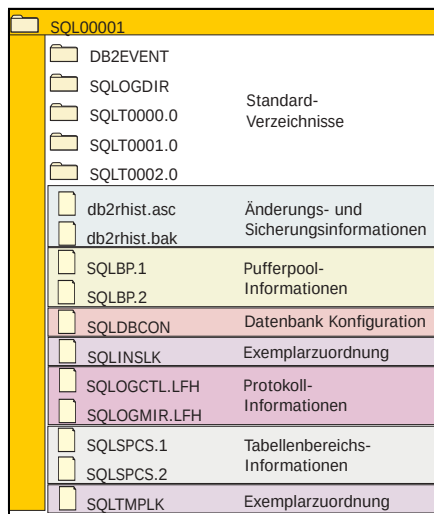


Abbildung 5.11: Verwaltungsdateien einer Datenbank

So dienen die Protokolldatei `db2rhist.asc` (recovery history file) und ihre Sicherungskopie zur Speicherung von Daten über Sicherungen und Wiederherstellungen, Ladeoperationen und andere Veränderungen der Tabellenbereiche oder der Datenbank.

`SQLDBCON` enthält die Datenbankkonfiguration, deren Anzeige, Inhalt und Veränderung wir in Abschnitt 5.6 „Die Datenbankkonfiguration“ näher unter die Lupe nehmen.

Die Datenbank-Protokoll-Konfigurationsdateien (`SQLOGCTL.LFH` und `SQLOGMIR.LFH`) bezeichnet man als Protokollsteuerdateien oder „log control files“. Sie enthalten Informationen über aktive Protokolldateien. Diese Dateien spielen bei der Wiederherstellung einer Datenbank eine sehr wichtige Rolle.

Die internen Dateien `SQLBP.1` und `SQLBP.2` enthalten Informationen zum Pufferpool; `SQLSPCS.1` und `SQLSPCS.2` enthalten Tabellenbereichsinformationen.

Sowohl `SQLINSLK` als auch `SQLTMPLK` stellen sicher, dass eine Datenbank nur zu einem Exemplar gehört bzw. darin liegen kann. Das schließt nicht aus, dass man eine Datenbank aus unterschiedlichen Exemplaren ansprechen und abfragen kann.

Der Platzverbrauch

Insgesamt werden ca. 20 Mbyte Speicher pro Datenbank benötigt, die sich in etwa so verteilen:

- ▶ 16,5 Mbyte für den Systemkatalog
- ▶ 3 Mbyte für die Protokolldateien
- ▶ weniger als 500 Kbyte für die Verwaltungsdateien

5.5 Aktivieren und Deaktivieren einer Datenbank

Nach dem Erstellen ist die Datenbank inaktiv. Um mit einer Datenbank bzw. deren Inhalten arbeiten zu können, müssen Sie sich mit dem `CONNECT`-Befehl mit der Datenbank verbinden.

5.5.1 Verbindung zu einer Datenbank herstellen

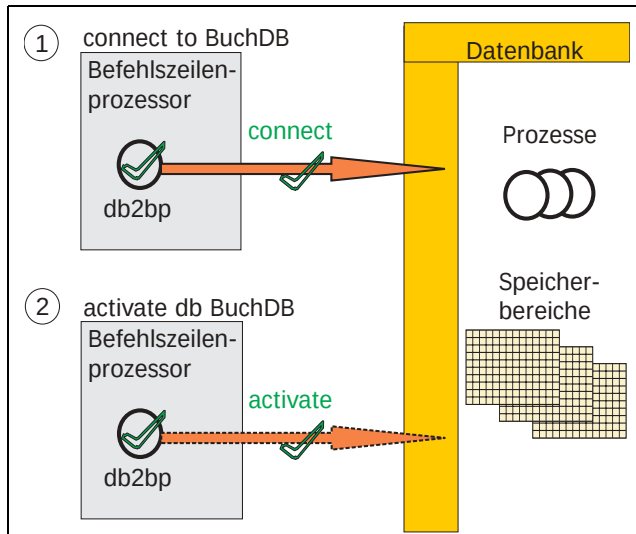


Abbildung 5.12: connect-Befehl und Aktivieren der Datenbank

Abbildung 5.12 zeigt bei Punkt ①, wie von einer Befehlszeile (oder durch eine Anwendung) eine Verbindung zur Datenbank hergestellt wird. Dabei wird, neben der eigentlichen Verbindung, ein so genannter „Backend“-Prozess gestartet, der Client-seitig die Verbindung zur Datenbank verwaltet.

Falls dieser Verbindungswunsch auf eine inaktive Datenbank trifft, wird die Datenbank durch diesen ersten `CONNECT` gestartet. Dabei werden Datenbankprozesse erzeugt und Speicherbereiche zugewiesen. Dieser Zusatzaufwand des Startvorgangs kann durch eine Aktivierung der Datenbank – per `ACTIVATE`-Befehl – vorweggenommen werden. Der `ACTIVATE`-Befehl stellt keine eigentliche Verbindung zur Datenbank her, trifft aber die gleichen Startvorbereitungen wie der Befehl `CONNECT`.

Ist die Datenbank aktiv, entsteht für weitere Verbindungen kein (bzw. kaum) Zusatzaufwand, da Speicherbereiche bereits reserviert und alle Prozesse aktiv sind und nun nur noch der Verbindung zugewiesen werden müssen.

Hinweis:

SQL-Abfragen und verschiedene Datenbankbefehle benötigen eine aktive Datenbankverbindung (CONNECT). Wenn Sie hauptsächlich mit einer bestimmten Datenbank arbeiten und den expliziten CONNECT-Befehl sparen wollen, können Sie die Registriervariable DB2DBDFT setzen:

```
db2set DB2DBDFT=<datenbankaliasname>
```

Dadurch wird bei jedem Befehl und jeder SQL-Abfrage, die eine Datenbankverbindung benötigt, ein impliziter CONNECT zur angegebenen Datenbank erstellt, falls keine Datenbankverbindung besteht, also explizit ein „CONNECT TO ...“ durchgeführt wurde.

5.5.2 Aktive Datenbanken und Verbindungen anzeigen

Um herauszufinden, welche Datenbanken – im aktuellen Exemplar – aktiv sind, führen Sie den Befehl LIST ACTIVE DATABASES aus:

```
D:\SQLLIB\BIN>db2 list active databases

           Aktive Datenbanken

Datenbankname           = BUCHDB
Aktuell verbundene Anwendungen = 2
Datenbankpfad           = C:\DB2\NODE0000\SQL00001\

Datenbankname           = TBC_MD
Aktuell verbundene Anwendungen = 2
Datenbankpfad           = D:\DB2\NODE0000\SQL00003\
```

Abbildung 5.13: list active databases-Befehl

Als „aktiv“ gelten dabei sowohl Datenbanken, die mit ACTIVATE DATABASE gestartet wurden, als auch diejenigen, die durch einen CONNECT-Befehl gestartet wurden. Eine Unterscheidung, durch welchen Befehl die Datenbank diesen Zustand erreicht, ist nicht möglich.

Die einzelnen Verbindungen zur Datenbank können Sie als DB2-(Exemplar-)Administrator über den Befehl LIST APPLICATIONS sehen.



Abbildung 5.14: list applications-Syntaxdiagramm

Dabei ist interessant, dass sich dieser Befehl auf die Exemplar-Ebene bezieht und so alle Datenbankverbindungen im aktuellen Exemplar aufzeigt, falls keine Einschränkung auf eine bestimmte Datenbank erfolgte.

```
D:\SQLLIB\BIN>db2 list applications
```

Ber.-Id	Anwendungs- name	Anwend.- kennzeich.	Anwendungs-ID	DB-Name	Anzahl Agenten
MTUSER	javaw.exe	97	*LOCAL.DB2.01AE41100437	BUCHDB	1
MTUSER	db2bp.exe	286	*LOCAL.DB2.006501081558	BUCHDB	1
DB2ADMIN	iwh2serv.exe	8	*LOCAL.DB2.016746163457	TBC_MD	1
DB2ADMIN	iwh2log.exe	7	*LOCAL.DB2.012F86163443	TBC_MD	1
DB2ADMIN	db2bp.exe	15	C0A83C2B.NE0C.013849195906	BUCHDB	1

Abbildung 5.15: list applications-Befehl

Jede Verbindung (connect) ist dabei eindeutig durch ein Anwendungskennzeichen (engl. application ID) identifiziert und kann einem Benutzer (über Ber.-Id) und einer Anwendung zugeordnet werden. Abbildung 5.15 zeigt beispielsweise in der ersten Zeile eine lokale Verbindung des Benutzers MTUSER zur Datenbank BUCHDB. Diese Verbindung hat das Anwendungskennzeichen „97“ und wird von der Anwendung javaw.exe (DB2-GUI) gehalten. Die Anwendung „286“ in der zweiten Zeile zeigt db2bp.exe als Anwendung. Dies ist der sog. Backend-Prozess des Befehlszeilenprozessors (command line).

Tipp:

Wenn mit LIST APPLICATIONS entfernte Verbindungen angezeigt werden, wie das in der letzten Zeile von Abbildung 5.15 der Fall ist, können Sie die IP-Adresse herausfinden, von der diese Verbindung ausgeht. Dazu müssen Sie den ersten Teil der Anwendungs-ID aus der Hexadezimaldarstellung ins Dezimalsystem umrechnen. In unserem Beispiel ergibt das: C0 A8 3C 2B = 192.168.60.43

5.5.3 Datenbankverbindungen beenden

Um eine Datenbankverbindung zu trennen, kann, wie bei Abbildung 5.16 in Punkt ① gezeigt, ein DISCONNECT-Befehl abgesetzt werden. Dabei wird nur die reine Datenbankverbindung getrennt; der Backend-Prozess (db2bp) aber bleibt bestehen, um für eine erneute Verbindung, eventuell auch zu einer anderen Datenbank, wieder zur Verfügung zu stehen.

Alternativ kann auch durch einen TERMINATE-Befehl eine Datenbankverbindung aufgehoben werden (Punkt ③). Neben der Verbindung wird dann auch der Backend-Prozess (db2bp) beendet – es handelt sich also um ein vollständiges Zurücksetzen des CONNECT.

Eine Datenbank wird inaktiv, wenn alle Verbindungen zu ihr beendet werden. Falls die Datenbank vorher durch ACTIVATE DATABASE aktiviert worden ist, muss zusätzlich ein DEACTIVATE DATABASE-Befehl (Punkt ②) abgesetzt werden, um die Datenbank inaktiv werden zu lassen. „Inaktiv“ bedeutet für die Datenbank einen „Ruhe-Zustand“, in dem die Speicherbereiche freigegeben und bereitgestellte Datenbankprozesse gestoppt werden.

Ein ACTIVATE DATABASE-Befehl kann nur mit einem DEACTIVATE DATABASE-Befehl rückgängig gemacht werden.

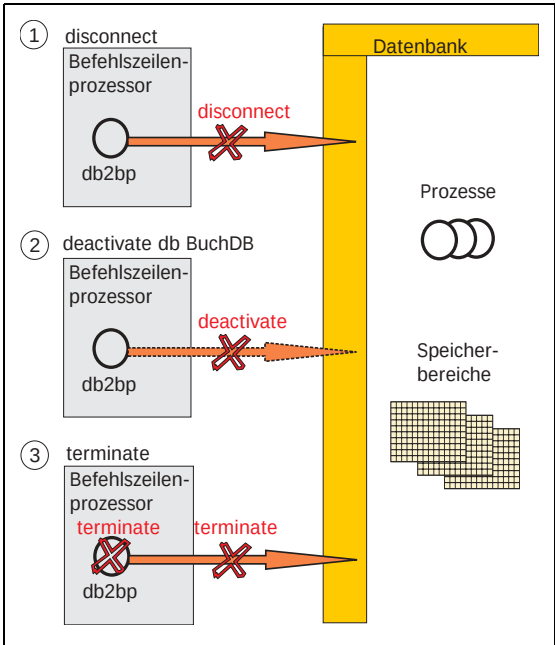


Abbildung 5.16: Datenbankverbindung beenden

Hinweis:

Die Trennung der Datenbankverbindung (siehe Abbildung 5.16) erfolgt aus der Umgebung heraus, aus der sie aufgebaut wurde, also in unserem Beispiel aus dem Befehlszeilenprozessor bzw. einer Anwendung. Dies setzt voraus, dass die Anwendung bzw. der Benutzer diese Befehle (TERMINATE, CONNECT RESET, DEACTIVATE) auf ihren bzw. seinen Systemen bzw. Datenbank-Clients absetzt.

Sie als DB2-Administrator können einen solchen Verbindungsabbruch aber auch erzwingen. Dafür steht Ihnen der FORCE APPLICATION-Befehl zur Verfügung.

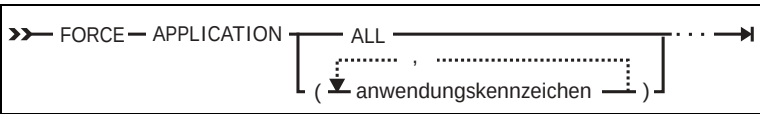


Abbildung 5.17: force application-Syntaxdiagramm

Das Anwendungskennzeichen ist die ID aus der „anwendungskennzeichen“-Spalte von Abbildung 5.15. So können Sie gezielt einzelne Benutzeraktionen bzw. Anwendungen abbrechen. Diese werden dann rückgängig gemacht (Rollback).

Besondere Vorsicht ist bei der Option `ALL` geboten, da diese alle Verbindungen zu allen Datenbanken im Exemplar beendet – durchaus also eine Gelegenheit, sich „Freunde“ zu schaffen.

5.6 Die Datenbankkonfiguration

Jede Datenbank kann individuell konfiguriert werden. Die Datenbankkonfiguration enthält neben Konfigurationsparametern auch informative Stati (z.B. den Datenbankstatus), die Rückmeldung über den Zustand der Datenbank geben.

Folgende Bereiche können in der Datenbankkonfiguration beeinflusst werden:

- ▶ Speicherbereiche der Datenbank
- ▶ Protokoll- und Wiederherstellungseinstellungen
- ▶ Konfiguration von Zusatz-Software
 - TSM-(Tivoli Storage Manager-)Parameter
 - Data Link-Parameter (DB2-Zusatzprodukt für Datei-Server-Verwaltung)
- ▶ Standardvorgaben

5.6.1 Anzeige der Datenbankkonfiguration

Die aktuellen Einstellungen für Ihre Datenbank können Sie über

```
get db cfg for <dbname>
```

abfragen. <dbname> ist dabei der Datenbank-Alias. Dieser muss angegeben werden, falls die Umgebungsvariable DB2DBDFT nicht gesetzt ist oder es keine bestehende Verbindung zur Datenbank gibt.

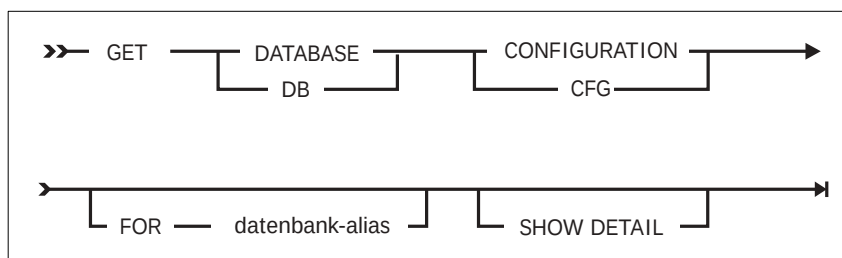


Abbildung 5.18: get database configuration-Syntaxdiagramm

Mit SHOW DETAIL bekommen Sie, neben dem aktuellen Wert, auch den sog. „verzögerten Wert“ angezeigt. Dieser wird relevant, falls Sie Parameter geändert haben, die nicht sofort wirksam werden können bzw. sollen. Somit ist es mit DB2 V8 möglich, neben dem aktuellen Wert auch den geänderten Wert zu sehen.

Die Abfrage sowie die Änderung der Konfiguration kann in der grafischen Oberfläche in der DB2-Steuerzentrale durchgeführt werden. Auf der Datenbank-Ebene wählen Sie auf der zu konfigurierenden Datenbank PARAMETER KONFIGURIEREN... aus dem Kontextmenü aus.

D:\SQLLIB\BIN>db2 get db cfg for BuchDB show detail

Konfiguration für Datenbank			
Beschreibung	Parameter	Aktueller Wert	Verzögerter Wert

Releasestand der Datenbankkonfiguration		= 0x0a00	
Releasestand der Datenbank		= 0x0a00	
Datenbankgebiet		= DE	
Codepage für Datenbank		= 819	
Codierter Zeichensatz der Datenbank		= ISO8859-1	
Landescode der Datenbank		= 49	
...			
Sicherung anstehend		= NO	
Datenbank ist konsistent		= YES	
Aktualisierende Wiederherstellung anstehend		= NO	
Wiederherstellung anstehend		= NO	
Zuordnung aus mehreren Seiten bestehender Datei aktiv		= NO	
Status der Protokollspeicherung für Wiederherstellung		= NO	
Status des Benutzerexits für Protokollierung		= NO	
...			
Zwischenspeicher der Datenbank (4 KB)	(DBHEAP)	= 1200	1200
Größe des gemeinsamen Datenbankspeichers (4 KB)	(DATABASE_MEMORY)	= AUTOMATIC(7488)	AUTOMATIC(7488)
Katalogcachelgröße (4 KB)	(CATALOGCACHE_SZ)	= (MAXAPPLS*4)	(MAXAPPLS*4)
Protokollpuffergröße (4 KB)	(LOGBUFSZ)	= 8	8
Zwischenspeicher für Dienstprogramme (4 KB)	(UTIL_HEAP_SZ)	= 5000	5000
...			

Abbildung 5.19: get database configuration show detail-Befehl

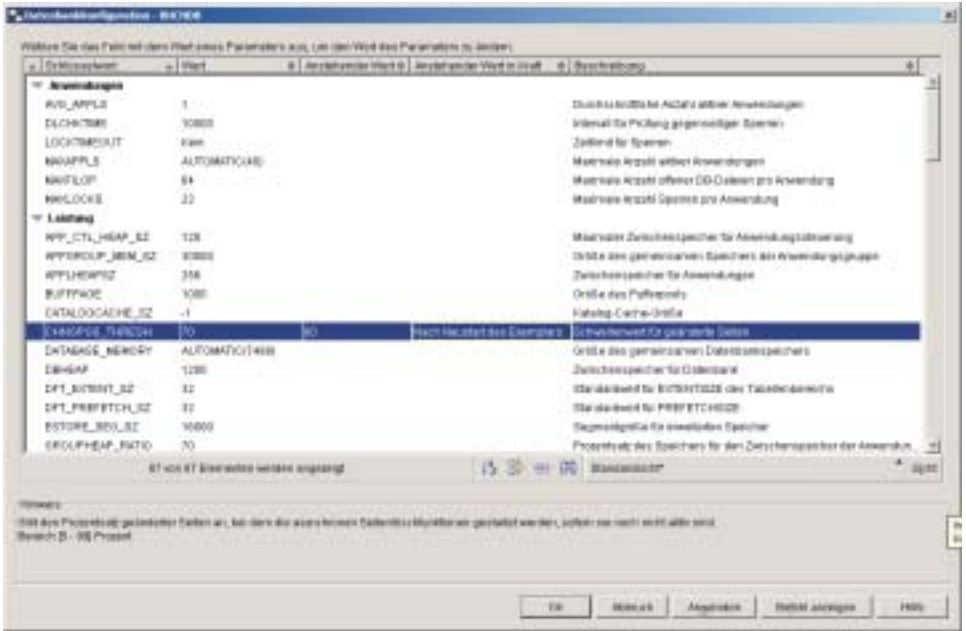


Abbildung 5.20: Datenbankkonfiguration in der Steuerzentrale

Neben dem Wert und dem sog. ANSTEHENDEN WERT ist in der grafischen Oberfläche auch der Zeitpunkt zu sehen, wann die neue Konfiguration wirksam wird (ANSTEHENDER WERT IN KRAFT).

Typen von Datenbankparametern

Folgende Konfigurationsparameter-Typen können in der Datenbankkonfiguration unterschieden werden:

► Informelle Parameter

Sie geben nur eine Statusinformation wieder und können nicht direkt geändert werden. Dazu gehören beispielsweise: SICHERUNG ANSTEHEND oder PFAD ZU DEN PROTOKOLLDATEN.

► Änderbare Konfigurationsparameter

- ohne Vorbelegung – wie z.B. NEWLOGPATH
- mit AUTOMATIC-Wert – dabei wird der Wert automatisch vom Datenbanksystem gesetzt, z.B. DATABASE_MEMORY
- mit errechnetem Wert – dabei hängt der Wert von einer Formel bzw. einem anderen Konfigurationsparameter ab. Ein Beispiel dafür ist CATALOGCACHE_SZ = (MAXAPPLS * 4).

5.6.2 Ändern der Datenbankkonfiguration

Anpassungen der Datenbankkonfiguration sind nötig, um die Standardwerte an Ihre individuelle Umgebung und Anforderungen anzupassen. Die Standardwerte der Datenbankkonfiguration sind für die „Sample“-Datenbank ausreichend und sind absichtlich so ausgelegt, dass keine Systemressourcen (hauptsächlich Speicherplatz) unnötig verschwendet werden. Eine Anpassung bestimmter Parameter für Ihre Produktionsdatenbanken ist somit unerlässlich.

In DB2 V8 werden viele Parameteränderungen sofort, d.h. im laufenden Betrieb, wirksam. Einige Parameter jedoch erfordern nach wie vor einen „Neustart“ der Datenbank, d.h. ein neues „Aktivwerden“ mit beispielsweise einem neuen ersten CONNECT. Da es keinen „Stop“-Befehl für die Datenbank gibt, müssen bestehende Verbindungen per CONNECT RESET oder TERMINATE beendet werden. Die momentan wirksame Konfiguration und die, die beim nächsten Start der Datenbank wirksam wird, sehen Sie bei der Abfrage der Konfiguration durch die SHOW DETAIL-Option. Diese Option erfordert allerdings eine Verbindung zur Datenbank.

Hinweis:

Um über die DB2-Steuerzentrale Änderungen vorzunehmen, wählen Sie PARAMETER KONFIGURIEREN... im Kontextmenü auf der Datenbank-Ebene.

Zum Ändern eines Eintrags klicken Sie diesen an und wählen den Button „...“ aus, der nun erscheint. Im Fenster DATENBANKKONFIGURATIONSPARAMETER ÄNDERN können Sie dann den neuen Wert eingeben.

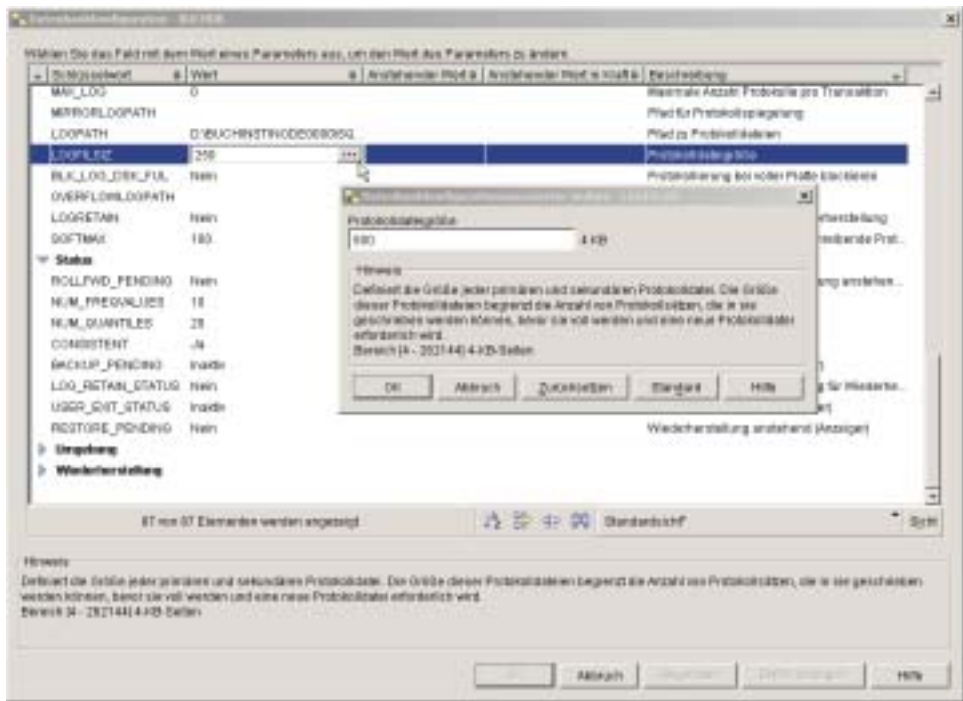


Abbildung 5.21: Steuerzentrale – Datenbankkonfiguration ändern

In der grafischen Oberfläche bekommen Sie zusätzliche nützliche Hinweise über die Einheit des zu ändernden Parameters, seine Bedeutung und den möglichen Wertebereich.

Eine Konfigurationsänderung können Sie auch über einen UPDATE DB CFG-Befehl vornehmen.

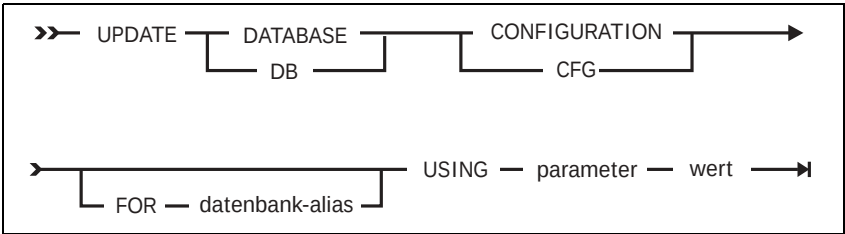


Abbildung 5.22: update database configuration-Syntaxdiagramm

Die Änderung aus Abbildung 5.21 würde beispielsweise folgendermaßen aussehen:
update db cfg for BuchDB using logfilsiz 500

Die Hinweise zu den Parametern sind – anders als beim GUI – hier nicht verfügbar.

5.6.3 Der Konfigurationsadvisor

Um es Ihnen zu erleichtern, die passende Datenbankkonfiguration für Ihre Anforderungen zu finden, ist in DB2 ein Konfigurationsadvisor enthalten. Wenn Sie acht grundsätzliche Fragen beantworten, z.B. über die Anzahl der Transaktionen und den für DB2 zur Verfügung stehenden Hauptspeicher, ermittelt ein komplexer Algorithmus eine Konfigurationsempfehlung. Diese empfohlenen Einstellungen können entweder in der Taskzentrale gespeichert oder sofort angewandt werden.

Sie erreichen diesen Advisor über den Punkt KONFIGURATIONSDADVISOR... des Kontextmenüs auf der Datenbank-Ebene.



Abbildung 5.23: Konfigurationsadvisor – Schritt 1 – Einführung

Folgende Fragen müssen beantwortet werden:

- ▶ Wie viel Hauptspeicher soll DB2 nutzen dürfen?
- ▶ Haben Sie aufwändige Abfragen (z.B. für komplexe Datenauswertungen), einfache Transaktionen oder einen Gemischtbetrieb?
- ▶ Wie viele Anweisungen pro Transaktion haben Sie etwa und wie viele pro Minute?
- ▶ Liegt Ihre Priorität auf einer hohen Transaktionsleistung, einer schnellen Wiederherstellbarkeit oder beidem?
- ▶ Enthält Ihre Datenbank bereits den vollen Umfang der Datenmenge?
- ▶ Mit wie vielen lokalen und entfernten Verbindungen rechnen Sie durchschnittlich?
- ▶ Welche Isolationsstufe wird verwendet?

Sollten Sie einige Fragen nicht beantworten können, wie z.B. die nach der Anzahl Ihrer Transaktionen in der Minute, kann Ihnen ein DB2-Monitor diese Werte liefern. Die Überwachung von DB2 behandeln wir in Kapitel 14.

Die Ergebnisse des Konfigurationsadvisors sind beachtlich. Trotz dieser wenigen, einfach gehaltenen Fragen werden durchaus gute Ergebnisse erzielt. Diese reichen zwar nicht an die eines erfahrenen Administrators bzw. Beraters hin, stellen aber eine brauchbare Ausgangsbasis dar, die der Standardkonfiguration weit überlegen ist.

Auch von der Befehlszeile aus steht uns dieses mächtige Hilfsmittel per `AUTOCONFIGURE`-Befehl zur Verfügung.

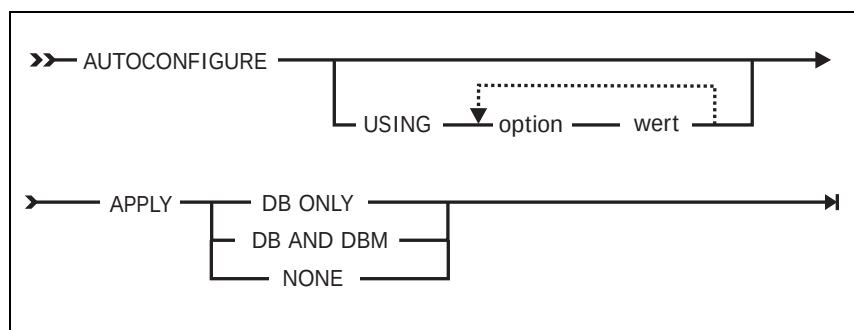


Abbildung 5.24: `autoconfigure`-Syntaxdiagramm

Dies ist besonders insofern interessant und nützlich, als damit eine Einbindung in Scriptdateien möglich wird. Auch direkt im `CREATE DATABASE`-Befehl gibt es bereits eine `AUTOCONFIGURE`-Option. Somit lassen sich Scripts für die automatische Datenbankerstellung auf entfernten Rechnern erzeugen, die die Datenbank optimal an die individuelle Hardware-Umgebung anpassen. Wird z.B. DB2 als Datenhaltungsgrundlage für andere Software benutzt, wird so ein automatisches Grund-Tuning der erzeugten Datenbank möglich.

Beispiel:

Abschnitt 5.7 „Partitionsgruppen“ zeigt die automatische Konfiguration einer Datenbank eines DB2-Servers. Die Datenbank soll dabei nur 40% des Hauptspeichers nutzen dürfen, um für andere Anwendungen oder andere Datenbanken auch noch Hauptspeicher zur Verfügung zu haben. Es wird mit ca. 100 Transaktionen pro Minute gerechnet, wobei diese transaktionsorientierte Last – mit durchschnittlich acht Befehlen – kleine Transaktionen verarbeitet. Alle Daten wurden bereits in die Datenbank geladen. Es werden durchschnittlich zwei lokale und sechs entfernte Verbindungen erwartet. Die Anwendungen sind größtenteils mit Lesestabilität (RS) gebunden (doch dazu später mehr).

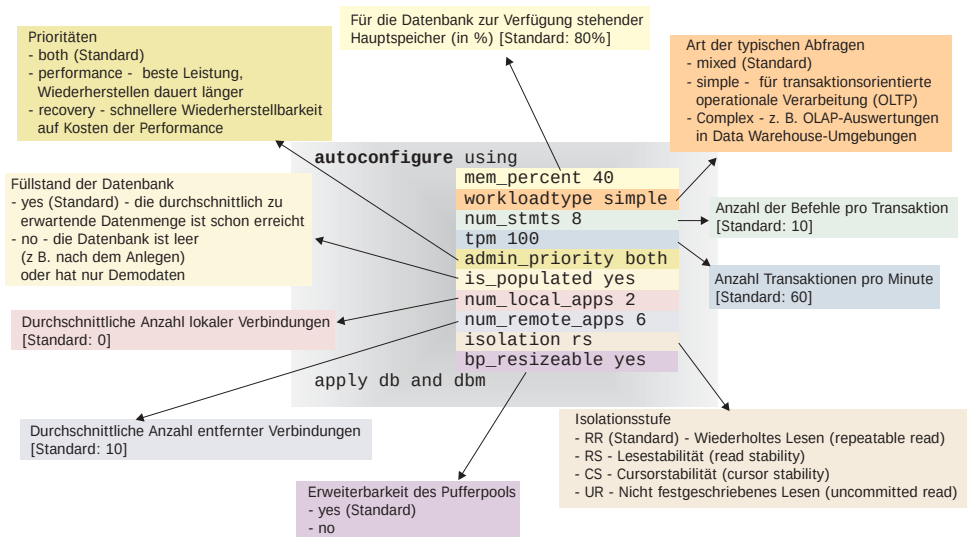


Abbildung 5.25: autoconfigure – Beispiel

5.7 Partitionsgruppen

In nicht-partitionierten DB2-Umgebungen spielen die Partitionsgruppen keine Rolle und werden auch nicht im GUI angezeigt.

Eine Partitionsgruppe kann eine, mehrere oder alle Partitionen umfassen. In der Partitionskonfigurationsdatei db2nodes.cfg sind alle verfügbaren Partitionen definiert. Die Gruppierung von Partitionen dient der einfacheren Verwaltung und vor allem der besseren Verteilung der Daten über die zusammengeschlossenen Partitionen.

Dabei ist eine Partitionsgruppe ein logisches Objekt, das immer einer Datenbank zugeordnet ist.

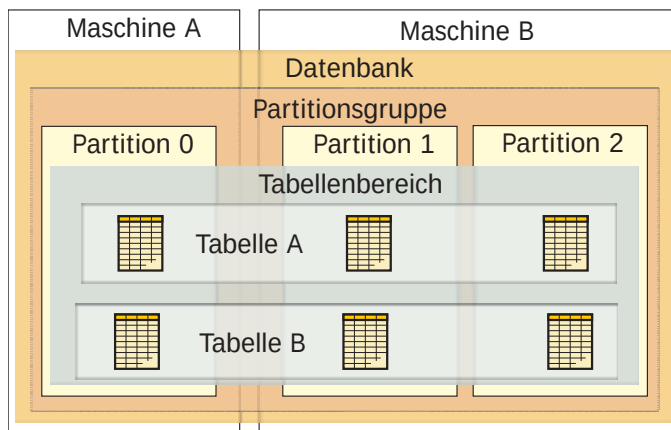


Abbildung 5.26: Die Partitionsgruppe in der Datenbankhierarchie

5.8 Die Tabellenbereiche

Durch das Anlegen der Datenbank sind drei Standard-Tabellenbereiche erstellt worden:

► SYSCATSPACE

Hier werden die Systemtabellen – also die Metadaten – der Datenbank gespeichert, mit Informationen über die Datenbankobjekte und Zugriffsrechte. Diese Tabellen werden vom Datenbanksystem verwaltet und können weder explizit erstellt noch gelöscht werden. Durch die Abfrage der Inhalte können Sie aber wichtige Strukturinformationen über Ihre Datenbank bekommen.

► TEMPSPACE1

Dieser Tabellenbereich wird für temporäre Daten und Tabellen benötigt, wie sie beispielsweise bei Reorganisationen, Sortierungen und anderen Verwaltungsaufgaben anfallen können, bei denen die definierten Speicherbereiche zu klein sind.

► USERSPACE1

In diesem Bereich werden die Objekte der Benutzer abgelegt, falls kein zusätzlicher Tabellenbereich erstellt wurde.

Daraus können Sie bereits erkennen, dass Objekte, wie z.B. Tabellen und Indizes, in der Datenbank Tabellenbereichen zugeordnet sind.

Wieso aber gibt es diese Tabellenbereiche, und warum legt man Tabellen nicht einfach so in der Datenbank ab?

Die Antwort: Als System-Administrator müssen Sie bestimmen können, auf welche Festplatten bzw. Bereiche die Tabellen abgespeichert werden. Mit der Tabellenbereichsdefinition wird eine gezielte Verteilung der Daten auf die physikalischen Speichermedien möglich. Durch eine gezielte Verteilung kann ein paralleler Zugriff ermöglicht und so die Leistung gesteigert werden. Der Tabellenbereich ist eine Speicherstruktur, die die „Verbindung“ von den logischen Objekten (Datenbank, Tabellen etc.) zu den physikalischen Objekten (Festplatten) darstellt.

Diese physikalische Struktur stellt – zusammen mit einem guten Datenmodell – das Fundament einer Datenbank dar. Wie bei einem Haus ist es sinnvoll, ein solides Fundament zu erstellen, da später, wenn das Haus steht, eine Änderung im Fundament sehr schwierig ist. Denken Sie dabei nur an den Schiefen Turm von Pisa. Dort hat selbst das Ausgleichen der Schiefe in einem der oberen Stockwerke die Schräglage nicht langfristig beseitigen können, sondern es ist eine aufwändige und teure Reparatur am Fundament notwendig geworden. Dies lässt sich direkt auf Datenbanken übertragen. Dass der Turm von Pisa durch die Schräglage eine Sehenswürdigkeit geworden ist, ist eine andere Sache.

5.8.1 Der Aufbau von Tabellenbereichen

Eine Datenbank enthält mehrere Tabellenbereiche. Neben den Standard-Tabellenbereichen (SYSCATSPACE, TEMPSPACE1 und USERSPACE1) kann der Administrator beliebige weitere Tabellenbereiche erstellen. Genauer betrachtet besteht eine Datenbank aus

mehreren Partitionen, und die Tabellenbereiche liegen ihrerseits in einer Partitionsgruppe (siehe Abbildung 5.26). In Einzelpartitionsumgebungen ist diese Zwischenstufe jedoch nicht relevant.

Die Daten eines Tabellenbereichs werden, wenn sie von einer Anwendung gelesen oder verändert werden, in den Hauptspeicher geladen und dort in einem Pufferpool (dem Datenbank-Cache) gehalten. Nur in diesem Pufferpool – es kann mehrere pro Datenbank geben – werden Daten geändert. Standardmäßig existiert ein Pufferpool (engl. bufferpool), und zwar der IBMDEFAULTBP. Mehrere Tabellenbereiche können einen Pufferpool benutzen, aber ein Tabellenbereich nutzt immer genau einen Pufferpool.

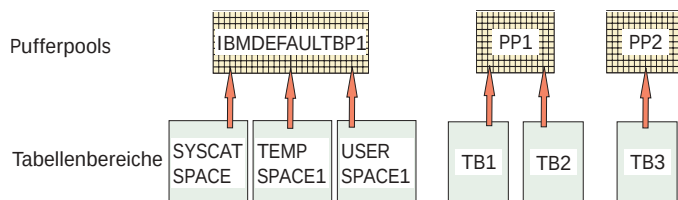


Abbildung 5.27: Zuordnung von Tabellenbereichen zu Pufferpools

Häufig stellt der Datentransfer – Eingabe/Ausgabe (E/A) oder im Englischen Input/Output (I/O) – von den Platten zum Hauptspeicher und umgekehrt einen Engpass dar. Die Zugriffszeiten auf die Platten liegen im Bereich von wenigen Millisekunden ($1 \text{ ms} = 10^{-3} \text{ s} = 0,001 \text{ s}$; in dieser Zeit legt das Licht ca. 300 km zurück). Im Vergleich zur Geschwindigkeit, mit der der Prozessor Daten verarbeiten kann, und zur Geschwindigkeit des Hauptspeicherzugriffs – diese beträgt einige Nanosekunden ($1 \text{ ns} = 10^{-9} \text{ s} = 0,000000001 \text{ s}$; in dieser Zeit legt selbst das Licht nur 30 cm zurück!) – ist dies sehr langsam. Diesen Engpass kann man durch den Einsatz mehrerer Festplatten und durch die Nutzung von Parallelität (paralleler E/A) entschärfen.

Verteilt man nun mittels Tabellenbereichen die Daten auf mehrere Festplatten, kann man durch die Nutzung von mehreren E/A-(I/O-)Prozessen den Schreib-Lese-Durchsatz wesentlich erhöhen und so maßgeblich zur Leistungssteigerung beitragen.

Der Begriff der *Verteilung* ist dabei entscheidend. Bei der Zuordnung einer Tabelle bzw. deren Daten auf eine Platte würde bei einer E/A-Anforderung einer Anwendung nur eine Festplatte arbeiten. Der Konjunktiv zeigt Ihnen schon, dass die Verteilungsmechanismen in den Tabellenbereichen nicht so arbeiten, sondern dass bessere Konzepte zum Einsatz kommen.

Abbildung 5.28 zeigt einen einzelnen Tabellenbereich in einer Datenbank und drei Festplatten: links die logischen Objekte, rechts die physikalischen Objekte. Der Tabellenbereich ist auf jeder Platte einem Behälter (Container) zugeordnet. In diesen Behälter werden die Objektdaten abgelegt. Unter Performance-Gesichtspunkten ist für einen Tabellenbereich ein Behälter pro Platte sinnvoll.

Eine gleichmäßige Verteilung der Tabellendaten, die für den parallelen Zugriff so wichtig ist, wird erst durch die so genannten *Speicherbereiche* (extents) möglich. Die Daten der Tabelle TabX (siehe Abbildung 5.28) werden „häppchenweise“ auf die

zugehörigen Behälter verteilt. Die Häppchen – also die DB2-Speicherbereiche – sind dabei im Bild durchnummeriert, um zu zeigen, dass die Daten „reihum“ (das Fachwort im Englischen hierfür heißt „round-robin“) verteilt werden. So wird eine relativ gleichmäßige Verteilung erreicht, die auch für eine einzelne Tabelle im Tabellenbereich einen parallelen Zugriff erlaubt.

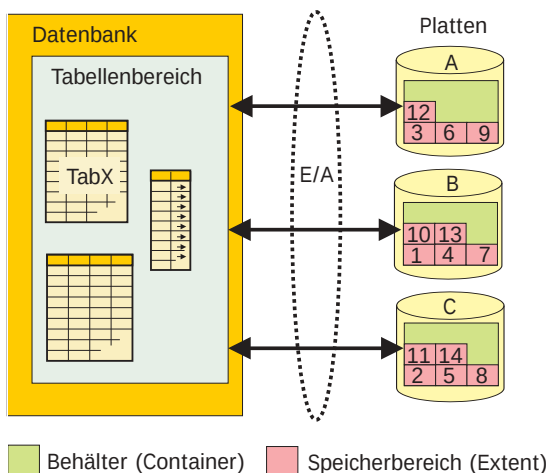


Abbildung 5.28: Physikalische Verteilung von Tabellenbereichsdaten

Der parallele Zugriff bedeutet, dass mehrere Plattenarme gleichzeitig positioniert werden können. Mehrere Lese-/Schreib-Köpfe der Festplatten können gleichzeitig Daten auslesen, und mehrere E/A-Prozesse können diese Daten in den Hauptspeicher, genauer in den Pufferpool der Datenbank, übertragen. In der gleichen Zeit können so, durch die Nutzung der Parallelität, mehr Daten übertragen werden, die dann, wenn sie benötigt werden, direkt aus dem Hauptspeicher gelesen werden können. Ein deutlicher Leistungsgewinn!

Im Beispiel aus Abbildung 5.28 können so – bei einer Leseanforderung für Inhalte der TabX – die Platten A, B und C gleichzeitig arbeiten, und jede Platte muss nur einen Teil der benötigten Tabellendaten zur Verfügung stellen.

Den Aufbau eines Behälters (container) zeigt Abbildung 5.29. Auch hier lässt sich sehr schön die Kette mit den 1:n-Beziehungen fortsetzen, die Sie schon bei der Beziehung zwischen Exemplar, Datenbank und Tabellenbereich gesehen haben (siehe Abbildung 5.1). Ein Behälter besteht aus mehreren Speicherbereichen (extents), die sich ihrerseits aus (Speicher-)Seiten (pages) zusammensetzen. Auf diesen Seiten sind, neben DB2-Verwaltungsinformationen, die Zeilen der Tabelle abgelegt.

Ein Speicherbereich – im Weiteren auch mit dem engl. Fachbegriff „Extent“ bezeichnet – ist also die *Einheit der Verteilung* der Daten über die Behälter. Die Standardgröße eines Extents sind 32 Seiten – bei der Standardseitengröße von 4 Kbyte ergibt das 128 Kbyte. In der Datenbankkonfiguration können Sie mit dem Parameter DFT_EXTENT_SZ diesen Standard abändern. Die Vorgabe „32“ ist aber für die meisten Anforderungen passend und performant.

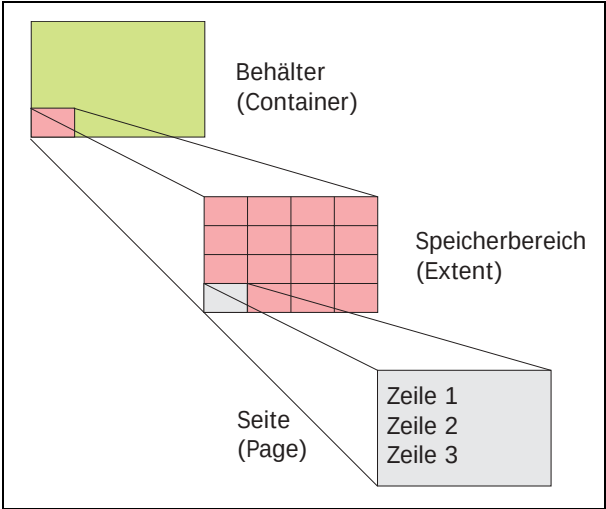


Abbildung 5.29: Aufbau eines Behälters

Größe der Speicherseiten

Speichergrößen werden in DB2 immer in der Einheit „Seiten“ angegeben. Der Standardwert für die Seitengröße (pagesize) ist 4 Kbyte. Es können aber auch andere Seitengrößen gewählt werden – zur Auswahl stehen, außer den 4-Kbyte-Seiten, auch 8, 16 oder 32 Kbyte.

Verschiedene Limitierungen sind von der gewählten Seitengröße abhängig:

Seitengröße	max. Spaltenanzahl	max. Zeilenlänge in Byte	max. Tabellengröße pro Partition in Gbyte
4	500	4.005	64
8	1.012	8.101	128
16	1.012	16.293	256
32	1.012	32.677	512

Tabelle 5.1: Tabelle mit DB2-Seitengrößen

Die Festlegung der Seitengröße geschieht auf Tabellenbereichsebene beim Befehl CREATE TABLESPACE. Das heißt, ein Tabellenbereich hat einerseits immer eine feste, nicht mehr veränderbare Seitengröße, andererseits kann es in einer Datenbank – durch mehrere Tabellenbereiche – mehrere verschiedene Seitengrößen nebeneinander geben.

DB2-E/A-Zugriffe erfolgen seitenweise (page) oder speicherbereichsweise (extent), d.h., auch wenn nur ein Teil einer Zeile von einer Anwendung benötigt wird, wird die ganze zugehörige Seite gelesen.

Die Entscheidung, welche Seitengröße zu wählen ist, hängt – wie so oft – von verschiedenen Faktoren ab:

- ▶ Anforderungen mit einem Größenbedarf, die eine bestimmte Seitengröße notwendig machen
- ▶ Art der Datenbankabfragen:
 - Analyseorientierte Abfragen und Anwendungen, wie sie in Data Warehouse-Umgebungen vorkommen, greifen – typischerweise – auf viele aufeinander folgende Daten zu. Es werden also viele Zeilen für die Verarbeitung benötigt. Größere Seiten können hier einen effizienteren E/A-Zugriff möglich machen, da weniger Leseanforderungen notwendig sind als bei kleinen Seiten.
 - Transaktionsorientierte Anwendungen (OLTP) benötigen durch ihre wahlfreien Zugriffsanforderungen meist nur wenige aufeinander folgende Daten. Eine eher kleinere Seitengröße verhindert, dass zu viele Daten in den Hauptspeicher gelesen werden, die dort zwischengespeicherte Daten überschreiben und dann selbst später eventuell nicht benötigt werden.
- ▶ Verhinderung von Platzverschwendung durch die Einschränkung, dass eine Speicherseite im DB2 maximal 255 Zeilen beinhalten kann. Auf jeder Seite mit Tabellendaten sind 76 Byte für DB2-Verwaltungsinformationen reserviert; auf einer 4-Kbyte-Seite stehen somit 4.020 Byte für Daten zur Verfügung. Ist die Seitengröße sehr groß gewählt (oder sind die Zeilen sehr „kurz“), wäre jeglicher Speicherplatz, der von 255 Reihen auf einer Seite nicht benutzt wird, verschwendet.

Ein zusätzlich zu beachtender Punkt bei Seitengrößen über 4 Kbyte ist, dass einem Tabellenbereich immer ein Pufferpool mit identischer Seitengröße zugeordnet werden muss. Es empfiehlt sich, pro Seitengröße zusätzlich jeweils einen temporären Tabellenbereich mit entsprechender Seitengröße anzulegen.

Der folgende Ablauf zeigt die ideale Vorgehensweise und Reihenfolge, wenn Seitengrößen ungleich der Standardgröße von 4 Kbyte benötigt werden – sozusagen Ihr „Kochbuch“ für ein „Seitengröße-Menü“:

1. Anlegen eines Pufferpools mit gewünschter Seitengröße (hier 16 Kbyte)

```
create bufferpool bp_16k size 10000 pagesize 16K
```

2. Anlegen des zugehörigen Tabellenbereichs

```
create tablespace tbsms_16k pagesize 16K managed by system  
using('/disk1/db/tbsms_16k/b1', '/disk2/db/tbsms_16k/b2')  
bufferpool bp_16k
```

3. Anlegen des zugehörigen temporären Tabellenbereichs

```
create temporary tablespace temptbsms_16k pagesize 16K managed by system  
using('/disk3/db/temptbsms_16k/b1', '/disk4/db/temptbsms_16k/b2')  
bufferpool bp_16k
```

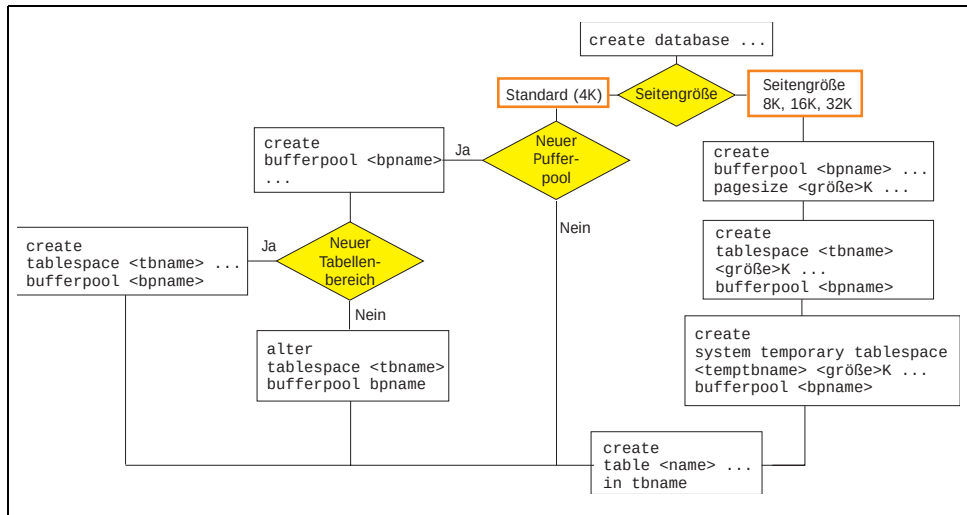


Abbildung 5.30: Sinnvolle Abfolge beim Aufbau einer Datenbank

Wird der Pufferpool als Erstes erzeugt, kann in Schritt 2 der Tabellenbereich gleich diesem zugeordnet werden. So kann ein zusätzlicher ALTER TABLESPACE-Befehl gespart werden, der sonst die Pufferpool-Zuordnung umsetzen müsste. Mehrere temporäre (USER- oder SYSTEM-) Tabellenbereiche mit gleicher Seitengröße sind nicht sinnvoll, da immer nur einer (gleichzeitig) verwendet wird. Mehrere Behälter für die temporären Tabellenbereiche können dagegen schon die Leistung steigern.

Fassen wir also zusammen: Um eine gute Leistung zu erreichen, werden die Tabellen, Indizes und LOBs Tabellenbereichen zugeordnet, deren Verteilung speicherbereichsweise auf die Behälter erfolgt. Je mehr Behälter einem Tabellenbereich zugewiesen werden, desto höher ist die erreichbare Parallelität für die E/A und somit die Performance. Dies setzt allerdings voraus, dass die Behälter auf verschiedenen Platten liegen und – idealerweise – die Platten auch an mehrere Festplatten-Controller angeschlossen sind, um einen „Flaschenhals“ dort zu vermeiden.

Was sind die Behälter physikalisch gesehen?

Die Antwort lautet: Es kommt darauf an! Diese Aussage führt in Schulungen häufig zu einem Lächeln oder kleinen Seufzer der Teilnehmer, ist aber wie so oft die einzig richtige Antwort.

In DB2 gibt es zwei verschiedene Arten von Tabellenbereichen – SMS und DMS. Je nach Art ist ein Behälter ein Verzeichnis oder eine Datei bzw. eine unformatierte (Raw-)Einheit. Es kommt also auf die Tabellenbereichsart an.

Die Verteilung der Daten per Speicherbereich (extent), abwechselnd über die Behälter, ist jedoch generell gültig – unabhängig von der Tabellenbereichsart!

Der obere Teil der Abbildung 5.31 zeigt einen Behälter eines SMS-Tabellenbereichs. Der untere Teil – ein DMS-Tabellenbereich – lässt die Wahl zwischen Dateibehältern und unformatierten (RAW-)Einheiten. Diese sind unformatierte Partitionen (Windows) bzw. logische Platten (volumes) ohne Dateisystem auf Unix-Systemen.

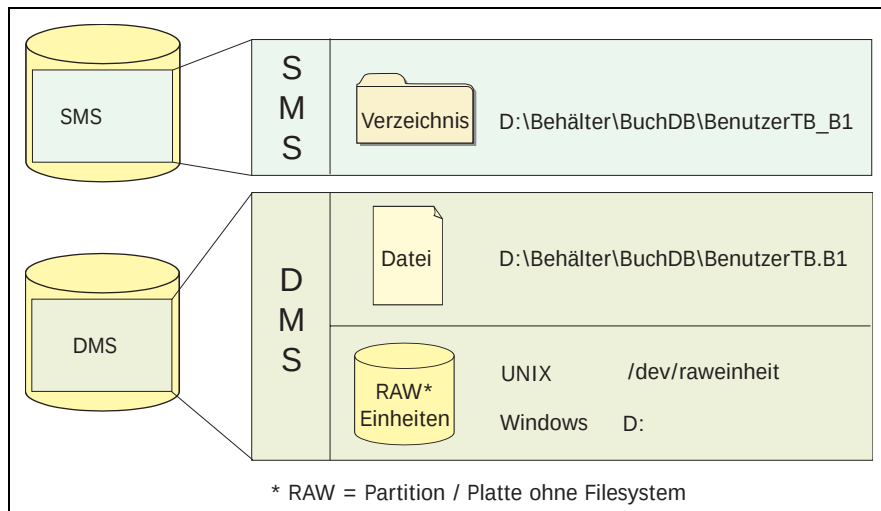


Abbildung 5.31: Behältertypen

Hinweis:

Behälter können im DB2 nicht auf Netzwerklaufwerke, NFS-gemountete Pfade oder GPFS-(General Parallel File System) Dateisysteme gelegt werden!

In einer Datenbank werden in der Regel DMS- und SMS-Tabellenbereiche verwendet. Bevor wir uns überlegen, welche Art von Tabellenbereichen warum eingesetzt werden soll, müssen wir erst die Unterschiede herausarbeiten.

5.8.2 Vom Betriebssystem verwaltete Tabellenbereiche – SMS

SMS – „System Managed Space“ – oder eben „vom Betriebssystem verwalteter Platz“ stellt die Standard-Tabellenbereichsart dar. Es werden bei der Datenbankerstellung alle Tabellenbereiche mit diesem Typ erstellt, falls es im Befehl nicht extra anders angegeben wird.

SMS zeichnet sich durch eine dynamische Speicherplatzverwaltung aus. Die Daten werden in verschiedenen Dateien in den Verzeichnisbehältern gespeichert. Das Betriebssystem erweitert dabei den Speicherplatz der Dateien, falls deren Datenmenge zunimmt, und verringert den belegten Platz, falls Daten aus den Objekten gelöscht werden. Somit ist nur der aktuell benötigte Speicherplatz belegt. Die Größe eines SMS-Behälters wächst somit dynamisch bis an die Grenze des Dateisystems bzw. der Platte.

Die einzelnen Dateien können im Verzeichnis aufgelistet werden. Für jede Tabelle und jeden Index gibt es mindestens eine Datei. Mit Hilfe der Systemtabellen bzw. System-Sichten ist es möglich, die Dateinamen bestehenden Objekten in der Datenbank zuzuordnen (vgl. Abbildung 5.32).

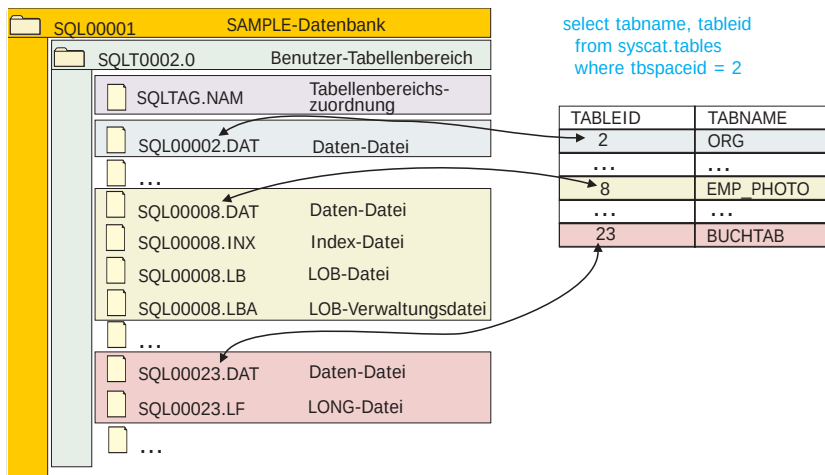


Abbildung 5.32: Verzeichnis eines SMS-Tabellenbereichs

5.8.3 Vom Datenbankmanager verwaltete Tabellenbereiche – DMS

DMS steht für „Database Managed Space“ – übersetzt „vom Datenbankmanager verwalteter Bereich“. Hier erzeugt also DB2 eigene Verwaltungsstrukturen und verlässt sich nicht auf das Betriebssystem bzw. das Dateisystem.

Bei DMS-Tabellenbereichen wird bereits beim Anlegen der Platz für die Behälter fest reserviert. Damit muss beim CREATE TABLESPACE-Befehl auch eine Größenangabe für die einzelnen Behälter mitgegeben werden. Dieser definierte Speicherbereich auf der Platte ist dann belegt, unabhängig davon, ob der Inhalt mit Daten gefüllt ist oder nicht. Diese Zuordnung (pre-allocation) der Speicherressourcen zu den Behältern ergibt einen Leistungsvorteil, besonders dann, wenn zusätzliche Daten eingefügt werden. In dieser Situation können die Daten sofort in den Tabellenbereich geschrieben werden, und es muss nicht – wie bei SMS – zuerst zusätzlicher Platz vom Dateisystem angefordert werden.

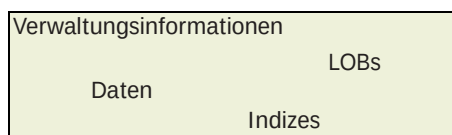


Abbildung 5.33: DMS-Behälter

Anders als bei SMS können auch die einzelnen Strukturen (Daten, Indizes und LOBs) eines DMS-Behälters nicht mit Mitteln des Betriebssystems angezeigt werden. Im Dateisystem sieht man nur eine große Datei (siehe Abbildung 5.33), falls es sich um einen Dateibehälter handelt. Im Falle einer unformatierten Einheit (raw device) als Behälter hat man keinerlei Zugriffsmöglichkeit über das Dateisystem, da „unformatiert“ genau das Fehlen des Dateisystems bezeichnet.

Die Performance-Vorteile aus der Vorreservierung des Speicherplatzes und der DB2-eigenen Verwaltungsstrukturen betragen ca. 10% bis 15% bei größeren Datenmengen. Der Wegfall des Dateisystems bei unformatierten Einheiten hat zweierlei leistungsrelevante Auswirkungen:

- ▶ Durch das Wegfallen des Dateisystemaufrufs, wie er beim Dateizugriff nötig ist, ergibt sich noch mal eine Leistungssteigerung von ca. 5% bis 10%.
- ▶ Der Dateisystem-Cache wird nicht mehr genutzt. Dadurch können negative Effekte entstehen, vor allem für LONG-Daten, die DB2-seitig nicht zwischengespeichert („gecached“) werden.

Zusätzliche Leistungsvorteile können durch die Verteilung von Daten, Indizes und LONG-Objekten auf verschiedene DMS-Tabellenbereiche erzielt werden. Beim Anlegen der Tabelle ist es möglich, wie im Folgenden gezeigt, eine solche Aufteilung zu erzielen:

```
create table Mitarb
( MitNr    integer not null,
  MitName  char (30),
  VgNr     integer not null,
  Bild     BLOB (1 M)
)
in         TB_DMS_Daten
index in   TB_DMS_Index
long in    TB_DMS_LOBS;
```

Abbildung 5.34: create table-Befehl mit Verteilung auf mehrere DMS-Tabellenbereiche

Somit ergibt sich die Möglichkeit, nicht nur die im Tabellenbereichskonzept gezeigte Parallelität beim Behälterzugriff, sondern eine zusätzliche Parallelität auf der Ebene der Daten- und Indexzugriffe zu nutzen.

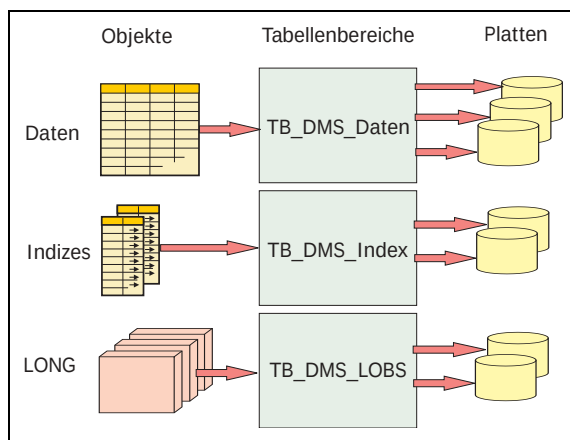


Abbildung 5.35: Verteilung von Daten, Indizes und LONG bei DMS-Tabellenbereichen

Abbildung 5.35 zeigt die durch den CREATE TABLE-Befehl (siehe Abbildung 5.34) erzeugte Verteilung auf mehrere DMS-Tabellenbereiche. Eine Aufteilung kann nur

nach diesen Strukturen erfolgen, d.h., die Daten einer Tabelle sind immer in einem Tabellenbereich, zugehörige LONG-Objekte sind – wenn gewünscht – in einem eigenen Tabellenbereich, und auch alle Indizes zu einer Tabelle sind immer in einem Tabellenbereich zu finden.

5.8.4 Wie erstelle ich einen Tabellenbereich?

Die ersten drei Tabellenbereiche werden beim Anlegen der Datenbank kreiert. Sie können mit dem CREATE TABLESPACE-Befehl zu jeder Zeit eigene Tabellenbereiche hinzufügen.

Das Syntaxdiagramm sieht folgendermaßen aus:

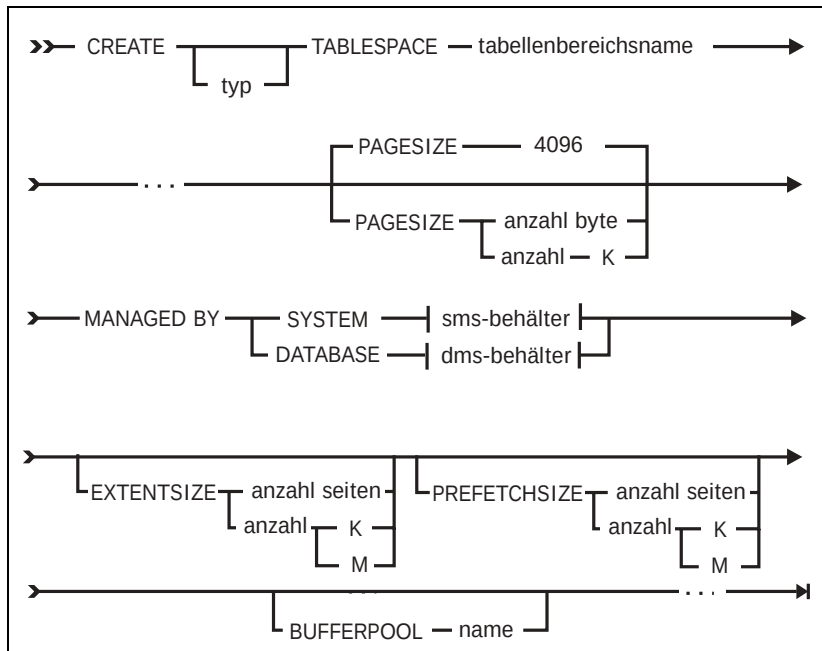


Abbildung 5.36: create tablespace-Syntaxdiagramm

Es gibt verschiedene Typen von Tabellenbereichen – damit sind jetzt nicht die Arten SMS und DMS gemeint –, sondern der Typ bezieht sich auf die Art der Daten, die im Tabellenbereich gespeichert werden sollen. Je nach Datenart sind die Verwaltungsstrukturen des Tabellenbereiches optimiert. Folgende Typen stehen zur Auswahl:

► **REGULAR**

Dieser Tabellenbereich enthält Benutzerdaten und ist der Standardtyp.

► **LARGE**

Die Verwaltungsstruktur dieses Typs ist speziell auf die Anforderungen dieser großen Objekte ausgelegt. LONG-Daten, LOBs und Indexdaten können hier gespeichert werden.

Dieser Typ ist nur für DMS-Tabellenbereiche verfügbar. Bei SMS werden die LOBs in speziellen Dateien im Behälterverzeichnis gespeichert (vgl. Abbildung 5.36).

► **SYSTEM TEMPORARY**

Temporäre Daten, wie sie z.B. bei Sortierungen anfallen, benötigen diesen Tabellenbereichstyp. Da jede Datenbank mindestens einen dieser Bereiche benötigt, wurde bei der Erstellung der Datenbank der TEMPSPACE1 in dieser Art angelegt.

► **USER TEMPORARY**

Dieser Typ muss erstellt werden, bevor sog. „globale temporäre Tabellen“ von den Benutzern erstellt werden können.

Nach dem Namen des Tabellenbereichs wird definiert, welche Seitengröße für den Tabellenbereich verwendet werden soll (vgl. Tabelle 5.1 auf Seite 186). Der Standardwert ist 4.096 Bytes. Zur Auswahl stehen noch 8.192 Bytes (8K), 16.384 (16K) oder 32.768 (32K).

MANAGED BY SYSTEM gibt an, dass ein SMS-Tabellenbereich erstellt werden soll, MANAGED BY DATABASE bedeutet, dass ein DMS-Tabellenbereich gewünscht wird.

Die EXTENTSIZE, also die Größe der Speicherbereiche bzw. „Häppchen“, wie die Daten abwechselnd über die Behälter verteilt werden, kann ebenfalls angegeben werden. Die Angabe erfolgt entweder in der Einheit Speicherseiten oder der Einfachheit halber in Kilobyte (K) oder Megabyte (M), die dann aber auf den nächsten Wert gerundet werden, der einem Vielfachen einer Speicherseite entspricht. Wenn die Option EXTENTSIZE hier nicht angegeben wird, wird die Einstellung des Parameters DFT_EXTENT_SZ aus der Datenbankkonfiguration übernommen.

Hinweis:

Die EXTENTSIZE-Einstellung kann später, wie die PAGESIZE, für diesen Tabellenbereich nicht mehr geändert werden!

Unter PREFETCHSIZE ist die Datenmenge anzugeben, die vom DB2 „vorausgelesen“ werden soll. Das heißt: Wenn eine Anwendung größere Datenmengen anfordert und DB2 erkennt, dass ein vorzeitiges Bereitstellen der Daten im Pufferpool Leistungsvorteile bringen würde, wird der Vorabsezugriff (prefetch) eingesetzt. Dabei werden dann neben den aktuell angeforderten Daten weitere Datenseiten vorab in den Pufferpool gestellt. Die Anzahl der Seiten wird über PREFETCHSIZE definiert.

Wie bei Abbildung 5.27 beschrieben, werden die Daten eines Tabellenbereichs im Hauptspeicher in einen DB2-Pufferpool geladen. Die Zuordnung erfolgt auf Tabellenbereichsebene. Wird die BUFFERPOOL-Option nicht angegeben, wird der Tabellenbereich mit dem IBMDEFAULTBP – also dem Standard-Pufferpool – verknüpft.

Wie die Behälter definiert werden, sehen wir uns getrennt nach SMS und DMS an.

Tipp:

Über den Parameter DFT_PREFETCH_SZ aus der Datenbankkonfiguration kann ein Standardwert vorgegeben werden. Sinnvoller ist jedoch in den meisten Fällen eine Abstimmung dieses Parameters anhand der Behälteranzahl und der gewählten EXTENTSIZE nach folgender Regel:

$$\text{PREFETCHSIZE} = \text{EXTENTSIZE} * \text{Anzahl der Behälter}$$

Bei SMS-Tabellenbereichen entspricht ein Behälter einem Verzeichnis. Diese Verzeichnisse werden in Hochkommata angegeben; mehrere Verzeichnisse – Behälter – werden mit Kommata getrennt. Eine Größenangabe ist nicht erforderlich, da SMS-Tabellenbereiche mit den Daten dynamisch wachsen.

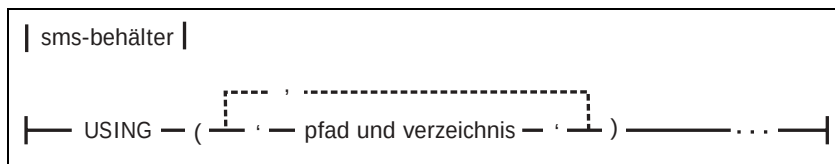


Abbildung 5.37: Syntax für SMS-Behälter

Der DB-Administrator ist dafür verantwortlich, dass sich die Verzeichnisse bzw. Dateien bei SMS bzw. DMS auf unterschiedlichen Laufwerken (Platten) befinden, da nur so der Leistungsvorteil durch die Parallelität voll genutzt werden kann.

Bei DMS-Tabellenbereichen muss angegeben werden, ob Dateien (FILE) oder unformatierte Einheiten (DEVICE) als Behälter dienen sollen. Eine Mischung von Dateien und „Devices“ für einen Tabellenbereich ist aus Performance-Gründen nicht zu empfehlen!

Wie bei SMS werden die Behälter (hier Dateien mit Pfad oder Einheiten) in Hochkommata gesetzt. Zusätzlich ist auch eine Größenangabe für jeden Behälter zu machen. Diese kann in der Einheit Speicherseiten oder Kilobyte (K), Megabyte (M) bzw. Gigabyte (G) angegeben werden, wobei alle Eingaben intern in Speicherseiten umgerechnet werden.

Tipp:

Es empfiehlt sich generell, die Behälter gleich groß zu machen, da nur so der ganze Speicherbereich genutzt werden kann (SMS) bzw. die optimale Parallelität für den Zugriff garantiert werden kann (DMS). Bei SMS ist ein Tabellenbereich voll, wenn einer der Behälter (Dateisystem) voll ist. Unterschiedlich große Behälter führen hier also zu Platzverschwendung. Bei DMS kann, auch wenn ein Behälter schon bis zum Limit gefüllt ist, weiter auf die anderen Behälter geschrieben werden, dann aber mit verringerter Parallelität.

Bei unformatierten Einheiten sollte die Behältergröße mit dem Platz auf der Einheit übereinstimmen, da der restliche Speicher nicht anders verwendet werden kann und so verschwendet wäre.



Abbildung 5.39: Assistent: Tabellenbereich erstellen – Seite 1



Abbildung 5.40: Assistent: Tabellenbereich erstellen – Schritt 2

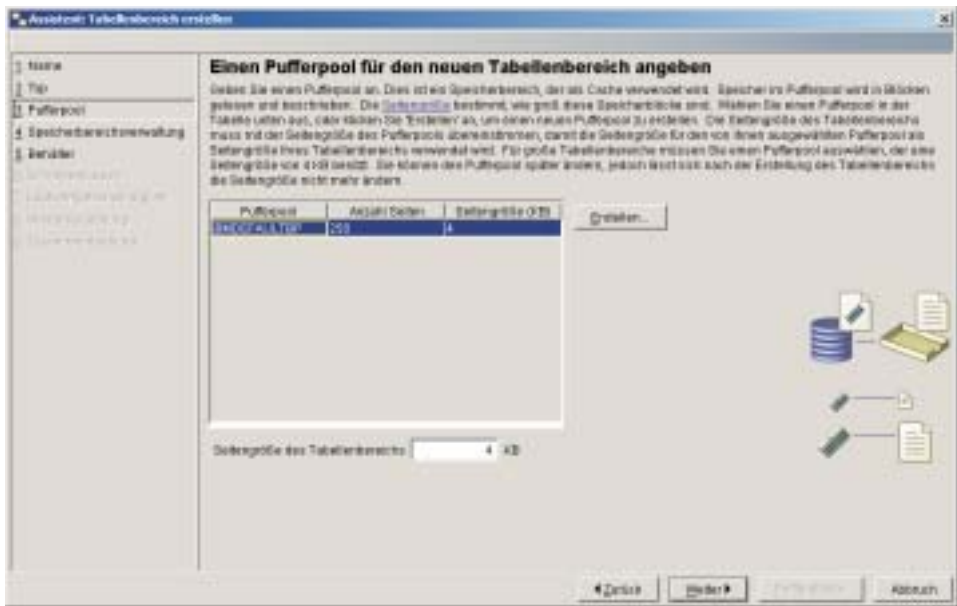


Abbildung 5.41: Assistent: Tabellenbereich erstellen – Schritt 3

Die Speicherbereichsverwaltung im Schritt 4 lässt Ihnen die Wahl zwischen SMS- und DMS-Tabellenbereichen, wobei hier SYSTEMVERWALTETER SPEICHERBEREICH für SMS und DATENBANKVERWALTETER SPEICHERBEREICH für DMS steht.

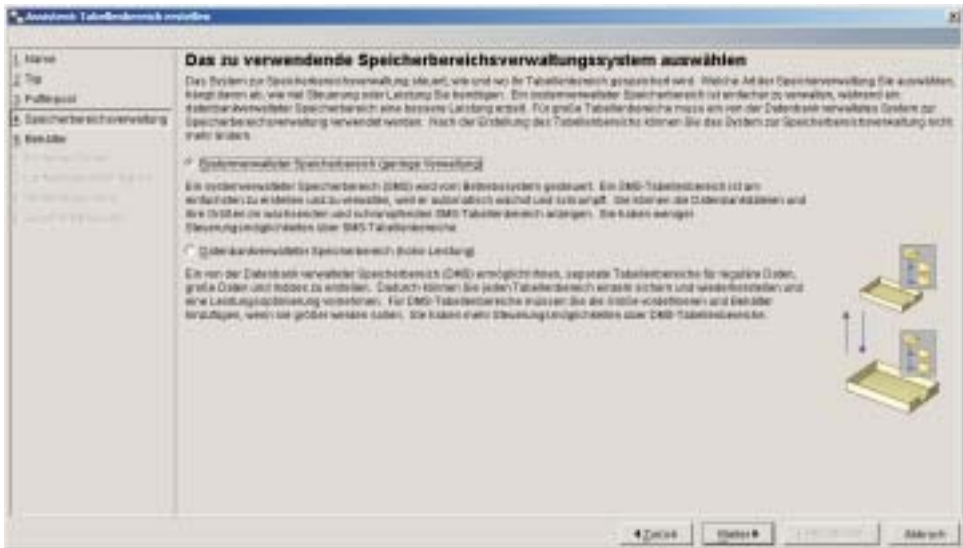


Abbildung 5.42: Assistent: Tabellenbereich erstellen – Schritt 4

Entsprechend Ihrer Auswahl hier können Sie die Behälter im nächsten Schritt – per HINZUFÜGEN-Button – definieren. Bei SMS können Sie dann Verzeichnisse wählen, bei DMS haben Sie – wie auch beim Befehl – die Auswahl zwischen Dateien und unformatierten Einheiten als Behälter. Die Abbildung 5.43 zeigt links die Behälterauswahl bei SMS- und rechts bei DMS-Tabellenbereichen.

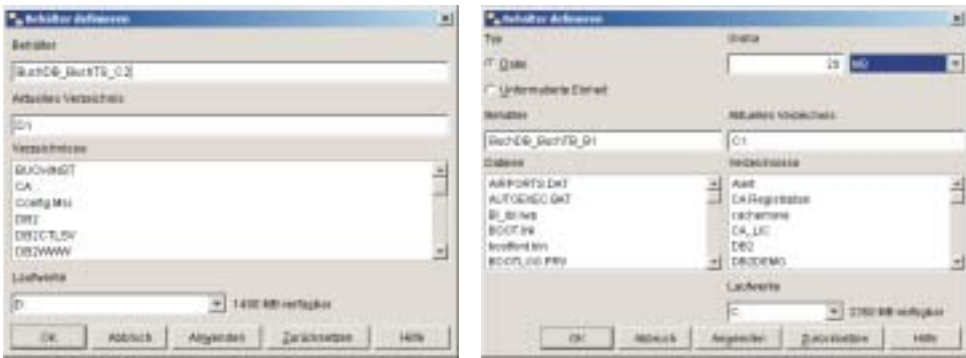


Abbildung 5.43: Assistent: Tabellenbereich erstellen – Behälter definieren

Alle weiteren Schritte sind optional bzw. mit Standardwerten vorbelegt; somit wird nun der FERTIG STELLEN...-Button aktiv. Sie können und sollten sich die weiteren Seiten durchlesen, um die zusätzlichen Optionen kennen zu lernen. Die letzte Seite – Schritt 9 – zeigt Ihnen die Zusammenfassung an und ermöglicht die Anzeige des im Hintergrund abgesetzten DB2-Befehls bei Bestätigung des Fensters mit FERTIG STELLEN.

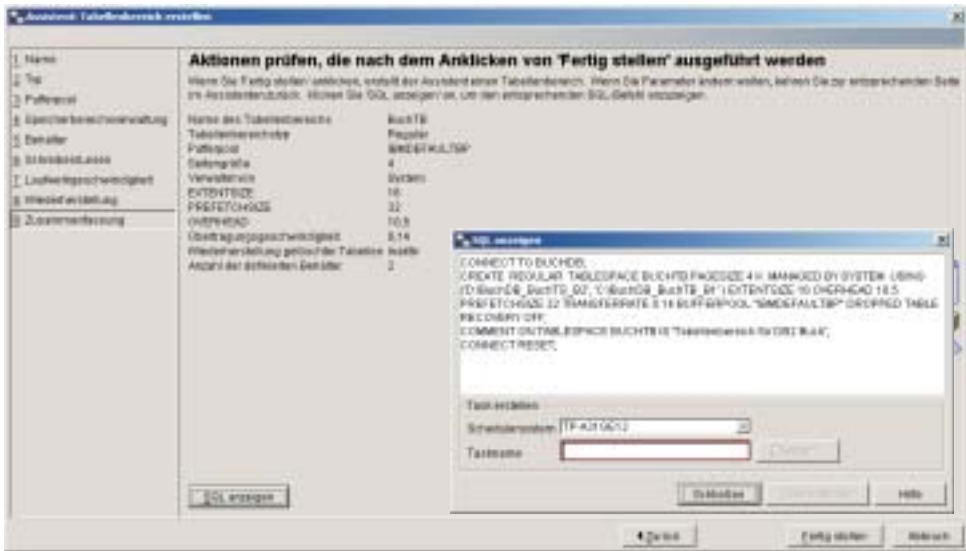


Abbildung 5.44: Assistent: Tabellenbereich erstellen – Schritt 9 – Zusammenfassung

5.8.5 Anzeige von Tabellenbereichen und Behältern

Den Aufbau einer Datenbank und ihrer Tabellenbereiche haben wir nun ausführlich beschrieben. Wie aber finden Sie sich in einem „fremden“ DB2-System zurecht? Welche Tabellenbereiche sind wie definiert? Wo hat der Kollege (oder man selbst) die Behälter platziert, um nicht zu sagen: „versteckt“?

Auf der grafischen Oberfläche gibt es in der Objektbaumstruktur unter der Datenbank einen TABELLENBEREICHE-Eintrag. Wenn Sie diesen anklicken, bekommen Sie auf der rechten Seite im Detailfenster eine Übersicht über die zur Verfügung stehenden Tabellenbereiche zu sehen (siehe Abbildung 5.45).

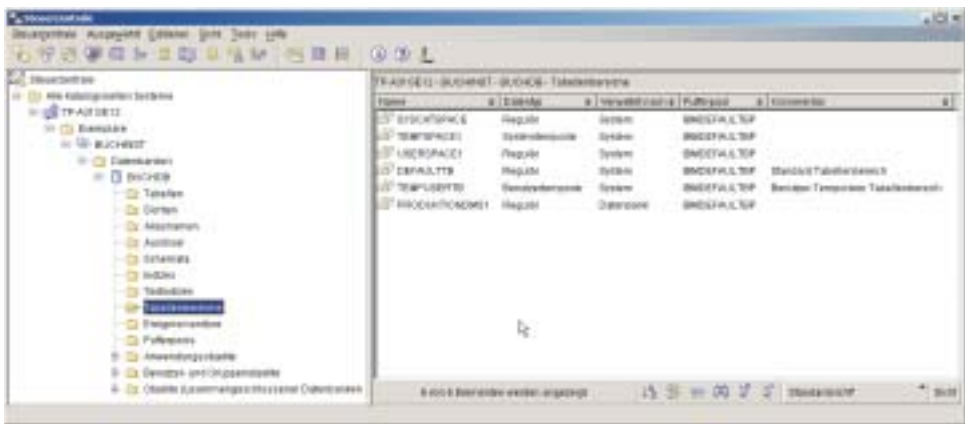


Abbildung 5.45: Tabellenbereichsübersicht in der Steuerzentrale

Die Tabellenbereichsinformationen sind in der Systemkatalog-Tabelle SYSIBM.SYSTABLESPACES der Datenbank gespeichert, die Sie sich auch über die SYSCAT.TABLESPACES-Sicht anzeigen lassen können.

Eine Anzeige der zugehörigen Behälter ist hier nicht vorgesehen. Die Informationen zu den Tabellenbereichsbehältern speichert DB2 nicht in den Systemtabellen, sondern in Verwaltungsdateien auf der Platte. Somit kann auch eine SQL-Abfrage der Katalogtabellen hier nicht weiterhelfen. Durch einen kleinen Trick bekommt man doch die benutzten Behälter im GUI angezeigt, wenn Sie nicht lieber den nachfolgend beschriebenen DB2-Befehl absetzen wollen.

Wählen Sie einen Tabellenbereich im rechten Detailfenster aus, und öffnen Sie das Kontextmenü mit der rechten Maustaste. Wenn Sie nun auf ÄNDERN... klicken, öffnet sich das TABELLENBEREICH ÄNDERN-Fenster (siehe Abbildung 5.46) mit einem auswählbaren BEHÄLTER-Reiter. Dort finden Sie die aktuell definierten Behälter dieses Tabellenbereichs mit Lage und Größe.

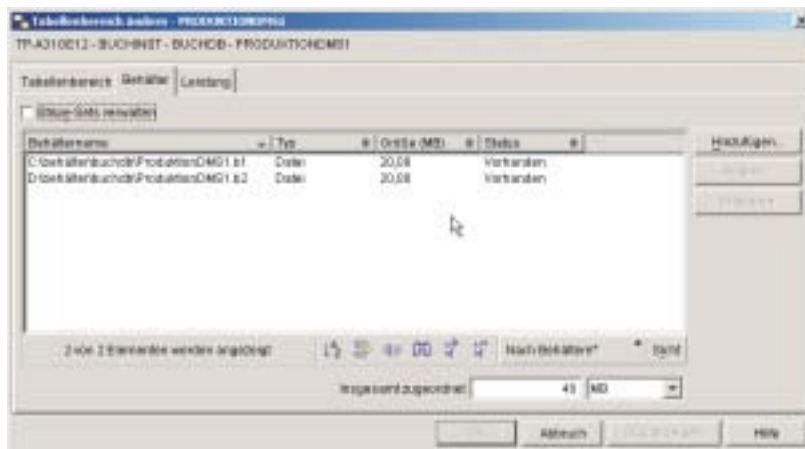


Abbildung 5.46: Tabellenbereich ändern-Fenster der Steuerzentrale

Aber wie gesagt, das ist ein Trick und nicht die gängige und empfohlene Vorgehensweise. Über einen DB2-Befehl können Sie wichtige Zusatzinformationen abfragen und bekommen eine Auflistung der Tabellenbereiche und auch der Behälter.

Sie verbinden sich dazu über den CONNECT-Befehl mit der Datenbank. So wird festgelegt, auf was sich die nächsten Befehle beziehen bzw. welche Tabellenbereiche gemeint sind. Der folgende Befehl zeigt Ihnen dann eine Liste der definierten Tabellenbereiche der aktuellen Datenbank an:

```
list tablespaces [show detail]
```

Die SHOW DETAIL-Option zeigt zusätzliche Informationen (diese sind in Abbildung 5.47 farbig hinterlegt) über die definierten Werte (z.B. EXTENTSIZE), den zur Verfügung stehenden Platz, den Füllgrad und die Anzahl der Behälter an.

Für eine bessere Übersicht wurden aus der Ausgabe einige Teile (Tabellenbereiche) entfernt. Jeder Tabellenbereich ist durch eine in der Datenbank eindeutige Tabellenbereichs-ID gekennzeichnet. So steht hier die ID 3 für einen explizit angelegten SMS-Tabellenbereich und die ID 5 für einen DMS-Tabellenbereich.

Schön zu erkennen ist auch, dass wegen der dynamischen Reservierung von Speicher bei SMS der Eintrag Seiten insgesamt und Verwendete Seiten gleich ist. Freie Seiten und Obere Grenze (Seiten) spielen bei diesem Tabellenbereichstyp ebenfalls keine Rolle.

Anders ist das im Falle des DMS-Tabellenbereichs – alle Felder sind hier gefüllt. Nähert sich hier der Wert der Verwendete Seiten dem der Verwendbare Seiten, ist ein Eingriff von Ihnen gefragt, um entweder Daten aus diesem Tabellenbereich zu entfernen oder – wie im folgenden Abschnitt „Administration von Tabellenbereichen“ beschrieben – diesen zu erweitern.

```
D:\SQLLIB\BIN>db2 list tablespaces show detail
```

Tabellenbereiche für aktuelle Datenbank

```

Tabellenbereichs-ID          = 0
Name                         = SYSCATSPACE
Typ                          = Vom System verwalteter Bereich
Inhalt                       = Beliebige Daten
Status                       = 0x0000
    Detaillierte Erläuterung:
        Normal
Seiten insgesamt              = 4262
Verwendbare Seiten            = 4262
Verwendete Seiten             = 4262
Freie Seiten                  = Nicht zutreffend
Obere Grenze (Seiten)         = Nicht zutreffend
Seitengröße (Byte)            = 4096
EXTENTSIZE (Seiten)           = 32
PREFETCHSIZE (Seiten)         = 16
Anzahl Behälter               = 1
...

Tabellenbereichs-ID          = 3
Name                         = DEFAULTTB
Typ                          = Vom System verwalteter Bereich
Inhalt                       = Beliebige Daten
Status                       = 0x0000
    Detaillierte Erläuterung:
        Normal
Seiten insgesamt              = 1
Verwendbare Seiten            = 1
Verwendete Seiten             = 1
Freie Seiten                  = Nicht zutreffend
Obere Grenze (Seiten)         = Nicht zutreffend
Seitengröße (Byte)            = 4096
EXTENTSIZE (Seiten)           = 16
PREFETCHSIZE (Seiten)         = 16
Anzahl Behälter               = 1
...

Tabellenbereichs-ID          = 5
Name                         = PRODUKTIONDMS1
Typ                          = Von der Datenbank verwalteter
Bereich
Inhalt                       = Beliebige Daten
Status                       = 0x0000
    Detaillierte Erläuterung:
        Normal
Seiten insgesamt              = 10240
Verwendbare Seiten            = 10208
Verwendete Seiten             = 48
Freie Seiten                  = 10160
Obere Grenze (Seiten)         = 48
Seitengröße (Byte)            = 4096
EXTENTSIZE (Seiten)           = 16
PREFETCHSIZE (Seiten)         = 16
Anzahl Behälter               = 2

```

Abbildung 5.47: Ausgabe des Befehls `list tablespaces show detail`

Der Eintrag Obere Grenze (Seiten) – auch „High Water Mark“ (HWM) genannt – zeigt den höchsten Füllgrad eines Tabellenbereichs an – es muss ja nicht unbedingt „Wasser“ sein, wie der englische Begriff andeutet. Diesem Wert kommt eine besondere Rolle zu, da sich verschiedene Tools bzw. Befehle bei der Größe eines Tabellenbereichs an diesem Wert orientieren. Die Größenreduzierung eines Behälters (Details siehe Abschnitt „Administration von Tabellenbereichen“) z.B. wird nur bis zur Oberen Grenze (Seiten) erlaubt, auch wenn aktuell weniger Daten (verwendete Seiten) im Tabellenbereich sind.

Die Behälter eines Tabellenbereichs werden aufgelistet durch:

```
list tablespace containers for <tablespace-id> [show detail]
```

Dabei müssen Sie die Tabellenbereichs-ID, die in Abbildung 5.47 ausgegeben wurde (und auch so im Systemkatalog der Datenbank zu finden ist), verwenden, um den Tabellenbereich auszuwählen, dessen Behälter (Container) Sie sehen wollen.

Im folgenden Beispiel in der Abbildung 5.48 wurde dafür der DMS-Tabellenbereich mit der ID 5 ausgewählt.

```
D:\SQLLIB\BIN>db2 list tablespace containers for 5 show detail
```

Tabellenbereichsbehälter für Tabellenbereich 5

Behälter-ID	= 0
Name	= C:\behälter\buchdb\ProduktionDMS1.b1
Typ	= Datei
Seiten insgesamt	= 5120
Verwendbare Seiten	= 5104
Im Zugriff	= Ja
Behälter-ID	= 1
Name	= D:\behälter\buchdb\ProduktionDMS1.b2
Typ	= Datei
Seiten insgesamt	= 5120
Verwendbare Seiten	= 5104
Im Zugriff	= Ja

Abbildung 5.48: Ausgabe des Befehls `list tablespace containers`

Auch hier wurden die Informationen farbig hinterlegt, die nur über die `SHOW DETAIL`-Option angezeigt werden.

Hinweis:

Die Tabellenbereichs- und Behälter-Informationen können Sie sich auch über den Snapshot-Monitor anzeigen lassen. Der zugehörige Befehl lautet:

```
get snapshot for tablespaces on <datenbankname>
```

Das weiteren Einzelheiten zur Überwachung (Monitoring) werden in Kapitel 14 genauer unter die Lupe genommen.

Was Ihnen kein Befehl und kein Tool abnehmen kann, ist die sinnvolle Platzierung der Behälter auf Ihrem System. Dabei ist es – je nach Betriebssystem – durchaus anspruchsvoll herauszufinden, welche Verzeichnisse welchen physikalischen Platten zugeordnet sind. Diese Information ist aber für eine leistungsorientierte Verteilung der Daten unerlässlich. Was uns dabei oft geholfen hat, ist die Anfertigung eines Schaubildes, das Platten, logische Platten (volumes), Dateisysteme etc. beinhaltet. So können Sie schnell Fehler lokalisieren oder Verzeichnisse für neue Behälter finden.

5.8.6 Administration von Tabellenbereichen

Die administrativen Aufgaben bei SMS-Tabellenbereichen halten sich sehr in Grenzen, da „nur“ ein Volllaufen der Behälter verhindert werden muss. Dies geschieht durch die Überwachung der Dateisysteme und gegebenenfalls durch die Erweiterung dieser mit Mitteln des Betriebssystems.

Bei DMS-Tabellenbereichen wurde die anfängliche Größe jedes Behälters bereits beim Anlegen definiert – idealerweise wurden alle Behälter gleich groß gewählt. Das erfordert schon vor dem CREATE TABLESPACE-Befehl, dass Sie sich als Datenbank-administrator mit den zu erwartenden Datenmengen auseinander setzen. Dabei geht es nicht um eine Berechnung mit einer Genauigkeit im Kilobyte-Bereich – besonders nicht bei den heutigen Kosten der Speichermedien –, sondern um die Abschätzung von Speicherplatz, der auch für die nächsten Monate ausreichend ist.

Wir alle kennen die Situation, dass uns trotz bester Vorhersagen die Realität mit meist viel schneller als erwartet wachsenden Daten einholt. Bei DMS-Tabellenbereichen können die Behälter (Container) über den ALTER TABLESPACE-Befehl in ihrer Größe angepasst werden.

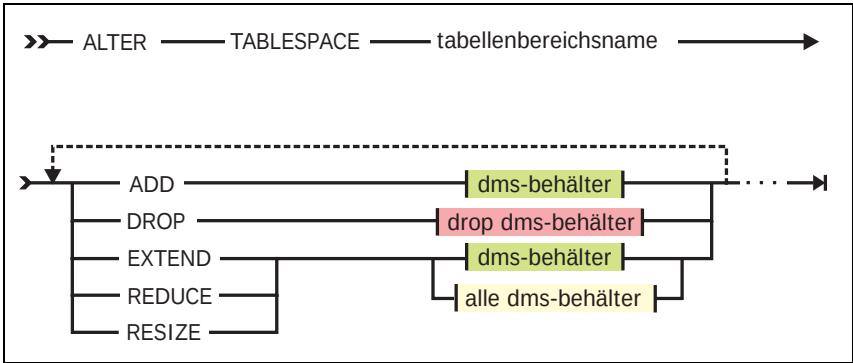


Abbildung 5.49: alter tablespace-Syntaxdiagramm

Entsprechend der gewählten Option werden dann entweder einzelne oder alle Behälter verändert. Die Behälter müssen – bis auf die DROP-Option – mit einer Größenangabe versehen sein (siehe Abbildung 5.50).

Während die ADD- und DROP-Option bei der Änderung eines Tabellenbereichs „sprechend“ sind, müssen wir uns die Unterschiede zwischen EXTEND, REDUCE und RESIZE etwas genauer anschauen. Sowohl EXTEND (erweitern) als auch REDUCE (verringern) erwarten eine Größenangabe, um die verringert bzw. erweitert werden soll. RESIZE dagegen hat als Größenangabe die neue Zielgröße des Behälters. Diese kann kleiner oder größer sein als die bisherige. Die Abbildung 5.51 verdeutlicht dies noch einmal grafisch.

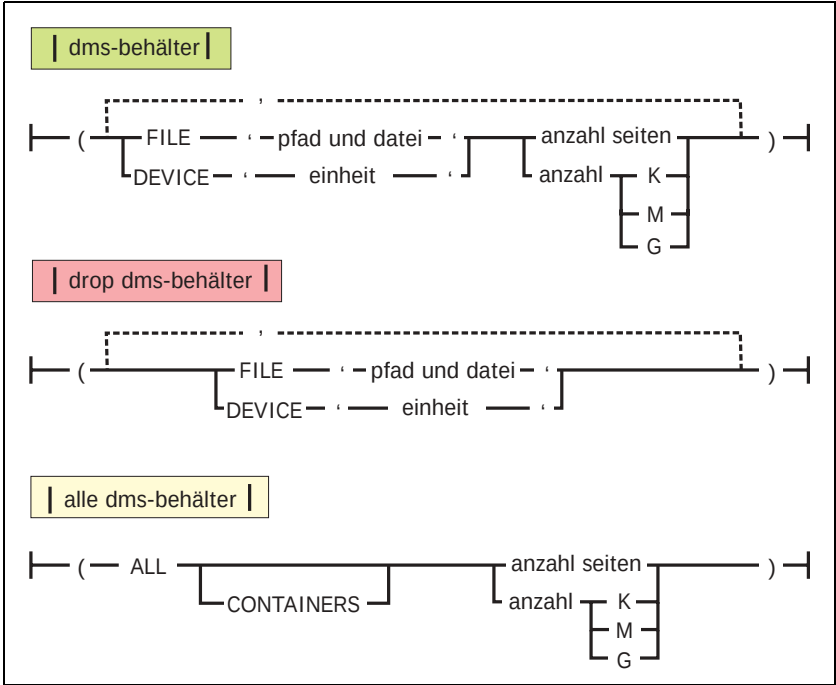


Abbildung 5.50: Syntaxdiagramm für die alter tablespace-Behälter-Option

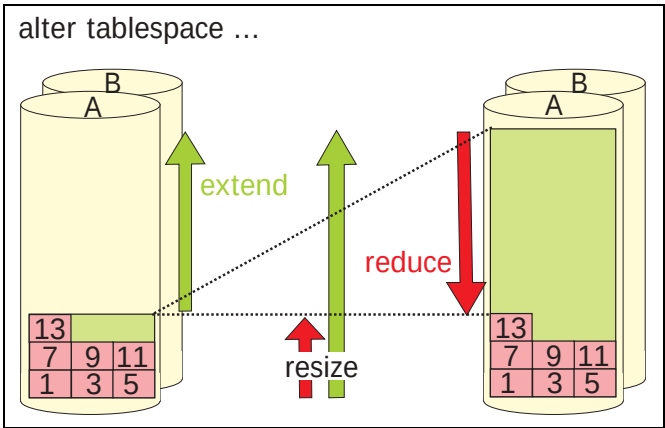


Abbildung 5.51: Größenänderung eines DMS-Tabellenbereichs

Bei diesen Anpassungen sind einige wichtige Auswirkungen zu beachten: Die Erweiterung eines Tabellenbereichs durch einen zusätzlichen Behälter hat zur Folge, dass die „alten“ Behälter relativ voll sind und der „neue“ zusätzliche Behälter leer ist. Werden dem Tabellenbereich nun neue Daten zugewiesen (z.B. durch Laden von Daten in eine neue Tabelle), würden diese hauptsächlich in den neuen Behälter wandern. So wäre eine Nutzung der Parallelität – wenigstens für dieses Objekt – nicht mehr möglich.

Würde – wäre – hätte ... kurzum: So darf es nicht laufen.

DB2 hat ausgefeilte Performance-Mechanismen, die einen solchen suboptimalen Zustand durch einen automatischen Ausgleich der Behälter-Füllstände vermeiden. Man nennt diesen Vorgang „Vorwärtsneuausgleich“ (engl. rebalancing bzw. forward rebalancing), wenn ein neuer Behälter hinzugefügt wird. Dabei werden, wie in Abbildung 5.52 – von oben nach unten gelesen – gezeigt, die Speicherbereiche (extents) neu über alle Behälter verteilt. So können ein gleichmäßiger Füllgrad und eine optimale Parallelität gewährleistet werden. Nach dem Neuausgleich kann nun nicht nur zweifach, sondern dreifach parallel gelesen bzw. geschrieben werden.

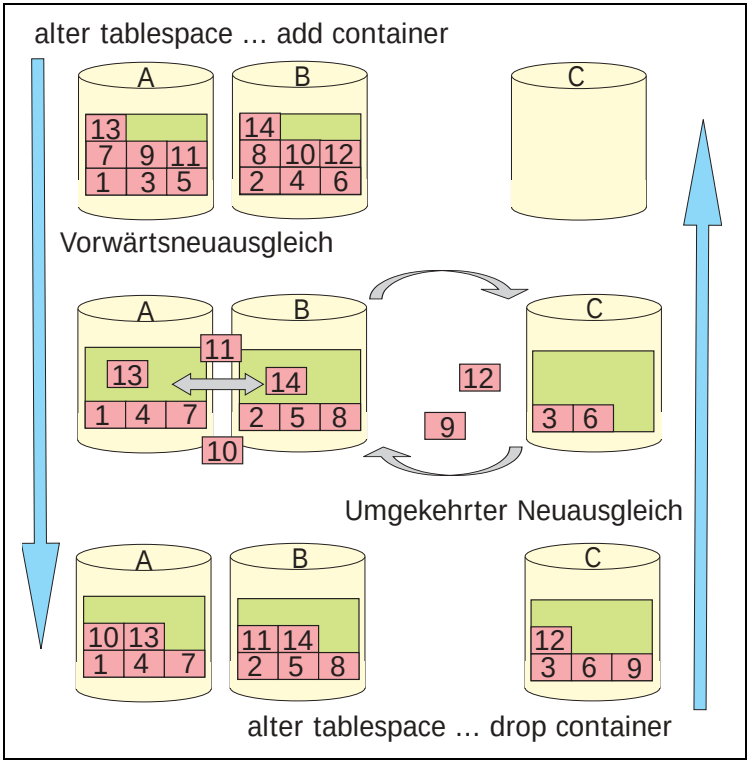


Abbildung 5.52: DMS-Neuausgleich (rebalancing)

Wenn ein Behälter entfernt werden soll, müssen die Speicherbereiche (extents) aus dem zu entfernenden Behälter in die anderen Behälter verteilt werden. Dieser Vorgang heißt „Umgekehrter Neuausgleich“ (reverse rebalancing). Abbildung 5.52 veranschaulicht diesen Vorgang, wenn Sie das Bild von unten nach oben betrachten.

Hinweis:

Generell stellt jeder Neuausgleich (engl. rebalancing) eine zusätzliche Systemlast durch die Umverteilung der Speicherbereiche (extents) dar. Dies schränkt zwar den Zugriff von Anwendungen nicht ein, hat aber merkbare Auswirkungen auf ihre Performance.

Das Hinzufügen (ADD) und Entfernen (DROP) sollte deshalb nur zu Zeiten stattfinden, in denen die Anwender nicht höchste Leistung benötigen. Idealerweise werden diese Befehle zeitgesteuert abgesetzt, um so diesen Ausgleich z.B. um Mitternacht durchzuführen. Dies setzt allerdings voraus, dass Sie als Administrator die Füllstände Ihrer Tabellenbereiche kennen und überwachen, um rechtzeitig eingreifen zu können.

Werden *mehrere Behälter* hinzugefügt oder entfernt, ist es unter Performance-Gesichtspunkten dringend anzuraten, dies in *einem Befehl* zu tun, um den Ausgleichsaufwand nur einmal zu haben.

Werden Behälter nur erweitert oder verkleinert, ist kein Neuausgleich notwendig.

5.9 Tipps aus der Praxis

Einige Problematiken aus diesem Themenbereich kommen in Schulungen und beim Kunden immer wieder auf den Tisch. Vieles davon ist in der Dokumentation sehr versteckt und somit unbekannt. Im Folgenden möchten wir Ihnen einige Punkte aufzeigen, die sich in der Praxis als sehr nützlich erwiesen haben.

5.9.1 Welche Art Tabellenbereich soll ich nun verwenden?

Dies ist natürlich eine berechtigte Frage, aber die Antwort ist nicht ganz so einfach. Versuchen wir sie einmal auf ein paar grundlegende Empfehlungen zu reduzieren.

Folgende Punkte beeinflussen die Entscheidung bei der Frage nach SMS oder DMS:

- Haben Sie als Administrator ständig die Möglichkeit, das System zu überwachen, und gegebenenfalls die Möglichkeit einzugreifen?

Bei DB2-Servern, die z.B. irgendwo in Filialen stehen, wo kein Datenbankadministrator vor Ort ist, empfehlen sich eher SMS-Tabellenbereiche, weil sie weniger administrative Eingriffe benötigen. Eine gewisse Leistungseinbuße wird in solchen Fällen oft gern in Kauf genommen.

- Haben Sie sehr große Datenmengen, und sind Sie auf sehr gute Performance angewiesen?

Auch wenn sich der Leistungsunterschied zwischen SMS- und DMS-Tabellenbereichen in den letzten Versionen, nicht zuletzt durch schnellere Platten und die Möglichkeit der performanteren Reservierung von Speicherplatz (siehe Abschnitt „Das Tool db2empfa“) immer weiter verringert hat, ist DMS in diesen Fällen schneller. In solchen Situationen bringen unformatierte Einheiten (raw devices) als Behälter noch mal eine zusätzliche Leistungssteigerung.

Bei großen Tabellen (Datenmengen) – also im Gigabyte-Bereich – tendiert man eher zu DMS, bei vielen kleinen Objekten kann man mit SMS Platz sparen.

► Welche Art von Daten wollen Sie speichern?

Für temporäre Daten ist es beispielsweise charakteristisch, dass zeitweise sehr viele Daten vorliegen, während zu anderen Zeiten kaum oder keine Daten zwischengespeichert werden. Bei temporären Tabellenbereichen ist eindeutig die SMS-Tabellenbereichsverwaltung vorteilhaft, da man keinen Platzbedarf kalkulieren muss und nicht unnötig Speicher vorhält, wenn keiner benötigt wird. DMS-Tabellenbereiche kommen evtl. hier nur für Tabellenbereiche für temporäre Benutzerdaten in Frage, wenn Sie sehr große temporäre Tabellen nutzen.

LOB-(Large Objects-) und LONG-Daten werden nicht im Pufferpool der Datenbank gecacht (zwischengespeichert). Liegen die Behälter aber in einem Dateisystem, so übernimmt das Betriebssystem über den Dateisystem-Cache diese Aufgabe, und das kann sich wiederum vorteilhaft auswirken. Dazu dürfen Sie aber keine unformatierten Einheiten (DMS raw devices) einsetzen.

► Haben Sie häufige leistungskritische Indexzugriffe?

Nur DMS-Tabellenbereiche erlauben Ihnen eine Aufteilung der Daten, LOBs und Indizes auf verschiedene Tabellenbereiche und so einen parallelen Zugriff auf Index und Daten, die ihrerseits wieder parallel ablaufen können. Dies ergibt ein ideales Leistungsverhalten, wenn die Behälter sauber auf mehrere unterschiedliche Festplatten verteilt wurden.

Sie sehen, dass auch hier wieder Kompromisse zu finden sind.

Abschließend können wir hier aber noch einmal feststellen, dass die drei Tabellenbereiche, die beim CREATE DATABASE-Befehl erstellt werden, mit der SMS-Tabellenbereichsverwaltung eigentlich gut beraten sind. Die generellen Empfehlungen für den SYSCATSPACE, also für die Systemtabellen, geht auch in Richtung SMS. Der temporäre Tabellenbereich TEMPSPACE1 sollte SMS sein, und der USERSPACE1 wird später sowieso nicht mehr genutzt, da eigene Tabellenbereiche mit mehreren gezielt verteilten Behältern vom Administrator angelegt werden. Der einzige Punkt, der aus Leistungssicht verbessert werden könnte, ist, dass der temporäre Tabellenbereich mit mehreren Behältern Parallelität beim E/A-Zugriff nutzen könnte. Diese Einschränkung kann man aber durch einen neuen, explizit erzeugten temporären Tabellenbereich mit eventuell größerer Seitengröße beheben.

5.9.2 Der Standard-Tabellenbereich

Über Tabellenbereiche haben wir in diesem Kapitel schon eingehend gesprochen. Die Frage aber, wo Tabellen (oder allgemeiner: Datenbankobjekte) standardmäßig abgelegt werden, falls kein Tabellenbereich explizit angegeben wird, ist noch offen oder nur vermeintlich klar.

Benutzer tendieren dazu, bei einem CREATE TABLE-Befehl, die IN <tabellenbereich>-Option zu „vergessen“ – nach dem Motto „wird schon gut gehen“. Aber auch einige Datenbank-Tools (z.B. Visual Explain) benötigen zusätzliche Tabellen, die bei Bedarf automatisch angelegt werden.

Wo bzw. in welchen Tabellenbereich werden diese abgelegt?

Sie werden sich vielleicht an den USERSPACE1 als einen der Tabellenbereiche erinnern, die standardmäßig beim Befehl CREATE DATABASE erstellt werden. Aber ist dies auch der Standard-(Default-)Tabellenbereich?

„Ja – aber...“ – hatten Sie die Antwort nicht so schon erwartet?

Die Tabellen werden standardmäßig nur dann im USERSPACE1 abgelegt, wenn bisher kein zusätzlicher Tabellenbereich erstellt worden ist. Ansonsten gilt die Regelung, die in Abbildung 5.53 dargestellt ist.

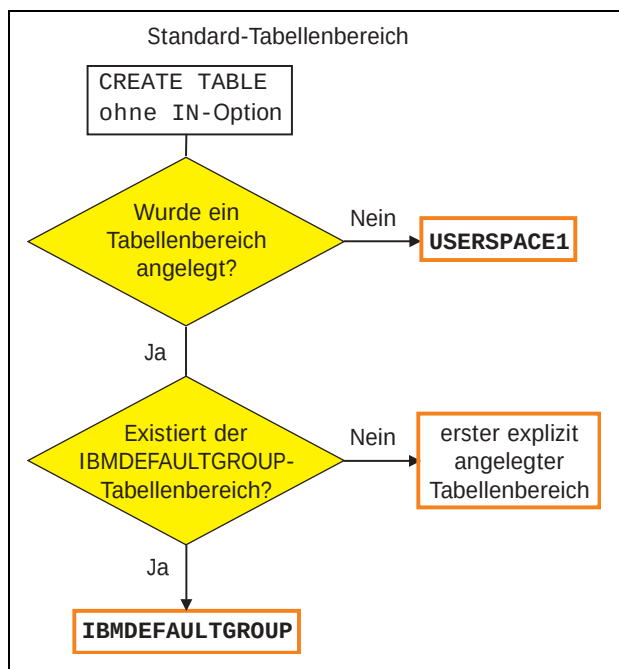


Abbildung 5.53: Standard-Tabellenbereich-Diagramm

Das bedeutet also, der Standard-Tabellenbereich ist der erste Tabellenbereich, der vom DB2-Datenbankadministrator erzeugt wird. Um hier keine großen Wartungsaufgaben bezüglich der Größenanpassung haben zu müssen, sollte der erste Tabellenbereich ein SMS-Tabellenbereich sein. Sinnvoll wäre es, diesen auch sprechend zu benennen, etwa „StandardTB“.

So können Sie als Datenbankadministrator regelmäßig überprüfen, welche Objekte ohne exakte Zielangabe angelegt worden sind, und dies gegebenenfalls bereinigen. Diese Technik verhindert, dass Ihnen DMS-Tabellenbereiche (weil zufällig als Erstes erstellt) volllaufen, wenn z.B. die Explain-Tabellen automatisch erstellt werden.

Ist das Kind schon in den Brunnen gefallen – d.h., existiert schon eine Datenbank mit zusätzlichen Tabellenbereichen –, können Sie die Situation retten, indem Sie einen SMS-Tabellenbereich mit dem Namen IBMDEFAULTGROUP erstellen.

5.9.3 Der Datenbank-Wartungsmodus

Eine Datenbank wird – wie in Abschnitt 5.5 besprochen – durch einen `CONNECT`-Befehl aktiviert. Bestimmte Wartungsarbeiten (z.B. ein Offline-Backup) erfordern aber einen exklusiven Zugriff auf die Datenbank. In produktiven Umgebungen kommt es immer wieder vor, dass dieser exklusive Zugriff dem Administrator Schwierigkeiten bereitet. Ständig werden neue Verbindungen aufgebaut, und so kann es sein, dass selbst ein `FORCE APPLICATIONS`-Befehl nicht zum Ziel führt, da sofort wieder neue Verbindungen erstellt werden, die die Datenbank aktivieren, bevor die eigentliche Wartungsarbeit gestartet werden konnte.

Ein sog. *Wartungsmodus* schafft hier in DB2 V8 Abhilfe. Vorausgesetzt, man ist berechtigt, diesen Befehl auszuführen (`SYSADM` oder `DBADM`), können Benutzer und Anwendungen von der Datenbank getrennt werden. Offene Transaktionen werden dabei zurückgefahren (`rollback`). Mit der Option `DEFER` soll DB2 warten, bis die Anwendungen ihre aktuelle Transaktion beendet haben. Dies ist aber aktuell noch nicht möglich!

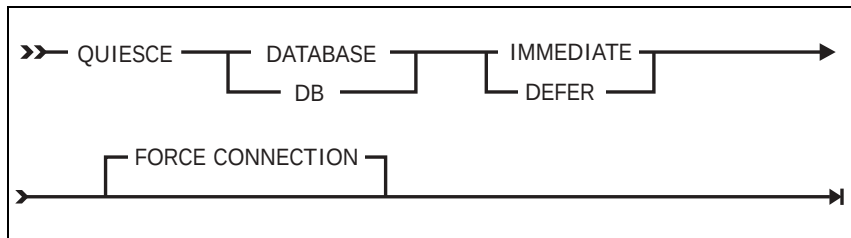


Abbildung 5.54: *quiesce database*-Syntaxdiagramm

Auf eine derart in den Wartemodus versetzte Datenbank haben nur noch Administratoren (`SYSADM`, `DBADM`, `SYSMAINT` oder `SYSCTRL`) oder über das `QUIESCE_CONNECT`-Recht explizit berechnete Benutzer Zugriff (weitere Details zu Berechtigungen siehe auch Kapitel 10.2 „Zugriffsrechte innerhalb einer Datenbank“).

Versucht nun ein unberechtigter Benutzer, sich mit der Datenbank zu verbinden, wird ein Fehler ausgegeben:

```

D:\SQLLIB\BIN>db2 connect to buchdb user user1 using user1
SQL20157N Der Benutzer "USER1" verfügt nicht über das Zugriffsrecht
QUIESCE_CONNECT. SQLSTATE=08004
  
```

Dieser Datenbank-Pflegemodus bleibt so lange bestehen, bis er explizit durch einen `UNQUIESCE DATABASE`-Befehl aufgehoben wird.

5.9.4 Unterstützung von RAID-Systemen

RAID steht für „Redundant Array of Independent Disks“ und ist ein Konzept zur Kopplung mehrerer Festplatten unter einem Controller, der die Daten selbstständig spiegelt (sog. RAID-1) oder automatisch verteilt (striping). Das bezeichnet man als RAID Level 0 oder kurz RAID-0. Es gibt noch mehrere RAID-Level, von denen aber nur noch zwei – RAID-5 und RAID-10 – eine wichtigere Rolle spielen, da sie die

zwei genannten Techniken kombinieren. Dort werden die Daten einerseits verteilt (striping), um bessere Schreib- und Lese-Leistungen zu erzielen, und andererseits auch redundant gehalten, um Ausfallsicherheit zu gewährleisten. RAID-5, das heute am meisten verwendete RAID-Verfahren, benutzt dazu eine sog. *Parity-Information*, eine Art Prüfsumme. Damit können, wenn eine Platte kaputt geht, die Daten aus den Daten der anderen Platten und der Parity-Information wiederhergestellt werden. RAID-10 dagegen setzt auf eine echte Spiegelung. Dabei werden die Daten wirklich doppelt gehalten. Diese Technik erfordert somit mehr Speicherplatz. So viel zu RAID-Systemen im Allgemeinen.

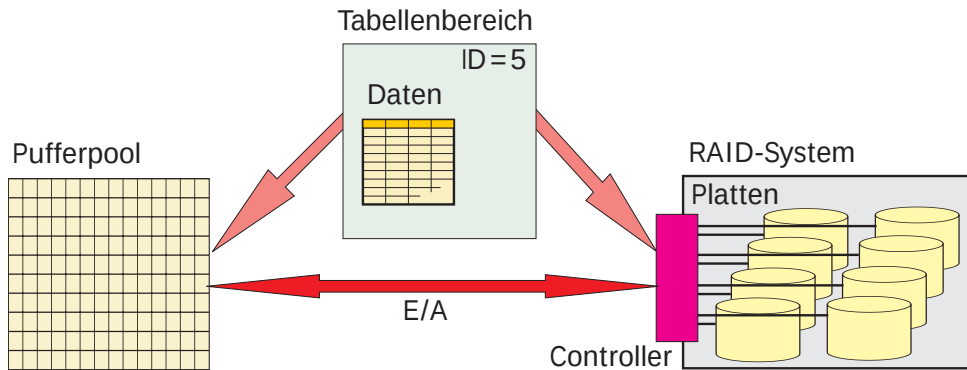


Abbildung 5.55: RAID-System

Für DB2 sieht ein RAID-System eigentlich wie eine – wenn auch schnelle – Festplatte aus, da nur ein Zugriffsweg besteht. Der RAID-Controller verwaltet die einzelnen Festplatten selbstständig, d.h., hier kann DB2 keinen Einfluss nehmen. Abbildung 5.55 zeigt ein RAID-System mit acht Festplatten. Bei der Definition des Tabellenbereichs (mit der Tabellenbereichs-ID 5) wird durch die Behälterdefinition angegeben, dass die Daten auf der RAID-Einheit gespeichert werden sollen.

Die DB2-Möglichkeiten des parallelen Zugriffs und der Datenverteilung durch das Tabellenbereichsbehälter-Konzept scheinen so ausgehebelt zu werden.

In der Überschrift wird ja schon von der „Unterstützung von RAID-Systemen“ gesprochen. Durch spezielle Einstellungen in DB2 können Sie auch diese Systeme optimal einbinden.

Folgende Punkte sind dabei zu beachten:

- Legen Sie einen Behälter pro RAID-System an.
- Stimmen Sie die RAID-Stripe-Größe, also die Größe der Datenportionen, in denen die RAID-Kontrolleinheit die Daten über die Platten verteilt, mit den DB2-Speicherbereichen ab. Dabei sollte die Speicherbereichsgröße (EXTENTSIZE) ein Vielfaches der RAID-Stripe-Größe sein.

- Aktivieren Sie den parallelen Zugriff (siehe Abbildung 5.56) trotz eines einzelnen Zugriffswegs (Behälters) über die Registrierungsvariable DB2_PARALLEL_IO. Mit dieser Variable wird definiert, für welchen Tabellenbereich dieser parallele Zugriff gelten soll – d.h., es werden die Tabellenbereiche angegeben, die Behälter auf RAID-Systemen haben. Die Tabellenbereiche werden dabei durch Ihre Tabellenbereichs-ID angegeben:

```
db2set DB2_PARALLEL_IO = 5,8
```

Alternativ kann auch „*“ verwendet werden, um parallele E/A für alle Behälter zu aktivieren.

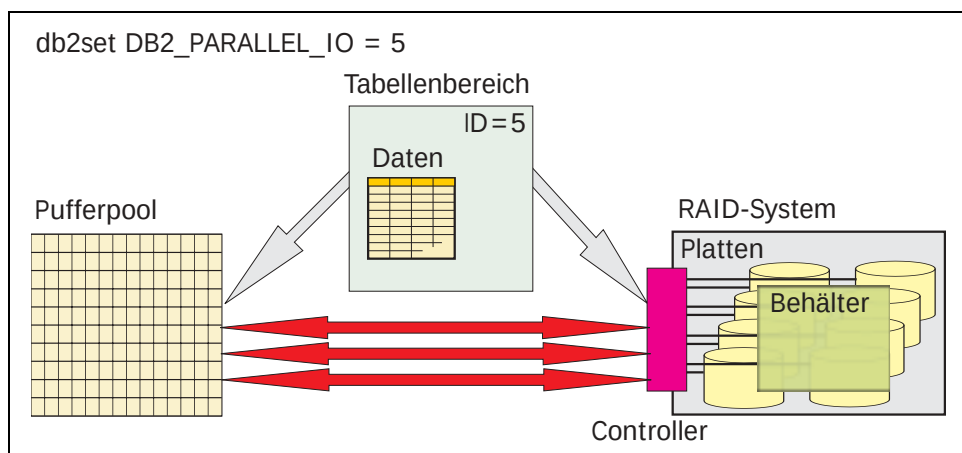


Abbildung 5.56: DB2-Anpassung für RAID-Systeme

Hinweis für DB2 V7 und frühere Versionen:

Bei älteren DB2-Versionen ist zusätzlich die DB2-Registrierungsvariable `DB2_STRIPED_CONTAINERS = YES` zu setzen, um nicht durch die DB2-Verwaltungs-informationen im DMS-Behälter eine Verschiebung zwischen RAID-Stripes und DB2-Speicherbereichen zu bekommen, die die Leistung beeinträchtigen würden.

In DB2 V8 ist dieses Verhalten die Standardeinstellung. Die Registrierungsvariable `DB2_STRIPED_CONTAINERS` ist hier weggefallen.

5.9.5 Das Tool db2empfa

„DB2 enable multi-page file allocation“ wird zu „db2empfa“ abgekürzt – sind sprechende Namen nicht schön? Übersetzt bedeutet das Ganze: „Speicherreservierung in DB2-Speicherbereichseinheiten“ („extent“-weise) für SMS-Tabellenbereiche.

DMS-Tabellenbereiche nutzen ihren reservierten Speicherplatz in den Behältern, wenn neue Daten hinzukommen. Bei SMS-Tabellenbereichen wird der Speicherplatz dynamisch reserviert, und das standardmäßig in Seiten-Einheiten. Kommen

nun viele Datensätze, z.B. durch das Importieren von Daten in eine Tabelle hinzu, wird eine Seite für dieses Objekt reserviert und dann mit Daten aufgefüllt. Dieses Reservieren und Auffüllen geschieht so lange, bis alle Daten gespeichert sind. Da eine Seite 4 Kbyte (oder maximal 32 Kbyte) umfasst, wird sich dieser Vorgang sehr oft wiederholen, bevor die ganze Datenmenge gespeichert ist.

Eine Leistungssteigerung kann nun dadurch erreicht werden, dass der Speicherplatz nicht für jede Seite einzeln angefordert wird, sondern dass ein ganzer DB2-Speicherbereich (extent) reserviert wird; das sind standardmäßig 32 Seiten auf einmal. Ist dieser dann mit Daten gefüllt, wird der nächste Speicherbereich reserviert usw. Dadurch muss das System zwar mehr Speicher auf einmal reservieren (z.B. 128 Kbyte statt 4 Kbyte), hat aber andererseits nur 1/32 der Reservierungsanforderungen.

Aktiviert wird dieses Verhalten durch folgenden Befehl:

```
db2empfa buchdb
```

Dieser Befehl benötigt einen exklusiven Zugriff auf die Datenbank – eine Gelegenheit also, den Wartungsmodus aus Abschnitt „Der Datenbank-Wartungsmodus“ auszuprobieren.

Hinweis:

Einmal aktiviert, gilt diese Reservierungsmethode für alle SMS-Tabellenbereiche in der Datenbank (bzw. Datenbankpartition). Ein „Deaktivieren“ dieser Funktion ist *nicht* möglich. Dies sollte Sie aber vom Einsatz dieses Tools nicht abhalten, da der einzige Nachteil ein etwas größerer Platzbedarf im Tabellenbereich ist, da eventuell nur Teile des DB2-Speicherbereichs (extent) mit Daten gefüllt sind.

Ob diese Reservierungstechnik in Ihrer Datenbank schon aktiviert wurde, können Sie über den informativen Parameter `MULTIPAGE_ALLOC` (im GUI) bzw. Zuordnung aus mehreren Seiten bestehender Datei aktiv (Name bei der Abfrage der DB-Konfiguration über die Befehlszeile) in der Datenbankkonfiguration sehen.

5.9.6 Der Befehl `db2relocate`

In diesem Zusammenhang sei auch kurz auf den Befehl `db2relocate` hingewiesen, der in DB2 V7.2 neu hinzugekommen ist. Dieses Tool ermöglicht Ihnen u.a. die Verlegung von Behältern oder das Umhängen einer Datenbank in ein anderes Exemplar.

Ein Beispiel für den Einsatz sowie einen Überblick über die Optionen finden Sie in Kapitel 9, „Sicherung und Wiederherstellung einer Datenbank“.

6 Aufbau verschiedener Datenbankverbindungen

In den vorangegangenen Kapiteln wurde immer wieder der Verbindungsaufbau zur Datenbank erwähnt. Was passiert aber genau, wenn ein `CONNECT TO ...` abgesetzt wird? Welche Voraussetzungen müssen erfüllt sein, und welche Sonderfälle gibt es?

Diese Fragen wollen wir in diesem Kapitel behandeln. In der Praxis haben wir relativ oft mit Problemen und Fehlersituationen zu tun, die mit diesem Bereich zusammenhängen. Geht alles, ist ein `CONNECT`-Befehl fast das Einfachste der Welt – funktioniert er nicht, muss man die Details kennen, die Sie auf den folgenden Seiten finden.

6.1 Konzept: DRDA

Bei der Verbindung zu Datenbanken spielen verschiedene Standards eine wichtige Rolle.

DRDA steht für *Distributed Relational Database Architecture*, also für eine Architektur für verteilte relationale Datenbanken. DRDA ist ein Standard der *Open Group* für die Verbindung von Anwendungen zu Datenbanken.

Über diesen, ursprünglich von IBM entwickelten Standard kommunizieren mit DB2 UDB V8 auch die DB2-Clients mit den DB2-Servern im Unix- und Windows-Umfeld. Vor DB2 UDB V8 wurde dieses Protokoll nur zur Kommunikation zum Großsystem – also zu DB2 für z/OS und DB2 für AS/400 – benutzt.

DRDA soll eine Verbindungsarchitektur implementieren, um auch in einer heterogenen Datenbankumgebung eine Kommunikation zu ermöglichen. Dabei sind zwei grundlegende Komponenten notwendig, ein so genannter *DRDA Application Requestor* (AR) und ein *DRDA Application Server* (AS).

In Abbildung 6.1 sehen Sie, wie eine Datenbankanwendung über die DB2-Client-Komponente (den DB2 RTC – Run Time Client) eine Verbindung zu einem DB2-Datenbankserver aufbaut. Die Netzverbindung basiert dabei auf dem Protokoll TCP/IP. DRDA setzt nun auf dieses Protokoll auf. Das heißt, die DRDA-Architektur nutzt das „darunter liegende“ TCP/IP-Protokoll, um vom DRDA AR, der durch den DB2-Client (RTC) implementiert ist, eine Verbindung mit dem DRDA AS herzustellen. Diese DRDA AS-Komponente ist in jede DB2-Server-Version integriert.

In älteren Versionen wurde DB2 Connect benötigt, um die DRDA AR-Funktionalität auf der Workstation-Seite zu implementieren. Erst dann konnte eine Verbindung z.B. zum DB2 für OS/390 hergestellt werden, das – wie die anderen Server auch – die DRDA AS-Funktionalität beinhaltet. Auch in DB2 UDB Version 8 ist DB2 Connect noch notwendig, um sich zum Großsystem zu verbinden, wenn auch der Grund nicht mehr die Implementierung der DRDA AR-Komponente ist.

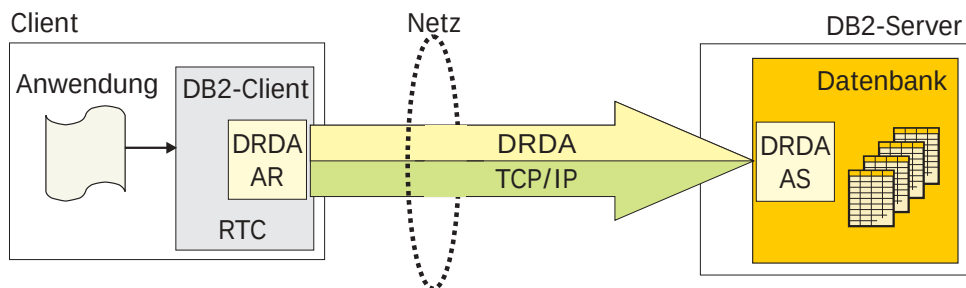


Abbildung 6.1: DRDA-Verbindung

DRDA ist nicht auf das TCP/IP-Protokoll beschränkt, sondern kann auch die heute nicht mehr so verbreitete SNA-Architektur mit dem APPC-Protokoll nutzen. Durch DRDA werden neben Sicherheitsmechanismen auch die verschiedenen Arten von Transaktionen unterstützt. Neben der „normalen“ Transaktion (Unit Of Work) werden auch die so genannten DUOW (Distributed Units Of Work) – also verteilte Transaktionen – unterstützt, bei denen innerhalb einer Transaktion Änderungen an mehreren Datenbanken vorgenommen werden können. Um auch in einem solchen Umfeld die Transaktionssicherheit gewährleisten zu können, braucht man ein ausgefeiltes Konzept – den so genannten *two-phase commit* oder, auf Deutsch, die *zweiphasige Festschreibung* (siehe auch Abschnitt 6.6 „Zweiphasige Festschreibung (two-phase commit)“).

6.2 Die lokale Datenbankverbindung

Für die *lokale* Verbindung zur Datenbank spielt das DRDA-Konzept erst mal keine Rolle.

Lokal bedeutet hier, dass am Datenbank-Server selbst eine Verbindung zu dem aktuellen Exemplar (instance) hergestellt wird, das in der Umgebungsvariable DB2INSTANCE gesetzt ist.

Eine lokale Datenbankverbindung wird über einen

```
connect to <db_alias_name>
```

initiiert und benötigt keine Benutzer-ID-Angabe und kein Kennwort, da beides standardmäßig aus der Betriebssystem-Umgebung übernommen wird. Die Datenbank wird dabei *immer* über ihren Aliasnamen angesprochen, der – falls nicht extra beim CREATE DATABASE angegeben – dem Datenbanknamen entspricht.

Was passiert nun genau bei diesem CONNECT-Befehl?

Als Erstes wird überprüft, ob der angegebene Datenbankalias überhaupt im Exemplar bekannt ist. Dazu gibt es ein so genanntes *Systemdatenbankverzeichnis*.

6.2.1 Das Systemdatenbankverzeichnis

Im Systemdatenbankverzeichnis (System Database Directory) sind alle Datenbanken registriert, auf die aus dem aktuellen Exemplar ein Zugriff möglich sein soll. Es gibt also ein Systemdatenbankverzeichnis pro Exemplar, sozusagen ein „Datenbankzugriffsverzeichnis“. Dabei unterscheidet man bei den Einträgen des Systemdatenbankverzeichnisses zwischen:

- lokalen Datenbanken
- entfernten (remote) Datenbanken

In diesem Kapitelabschnitt wollen wir uns mit den lokalen Datenbanken beschäftigen, während die entfernten Datenbanken dann im Abschnitt 6.4 „Datenbankverbindung zu einem entfernten DB2-Datenbank-Server“ folgen.

Schauen wir uns dieses Systemdatenbankverzeichnis an. Über den Befehl LIST DATABASE DIRECTORY bzw. LIST DB DIRECTORY wird das Systemdatenbankverzeichnis angezeigt (siehe Abbildung 6.2).

```
D:\SQLLIB\BIN>db2 list db directory
```

Systemdatenbankverzeichnis

Anzahl Einträge im Verzeichnis = 3

Eintrag für Datenbank 1:

Aliasname der Datenbank	= TOOLSDB
Datenbankname	= TOOLSDB
Knotenname	= TP_A30
Release-Stand der Datenbank	= a.00
Kommentar	= Toolskatalog(in DB2 Exemplar)
Verzeichniseintragsart	= Fern
Datenbankpartitionsnummer für Katalog	= -1

Eintrag für Datenbank 2:

Aliasname der Datenbank	= BUCHDB
Datenbankname	= BUCHDB
Datenbanklaufwerk	= D:\BUCHINST
Release-Stand der Datenbank	= a.00
Kommentar	= Datenbank für das DB2 Buch
Verzeichniseintragsart	= Indirekt
Datenbankpartitionsnummer für Katalog	= 0

Eintrag für Datenbank 3:

Aliasname der Datenbank	= SAMPLE
Datenbankname	= SAMPLE
Datenbanklaufwerk	= D:\BUCHINST
Release-Stand der Datenbank	= a.00
Kommentar	=
Verzeichniseintragsart	= Indirekt
Datenbankpartitionsnummer für Katalog	= 0

Abbildung 6.2: Beispiel: Systemdatenbankverzeichnis mit der lokalen Datenbank BUCHDB

Der mehrzeilige Eintrag für jede lokale Datenbank (hier z.B. für die Datenbank BUCHDB) beinhaltet:

- ▶ den Aliasnamen der Datenbank – dieser muss im CONNECT-Befehl angegeben werden
- ▶ den Datenbanknamen
- ▶ das Datenbanklaufwerk – der Pfad bzw. das Laufwerk der Datenbank
- ▶ den Release-Stand der Datenbank – das ist eine interne Bezeichnung für die Datenbankversion
- ▶ einen Kommentar – dieser kann vom Benutzer vergeben werden und Zusatzinformationen zur Datenbank beinhalten
- ▶ die Verzeichniseintragsart – dabei gibt es zwei Möglichkeiten:
 - *Indirekt* (indirect) bedeutet hierbei lokal im aktuellen Exemplar
 - *Fern* (remote) bedeutet hierbei, dass sich die Datenbank in einem anderen Exemplar – also eventuell auch auf einer anderen Maschine – befindet.
- ▶ die Datenbankpartitionsnummer für Katalog:
 - 0 ist hier Standard für nicht-partitionierte Exemplare.
 - -1 ist Standard für entfernte Exemplare.

Die Einträge in dieses Systemdatenbankverzeichnis erfolgen für lokale Datenbanken automatisch bei der Erstellung, also während des CREATE DATABASE-Befehls. Wenn eine Datenbank in einem Exemplar erstellt wird, soll es natürlich auch möglich sein, auf diese zuzugreifen.

Für „ferne“ Datenbanken muss dies explizit erfolgen, da es für das System keine Möglichkeit gibt, einen CREATE DATABASE-Befehl auf einem anderen System „mitzubekommen“. Die Registrierung einer entfernten Datenbank im Systemdatenbankverzeichnis erfolgt über einen CATALOG DATABASE -Befehl, den wir später genauer besprechen.

Indirekt bzw. lokal bedeutet ja – wie oben beschrieben –, dass die Datenbank sich im aktuellen Exemplar befindet. Da aber das Exemplar und die Datenbank logische Einheiten sind, ist damit noch nicht klar, wo sich die Datenbank physikalisch befindet. Selbst der Eintrag Datenbanklaufwerk genügt nicht, um die physikalische Platzierung einer Datenbank eindeutig zu bestimmen. In Abbildung 6.2 haben z.B. die Datenbank BUCHDB und SAMPLE genau die gleichen Einträge.

Es kann also beim CONNECT auf das Systemdatenbankverzeichnis nicht sofort auf die Datenbank zugegriffen werden, sondern zuerst muss noch der genaue physikalische Ort bestimmt werden.

Warum dieser Aufwand?

Die Antwort hat mit der Flexibilität zu tun, mit der Sie Datenbanken gezielt auf Ihre Festplatten verteilen bzw. platzieren können. Beim CREATE DATABASE-Befehl haben Sie die Möglichkeit, durch die ON <path/laufwerk>-Option zu bestimmen, wo – also in welches Dateisystem und somit auf welche Platte – die Datenbank erstellt werden soll.

Der CONNECT kann nun nicht Ihr ganzes System nach der gewünschten Datenbank durchsuchen, sondern schaut zentral – an einer definierten Stelle (z.B. ...\\SQL-LIB\\BUCHINST\\SQLDBDIR\\SQLDBDIR) – in dem Systemdatenbankverzeichnis nach, ob die Datenbank „bekannt“ ist. Danach wird über den Eintrag Datenbanklaufwerk in einem weiteren Verzeichnis – dem *lokalen Datenbankverzeichnis* – versucht, die Stelle zu finden, wo die Datenbank tatsächlich physikalisch gespeichert ist.

Die Antwort ist also:

Um Datenbanken individuell platzieren zu können, ist eine zusätzliche Stufe (Indirektion) notwendig.

Handelt es sich um eine lokale Datenbank mit dem Eintrag Verzeichniseintragsart = Indirekt, springt DB2 mit Hilfe des Eintrags Datenbanklaufwerk in das *lokale Datenbankverzeichnis*.

6.2.2 Das lokale Datenbankverzeichnis

Das lokale Datenbankverzeichnis enthält alle Datenbanken, die auf dem jeweiligen Laufwerk (z.B. D:) bzw. auf dem jeweils angegebenen Pfad liegen.

Sie erreichen das lokale Datenbankverzeichnis über den Befehl:

```
LIST DATABASE DIRECTORY ON <datenbanklaufwerk/pfad>
```

Dabei geben Sie dem Befehl LIST DB DIRECTORY den Eintrag Datenbanklaufwerk des Systemdatenbankverzeichnisses mit.

```
D:\\SQLLIB\\BIN>db2 list db directory on D:\\buchinst

Lokales Datenbankverzeichnis auf D:\\buchinst

Anzahl Einträge im Verzeichnis = 2

Eintrag für Datenbank 1:

Aliasname der Datenbank           = BUCHDB
Datenbankname                     = BUCHDB
Datenbankverzeichnis              = SQL00001
Release-Stand der Datenbank       = a.00
Kommentar                         = Datenbank für das DB2 Buch
Verzeichniseintragsart            = Lokal
Datenbankpartitionsnummer für Katalog = 0
Datenbankpartitionsnummer        = 0

Eintrag für Datenbank 2:

Aliasname der Datenbank           = SAMPLE
Datenbankname                     = SAMPLE
Datenbankverzeichnis              = SQL00002
Release-Stand der Datenbank       = a.00
Kommentar                         =
Verzeichniseintragsart            = Lokal
Datenbankpartitionsnummer für Katalog = 0
Datenbankpartitionsnummer        = 0
```

Abbildung 6.3: Beispiel für ein lokales Datenbankverzeichnis

Im lokalen Datenbankverzeichnis finden Sie folgende Einträge:

- ▶ Aliasname der Datenbank – entspricht hier immer dem Datenbanknamen und hat keine weitere Bedeutung.
- ▶ Datenbankname – Dieser Name entspricht dem Datenbanknamen im Systemdatenbankverzeichnis.
- ▶ Datenbankverzeichnis – der Name des physikalischen Verzeichnisses, in dem sich die Datenbankstruktur befindet – das so genannte Stammverzeichnis.
- ▶ Release-Stand der Datenbank – zeigt die interne Version der Datenbank.
- ▶ Kommentar – Zusatzinformation, die beim CREATE DATABASE angegeben wurde.
- ▶ Verzeichniseintragungsart – *lokal* weist hier darauf hin, dass sich die Datenbank auf diesem Laufwerk befindet.
- ▶ Datenbankpartitionsnummer für Katalog und Datenbankpartitionsnummer spielen nur in partitionierten Umgebungen eine Rolle.

Das Stammverzeichnis bzw. das Datenbankverzeichnis befindet sich in der Verzeichnishierarchie, die wir in Kapitel 5.4 vorgestellt haben, also unter einem Verzeichnis mit dem Exemplarnamen und einem NODE0000-Verzeichnis – in unserem Beispiel unter:

```
D:\BUCHINST\NODE0000\SQL00001
```

Das lokale Datenbankverzeichnis wird bei der Erstellung der Datenbank angelegt bzw. um die neue Datenbank erweitert. Eine manuelle Anpassung ist nicht möglich. Dieses Datenbankverzeichnis existiert nur auf dem Datenbank-Server, der die Datenbank enthält.

6.2.3 Schaubild einer lokalen Datenbankverbindung

Abbildung 6.4 fasst den Ablauf eines lokalen Verbindungsaufbaus und der notwendigen Übereinstimmungen der Datenbankverzeichnisse noch einmal zusammen.

Im CONNECT-Befehl wird der Datenbankaliasname angegeben – Punkt ①. Über diesen Namen wird der zugehörige Eintrag im Systemdatenbankverzeichnis (①) gefunden. Durch den Eintrag Indirekt (Punkt ②) wird auf ein lokales Datenbankverzeichnis verwiesen, das sich in einer von DB2 vorgegebenen Verzeichnisstruktur auf dem angegebenen Datenbanklaufwerk (ebenfalls Punkt ②) befindet.

Der Datenbankname des Systemdatenbankverzeichnisses verweist auf den zugehörigen Eintrag im lokalen Datenbankverzeichnis – Punkt ③. Damit kann nun auf das physikalische Verzeichnis der Datenbank zugegriffen werden. Der Verbindungsaufbau ist erfolgreich!

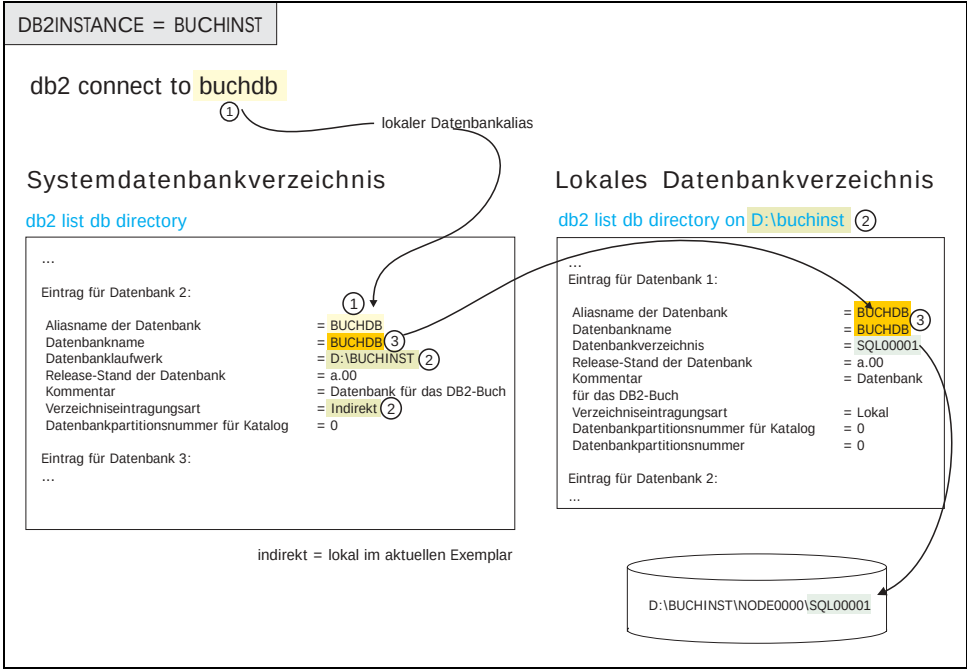


Abbildung 6.4: Schaubild – lokale Datenbankverbindung

6.3 Authentifizierung

Bevor wir den Verbindungsaufbau zu einer entfernten Datenbank betrachten, müssen wir noch einige Sicherheitsüberlegungen anstellen, die die Authentifizierung betreffen.

Unter Authentifizierung versteht man die Identifikation eines Benutzers unter Sicherheitsaspekten. DB2 benutzt dazu keine eigene Benutzerverwaltung, sondern greift auf die Benutzerverwaltung des Betriebssystems bzw. auf die Benutzerverwaltung eines separaten Produkts – wie Kerberos – zurück.

Allen ist gemeinsam, dass ein Benutzer eine gültige Kombination aus Benutzer-ID und Kennwort haben muss, um an einem System und mit DB2 arbeiten zu können.

Ist ein Benutzer erst einmal angemeldet – also authentifiziert –, wird seine Benutzer-ID als SQL-Berechtigungsname übernommen. Er unterliegt zusätzlich den Sicherheitsrichtlinien der Datenbank, die definieren, was ein authentifizierter Benutzer „darf“. Eine Benutzer-ID und ein Passwort berechtigen damit einen Benutzer noch nicht dazu, Daten einzusehen, zu erstellen oder zu verändern! Dazu sind zusätzlich die entsprechenden SQL-Berechtigungen in der Datenbank notwendig, die wir im Sicherheitskapitel behandeln.

Für die Authentifizierung stehen im DB2 mehrere Möglichkeiten zur Verfügung, von denen wir zunächst die Hauptarten aufgreifen:

- ▶ Authentifizierung am Server
- ▶ Authentifizierung am Client
- ▶ Authentifizierung mit Kerberos

Die jeweilige Authentifizierungsart wird auf Exemplarebene in der Datenbankmanager-Konfigurationsdatei – im Parameter `AUTHENTICATION` – festgelegt.

6.3.1 Server-Authentifizierung

Der Eintrag in der Datenbankmanager-Konfiguration

Authentifizierung des Datenbankmanagers (`AUTHENTICATION`) = `SERVER`

ist die Standardeinstellung für jedes Exemplar. Er bedeutet, dass für die Authentifizierung am Datenbank-Server eine – am Server – gültige Benutzer-ID mit zugehörigem Kennwort angegeben werden muss.

Für Ihren Verbindungsaufbau heißt das, dass Sie den `CONNECT`-Befehl folgendermaßen absetzen müssen:

```
connect to buchdb user <benutzer-id> using <kennwort>
```

Alternativ können Sie den Befehl so eingeben, dass Ihr Kennwort nicht im „Klartext“ zu sehen ist:

```
connect to buchdb user <benutzer-id> [Eingabe]
```

Aktuelles Kennwort für <benutzer-id> eingeben:

Die zweite Zeile mit der Aufforderung, das Kennwort einzugeben, taucht auf, sobald Sie den Befehl mit der Eingabe-Taste bestätigt haben. Die Kennworтеingabe erfolgt dann nicht sichtbar.

Eine Ausnahme bei diesem Verfahren ist die lokale Datenbankverbindung. Bei ihr werden automatisch der Benutzername und das Kennwort übernommen, mit denen Sie sich lokal am System angemeldet haben. Die `USER`-Option beim `CONNECT` ist somit nicht notwendig, kann aber z.B. benutzt werden, um gegenüber der Datenbank als ein „anderer“ Benutzer aufzutreten. Dies setzt allerdings voraus, dass Sie mehrere gültige Benutzer-ID-Kennwort-Kombinationen auf dem System haben.

Maßgeblich ist also der `AUTHENTICATION`-Konfigurationsparameter des Zielexemplars, in dem sich die (entfernte) Datenbank befindet.

Bei dieser Technik werden die Benutzer-ID und das Kennwort im Klartext – also unverschlüsselt – übertragen. Das stellt ein Sicherheitsrisiko dar, wenn das Netz nicht anderweitig geschützt ist.

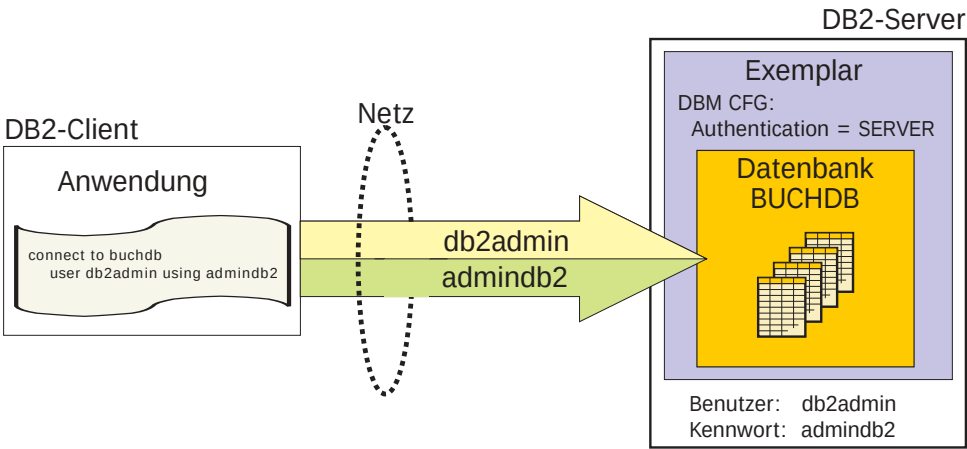


Abbildung 6.5: Funktionsweise von AUTHENTICATION = SERVER

SERVER_ENCRYPT

Eine spezielle Server-Authentifizierung – Authentication = SERVER_ENCRYPT – stellt sicher, dass diese Übertragung vom Client zum Server verschlüsselt erfolgt. Dazu wird auf dem Client das Kennwort verschlüsselt, bevor es zum Server übertragen wird. Auf dem Server wird das Kennwort dann entschlüsselt und überprüft.

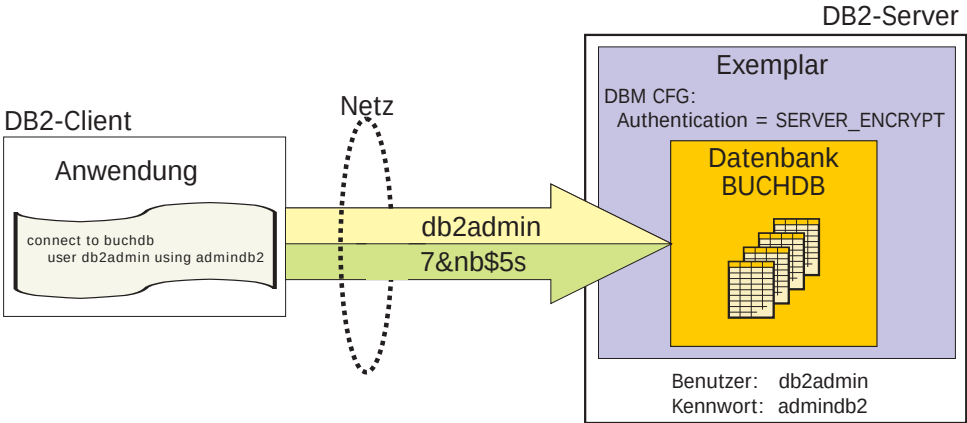


Abbildung 6.6: Funktionsweise von AUTHENTICATION = SERVER_ENCRYPT

6.3.2 Client-Authentifizierung

Bei dieser Einstellung – also wenn auf dem Zielexemplar die Datenbankmanager-Konfiguration den Wert **AUTHENTICATION = CLIENT** aufweist – erfolgt die Authentifizierung im Betriebssystem des Clients (siehe Abbildung 6.7). Der Server führt keine zusätzliche eigene Authentifizierung mehr durch. Dies gilt allerdings nur, wenn die anderen Sicherheitsparameter (**TRUST_ALLCLNTS** und **TRUST_CLNTAUTH**), die mit der Authentifizierung zusammenhängen, die Standardwerte haben.

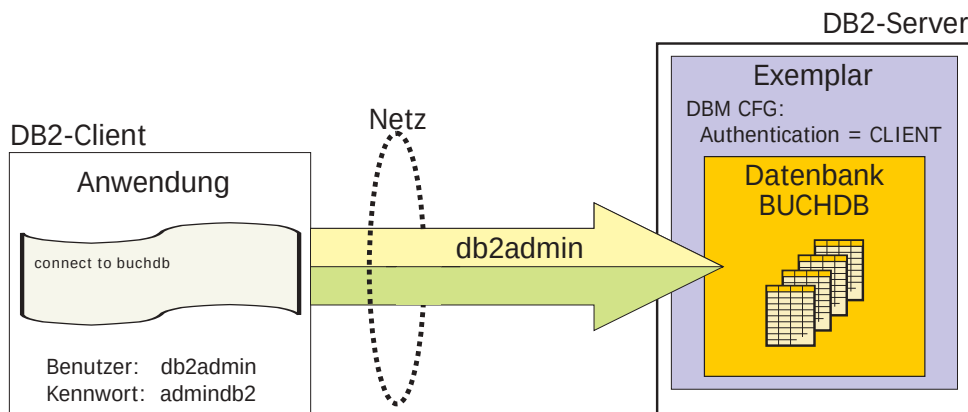


Abbildung 6.7: Funktionsweise von AUTHENTICATION = CLIENT

Mit der Client-Authentifizierung (AUTHENTICATION = CLIENT) werden auch die Datenbankmanager-Parameter TRUST_ALLCLNTS und TRUST_CLNTAUTH wirksam, während sie bei der Server-Authentifizierung unwirksam sind. Durch diese Parameter kann die Client-Authentifizierung noch genauer definiert werden, um diese – vom Prinzip her – relativ unsichere Architektur zu verwalten.

TRUST_ALLCLNTS

Da die Clients die unterschiedlichsten Betriebssysteme aufweisen können, kann nicht sichergestellt werden, dass auch eine vertrauenswürdige Authentifizierung durchgeführt werden kann. Betriebssysteme wie Windows 98 oder Windows ME haben keine integrierte sichere Benutzerverwaltung und werden daher von DB2 als „unsicher“ eingestuft.

Der Datenbankmanager-Parameter TRUST_ALLCLNTS kann auf NO gesetzt werden, um anzuzeigen, dass diese unsicheren Clients – trotz Client-Authentifizierung – ihr Kennwort mitsenden müssen, um sich am Server zu authentifizieren.

Der Standardwert ist TRUST_ALLCLNTS = YES, d.h., auch den unsicheren Clients wird „vertraut“.

Eine weitere Einstellmöglichkeit ist TRUST_CLNTAUTH = DRDAONLY. Dabei wird nur folgenden Clients „vertraut“: DB2 für OS/390 und z/OS, DB2 für VM und VSE und DB2 für OS/400. Alle anderen benötigen eine Benutzer-ID und ein Kennwort für die Authentifizierung am Server.

Sichere Betriebssysteme – wie Unix-Systeme oder Windows-Server-Betriebssysteme (Windows NT, Windows 2000 etc.) – haben eine eigene Benutzerverwaltung und sind von diesem Parameter nicht betroffen.

TRUST_CLNTAUTH

Dieser Parameter wird ebenfalls in der Datenbankmanager-Konfiguration des Ziel-Exemplars gesetzt und ist standardmäßig mit dem Wert CLIENT vorbelegt.

Die Client-Authentifizierung bedeutet, dass keine Angabe der Benutzer-ID und des Kennworts für einen CONNECT notwendig ist.

Über diese Konfiguration kann angegeben werden, was passiert, also wo authentifiziert wird, wenn trotzdem eine Benutzer-ID und ein Kennwort angegeben wird. Im Standardfall – mit `TRUST_CLNAUTH = CLIENT` – wird dies am Client durchgeführt. Bei der Einstellung `TRUST_CLNAUTH = SERVER` wird trotz Client-Authentifizierung (`AUTHENTICATION = CLIENT`) das Kennwort zum Datenbank-Server übertragen und dort authentifiziert (siehe Abbildung 6.8).

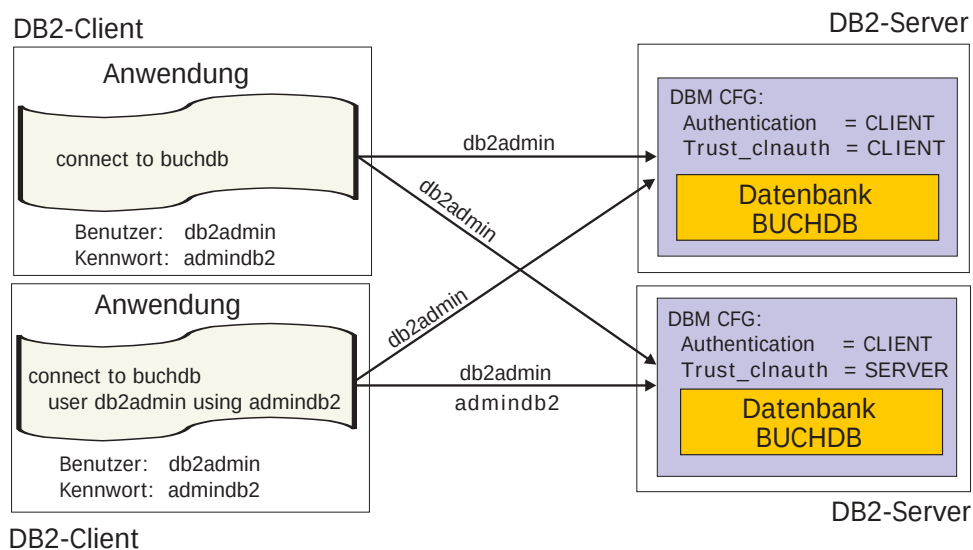


Abbildung 6.8: Funktionsweise von `TRUST_CLNAUTH`

6.3.3 Kerberos-Authentifizierung

Kerberos ist eine Authentifizierungstechnik, die eingesetzt wird, um eine sichere Authentifizierung zwischen Clients und Servern im Netz zu gewährleisten, bei der keine Kennwörter unverschlüsselt übertragen werden. Dafür wird ein separater Kerberos-Server verwendet. Das Kerberos-Protokoll ist sowohl als freie wie auch als kommerzielle Software verfügbar.

Die DBM-Konfiguration `AUTHENTICATION = KERBEROS` ist momentan nur auf Windows 2000-Servern verfügbar! Sowohl der Server als auch der Client müssen dabei das Kerberos-Protokoll unterstützen.

Durch die Einstellung `AUTHENTICATION = KRB_SERVER_ENCRYPT` wird für Clients, die kein Kerberos unterstützen, die Authentifizierungstechnik `SERVER_ENCRYPT` durchgeführt, also die verschlüsselte Übertragung und Authentifizierung am Server.

6.3.4 Authentifizierungskonfiguration auf dem Client

Obwohl die Konfiguration der Authentifizierung im Exemplar auf dem DB2-Datenbank-Server festgelegt wird, kann diese Einstellung bei der Einrichtung von Clients noch einmal explizit angegeben werden.

Diese Einstellung sehen Sie später bei der Client-Konfiguration.

Warum ist das notwendig bzw. was für Vorteile bringt es?

Als Erstes ist diese Prozedur nicht unbedingt gefordert, da, wenn keine Angabe erfolgt, automatisch die Einstellung vom Server übernommen wird und dies auch die gültige Definition ist. Sonst könnte jemand auf einem Client ja die CLIENT-Authentifizierung angeben, um sich lokal zu authentifizieren, wenn eigentlich im Exemplar `Authentication = SERVER` gesetzt ist. So etwas geht natürlich nicht.

Der Vorteil besteht darin, dass der Client beim Verbindungsaufbau nicht erst die Authentifizierungsmethode vom Server abfragen muss, was einen Zusatzaufwand darstellt der unnötig ist, wenn der Client die Authentifizierungsart des Datenbank-servers kennt. Dies stellt somit einen Performance-Vorteil dar.

Hinweis:

Dadurch kann es aber zu Problemen kommen, wenn die Angabe am Client falsch – also nicht mit dem Server übereinstimmend – gesetzt wird. Folgende Kombination führt zu einem Verbindungsfehler beim `CONNECT`-Versuch:

Client: `AUTHENTICATION = SERVER_ENCRYPT`

Server: `AUTHENTICATION = SERVER`

In diesem Fall kann der Server das verschlüsselte Kennwort des Clients nicht als gültig erkennen.

6.4 Datenbankverbindung zu einem entfernten DB2-Datenbank-Server

Die Verbindung zu einem entfernten Datenbank-Server erfordert Konfigurationen sowohl am Client als auch am Server. Die notwendigen Einstellungen betrachten wir aber erst, nachdem wir das Konzept des Verbindungsaufbaus besprochen haben.

Wie im Abschnitt 6.2 „Die lokale Datenbankverbindung“ für die lokale Datenbankverbindung erwähnt wurde, setzt der `CONNECT`-Befehl zum entfernten DB2-Server auf das `Systemdatenbankverzeichnis` auf. In diesem Verzeichnis werden neben den lokal verfügbaren Datenbanken am Server auch die fernen Datenbanken aufgeführt. Es stellt also eine vollständige Liste der Datenbanken dar, zu denen eine Verbindung aufgebaut werden kann.

Abbildung 6.9 zeigt das `Systemdatenbankverzeichnis` eines Clients (RTC oder Datenbank-Server), der auf den entfernten DB2-Server zugreifen will, der die Datenbank `BUCHDB` beinhaltet. Das `Systemdatenbankverzeichnis` existiert auf jedem

Server und jedem Client, sobald eine Datenbank „katalogisiert“, also eingetragen, wird. Dies passiert manuell am Client oder am Datenbank-Server auch automatisch, falls eine lokale Datenbank erstellt wird.

```
D:\Programme\IBM\SQLLIB\BIN>db2 list db directory

Systemdatenbankverzeichnis

Anzahl Einträge im Verzeichnis = 1

Eintrag für Datenbank 1:

Aliasname der Datenbank           = BUCHDB
Datenbankname                     = BUCHDB
Knotenname                       = TCPA590D
Release-Stand der Datenbank       = a.00
Kommentar                        = Datenbank für das DB2 Buch
Verzeichniseintragsart           = Fern
Datenbankpartitionsnummer für Katalog = -1
```

Abbildung 6.9: Beispiel: Systemdatenbankverzeichnis mit der fernen Datenbank BUCHDB

Im Gegensatz zu einer lokalen Datenbank (vgl. Abbildung 6.2) betrachten wir nun die Verzeichniseintragsart = Fern. Dies bedeutet, die Datenbank BUCHDB befindet sich nicht im aktuellen Exemplar.

Außerdem wird der zusätzliche Eintrag Knotenname zur Datenbank angezeigt. Dieser ist der Verweis auf ein sog. *Knotenverzeichnis* (oder *node directory*), das die weiteren Verbindungsinformationen beinhaltet, um sich zum entfernten Datenbank-Server verbinden zu können. Denn nur zu sagen: Bitte eine Verbindung zur Datenbank BUCHDB aufbauen, die auf einem fernen Server liegt! genügt natürlich nicht.

Sie können das gut mit einem Adressbuch vergleichen. Wenn Sie z.B. im Urlaub sind und zu dem – nun entfernten – Freund Kontakt aufnehmen wollen, müssen Sie einen Blick in Ihr Adressbuch werfen, um die genauen Adressdaten wie Straße, Hausnummer und Postleitzahl herauszufinden. Mit diesen Adressdaten können Sie ihm dann eine Postkarte schicken.

Der Eintrag Verzeichniseintragsart = Fern bedeutet, dass im Knotenverzeichnis weitere Daten für die Verbindung zur Verfügung stehen und dort unter dem Knotennamen TCPA590D zu finden sind. Dieser Knotenname kann beliebig gewählt werden und muss nur zwischen dem Systemdatenbankverzeichnis und dem Knotenverzeichnis übereinstimmen.

Werfen wir einen Blick in das zugehörige Knotenverzeichnis dieses Clients (siehe Abbildung 6.10).

Das Knotenverzeichnis wird über den Befehl LIST NODE DIRECTORY aufgerufen. Es existiert einmal pro Exemplar bzw. pro Client und kann beliebig viele Einträge (Knoteneinträge) beinhalten.

```
D:\Programme\IBM\SQLLIB\BIN>db2 list node directory
```

Knotenverzeichnis

Anzahl Einträge im Verzeichnis = 1

Eintrag für Knoten 1:

Knotenname	= TCPA590D
Kommentar	=
Verzeichniseintragungsart	= LOCAL
Protokoll	= TCPIP
Host-Name	= 192.168.60.43
Servicename	= 55000

Abbildung 6.10: Beispiel für ein Knotenverzeichnis (node directory)

In unserem Beispiel finden wir nur einen Eintrag mit dem Namen TCPA590D. Er muss wie gesagt mit dem Knotennamen (ein- bis achtstellig) im Systemdatenbankverzeichnis übereinstimmen. Ihm sind nun folgende weitere Werte zugeordnet:

- ▶ **Kommentar** – Hier kann optional ein beliebiger Kommentar zur Identifizierung des entfernten Servers angegeben werden.
 - ▶ **Verzeichniseintragungsart** – zwei Möglichkeiten stehen zur Auswahl:
 - LOCAL – Es handelt sich um ein lokales (Knoten-)Verzeichnis.
 - LDAP – Das Verzeichnis liegt auf einem LDAP-Server (mehr dazu in Abschnitt 6.8.3).
 - ▶ **Protokoll** – Mit welchem Protokoll soll die Verbindung zum Datenbank-Server aufgebaut werden? Zur Auswahl stehen:
 - TCPIP – TCP/IP wird weitaus am häufigsten verwendet und im Folgenden vorausgesetzt.
 - APPC – Die „Advanced Program-to-Program Communication“ wird für die SNA-Architektur genutzt.
 - NETBIOS oder NPIPE – Verfahren nur für die Kommunikation zwischen Windows-Maschinen
 - ▶ **Host-Name** – Adresse oder Name des Servers
- Beim Eintrag Host-Name kann ein Servername angegeben werden. Dieser muss dann aber im System bekannt sein bzw. aufgelöst werden können, muss also einer gültigen IP-Adresse zugeordnet werden können. Alternativ kann – wie im Beispiel – die IP-Adresse direkt verwendet werden.
- ▶ **Servicename** – Port oder Portname des entfernten Exemplars
- Der Servicename entspricht der TCP/IP-Portnummer und kann ebenfalls entweder, wie hier gezeigt, direkt als Port oder über einen Namen eingetragen werden (der in der Datei etc/services in die Portnummer aufgelöst wird).

Details zu TCP/IP-Host-Namen und Servicenamen

Wir Menschen können uns Computernamen besser merken als irgendwelche Zahlen. Der Computer braucht für die Identifikation eines Rechners jedoch einen Zahlencode – die so genannte IP-Adresse. Sie besteht aus vier Zahlen zwischen 0 und 255, die jeweils durch einen Punkt voneinander getrennt sind (z.B. 192.168.1.10).

Sie können nun im Knotenverzeichnis entweder direkt die IP-Adresse oder den Computernamen eintragen. Dieser muss dann aber vom System in die richtige zugehörige IP-Adresse „aufgelöst“ werden.

Diese Auflösung kann lokal über die hosts-Datei am Rechner erfolgen. Die „hosts“-Datei – der Name ist standardisiert – liegt im Verzeichnis etc, unter Unix in /etc/hosts und unter Windows in %WINDIR%\system32\drivers\etc\hosts. Alternativ kann der Rechnername im Netz durch einen so genannten DNS (Domain Name Server) aufgelöst werden, der genau für diese Aufgabe zur Verfügung steht und in der Netzkonfiguration des Rechners angegeben sein muss.

Die Struktur der hosts-Datei, die im ASCII-Format vorliegt, sieht so aus:

```
127.0.0.1      localhost
192.168.60.43 mtserver1
```

Der einzelnen IP-Adresse wird ein System- bzw. Rechnername gegenübergestellt – so kann eine Zuordnung stattfinden.

Der Servicename wird der so genannten TCP/IP-Portnummer zugeordnet. Die Portnummer wird dazu verwendet, verschiedene Kommunikationsverbindungen an einem Server zu unterscheiden. So hat z.B. die DB2-Kommunikation einen anderen Port als eine Web-Verbindung (http-Verbindung) oder eine Telnet-Verbindung. Genau wie beim Hostnamen kann hier direkt eine Portnummer angegeben werden – in unserem Beispiel ist das 55000 – oder ein Name, der dann in der Datei services aufgelöst wird. Der Speicherort dieser Datei entspricht den Verzeichnissen in der hosts-Datei.

Die Struktur der services-Datei, die ebenfalls im ASCII-Format vorliegt, sieht so aus:

```
...
db2cDB2      50000/tcp      #connection port for the DB2 instance DB2
db2cDB2CTLSV 50002/tcp      #connection port for the DB2 instance DB2CTLSV
...
```

In dieser Datei liegen sehr viele Einträge vor. Die unteren (kleinen) Portnummern sind alle standardisiert (Details finden Sie unter: <http://www.iana.org/assignments/port-numbers>), die restlichen Portnummern (bis max. 65.535) können frei vergeben werden, müssen aber eindeutig sein.

Der Hostname und der Servicename fungieren im TCP/IP-Protokoll in etwa wie die Postleitzahl bzw. der Ortsname und die Hausnummer in Ihrem Adressbuch.

Mit dieser Kombination aus Systemdatenbankverzeichnis (System Database Directory) und Knotenverzeichnis (node directory) kann nun vom Client oder einem anderen DB2-Datenbank-Server eine Verbindung zum Server aufgebaut werden – vorausgesetzt, dort sind alle für die Kommunikation notwendigen Einstellungen vorgenommen worden.

6.4.1 Vorbereitung eines DB2-Servers

Nach der Erstellung eines Exemplars (siehe Kapitel 3, „Installation der DB2 UDB“) müssen zusätzliche Konfigurationen durchgeführt werden, um Verbindungen entfernter Clients verarbeiten zu können, während lokale Verbindungen sofort und ohne weiteres Zutun funktionieren.

Die Kommunikation wird in DB2 auf Exemplarebene konfiguriert. Wir beschränken uns im Folgenden auf die Verbindung mit TCP/IP, da diese Variante mit Abstand am häufigsten in der Praxis vorkommt.

Protokollunterstützung – DB2COMM

Für jedes unterstützte Protokoll (TCP/IP, APPC, NETBIOS, NPIPE) gibt es in DB2 einen eigenen Verbindungsmanager. Dieser muss beim Start des Exemplars mit gestartet werden, um für eingehende Anfragen über das entsprechende Protokoll bereitzustehen.

DB2 startet die Protokollunterstützung bzw. die Verbindungsmanager für alle in der Registriervariable DB2COMM angegebenen Protokolle.

Über folgenden Befehl wird die – standardmäßig nicht belegte – Variable gesetzt:

```
db2set DB2COMM=TCPIP
```

Wird die Unterstützung mehrerer Protokolle gewünscht, können diese – durch Kommata getrennt – hintereinander angegeben werden, also so:

```
db2set DB2COMM=TCPIP, NETBIOS, NPIPE
```

Bei dem nächsten Start des Exemplars werden nun diese Protokoll-Verbindungsmanager gestartet.

Zuordnung des TCP/IP-Ports

Wie Sie in Abbildung 6.9 und Abbildung 6.10 gesehen haben, soll von einem Client die Verbindung zur Datenbank BUCHDB hergestellt werden. Dazu wurde neben dem Datenbanknamen auch die IP-Adresse des Servers angegeben. Wie aber findet der Client das „richtige“ Exemplar – also das Exemplar mit der BUCHDB-Datenbank, die angesprochen werden soll?

Das funktioniert über den Servicenamen (siehe Abbildung 6.10) im Knotenverzeichnis (node directory). Am DB2-Server muss jedem Exemplar ein eigener TCP/IP-Port zugeteilt werden. Somit wird also das Exemplar über die Portnummer bestimmt.

Die Einstellung bzw. Konfiguration dieser Portnummer geschieht in der Datenbankmanager-Konfiguration – der Konfiguration des Exemplars. Dort wird der Parameter SVCENAME auf die Portnummer oder den entsprechenden Portnamen gesetzt, der über die Datei etc/services aufgelöst werden kann.

Der zugehörige Befehl, um für das aktuelle Exemplar BUCHINST den Portnamen db2cBUCHINST zu setzen, lautet:

```
update dbm cfg using svcename db2cBUCHINST
```

Genauso könnte eine Portnummer gesetzt werden.

Tipp:

Haben Sie mehrere Exemplare auf einem Server, empfiehlt es sich, Portnamen zu verwenden und diesen in der Datei etc/services entsprechende Portnummern zuzuweisen, um systemweit die Eindeutigkeit dieser Portnummer gewährleisten zu können. So vermeiden Sie es, bei der Fehlersuche jede Exemplarkonfiguration auf eventuell mehrfach verwendete Ports untersuchen zu müssen.

Für unser Beispiel muss nun zusätzlich die Datei etc/services um folgenden Eintrag erweitert werden:

```
db2cBUCHINST 55000/tcp #DB2 Exemplar BUCHINST - Eintrag von: MT
```

Das Zeichen „#“ markiert einen Kommentar. Kommentare sind optional, aber hilfreich, wenn der entsprechende Text dem Administrator bei der Zuordnung helfen kann.

6.4.2 Konfiguration des Clients mit der grafischen Oberfläche

Auf der Client-Seite muss es u.a. die entsprechenden Einträge im Systemdatenbankverzeichnis und Knotenverzeichnis geben, die wir in Abbildung 6.9 und Abbildung 6.10 gezeigt haben. Da diese nicht „von selbst“ dort auftauchen, besteht auch hier Handlungsbedarf.

Um eine solche Client-Server-Verbindung zu konfigurieren, stehen Ihnen drei Möglichkeiten offen:

- ▶ Konfiguration mit Hilfe des GUIs – über den Konfigurationsassistenten oder die Steuerzentrale
- ▶ manuelle Konfiguration mit CATALOG-Befehlen
- ▶ Verwendung vorher erstellter Profile

Da die Server-Konfiguration nur einmal (pro Exemplar) erfolgt, Sie aber wahrscheinlich viele Clients einrichten müssen, zeigen wir Ihnen alle diese Methoden etwas genauer.

Zwei Möglichkeiten stehen Ihnen im GUI zur Verfügung: der Konfigurationsassistent und die Steuerzentrale. Für beide ist ein zusätzliches Verzeichnis – das *administrative Knotenverzeichnis* – wichtig.

Das administrative Knotenverzeichnis (admin node directory)

In der grafischen Oberfläche, z.B. in der Steuerzentrale, werden im Objektbaum Systeme mit einem Computer-Icon dargestellt. Diese DB2-Systeme sind genau genommen nicht die tatsächlichen Maschinen, sondern der DAS (DB2 Administration Server) auf dem (entfernten) DB2-Server, der sozusagen als System-Stellvertreter verwendet wird. In den Objektbaum können so nur DB2-Server aufgenommen werden und keine anderen Maschinen.

Die Systeminformationen der entfernten DB2-Server werden im administrativen Knotenverzeichnis am Client abgelegt. In diesem Zusammenhang kann natürlich auch ein DB2-Server als „Client“ eines anderen DB2-Servers angesehen werden.

Sie können sich die Inhalte durch den Befehl

```
list admin node directory
```

anzeigen lassen, dessen Ergebnis so aussehen kann, wie in Abbildung 6.11 gezeigt.

```
D:\Programme\IBM\SQLLIB\BIN>db2 list admin node directory

Knotenverzeichnis

Anzahl Einträge im Verzeichnis = 2

Eintrag für Knoten 1:

Knotenname           = DASAA90F
Kommentar             =
Verzeichniseintragungsart = LOCAL
Protokoll             = TCPIP
Host-Name             = MTServer1
Servicename           = 523

Eintrag für Knoten 2:

Knotenname           = RS6000
Kommentar             = Forum DB Server
Verzeichniseintragungsart = LOCAL
Protokoll             = TCPIP
Host-Name             = 192.168.62.23
Servicename           = 523
```

Abbildung 6.11: Beispiel für ein administratives Knotenverzeichnis

Der Servicename ist hier immer 523 – ein standardisierter Port für die Kommunikation des IBM DB2-DAS.

Diese Konfigurationsebene spielt nur für die Administration entfernter Systeme mit der grafischen Oberfläche eine Rolle und wird bei der manuellen Konfiguration nicht benötigt, die nur das Ziel hat, eine Datenbankverbindung aufbauen zu können.

Den Grund dafür können Sie in Abbildung 6.12 sehen.

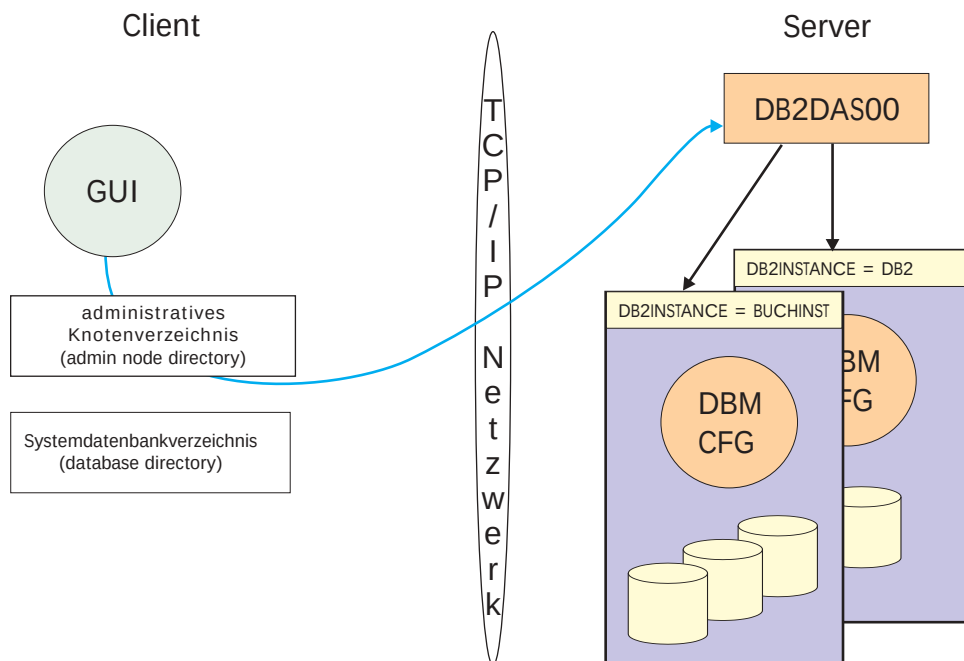


Abbildung 6.12: Funktionsweise des administrativen Knotenverzeichnisses

Durch die Verbindung zum DAS am Server, der dann die administrativen Aufgaben an die Exemplare auf dem DB2-Server weiterreicht, wird wesentlich mehr Flexibilität geschaffen. So kann beispielsweise ein Exemplar auf dem Server gestoppt und gestartet werden, ohne dass es Auswirkungen auf die Client/Server-Verbindung hat.

Die Alternative wäre – wie es übrigens bei der Verwaltung entfernter Server von der Befehlszeile durch den ATTACH-Befehl passiert –, dass der Client die Verbindung *direkt* zum Exemplar aufbaut. So bräuchte man aber nicht nur für jedes Exemplar eine eigene Verbindung, sondern würde sich auch bei einem db2stop sozusagen „den Ast absägen, auf dem man sitzt“.

Mit dem Konfigurationsassistenten

Durch die Eingabe von db2ca auf der Befehlszeile oder über das Windows-Startmenü (START/PROGRAMME/IBM DB2/INSTALLATION- UND KONFIGURATIONSTOOLS/KONFIGURATIONSSASSISTENT) rufen Sie den Konfigurationsassistenten auf.

Abbildung 6.13 zeigt die Oberfläche des Konfigurationsassistenten mit einem bestehenden Eintrag. Bei „neuen“ Datenbank-Clients wäre kein Eintrag vorhanden.

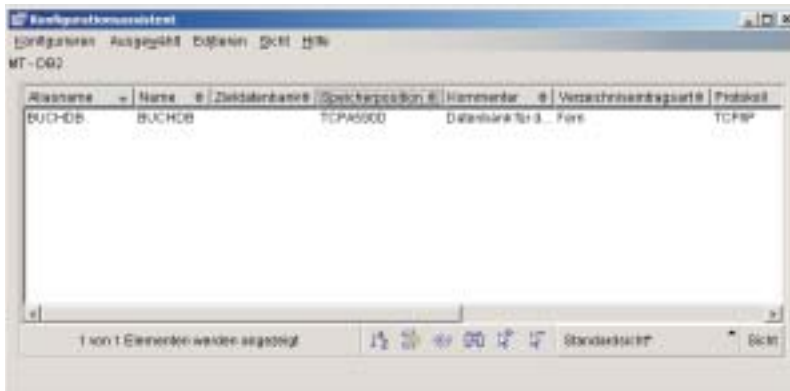


Abbildung 6.13: Der Konfigurationsassistent

Die Möglichkeiten des Menüs KONFIGURIEREN haben wir in Kapitel 4.14 gezeigt. Auch wenn Sie dieses GUI auf einem administrativen Client aufrufen (auf einem „normalen“ Client ist dieses GUI *nicht* enthalten), haben Sie die Möglichkeit, die DBM-Konfiguration und DB2-Registrierdatenbank anzupassen, aber eben nur für die eingeschränkten Möglichkeiten eines DB2-Clients, der ja keine lokalen Datenbanken hat.

Um eine neue ferne Datenbank hinzuzufügen, wählen Sie den Eintrag DATENBANK MIT ASSISTENT HINZUFÜGEN... aus dem AUSGEWÄHLT-Menü aus. Der ASSISTENT: DATENBANK HINZUFÜGEN erscheint (siehe Abbildung 6.14) mit dem ersten Schritt, in dem Sie die Art und Weise der zu tätigenen Konfiguration auswählen können. Zur Auswahl stehen:

- ▶ PROFIL VERWENDEN – Dazu haben wir ein eigenes Teilkapitel – Abschnitt 6.4.5 „Konfiguration des Clients über Profile“.
- ▶ NETZWERK DURCHSUCHEN – Diese Möglichkeit schauen wir uns nun genauer an.
- ▶ VERBINDUNG ZUR DATENBANK MANUELL KONFIGURIEREN – Dieser Teil besteht aus acht Schritten, die grafisch das nachbilden, was wir ebenfalls über die Befehle in Abschnitt 6.4.3 „Konfiguration des Clients durch Befehle“ erreichen.

Greifen wir also zunächst den Punkt NETZWERK DURCHSUCHEN auf. Sobald dieser ausgewählt worden ist, erscheint der zweite von insgesamt vier Schritten dieser Variante.



Abbildung 6.14: Assistent: Datenbank hinzufügen – Schritt 1 – Quelle

Über den WEITER-Button kommen Sie zu Abbildung 6.15.



Abbildung 6.15: Assistent: Datenbank hinzufügen – Schritt 2 – Netzwerk

BEKANNTE SYSTEME sind Systeme, von denen bereits eine Datenbank katalogisiert – also aufgenommen – wurde. Ein Doppelklick auf ANDERE SYSTEME (NETZWERK DURCHSUCHEN) bedeutet, dass DB2 durch einen so genannten „Broadcast“ im Netz versucht, DB2-Server selbstständig zu finden. Ein Broadcast ist eine Anfrage an alle im lokalen Netz angeschlossenen Systeme, ob sie DB2-Server sind. Der DAS (DB2 Administration Server) meldet sich dann auf diese Anfrage hin, falls es nicht anders konfiguriert wurde.

Hinweis:

Die Variante mit ANDERE SYSTEME (NETZWERK DURCHSUCHEN) kann in einem größeren Netzwerk länger dauern und nicht unterbrochen werden.

Diese automatische Suche wird bei sehr sicherheitsrelevanten Systemen, Exemplaren oder Datenbanken dem Administrator ein Dorn im Auge sein.

Um zu vermeiden, dass jemand auf diese Weise bewusst Systeme in einem Firmennetz finden kann, um sich unerlaubt „Zutritt“ zu verschaffen, können Sie Einschränkungen vornehmen. Folgende Konfigurationsparameter dienen dazu, auf den entsprechenden Ebenen eine automatische Rückmeldung auf die Suche zu unterdrücken:

► System-Ebene (DAS) – DISCOVER

Der Standardwert SEARCH erlaubt es, dass dieses System gefunden wird. Ist DISCOVER = KNOWN gesetzt, können nur Clients, die dieses System schon „benutzen“ – denen also dieses System schon bekannt ist –, weiter die Such-Funktionalität nutzen. Die Einstellung DISABLE verhindert das automatische Auffinden dieses DAS-Systems.

► Exemplar-Ebene – DISCOVER

Auf Exemplar-Ebene ist das Verhalten exakt so wie auf System-Ebene, nur dass der DISCOVER-Parameter eben nicht in der DAS-Konfiguration (get admin cfg) gesetzt bzw. geändert werden muss, sondern in der Exemplarkonfiguration. Der Parameter kann die gleichen Werte (SEARCH, KNOWN, DISABLE) annehmen.

► Datenbank-Ebene – DISCOVER_DB

Auf Datenbank-Ebene – also in der Datenbankkonfiguration – kann der Parameter DISCOVER_DB entweder auf dem Standardwert ENABLE bleiben oder auf DISABLE gesetzt werden, was ein automatisches Auffinden der Datenbank verhindert.

Um sich mit einem entfernten Datenbank-Server zu verbinden, sollten Sie schon seine „Kontaktdaten“ kennen. Sie schreiben ja auch keinen Brief an jemanden, von dem Sie keine Adresse haben.

Die notwendigen Informationen – und somit die Vorbereitung einer DB2-Server-Verbindung – sind:

- Rechnername (hostname) und/oder IP-Adresse des Servers
- Exemplarname

- ▶ zugehörige Portnummer – diese kann in diesem GUI auch automatisch erkannt werden
- ▶ Datenbank-Aliasname der Zieldatenbank auf dem DB2-Server

Haben Sie diese Informationen vom Exemplar-Administrator des entfernten Systems bekommen, können Sie den SYSTEM HINZUFÜGEN...-Button auswählen.

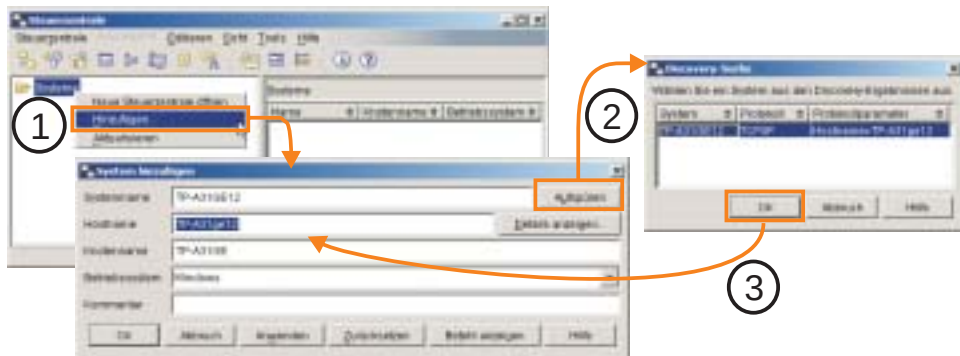


Abbildung 6.16: Assistent: Datenbank hinzufügen – System hinzufügen-Fenster

Der Systemname wird bei der Eingabe auch gleich als Knotenname vergeben – wie erwähnt, spielt die Namenswahl keine wesentliche Rolle. Unter dem HOSTNAMEN geben Sie den Servernamen (Hostnamen) des entfernten Servers ein. Dabei sollte aber sichergestellt sein, dass dieser bekannt ist, so dass der Rechner den Namen in die entsprechende IP-Adresse auflösen kann.

So können Sie das am besten überprüfen:

- ▶ Sitzen Sie lokal am Server, bekommen Sie durch den Befehl `hostname` den Hostnamen der Maschine angezeigt.
- ▶ Um die Namensauflösung zu überprüfen, geben Sie am Client folgenden Befehl ein:

```
ping <hostname>
```

Bekommen Sie daraufhin eine Anzeige, die die IP-Adresse beinhaltet, ist alles in Ordnung.

Kommt die Meldung, dass der Server bzw. der Name nicht gefunden werden kann, müssen Sie die Datei `etc/hosts` um folgende Zeile erweitern:

```
<IP-Adresse>    <hostname>
```

also z.B.:

```
192.168.60.43    mtserver1
```

Alternativ können Sie, wie in Abbildung 6.16 gezeigt, auch die IP-Adresse direkt eingeben.

Haben Sie dieses Fenster mit OK beendet, bekommen Sie das neu hinzugefügte System angezeigt und können sich auf die Exemplar- und Datenbank-Ebene vorlicken und die gewünschte Datenbank auswählen (siehe Abbildung 6.17).



Abbildung 6.17: Assistent: Datenbank hinzufügen – mit bekannten Systemen

Über Schritt 3 können Sie noch einen Aliasnamen für die Datenbank vergeben, unter dem die Datenbank dann lokal – also vom Client aus – angesprochen wird.

Schritt 4 ermöglicht es, die entfernte Datenbank für ODBC (Open Database Connectivity) zu registrieren, um so eventuell auch über Programme wie Microsoft Access, Lotus Approach oder Excel auf die Datenbank zuzugreifen.

FERTIG STELLEN öffnet ein kleines Fenster, in dem Sie die Verbindung testen können, nachdem Sie eine gültige Benutzer-ID und ein Kennwort eingegeben haben. Außerdem wird dann ein weiterer Eintrag im Konfigurationsassistenten-Fenster erzeugt. Jetzt ist die neue – entfernte – Datenbank am Client katalogisiert und kann im CONNECT-Befehl angegeben werden. Falls Sie in Schritt 3 einen Aliasnamen für die entfernte Datenbank angegeben hatten, müssen Sie diesen im CONNECT-Befehl angeben.

Hinweis:**Servername kann nicht aufgelöst werden**

Haben Sie direkt die IP-Adresse des Servers angegeben, weil die Namensauflösung (der Test mit `ping <hostname>`) nicht funktioniert hat, sollten Sie auf jeden Fall den Verbindungsaufbau testen. Schlägt dieser fehl, hilft es, die Datenbank bzw. die Verbindung im Konfigurationsassistenten zu selektieren und im Kontextmenü DATENBANK ÄNDERN... auszuwählen.



Abbildung 6.18: Assistent: Datenbank ändern – Schritt 2 – TCP/IP

Dort können Sie in Schritt 2 – wie in Abbildung 6.18 gezeigt – den Hostnamen mit der tatsächlichen IP-Adresse überschreiben.

Der Fehler rührt daher, dass DB2 dort automatisch den Rechnernamen eingetragen hatte, der aber nicht aufgelöst werden konnte. Dieser Trick stellt sicher, dass Ihre Verbindung nun funktionieren sollte (falls sonst alles andere richtig konfiguriert ist). Das Fenster VERBINDUNG TESTEN wird wieder nach dem Anklicken des FERTIG STELLEN-Buttons angezeigt und bietet Ihnen gleich eine gute Möglichkeit, den Verbindungstest durchzuführen.

Mit der Steuerzentrale

Die gleiche Verbindung von einem Client zu dem Server der Datenbank BUCHDB können wir nun auch in Einzelschritten über die Steuerzentrale herstellen.

Um dabei wieder den vollständigen Vorgang zeigen zu können, gehen wir von einem „neuen“ Client aus, einem Client also, von dem noch keine Verbindungsdefinitionen zu anderen Systemen bzw. Datenbanken bestehen.

Die Steuerzentrale zeigt sich ohne Einträge – bis auf die „Systeme-Wurzel“ des Objektbaums. Das Kontextmenü des SYSTEME-Eintrags enthält die Auswahl HINZUFÜGEN..., die Sie im Punkt ① der Abbildung 6.19 sehen.

Es erscheint zunächst ein leeres SYSTEME HINZUFÜGEN-Fenster, das Sie manuell füllen können. Alternativ können Sie sich über die Discovery-Funktionalität – also über die automatische Suche im Netz durch den AUFSPÜREN-Button (Punkt ②) – eine Auswahl verfügbarer Systeme anzeigen lassen.

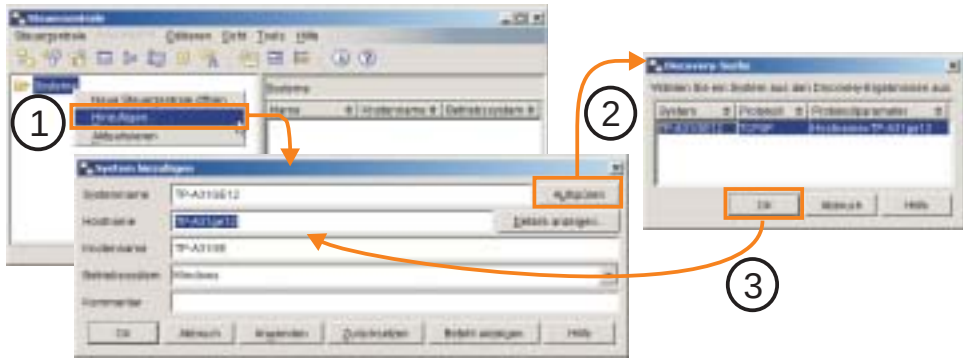


Abbildung 6.19: Steuerzentrale – System hinzufügen

Im DISCOVERY-SUCHE-Fenster wählen Sie eines der gezeigten Systeme aus. In unserem Beispiel wird nur ein System angezeigt, es könnten aber beliebig viele sein. Durch den OK-Button werden die Systemdaten auf das SYSTEM HINZUFÜGEN-Fenster übertragen – siehe Punkt ③. Dabei wird gleich automatisch ein Test der Kommunikationsverbindung durchgeführt, deren Status Sie mitgeteilt bekommen. Hier ist es wieder notwendig, dass der Client den HOSTNAME auflösen – also der gültigen IP-Adresse zuordnen – kann. Ist dies nicht möglich, schlägt der Verbindungstest fehl, und Sie sollten entweder die Namensauflösung sicherstellen oder den Hostnamen manuell im SYSTEM HINZUFÜGEN-Fenster mit der IP-Adresse des Servers überschreiben.

Damit haben Sie das entfernte System in Ihre Steuerzentrale aufgenommen. Genauer betrachtet, müsste man aus DB2-Sicht eigentlich sagen, dass Sie den DAS des entfernten Servers katalogisiert haben, der als „Vertreter“ des Systems bzw. der Servermaschine fungiert. Der DAS existiert ja nur einmal pro Server bzw. es kann nur ein DAS-Prozess pro Server aktiv sein.

Nachdem Sie das System selektiert haben, können Sie in der Steuerzentrale ein Exemplar hinzufügen. Dies funktioniert vom Prinzip her genauso wie das Hinzufügen eines Systems. Auf der Exemplarebene wählen Sie nun wieder HINZUFÜGEN... aus dem Kontextmenü aus, und das EXEMPLAR HINZUFÜGEN-Fenster erscheint (siehe Abbildung 6.20 – Punkt ①).

Über den AUFSPÜREN-Button – Punkt ② – bekommen Sie das DISCOVERY-SUCHE-Fenster mit allen auf dem Ziel-Server verfügbaren Exemplaren angezeigt. Dabei wird jedes Exemplar einmal pro Protokoll angezeigt, mit dem dieses Exemplar erreicht werden kann. Sie erinnern sich: Die Kommunikationsdefinition geschieht auf Exemplar-Ebene. In unserem Beispiel kann das DB2CTLSV-Exemplar sowohl über das Protokoll TCP/IP als auch über NPIPE erreicht werden. Es sind also dort diese zwei Verbindungsmanager mit der Registriervariable DB2COMM=TCPIP, NPIPE definiert worden.

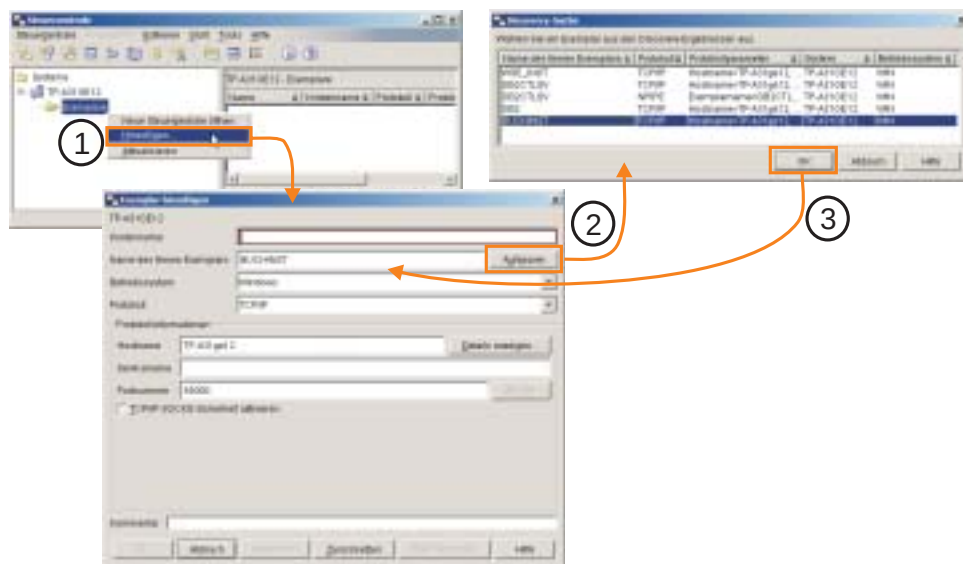


Abbildung 6.20: Steuerzentrale – Exemplar hinzufügen

Nach der Auswahl des gewünschten Exemplars werden die entsprechenden Daten bei der Bestätigung mit dem OK-Button in das EXEMPLAR HINZUFÜGEN-Fenster übertragen (Punkt ③). Auch hier gilt wieder der Hinweis mit der Namensauflösung des Hostnamens. Durch die Datenübernahme aus dem DISCOVERY-SUCHE-Fenster wird die vorher im Hostnamen-Feld enthaltene IP-Adresse durch den Servernamen überschrieben. Dies muss nun nochmals manuell korrigiert – also auf die IP-Adresse „zurück geändert“ – werden.

Generell gilt also, dass es bei der Server-Definition von einem DB2-Client aus wesentlich einfacher ist, über die etc/hosts-Datei eine lokale Namensauflösung zu definieren, falls kein DNS (Domain Name Server) im Netz vorhanden ist, als bei jedem Einzelschritt immer wieder diese Daten angeben zu müssen.

Jetzt muss im EXEMPLAR HINZUFÜGEN-Fenster (siehe Abbildung 6.20) nur noch ein Knotenname vergeben werden – diese Pflichteingabe wird auch durch eine rote Umrandung des Feldes angezeigt. Der Knotenname kann maximal acht Stellen lang sein und darf keine Sonderzeichen beinhalten, ist sonst aber frei wählbar. Es empfiehlt sich, den Namen des Exemplars zu übernehmen, um eine bessere Zuordenbarkeit zu bekommen.

Der nächste Schritt ist das Hinzufügen der Datenbank, das ebenfalls nach dem gleichen „Strickmuster“ abläuft.

Das Kontextmenü auf Datenbank-Ebene enthält einen HINZUFÜGEN...-Eintrag (Punkt ① in Abbildung 6.21). Das DATENBANK HINZUFÜGEN-Fenster hat auch wieder einen AUFSPÜREN-Button, der Ihnen die am DB2-Server verfügbare Datenbank anzeigt (Punkt ②). Ihre Auswahl wird mit dem OK-Button (Punkt ③) übernommen und kann durch zusätzliche Informationen ergänzt werden.

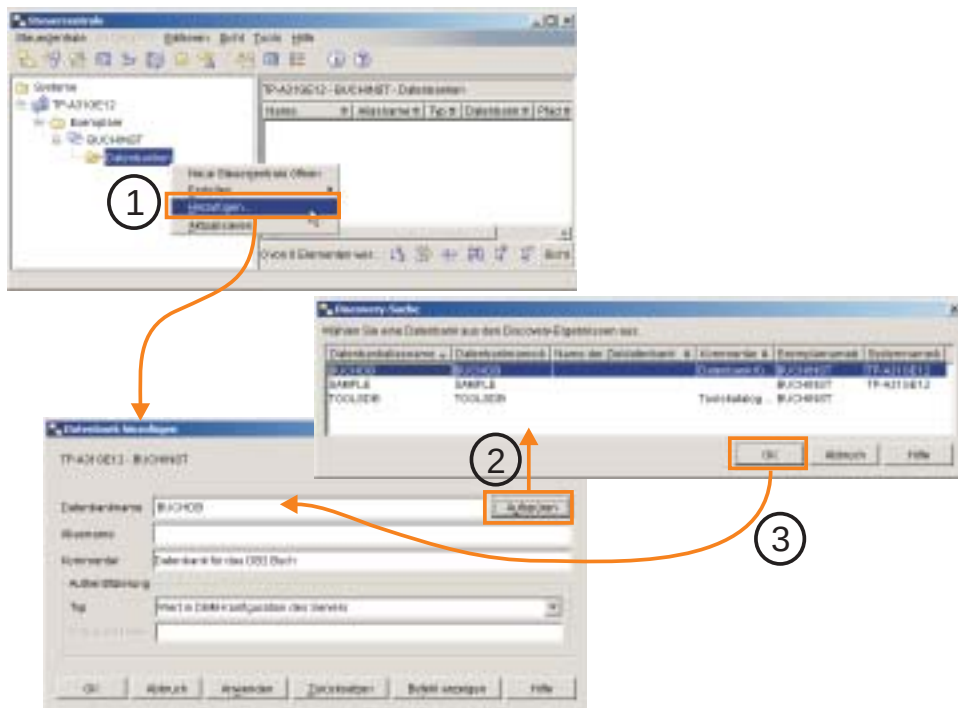


Abbildung 6.21: Steuerzentrale – Datenbank hinzufügen

So können Sie beispielsweise im DATENBANK HINZUFÜGEN-Fenster einen Aliasnamen für die Datenbank vergeben. Dieser muss dann im CONNECT-Befehl am Client angegeben werden. Wie bereits erwähnt wurde, wird die Datenbank *immer* über ihren Aliasnamen angesprochen. Das kann aber auf dem Client ein anderer Aliasname sein als auf dem DB2-Server oder einem der anderen Clients.

Auch der übernommene Kommentar kann angepasst werden. Die Authentifizierung kann hier – wie in Abschnitt 6.3.4 „Authentifizierungskonfiguration auf dem Client“ beschrieben – explizit am Client angegeben werden und sollte mit der Authentifizierung am Server bzw. der Authentifizierung des Exemplars übereinstimmen.

Mit diesem Schritt ist Ihr Client für die entfernte Datenbank konfiguriert, und Sie können nun die Tabellen dieser Datenbank in der Steuerzentrale anzeigen lassen und auch sonst alle Konfigurationen und Aufgaben durchführen. Bei dem ersten Klick auf die Datenbank werden Sie jedoch noch aufgefordert, eine gültige Benutzer-ID/Kennwort-Kombination einzugeben, um Sie für diesen Server zu authentifizieren. Dann steht der Administration des entfernten Systems nichts mehr im Wege.

6.4.3 Konfiguration des Clients durch Befehle

Die Methode, einen Client über die Steuerzentrale bzw. den DB2-Konfigurationsassistenten einzurichten, steht Ihnen nur zur Verfügung, wenn Sie den DB2-Client inklusive der grafischen Oberfläche installiert haben, also den administrativen

Client. Der pure RTC (Run-Time Client) enthält keine dieser grafischen Möglichkeiten, und so steht nur die manuelle Methode zur Verfügung – oder Sie arbeiten befehlsbasiert mit Profilen. Diese Alternative stellen wir aber erst im nächsten Abschnitt genauer dar.

In der Praxis hat es sich oft als sehr nützlich erwiesen, diese Technik zu beherrschen, weil Sie so am schnellsten eventuelle (Verbindungs-)Fehler entdecken und auch beheben können.

Dabei müssen wir folgende Aufgaben abarbeiten:

1. Konfigurations- und Verbindungsdaten des Servers ermitteln
2. das Knotenverzeichnis auf dem Client um den entfernten Server erweitern
3. das Systemdatenbankverzeichnis um die entfernte Datenbank erweitern
4. die Verbindung testen.

Beginnen wir Schritt für Schritt von vorn.

Sie sollten folgende Daten zur Verfügung haben, bevor Sie beginnen – denken Sie daran, dass Ihnen hier eine „Aufspüren“-Funktion wie im GUI nicht zur Verfügung steht:

- den Servernamen und/oder die IP-Adresse des Servers

Den Servernamen bekommen Sie vom Systemadministrator oder durch den Befehl `hostname` am Server (gilt für alle Betriebssysteme).

Die IP-Adresse bekommen Sie in Unix-Systemen durch den Befehl `ifconfig` und durch `ipconfig` in Windows-Umgebungen.

- die Portnummer des entfernten Exemplars

Diese erfragen Sie beim Exemplar-Administrator des entfernten DB2-Servers oder ermitteln sie lokal am Server durch die Abfrage des Parameters `SVCENAME` aus der Exemplarkonfiguration (durch den Befehl: `get dbm cfg`). Ist dort ein Name eingetragen, müssen Sie die zugehörige Portnummer aus der Datei `etc/services` auslesen. Diese befindet sich in Unix-Systemen im Verzeichnis `/etc/`, in Windows-Umgebungen finden Sie die Datei im Verzeichnis `%WINDIR%\System32\drivers\etc\`. Wobei `%WINDIR%` für das Windows-Installationsverzeichnis steht – also typischerweise für `c:\winnt` bzw. `c:\windows`.

- den Namen des Exemplars, das die gewünschte Datenbank beinhaltet

Diesen Namen finden Sie durch den Befehl `get instance` heraus, wenn Sie in einer Umgebung am Server arbeiten, in der die Datenbank lokal ist.

- den Datenbanknamen

- optional, aber zu empfehlen: die definierte Authentifizierungsmethode, die für das Exemplar gewählt wurde

Diese finden Sie in der Exemplar-Konfiguration unter dem Parameter `AUTHENTICATION`.

Nachdem Sie alle Daten zusammengetragen haben, können Sie sich an die Einrichtung des Clients machen.

Schritt 1 – Test der TCP/IP-Verbindung

Öffnen Sie eine Befehlszeile am Client, und testen Sie am besten erst einmal die reine TCP/IP-Verbindung DB2-unabhängig, um die Grundvoraussetzung für eine Verbindungsmöglichkeit sicherzustellen. Verwenden Sie einen ping-Befehl:

```
ping <hostname_des_servers>
```

Funktioniert dies, ist alles in bester Ordnung, und Sie können mit dem nächsten Schritt weitermachen. Funktioniert es nicht, probieren Sie es mit der IP-Adresse des Servers:

```
ping <ip-adresse>
```

Ist dieser Versuch erfolgreich und bekommen Sie Rückmeldungen über die hergestellte Verbindung, kann nur der Servername nicht aufgelöst werden. Das heißt für Sie, dass Sie entweder einen Eintrag in der Datei `etc/hosts` am Client vornehmen müssen, um die lokale Namensauflösung zu ermöglichen, oder im Weiteren mit der IP-Adresse direkt arbeiten müssen.

Sollte die zweite Variante ebenfalls nicht funktionieren, besteht momentan keine TCP/IP-Verbindungsmöglichkeit zum Server, und Sie sollten den Netzadministrator und den Systemadministrator des entfernten Servers kontaktieren und um Hilfe bitten.

Schritt 2 – Knotenverzeichnis-Eintrag erstellen

Die manuelle Anpassung des Knotenverzeichnisses (node directory) geschieht mit einem CATALOG-Befehl – für TCP/IP-Verbindungen mit dem Befehl `CATALOG TCPIP NODE`.

Dieser ist nach folgendem Schema aufgebaut:

```
catalog tcpip node <knotenname>  
remote <hostname oder ip-adresse>  
server <portname oder portnummer>
```

Er könnte also so lauten, um einen Eintrag für den DB2-Server `MTServer1` zu erzeugen, auf dem für das Ziel-Exemplar die Portnummer (bzw. Servicenummer) `55000` zugeordnet ist:

```
catalog tcpip node buchinst server remote mtserver1 server 55000
```

Alternativ können Sie auch einen Port- bzw. Servicenamen angeben. Dann müssen Sie diesen aber noch mit der entsprechenden Portnummer in der Datei `etc/services` eintragen.

Auch für die anderen Protokolle stehen entsprechende CATALOG-Befehle bereit, so z.B. `CATALOG NETBIOS NODE` etc.

Hinweis:

Ein Sonderfall ist gegeben, wenn Sie die *Datenbank eines anderen Exemplars* auf der *selben Maschine* ansprechen wollen. Der zugehörige CATALOG-Befehl lautet dann:

```
catalog local node <knotenname> instance <exemplarname>
```

Das Ergebnis Ihres Befehls können Sie sich über den Befehl LIST NODE DIRECTORY anschauen (siehe Abbildung 6.22).

```
D:\Programme\IBM\SQLLIB\BIN>db2 list node directory

Knotenverzeichnis

Anzahl Einträge im Verzeichnis = 1

Eintrag für Knoten 1:

Knotenname           = BUCHINST
Kommentar             =
Verzeichniseintragungsart = LOCAL
Protokoll             = TCPIP
Host-Name             = MTServer1
Servicename           = 55000
```

Abbildung 6.22: Beispiel für ein Knotenverzeichnis am Client

Der CATALOG-Befehl hätte um einige zusätzliche, informelle Parameter erweitert werden können. So wäre es möglich gewesen, einen Kommentar mit der WITH-Option oder durch die OSTYPE-Option das Betriebssystem des entfernten Rechners anzugeben. Der Systemname könnte mit der SYSTEM-Option spezifiziert werden, falls der Host-Name als IP-Adresse angegeben wurde. Durch die REMOTE_INSTANCE-Option könnte zusätzlich der Exemplarname mitgegeben werden.

Die maximale Ausbaustufe dieses Befehls würde so aussehen; sie ist aber rein funktionell mit dem obigen Befehl identisch:

```
catalog tcpip node buchinst server remote mtserver1 server 55000
remote_instance buchinst system mtserver1 ostype win
with „Buchinst am Buch-Server“
```

Tipp:

Ein kleiner Test kann auch für diesen Schritt schon durchgeführt werden. Was für die Datenbank ein CONNECT ist, ist für ein Exemplar ein ATTACH. Der ATTACH zu einem Exemplar dient zur entfernten Konfiguration eines Exemplars über die Befehlszeile, ist aber sehr unkomfortabel und wird nicht empfohlen, da über die Steuerzentrale mehr Möglichkeiten bestehen und die Übersicht wesentlich besser ist. Für einen Test an dieser Stelle der Client-Konfiguration kann er jedoch sehr hilfreich sein. Sie können sich also wie folgt über die Befehlszeile zum entfernten Exemplar „attachen“:

```
attach to <knotenname> user <benutzer-id> using <kennwort>
```

Bekommen Sie an dieser Stelle einen Fehler angezeigt, bleibt Ihnen nichts anderes übrig, als den Fehlercode genau zu studieren und sonst alle Einstellungen auf dem Server bzw. den Test der TCP/IP-Verbindung noch einmal durchzugehen.

Schritt 3 – Eintrag im Systemdatenbankverzeichnis erstellen

Da der CONNECT-Befehl immer zuerst im Systemdatenbankverzeichnis nachschaut, ob der Datenbank-Aliasname in der Umgebung bzw. auf dem Client bekannt ist, muss auch hier noch ein Eintrag erstellt werden. Man sagt hier: Die Datenbank muss katalogisiert werden.

Der zugehörige CATALOG-Befehl hat die in Abbildung 6.23 gezeigte Struktur.

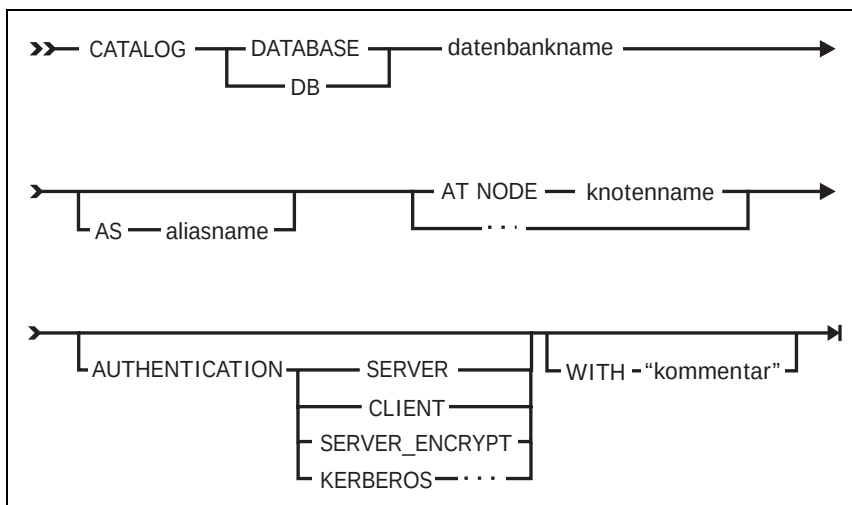


Abbildung 6.23: catalog database-Syntaxdiagramm

Das Syntaxdiagramm wurde hierbei an die vorgesehene Aufgabe angepasst, da die AT NODE-Option zwar für diese Aufgabe hier, wie angezeigt, benötigt wird, aber eigentlich optional ist.

Der hier angegebene knotenname *muss* mit dem Knotennamen des vorhergehenden CATALOG NODE-Befehls übereinstimmen!

Optional kann ein Aliasname für die Datenbank angegeben werden, der – wie schon öfter erwähnt – dann im CONNECT verwendet werden muss und lokal, also auf dem Client, gilt.

Die Spezifikation der Authentifizierungsmethode ist optional, aber leistungsrelevant und muss mit der Einstellung im Ziel-Exemplar übereinstimmen.

Das Beispiel für das Katalogisieren der BUCHDB-Datenbank auf unserem Client sieht dann so aus:

```
catalog db buchdb as buch at node buchinst
```

Da der Aliasname BUCH für die Datenbank angegeben wurde, kann nun mit folgendem Befehl eine Datenbankverbindung zum entfernten Server aufgebaut werden:

```
connect to buch user <benutzer-id> using <password>
```


Das sieht dann so aus, wie in Abbildung 6.24 gezeigt.

```
D:\Programme\IBM\SQLLIB\BIN>db2 connect to buch user db2admin
Aktuelles Kennwort für db2admin eingeben:

Datenbankverbindungsinformationen

Datenbank-Server                = DB2/NT 8.1.3
SQL-Berechtigungs-ID           = DB2ADMIN
Aliasname der lokalen Datenbank = BUCH

D:\Programme\IBM\SQLLIB\BIN>db2 list db directory

Systemdatenbankverzeichnis

Anzahl Einträge im Verzeichnis = 1

Eintrag für Datenbank 1:

Aliasname der Datenbank        = BUCH
Datenbankname                  = BUCHDB
Knotenname                     = BUCHINST
Release-Stand der Datenbank    = a.00
Kommentar                      =
Verzeichniseintragungsart      = Fern
Datenbankpartitionsnummer für Katalog = -1
```

Abbildung 6.24: Beispiel – connect und Systemdatenbankverzeichnis eines Clients

Abbildung 6.24 zeigt zusätzlich die Anzeige des Systemdatenbankverzeichnisses des Clients, der durch den Eintrag Verzeichniseintragungsart = Fern und Knotenname = BUCHINST auf den entsprechenden Eintrag des Knotenverzeichnisses verweist.

Der im Systemdatenbankverzeichnis des Clients angezeigte Datenbankname (hier BUCHDB) ist eigentlich der Datenbank-Aliasname der Datenbank am Server.

Passiert Ihnen bei den CATALOG-Befehlen einmal ein Fehler, können Sie die Einträge mit einem UNCATALOG-Befehl – also UNCATALOG NODE <knotenname> oder UNCATALOG DATABASE <db_aliasname> – wieder entfernen.

Um diese, inzwischen doch komplexere Abfolge einer Verbindung noch einmal im Zusammenhang zu sehen, haben wir auch für diese Art der Verbindung ein Schaubild vorbereitet.

6.4.4 Schaubild einer Verbindung zu einer entfernten Datenbank

In Abbildung 6.4 haben Sie den Verbindungsaufbau auf der Server-Seite gesehen. Abbildung 6.25 zeigt die Client-Seite vom CONNECT-Befehl über das Systemdatenbankverzeichnis zum Knotenverzeichnis. Während die gestrichelte Linie den logischen Verbindungsaufbau darstellt, zeigen die durchgezogenen Linien Abhängigkeiten bzw. geforderte Übereinstimmungen.

So muss insbesondere die Portnummer auf dem Client und dem Server übereinstimmen und auch der Datenbankname als Datenbank-Aliasname auf dem Server im Systemdatenbankverzeichnis gefunden werden können.

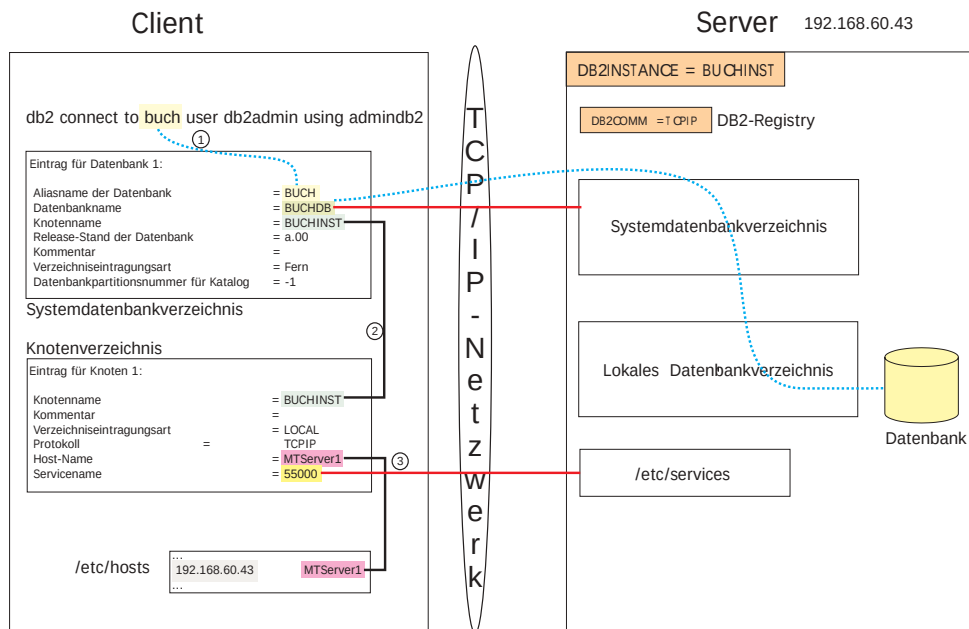


Abbildung 6.25: Schaubild – entfernte Datenbankverbindung

Auch bei einer entfernten Verbindung wird auf dem Server das Systemdatenbankverzeichnis und dann das lokale Datenbankverzeichnis durchlaufen – wie beim Aufbau der lokalen Verbindung.

6.4.5 Konfiguration des Clients über Profile

Profile sind in Textdateien gespeicherte Verbindungsdefinitionen. Sie haben damit die Möglichkeit, die Definition, die Sie einmal vorgenommen haben, auf andere Systeme zu übertragen und dort wieder einzuspielen.

Die DB2-Dokumentation zu diesem Thema ist sehr knapp. Deshalb bieten wir Ihnen hier eine Zusammenstellung, wie Sie die Profile in der Praxis verwenden können.

Es werden dabei zwei Arten von Profilen unterschieden:

► Clientprofile

Das Clientprofil enthält Informationen über *das aktuelle* Exemplar. In welchem Umfang diese Informationen in das Profil gespeichert bzw. exportiert werden, kommt auf Ihre Parameterwahl und auf das verwendete Tool an, mit dem das Profil erstellt wird. Clientprofile können von einem Client oder auch direkt an einem Server erstellt werden. Allerdings kann ein Clientprofil, das an einem Server erstellt wurde, *nicht* dazu verwendet werden, am Client eine Datenbankverbindungsdefinition zu erstellen.

Hinweis:

Das Detailbild (Abbildung 6.26) ist für Fortgeschrittene gedacht bzw. für alle, die sich mit diesem Thema eingehender beschäftigen wollen oder müssen und vor einem weiteren Detaillierungsgrad nicht zurückschrecken. Es ist nicht schlimm, wenn Sie sich dieses Bild erst nach ein paar praktischen Versuchen wieder genauer anschauen – aber wir wollten es Ihnen nicht vorenthalten.

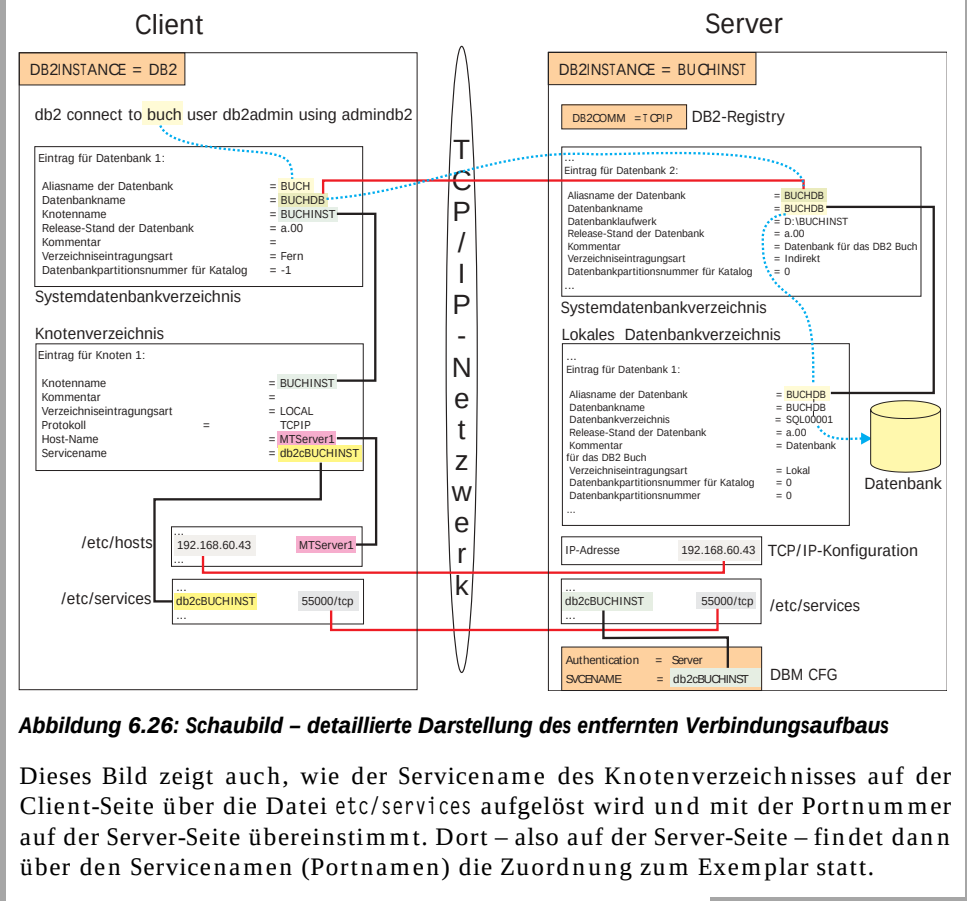


Abbildung 6.26: Schaubild – detaillierte Darstellung des entfernten Verbindungsaufbaus

Dieses Bild zeigt auch, wie der Servicename des Knotenverzeichnisses auf der Client-Seite über die Datei etc/services aufgelöst wird und mit der Portnummer auf der Server-Seite übereinstimmt. Dort – also auf der Server-Seite – findet dann über den Servicennamen (Portnamen) die Zuordnung zum Exemplar statt.

► Serverprofile

Serverprofile enthalten die Daten des Admin-, Knoten- und Systemdatenbank-Verzeichnisses *aller* Exemplare auf einem Server.

Sie werden am Server erstellt und ermöglichen es, diese Daten als Verbindungsdefinitionen für Clients zu verwenden.

Profile müssen durch einen Export-Vorgang erstellt werden und werden dann auf dem Zielsystem importiert.

Beispiel-Szenario

Anhand von Abbildung 6.27 möchten wir Ihnen die Verwendung der Profile an einem Beispiel zeigen.

Zuerst wird am Server jeweils ein Serverprofil erstellt. In diesem sind, wie oben beschrieben, die Verbindungsdefinitionen aller Exemplare und Datenbanken enthalten. Diese Profile (Serverprofil A und B) können – wie Punkt ① zeigt – auf einem Client 1 eingespielt werden. Dabei ist es möglich, nur bestimmte Datenbanken bzw. Verbindungsdefinitionen auf den Client zu übernehmen.

Haben Sie viele gleichartige Clients zu installieren, erstellen Sie nun am besten ein Clientprofil vom Client (Client 1). Da dieser alle Verbindungsdefinitionen zu den gewünschten Ziel-Datenbanken aufweist – die er entweder durch den Import der Serverprofile oder durch manuelle Anpassungen bekommen hat –, ist diese Definition eine ideale Ausgangsbasis.

Dieses Clientprofil kann auf die anderen Clients (Punkt ②) übertragen und dort importiert werden. Damit haben alle Clients die gleichen Verbindungsinformationen und können bei Bedarf Verbindungen zu Datenbanken in den Exemplaren A, B oder C aufbauen (Punkt ③).

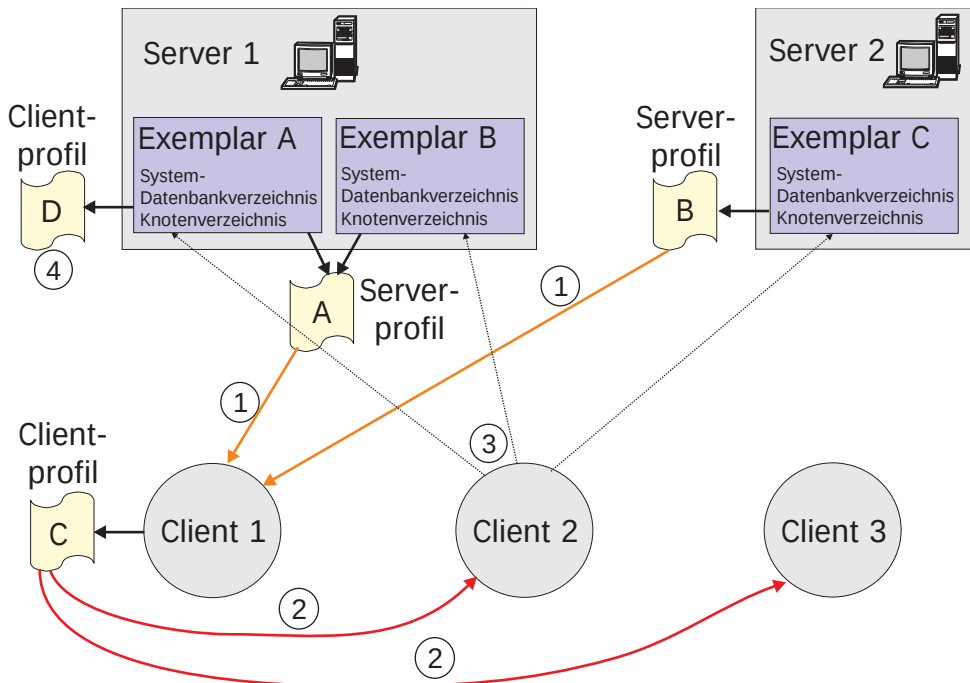


Abbildung 6.27: Beispiel-Szenario mit Client- und Serverprofilen

Sollten Sie keine gleichartigen Clients konfigurieren wollen – gleichartig im Bezug auf die Verbindungsdefinitionen –, können Sie natürlich weiterhin die Methode über die verschiedenen Serverprofile wählen. Das Importieren mehrerer Profile auf

einem System ist unproblematisch, da die Daten immer hinzugefügt werden und bestehende Definitionen nicht löschen.

Alternativ haben Sie auch die Möglichkeit, ein Clientprofil am Server zu erstellen (Punkt ④). Dieses kann z.B. als Sicherung der Serverdefinitionen verwendet werden und auch – zusätzlich zu den (lokalen) Verbindungsdaten – die DB2-Registrierungsvariablen und Exemplar-Konfiguration beinhalten. Dieses Profil kann allerdings nicht für die Übertragung der Verbindungsdefinitionen auf Clients benutzt werden, da in dieser Datei die Datenbanken als „lokal“ beschrieben sind.

Schauen wir uns also an, wie diese Profile zu erstellen bzw. zu importieren sind.

Erstellen eines Serverprofils

Ein Serverprofil wird aus der Steuerzentrale erstellt, indem Sie im Kontextmenü des (zu exportierenden Server-)Systems SERVERPROFIL EXPORTIEREN... auswählen (siehe Abbildung 6.28). Im SERVERPROFIL EXPORTIEREN-Fenster geben Sie dann den Zielpfad und den Namen der Serverprofil-Datei an. Durch Bestätigung mit OK wird das Serverprofil exportiert.

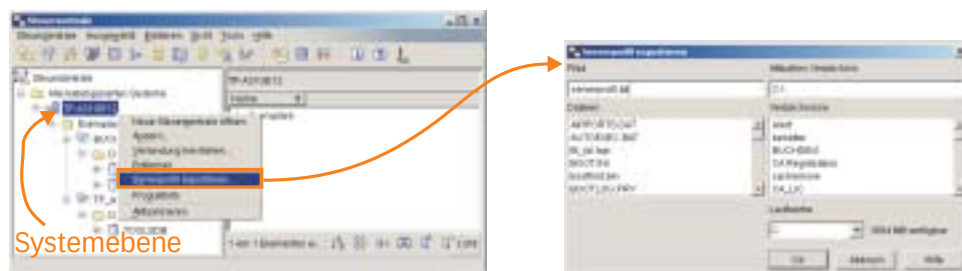


Abbildung 6.28: Serverprofile aus der Steuerzentrale erstellen

Importieren eines Serverprofils

Der Import eines Serverprofils geschieht auf Exemplar-Ebene, wie Abbildung 6.29 zeigt. Über den SERVERPROFIL IMPORTIEREN...-Eintrag des Kontextmenüs öffnet sich der SERVER-PROFIL IMPORTIEREN-Datei-Dialog. Weitere Auswahlmöglichkeiten gibt es hier nicht.

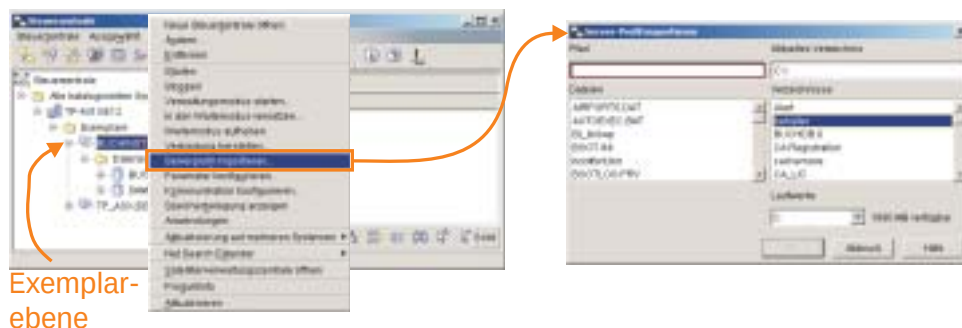


Abbildung 6.29: Serverprofil aus der Steuerzentrale importieren

Ein Serverprofil kann aber auch wie ein Clientprofil importiert werden. Dann stehen Ihnen weitere Möglichkeiten offen, nur bestimmte Inhalte zu selektieren – doch dazu mehr im Abschnitt „Das Clientprofil einspielen (importieren)“ auf Seite 252.

Das Clientprofil erstellen (exportieren)

Schauen wir uns also zuerst die Möglichkeiten für den Export an. Folgende Tools stehen für das Erstellen eines Clientprofils zur Auswahl:

- Das Befehlszeilen-Tool: db2cfexp
Dieses Tool ist auch auf RTCs (run-time clients) verfügbar, auf denen der Konfigurationsassistent und die anderen GUIs nicht installiert sind. In drei Einstellungsstufen lässt sich bestimmen, was exportiert werden soll.
- Der Konfigurationsassistent
Dieser bietet sehr flexible Auswahl- und Einstellungsmöglichkeiten, und sogar individuelle Anpassungen der Verzeichnisse (Systemdatenbank- und Knotenverzeichnis) sind möglich.

Schauen wir uns zuerst den Export mit dem db2cfexp-Tool an. db2cfexp steht für „connectivity configuration export tool command“ – erwähnenswert, wie wir finden.

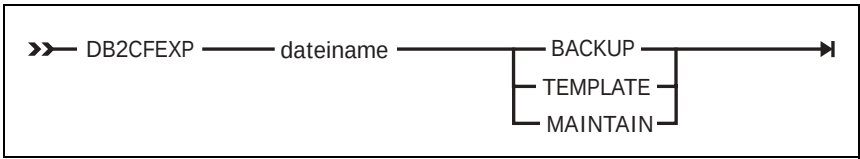


Abbildung 6.30: db2cfexp-Syntaxdiagramm

Neben dem Dateinamen, der für das zu erzeugende Profil notwendig ist, gibt es drei Ausprägungen, die über den Inhalt des Profils bestimmen:

Inhalt	MAINTAIN	TEMPLATE	BACKUP
Exemplar-Grunddaten	Ja	Ja	Ja
Systemdatenbankverzeichnis	Ja	Ja	Ja
Knotenverzeichnis	Ja	Ja	Ja
Admin. Knotenverzeichnis	-	Ja	Ja
ODBC/CLI-Information	-	Ja	Ja
Lokale Registrierungsvariablen	-	Ja	Ja
Exemplar-Konfiguration (DBM CFG)	-	Ja	Ja
Globale Registrierungsvariablen	-	-	Ja

Tabelle 6.1: Profil-Inhalt des db2cfexp-Tools

Sie sehen, dass neben den reinen Verbindungsdaten auch die Konfigurationswerte des Exemplars und die Registrierungsvariablen gesichert werden können. Wie die BACKUP-Option schon vermuten lässt, dient das weniger der Konfiguration anderer Systeme als vielmehr der Sicherung der lokalen Umgebung.

Die Option MAINTAIN ist dafür gedacht, Einstellungen eines Clients auf andere Clients zu übertragen. Haben Sie z.B. in Ihrer Firma viele gleichartige Clients einzurichten, die alle eine Verbindung zu den gleichen Servern herstellen können sollen, ist das die geeignete Variante.

Die zweite Art – also über den DB2-Konfigurationsassistenten – erlaubt eine noch feinere Abstufung bzw. Zusammenstellung des Profil-Inhalts, benötigt aber andererseits mindestens die Installation eines administrativen Clients.

Starten Sie den Konfigurationsassistenten über das Startmenü oder den Befehl `db2ca` von der Befehlszeile aus. Im Menüpunkt KONFIGURIEREN (siehe Abbildung 6.31) des Konfigurationsassistenten finden Sie den Eintrag PROFIL EXPORTIEREN, der Ihnen drei Untermenüpunkte zur Auswahl bietet:

► ALLE...

Alle möglichen Informationen werden exportiert. Diese Auswahl entspricht der BACKUP-Option des `db2cfexp`-Befehls. Durch diese Auswahl rufen Sie den Datei-dialog SICHERN (Punkt ① in Abbildung 6.31) auf.

► DATENBANKVERBINDUNGEN...

Nur die Datenbankverbindungen, also die Einträge des Systemdatenbankverzeichnis und Knotenverzeichnisses, werden in das Profil geschrieben. Dies entspricht der MAINTAIN-Option des `db2cfexp`-Befehls. Auch hier bekommen Sie gleich den Datei-Dialog, um den Dateinamen und Speicherpfad des Profils angeben zu können.

► ANPASSEN...

In dieser Auswahl können Sie sich den Inhalt des gewünschten Profils selbst zusammenstellen. Statt eines Datei-Diialogs wird das Fenster EXPORTPROFIL ANPASSEN geöffnet (siehe Punkt ② in Abbildung 6.31).

Die flexibelste Auswahl ist meist auch die geeignetste. Deshalb schauen wir uns das Fenster EXPORTPROFIL ANPASSEN noch etwas genauer an.

Als Erstes sollten Sie die Ziel-Datei angeben bzw. über den „...“-Button auswählen (Punkt ③ in Abbildung 6.31). Über die Auswahl (Punkt ④) stellen Sie sich den Profil-Inhalt zusammen. Wenn Sie DATENBANKVERBINDUNGEN auswählen, können Sie die gewünschten Datenbanken selektieren; die zugehörigen Knoten-Informationen für die Verbindung werden automatisch mit gespeichert. Die VERWALTUNGS-KNOTEN sind die Einträge des administrativen Knotenverzeichnisses, das Sie über LIST ADMIN NODE DIRECTORY angezeigt bekommen. Sie dienen – wie auch auf Seite 230 beschrieben – zur Verwaltung entfernter Systeme mit der grafischen Oberfläche.

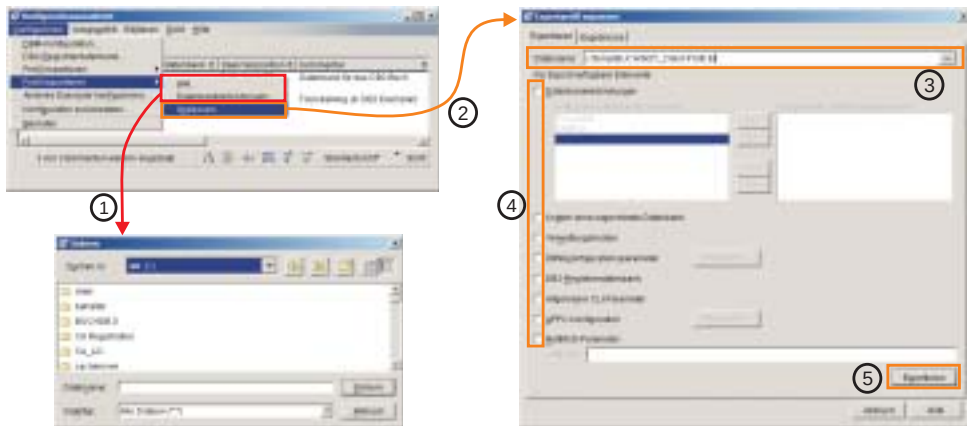


Abbildung 6.31: Profil mit dem Konfigurationsassistenten exportieren

Tipp:

In diesem EXPORTPROFIL ANPASSEN-Fenster können Sie auch „nur“ die DBM-KONFIGURATIONSPARAMETER oder DB2-REGISTRIERDATENBANK auswählen, um diese zu sichern, ohne irgendwelche Verbindungsinformationen exportieren zu müssen.

Das ermöglicht die Sicherung der Konfiguration, die in DB2 *sonst nicht* möglich ist! Der Vorteil gegenüber einer in eine Datei umgeleiteten („geipten“) DBM-Konfiguration ist, dass Sie dieses Profil wieder über das GUI bzw. den Befehl `db2cfimp` einspielen können, ohne die Parameter einzeln einzugeben.

Durch den EXPORTIEREN-Button (Punkt ⑤ der Abbildung 6.31) werden die gewählten Daten in das angegebene Profil exportiert, und Sie bekommen im Reiter ERGEBNISSE die Rückmeldung, welche Exporte durchgeführt worden sind und ob diese erfolgreich waren.

Das Clientprofil einspielen (importieren)

Das erzeugte Profil (Clientprofil) können Sie nun auf den gewünschten Zielrechner übertragen und dort über folgende Techniken einspielen:

- ▶ mit dem `db2cfimp`-Tool über die Befehlszeile

Der Vorteil hierbei ist die Verfügbarkeit dieses Tools auf allen DB2-Systemen – inklusive RTCs. Allerdings wird das Profil 1:1 übernommen, eine Einschränkung dessen, was eingespielt werden soll, ist durch dieses Tool nicht möglich.

- ▶ mit dem Konfigurationsassistent-GUI

Wie gehabt wird mindestens der administrative Client für dieses grafische Tool vorausgesetzt. Dafür können aber hier auch beim Import die zu importierenden Daten angepasst werden.

Das Tool db2cfimp besitzt keine Optionen, es muss nur das Profil angegeben werden. Durch db2cfimp <profilname> wird der Import gestartet.

Grafisch kann der Import auch wieder über den Konfigurationsassistenten erfolgen. Das KONFIGURIEREN-Menü (siehe Abbildung 6.32) bietet dazu den Eintrag PROFIL IMPORTIEREN. Zwei Untermenüpunkte stehen zur Auswahl:

► ALLE...

Bei dieser Auswahl öffnet sich direkt der Datei-Dialog ÖFFNEN (Punkt ① der Abbildung 6.32), und es muss nur noch das Profil ausgesucht werden.

► ANPASSEN...

Wie Sie in Punkt ② der Abbildung 6.32 sehen, wird das IMPORTPROFIL ANPASSEN-Fenster mit weiteren Auswahlmöglichkeiten geöffnet.

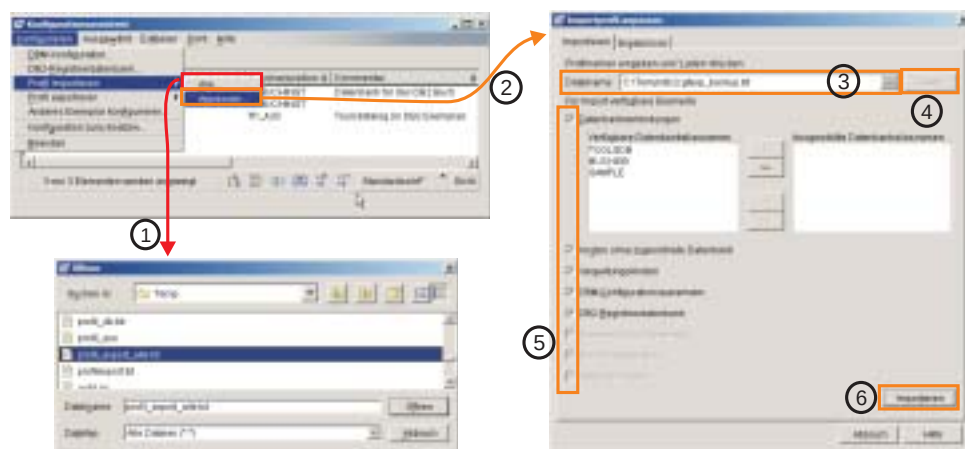


Abbildung 6.32: Profil mit dem Konfigurationsassistenten importieren

In dem Fenster IMPORTPROFIL ANPASSEN wählen Sie zuerst das zu importierende Profil aus, indem Sie es direkt in die DATEINAME-Zeile eintippen oder über den „...“-Button einen Datei-Dialog zur Auswahl des Profils öffnen (Punkt ③). Als Nächstes klicken Sie auf den LADEN-Button (Punkt ④) um den Inhalt des Profils zu lesen. Daraufhin werden Ihnen die Inhalte bzw. die für den Import verfügbaren Elemente angezeigt, und Sie können Einschränkungen vornehmen (Punkt ⑤). Dabei werden nur Inhalte zur Auswahl gestellt, die im Profil vorhanden sind; die anderen Elemente bleiben grau und sind somit nicht selektierbar. Auch die Verbindungsdefinitionen einzelner Datenbanken können so ausgewählt werden. Erst durch die Auswahl des IMPORTIEREN-Buttons werden die Inhalte in Ihr System bzw. in Ihr Exemplar übernommen.

Dabei können auch Profile, die mit dem db2cfexp-Tool erstellt wurden, problemlos über den Konfigurationsassistenten importiert werden.

Hinweis:

Die importierten Verbindungsdaten werden dabei immer nur hinzugefügt! Vorhandene Verbindungsdaten bleiben bestehen. Das ermöglicht es, auch auf einem Client zwei oder mehr Profile mit verschiedenen Verbindungsdefinitionen nacheinander einzuspielen.

Tipp:

Diese Technik erlaubt es auch, aus einem Profil nur die Datenbankmanager-Konfiguration (DBM CFG) zurückzuspielen, falls diese im Profil gespeichert ist: eine komfortable Art, Exemplar-Konfigurationen mit einem Profil-Export zu speichern und auf diese Art und Weise wieder einzuspielen.

Alternativ kann im Konfigurationsassistent auch der ASSISTENT: DATENBANK HINZUFÜGEN für den Import genutzt werden (siehe Abbildung 6.33).



Abbildung 6.33: Profil-Import mit Assistent: Datenbank hinzufügen

Das weitere Vorgehen entspricht fast der gezeigten „Profil importieren“-Technik. Es muss ein Profil ausgewählt und mit dem LADEN-Button eingelesen werden, und dann kann eine Datenbank selektiert werden. Weitere Einschränkungen gibt es hier nicht, und es kann nur *eine* Datenbank ausgewählt werden – der Assistent heißt ja auch „Datenbank hinzufügen“.

6.5 Verbindungen über DB2 Connect

DB2 Connect wird benötigt, um Datenbankverbindungen zu Host-Datenbanken, also zu DB2 für z/OS, DB2 für VSE und VM oder DB2 für AS/400, herzustellen. Dabei ist es unerheblich, ob Sie DB2 Connect als separates Produkt installiert haben oder als Bestandteil der DB2 ESE nutzen.

DB2 Connect stellt sicher, dass eine Kommunikation zwischen der DB2 UDB auf den Workstation-Plattformen (also Unix- und Windows-Systemen) und den Host-Systemen funktionieren kann.

Soll eine entfernte Host-Datenbank angesprochen werden – bei DB2 für z/OS spricht man von einem Subsystem –, müssen Sie durch einen Eintrag in das so genannte *DCS-Verzeichnis* sicherstellen, dass die Verbindung über die DB2 Connect-Funktionalität aufgebaut wird. DCS steht dabei für „Database Connection Services“.

Abbildung 6.34 zeigt, dass dieses DCS-Verzeichnis zusätzlich zum Systemdatenbankverzeichnis und dem Knotenverzeichnis notwendig ist.

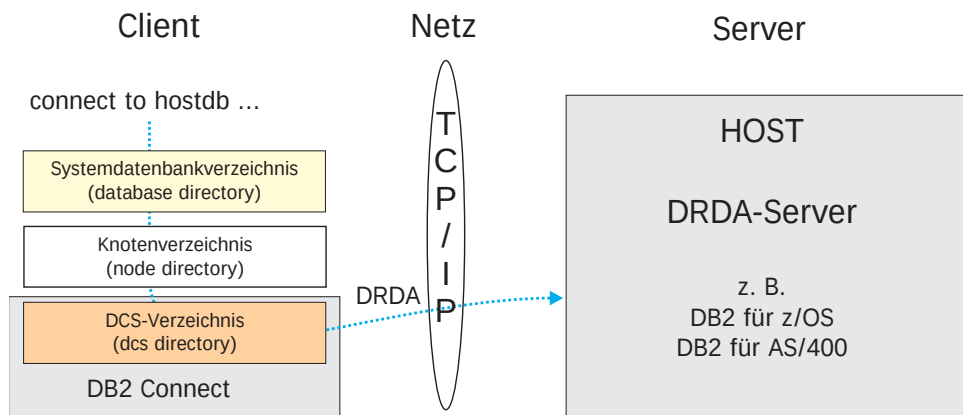


Abbildung 6.34: Host-Verbindung mit dem DCS-Verzeichnis

Die Einträge in das DCS-Verzeichnis können über den Befehl `CATALOG DCS DATABASE` manuell vorgenommen werden oder werden mit Hilfe des Konfigurationsassistenten automatisch erstellt.

Dabei hat der Befehl `CATALOG DCS DATABASE` folgende Struktur:

```
catalog database <datenbankname> as <ziel-datenbankname>
```

Der Datenbankname ist dabei der Name, der beim `CONNECT`-Befehl angegeben wird. Dieser muss mit dem Eintrag im Systemdatenbankverzeichnis übereinstimmen. Der Ziel-Datenbankname ist der Datenbankname auf dem Zielsystem. Für Verbindungen zu DB2 für z/OS bzw. OS/390 entspricht der Ziel-Datenbankname dem so genannten „Location“-Namen. Außerdem könnten noch zusätzliche Parameter für den DRDA AR (Application Requestor) mit angegeben werden.

Um sich den Inhalt des DCS-Verzeichnisses anzeigen zu lassen, benutzen Sie den Befehl `LIST DCS DIRECTORY`, und um Einträge zu löschen, verwenden Sie den Befehl `UNCATALOG DCS DATABASE`.

6.6 Zweiphasige Festschreibung (two-phase commit)

Bei verteilten Transaktionen (DUOW – Distributed Unit Of Work) können innerhalb einer einzelnen Transaktion mehrere (verteilte) Datenbanken geändert werden. Auch für diese Transaktion muss die Konsistenz gewährleistet werden können – und hier beginnt die Problematik. Es muss hier über Datenbank- und Systemgrenzen hinweg das „Alles-oder-nichts“-Konzept der Transaktion gewährleistet werden können.

Schauen wir uns das mit Hilfe des klassischen Beispiels einer Kontobewegung an. Abbildung 6.35 zeigt eine Anwendung, die eine Umbuchung von einem Konto auf ein anderes erledigen soll. Dazu wird auf dem Datenbank-Server (DB Server 1) vom Kontostand in der Datenbank A ein bestimmter Betrag abgezogen. Diese Transaktion wird in der Datenbank B vermerkt, indem durch die Anwendung eine Umbuchungsinformation geschrieben wird. Anschließend muss der Betrag auf dem anderen Konto, das sich in der Datenbank C auf dem DB Server 2 befindet, „gutgeschrieben“ werden.

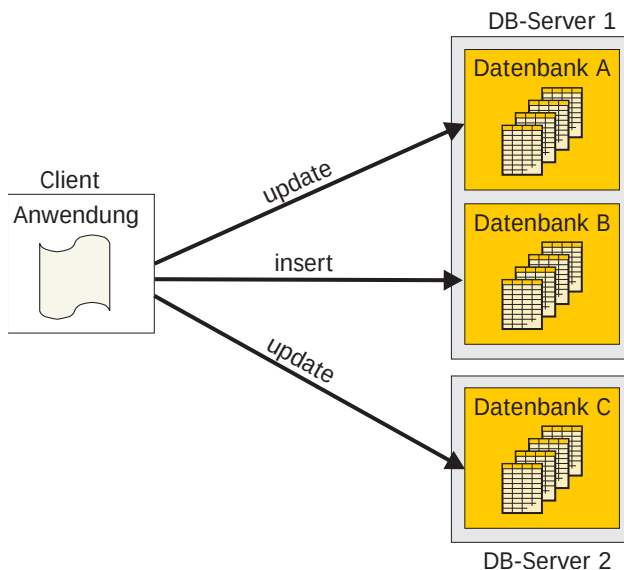


Abbildung 6.35: DUOW-Beispiel

Funktioniert alles, ist es natürlich kein Problem; die Transaktion wird mit einem Commit festgeschrieben. Sie kennen jedoch bestimmt „Murphys Gesetze“ – wenn etwas schief gehen kann, geht es auch schief –, daher muss man für diese Situationen Vorkehrungen treffen. Sollte in unserem Beispiel die Gutschrift auf das zweite Konto nicht erfolgen können, weil z.B. die Netzverbindung zum DB Server 2 momentan nicht funktioniert, darf diese Transaktion nicht bestätigt werden. Sonst könnte es sein, dass das Geld einfach „verschwindet“, wenn es zwar abgebucht

worden ist, aber nie am Zielkonto erschienen ist. Das wollen wir doch alle vermeiden. Die Lösung wäre also, sowohl den Insert als auch die Abbuchung – also den Update – rückgängig zu machen und die Umbuchung später noch einmal zu probieren. So wird es ja auch bei einer „normalen“ Transaktion in der Datenbank gemacht. Hier ist die Schwierigkeit jedoch durch die unterschiedlichen Datenbanken bedingt, die ja alle ihr Commit unabhängig voneinander machen und nicht wissen, wer (welche Datenbank) noch an der Transaktion beteiligt war.

Wie der Name schon sagt, hat dieses Konzept zwei Phasen:

- eine Prepare-Phase
- eine Commit-Phase

Zusätzlich wird ein Transaktionsmanager eingesetzt, um den Vorgang datenbankübergreifend zu überwachen.

Ein Transaktionsmanager wird informiert, dass eine DUOW stattfinden soll. Diese Transaktionsmanager-Funktionalität kann z.B. eine DB2-Datenbank übernehmen (siehe Abbildung 6.36). Es werden alle Änderungen der DUOW ausgeführt. Der Client schickt nun eine „Prepare“-Anforderung an alle beteiligten Datenbanken, auf die diese mit einer positiven Bestätigung antworten. Sollte bis zu diesem Zeitpunkt ein Fehler auftreten, kann die DUOW einfach zurückgerollt werden. Die zweite Phase entspricht dann sozusagen dem Rollback. Die Antwort der beteiligten Datenbanken, dass die Prepare-Phase erfolgreich war, wird nun auch dem Transaktionsmanager mitgeteilt, damit dieser von dem aktuellen Stand der Transaktion informiert ist. In der nun beginnenden zweiten Phase – der Commit-Phase – wird eine „Commit“-Anforderung an alle beteiligten Datenbanken gesendet, die sie darüber informiert, dass alle bereit für einen Commit seien. Auf allen Datenbanken wird dieser Commit durchgeführt und wieder mit einer Rückmeldung an den Client bestätigt.

Empfängt der Client alle Commit-Bestätigungen, wurde die DUOW erfolgreich durchgeführt und durch eine weitere Information an den Transaktionsmanager beendet.

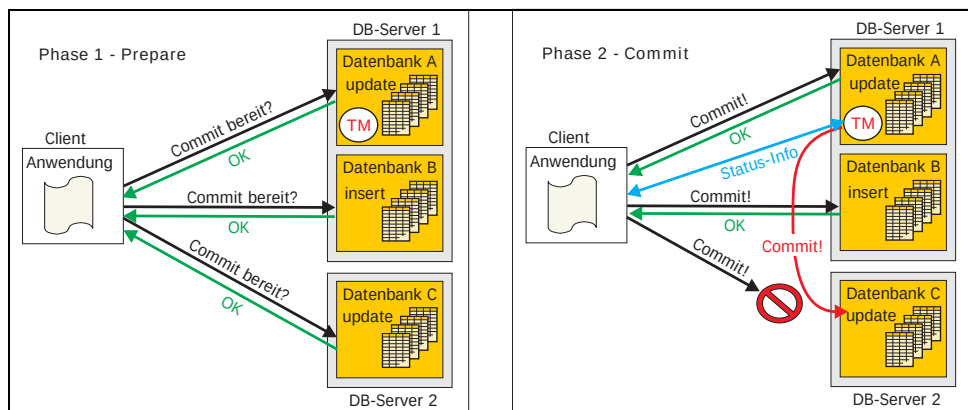


Abbildung 6.36: Phasen der zweiphasigen Festschreibung

Kommt eine Meldung nicht zurück, weil entweder die beteiligte Datenbank die Commit-Aufforderung gar nicht erhalten hat (siehe Abbildung 6.36, Datenbank C) und zu diesem Zeitpunkt nicht verfügbar war, oder ging die Bestätigungsmeldung im Netz verloren, greift der Transaktionsmanager ein. Ein „Rollback“ ist in dieser Phase nicht mehr möglich, da beteiligte Datenbanken schon die Freigabe zur Festschreibung empfangen haben und durch Commit festgeschriebene Daten unter keinen Umständen mehr per Rollback rückgängig gemacht werden können. Der Transaktionsmanager versucht nun periodisch die fehlende Bestätigung zu bekommen, indem er regelmäßig Commit-Anforderungen an die Datenbank schickt, die dies noch nicht bestätigt hat. So wird selbst bei einer Datenbank, die durch z.B. einen Systemausfall zeitweise nicht verfügbar war, sichergestellt, dass sie diese Commit-Anweisung durchführt und ihren Teil der Transaktion festschreibt.

Mit dem Befehl `LIST INDOUBT TRANSACTIONS` können diese „zweifelhaften“ Transaktionen, also die, die noch keine positive Rückmeldung gesendet haben, aufgelistet werden. Weitere Optionen stehen zur Verfügung, um den Transaktionsmanager (`tm_database`) bzw. das Resynchronisationsintervall (`resync_interval`) zu bestimmen sowie weitere Parameter festzulegen, die meist auf Exemplar-Ebene zu konfigurieren sind.

Auch eine DUOW kann so über mehrere Produkte und/oder Datenbanken sichergestellt werden.

Hinweis:

Die zweiphasige Festschreibung kann derzeit nicht mit der DB2 Federated-Eigenschaft genutzt werden, mit der in DB2-Datenbanken Tabellen von Datenbanken anderer Hersteller eingebunden werden.

6.6.1 Aktivierung der zweiphasigen Festschreibung

In der Steuerzentrale steht Ihnen über die Auswahl `AKTUALISIERUNG AUF MEHREREN SYSTEMEN` des Exemplar-Kontextmenüs ein Assistent zur Verfügung, um die notwendigen serverseitigen Einstellungen zu konfigurieren und zu testen. Der `ASSISTENT: AKTUALISIERUNG AUF MEHREREN SYSTEMEN KONFIGURIEREN` ermöglicht die Konfiguration der Transaktionsmanager-Datenbank (`tm_database`). Der Standard ist hier `1ST_CONN`. Das bedeutet, die Datenbank, zu der die erste Verbindung aufgebaut wird, übernimmt zusätzlich die Funktion des Transaktionsmanagers. Es kann hier auch jede andere Datenbank verwendet werden, die eine Verbindung zu den anderen beteiligten Datenbanken erstellen kann.

Um die Befehlszeile auf diese DUOW vorzubereiten, gibt es den `SET CLIENT`-Befehl, der dann so aussehen müsste:

```
set client connect 2 syncpoint twophase
```

Weitere Optionen in diesem Befehl erlauben eine noch feinere Anpassung des „Verhaltens“ der Befehlszeile. Diese Einstellung gilt dann für die Befehlszeile, also den so genannten Backend-Prozess, bis dieser durch einen `TERMINATE`-Befehl beendet wird.

Über den Befehl `QUERY CLIENT` kann man die aktuellen Einstellungen der Befehlszeile abfragen.

6.7 Verbindungen zu anderen Datenbank-Servern

DB2 kann auf Objekte (typischerweise Tabellen) aus anderen Datenbanken und – über das Zusatzprodukt „DB2 Information Integrator“ – sogar auf Datenbanken anderer Hersteller zugreifen. Diese Funktionalität heißt *Federated System* oder zu Deutsch „System mit zusammengeschlossenen Datenbanken“.

Damit können Sie in *einem* SQL-Befehl Daten aus der DB2-Datenbank direkt mit Daten anderer Datenbanken verknüpfen. Somit ist beispielsweise ein JOIN einer DB2-Tabelle mit einer *DB2 für z/OS*-Tabelle und mit einer Oracle-Tabelle möglich.

Dieses Verfahren ist also überall dort nützlich, wo Daten auf mehrere (heterogene) Systeme verteilt sind – wie es heute oft in „gewachsenen“ Unternehmen der Fall ist – und diese gemeinsam ausgewertet und verknüpft werden sollen.

6.7.1 Das „Federated“-Konzept

An dieser Stelle wollen wir das Konzept der zusammengeschlossenen Systeme bzw. Datenbanken vorstellen. Der Zugriff auf die Datenbanken anderer Hersteller wird aber in diesem Kapitel nicht weiter behandelt, da dafür einerseits zusätzliche, kostenpflichtige Software – der *DB2 Information Integrator* – notwendig wäre und die Erläuterungen andererseits den Rahmen dieses Kapitels bzw. Buches sprengen würden. Die hier besprochene Funktionsweise gilt jedoch so auch für die anderen Systeme.

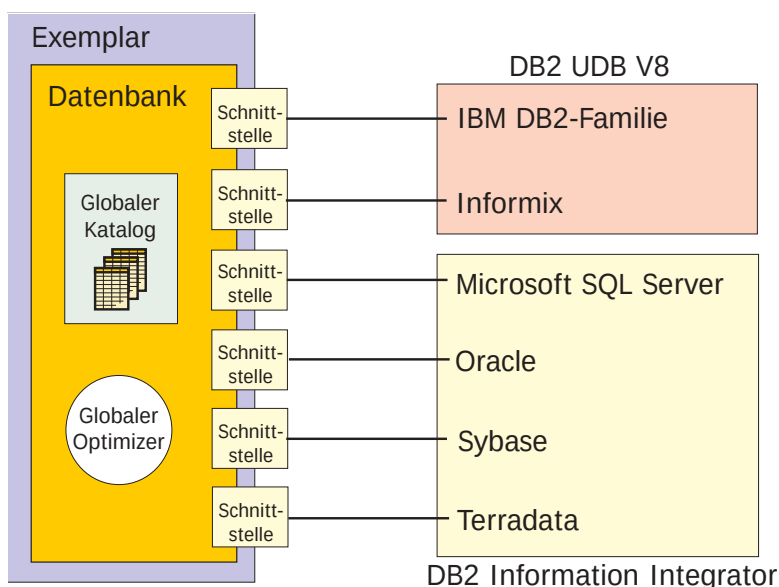


Abbildung 6.37: Das „Federated“-Konzept

Abbildung 6.37 zeigt, dass der Zugriff auf die angeschlossenen Systeme über eine Schnittstelle funktioniert, die in das DB2 eingebunden wird. Diese Schnittstelle ist ein Programm, das den Zugriff auf das entsprechende Ziel-System ermöglicht – hier wird üblicherweise die Client-Komponente des jeweiligen Zielsystems (also z.B.

Net8 für Oracle-Zugriffe) eingebunden. In DB2 wird eine derartige Schnittstelle registriert – ein so genannter Wrapper. Fremdsysteme sind über die Zusatzsoftware zu erreichen, während der Zugriff auf die IBM-eigenen Produkte DB2 und Informix bereits im Standard-DB2 möglich ist.

Die eigentlichen „Highlights“ dieser Funktionalität sind folgende:

- ▶ Ein *globaler Katalog* enthält Informationen über die Datenquellen. Neben den lokalen Datenbankobjekten sind dort also auch Daten über die Quellsysteme bzw. Tabellen enthalten. Eine Anwendung, die sich zu dieser DB2-Datenbank verbindet, „sieht“ alle Strukturen, unabhängig davon, wo sie tatsächlich physikalisch gespeichert sind.
- ▶ Durch den globalen Katalog wird es möglich, eine systemübergreifende, also *globale Optimierung* durchzuführen. Bei der Verknüpfung von Daten aus unterschiedlichen Systemen, die auch auf unterschiedlichen Servern liegen können, stellt das eine große Herausforderung dar. Hier spielen die Rechenleistung des Systems, Netzverbindungen zu den angeschlossenen Systemen und natürlich die Datenmengen eine entscheidende Rolle für die Leistungsfähigkeit eines solchen Systems.
- ▶ *Transparenz* für die Anwendung – d.h., eine Anwendung kann einen Zugriff nicht gleichzeitig im Oracle- und Sybase- und DB2-Dialekt absetzen, sondern muss eine Programmier Technik und einen SQL-Dialekt aufweisen. In diesem zusammengeschlossenen System kann eine Anwendung im „DB2-Stil“ erstellt werden, auch dann, wenn sie auf die anderen Systeme durchgreift. Dazu wird in DB2 die Funktionalität emuliert, die zwar DB2 bietet, aber das verbundene System so vielleicht nicht unterstützt, wodurch die korrekte Funktionsweise sichergestellt wird. Dass das nicht ganz so einfach ist, ist wohl leicht einzusehen.
- ▶ Eine weitere Möglichkeit erlaubt auch den „Durchgriff“ auf die angeschlossenen Systeme, um Befehle in der Syntax des Zielsystems abzusetzen. Auf diese Weise lassen sich dann z.B. auch Tabellen auf dem angeschlossenen System erstellen.

Im Folgenden wollen wir die Technik und Vorgehensweise zur Definition zusammengeschlossener Datenbanken anhand eines Zugriffs auf eine Tabelle einer anderen DB2-Datenbank behandeln.

6.7.2 Zugriff auf eine andere DB2-Datenbank

Von einer DB2-Datenbank soll der Zugriff auf eine Tabelle einer anderen DB2-Datenbank – ohne separaten CONNECT – ermöglicht werden, um so die lokale und entfernte Tabelle z.B. verknüpfen zu können.

Diese Schritte müssen in der aufgeführten Reihenfolge durchgeführt werden:

1. Ausgangsexemplar für die Definition zusammengeschlossener Systeme vorbereiten
2. Definition eines so genannten *Wrappers*
3. Anlegen eines Servers
4. Definition einer so genannten Benutzerzuordnung (user mapping)
5. Erstellen eines Kurznamens (nickname)

Damit kann dann der in Abbildung 6.38 gezeigte Zugriff umgesetzt werden.

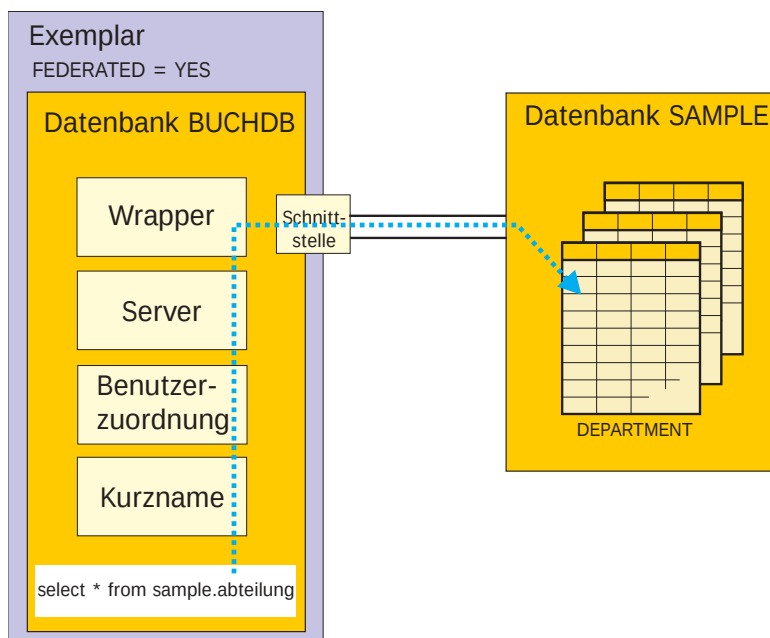


Abbildung 6.38: Zugriff auf eine Tabelle über die „Federated“-Funktionalität

Der folgende Ablauf wird nur über die Befehle beschrieben, ist aber auch über die Steuerzentrale sehr komfortabel möglich.

Hinweis:

Sollte beim Aufruf der (deutschsprachigen) grafischen Oberfläche (Steuerzentrale) bei der Erstellung eines Wrappers ein Fehler auftreten, prüfen bzw. kopieren Sie den kompletten Inhalt des Verzeichnisses `...\sqllib\tools\en_US` in das Verzeichnis `...\sqllib\tools\de_DE`. Dies sollte diesen Fehler beheben.

Zuerst muss sichergestellt werden, dass das lokale Exemplar, in dem die Ausgangsdatenbank liegt, für Objekte zusammengesetzener Datenbanken aktiviert ist. Setzen Sie dazu über die Exemplarkonfiguration den Parameter `FEDERATED` auf `YES` bzw. prüfen Sie, dass er auf `YES` gesetzt ist.

Anschließend muss ein so genannter *Wrapper* angelegt werden. Dies ist ein Modul, das die Interaktionen mit der angeschlossenen Datenquelle erlaubt. Für jede Art einer Datenquelle muss einmal in der Datenbank ein Wrapper angelegt werden, und zwar über die Steuerzentrale (Eintrag `OBJEKTE ZUSAMMENGESCHLOSSENER DATENBANKEN` im Objektbaum) oder folgenden Befehl:

```
create wrapper drda library 'db2drda.dll'
```

DRDA steht dabei für den Wrapper-Namen, und db2drda.dll ist das Modul zur Verbindung zu Datenbanken der DB2-Familie. Weitere Module für die in Abbildung 6.37 gezeigten Datenbanken bzw. auch ODBC können hier spezifiziert werden. Somit wird die Art der Quelle definiert.

Im nächsten Schritt werden durch eine Serverdefinition der spezielle Server und die Datenbank definiert, auf die zugegriffen werden soll. Die Datenbank muss dabei dem „Wrapper-Typ“ entsprechen, der oben definiert wurde. In unserem Fall müssen wir also den Server mit der anderen DB2-Datenbank spezifizieren.

In der Steuerzentrale können Sie das über das Kontextmenü des Server-Eintrags tun, der unter Ihrem neu erstellten Wrapper angezeigt wird. Alternativ verwenden Sie folgende Befehlsstruktur:

```
create server <servername> type <db-typ> version '<versionsnr>' wrapper <wrappername>  
authid "<benutzer-id>" password "<kennwort>" options ([node 'knotenname',] dbname  
'<datenbankname>', password 'Y')
```

Folgende Punkte müssen also enthalten sein:

- ▶ **servername** – Dieser kann beliebig gewählt werden.
- ▶ **type** – ist für Zugriffe auf andere DB2-Workstation-Datenbanken immer DB2/UDB.
- ▶ **version** – gibt die Version der Zieldatenbank an.
- ▶ **wrapper** – Der Wrapper-Name muss mit dem Namen des vorher definierten Wrappers übereinstimmen.
- ▶ **authid** – Benutzer-ID eines zum Anlegen berechtigten Benutzers.
- ▶ **password** – zur Benutzer-ID gehörendes Kennwort.
- ▶ **node** – Ein Knotenname muss angegeben werden, wenn sich die Ziel-Datenbank in einem anderen Exemplar befindet.
- ▶ **dbname** – Name der anderen (entfernten bzw. angeschlossenen) Datenbank.
- ▶ **password** – gibt an, dass das Kennwort mit übertragen werden soll.

In unserem Beispiel sähe das für eine Datenbank des gleichen Exemplars konkret so aus:

```
db2 "create server db2serv1 type DB2/UDB version '8.1' wrapper DRDA authorization  
\"db2admin\" password \"admindb2\" options(dbname 'sample', PASSWORD 'Y')"
```

Hinweis:

Um doppelte Anführungszeichen in einem Befehl verwenden zu können, der selbst in doppelte Anführungszeichen gekapselt ist, müssen diese *maskiert* werden. Die Maskierung erfolgt über „\“ (Backslash). Fehlt diese Maskierung, schlägt der Befehl fehl.

Für den Zugriff auf eine DB2-Datenbank eines anderen Exemplars ist das Katalogisieren eines Knotens eine wichtige Voraussetzung (wie auch weiter vorn für den Zugriff auf entfernte Datenbanken beschrieben wurde). Das Katalogisieren des Knotens dient dazu, die Verbindungsdaten bereitzustellen und den Zugriff zu ermöglichen.

```
db2 "create server db2serv2 type DB2/UDB version '8.1' wrapper DRDA authorization  
\"db2admin\" password \"admindb2\" options(node 'db2inst', dbname 'sample',  
PASSWORD 'Y')"
```

Der nächste Schritt – das Anlegen einer Benutzerzuordnung – dient dazu, eine lokale Benutzer-ID einer berechtigten Benutzer-ID auf dem entfernten Ziel-Datenbankserver zuzuordnen. Dieser Schritt ist immer notwendig – auch wenn die Datenbank in dem gleichen Exemplar auf dem lokalen Rechner liegt.

Somit definieren wir für unser Beispiel folgende Benutzerzuordnung:

```
create user mapping for db2admin server db2serv1 options (remote_authid 'db2admin',  
remote_password 'admindb2')
```

Der letzte Schritt besteht nun darin, einen so genannten Nickname oder eben Kurznamen für die Tabelle bzw. das Objekt in der nun angeschlossenen Datenbank zu vergeben. Während Sie sich im GUI eine Auswahl der Tabellen anzeigen lassen können, verwenden Sie beim manuellen Weg diesen Befehl:

```
create nickname <schema_lokal>.<tabellenname_lokal> for <server>.<schema_fern>.  
<tabellenname_fern>
```

Um auf die DEPARTMENT-Tabelle der angeschlossenen SAMPLE-Datenbank zugreifen zu können und diese lokal unter dem Namen SAMPLE.ABTEILUNG bekannt zu machen, ist folgender Befehl notwendig:

```
db2 create nickname sample.abteilung for db2serv1.db2admin.employee
```

Nun können Sie diese (Federated-)Tabelle wie eine normale, lokale Tabelle nutzen und bekommen also über

```
select * from sample.abteilung
```

den Inhalt der Tabelle DB2ADMIN.DEPARTMENT aus der SAMPLE-Datenbank angezeigt, obwohl Sie mit der Datenbank BUCHDB verbunden sind und die Tabellendaten nicht lokal in dieser Datenbank gespeichert sind (vgl. Abbildung 6.38).

Für diesen Zugriff auf zusammengeschlossene Systeme gelten einige Einschränkungen. So ist z.B. eine zweiphasige Festschreibung über diese Funktionalität nicht möglich. Trotzdem ist dies eine sehr praktische Integrationsmöglichkeit von Daten aus anderen Systemen, die weder den Import der Daten noch einen Replikationsvorgang erfordert. Über das Produkt DB2 Information Integrator können dann auch noch zahlreiche andere – auch nicht-relationale – Datenquellen angeschlossen werden.

6.8 Tipps aus der Praxis

6.8.1 Fehlerdiagnose für den Verbindungsaufbau

Mit dem in diesem Kapitel vermittelten Stoff können Sie schon sehr viele Probleme lösen, wenn einmal ein Verbindungsaufbau einfach nicht funktionieren will. Leider sind die Fehlermeldungen nicht immer wirklich aussagekräftig, um sofort erkennen zu lassen, welche Einstellung nicht stimmt. Einige Befehle und Tools können Ihnen in solchen Fällen noch hilfreich sein.

Folgende Punkte sind dabei zu überprüfen:

- ▶ genaue Analyse des gemeldeten Fehlers inklusive des protokollspezifischen Fehlercodes, z.B. mittels `db2 ? <SQLFehlercode>`.
- ▶ Stellen Sie sicher, dass der entfernte Server läuft und erreicht werden kann. Hier leistet der `ping`-Befehl gute Dienste.
- ▶ Das ferne Exemplar muss gestartet und für die Kommunikation vorbereitet worden sein.
- ▶ Stimmen die Einträge in dem Systemdatenbankverzeichnis, im Knotenverzeichnis und eventuell im DCS-Verzeichnis?

Dabei kann die `SHOW DETAIL`-Option für die Anzeige des Knotenverzeichnisses schnell die notwendigen Daten mit zusätzlichen Details auflisten (siehe Abbildung 6.39). Folgende zusätzliche Informationen werden angezeigt:

- Name des fernen Exemplars
- System
- Betriebssystem

```
D:\Programme\IBM\SQLLIB\BIN>db2 list node directory show detail
```

Knotenverzeichnis

Anzahl Einträge im Verzeichnis = 1

Eintrag für Knoten 1:

Knotenname	= TCPA590D
Kommentar	=
Verzeichniseintragungsart	= LOCAL
Protokoll	= TCPIP
Host-Name	= 192.168.60.43
Servicename	= 55000
Name des fernen Exemplars	= BUCHINST
System	= 192.168.60.43
Betriebssystemtyp	= WIN

Abbildung 6.39: Beispiel für ein Knotenverzeichnis mit `show detail`-Option

Auch das folgende Übereinstimmungsschaubild (siehe Abbildung 6.40) soll noch einmal die wichtigsten Einträge zeigen, bei denen eine Übereinstimmung gefordert ist.

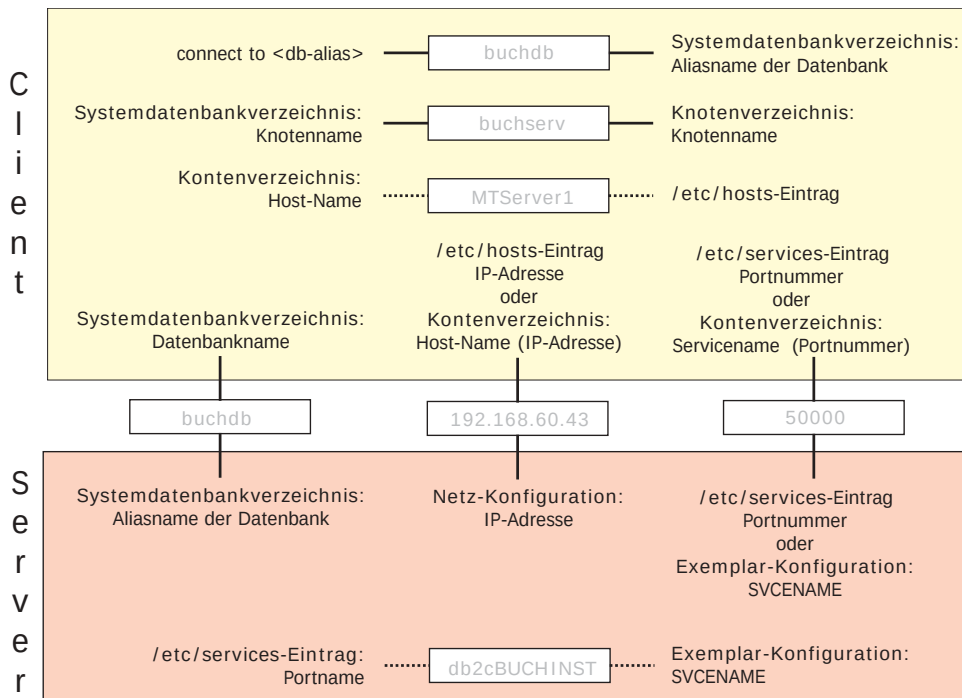


Abbildung 6.40: Schaubild Verbindungsübereinstimmung

- Auch die „Rückwärtsauflösung“ des Rechnernamens muss funktionieren. Das bedeutet: Nicht nur der Client muss den Server über den Namen bzw. die IP-Adresse „finden“, sondern auch der Server muss den Client kennen. „Normalerweise“ ist das kein Problem – aber was ist in der Praxis schon „normal“?

Hat der Server nämlich lokal dem Client-Rechnernamen eine andere IP-Adresse (in der /etc/hosts-Datei des Servers) zugeordnet, als dieser tatsächlich hat, schlägt der Verbindungsaufbau fehl.

6.8.2 Passwortänderung vom Client

Sie haben die Möglichkeit, Ihr Passwort am Server vom Client aus zu ändern. Dies geht durch einen erweiterten CONNECT-Befehl mit der Struktur:

```
connect to <db-aliasname> user <benutzer-id> using <altes-kennwort>
new <neues-kennwort> confirm <neues-kennwort>
```

Hinweis:

Bei der Passwortänderung im CONNECT-Befehl wird das neue Kennwort immer unverschlüsselt übertragen, auch wenn der Server mit der Authentifizierungsmethode SERVER_ENCRYPT konfiguriert ist.

6.8.3 LDAP – Lightweight Directory Access Protocol

LDAP ist eine Standard-Zugriffsmethode für einen Verzeichnisdienst. Für DB2 bedeutet das, dass Sie die für den Zugriff notwendigen Verzeichnisse, wie das Systemdatenbankverzeichnis und das Knotenverzeichnis, nicht mehr lokal auf jeder Maschine halten, sondern *zentral* auf einem LDAP-Server.

Im DB2 stehen zwei Cache-Mechanismen zur Verfügung, um die einmal vom LDAP-Server gelesenen Informationen lokal zwischenspeichern zu können und einen solchen Leistungsvorteil bei einem erneuten Zugriff gewährleisten zu können.

Um eine LDAP-Umgebung einzurichten, sind mehrere Schritte notwendig. Unter anderem muss jeder DB2-Server auf dem LDAP-Server registriert werden, und auch eigene CATALOG LDAP-Befehle stehen zur Verfügung, um die Verzeichniseinträge zu verwalten.

Eine Beschreibung, wie eine solche Umgebung aufzubauen ist, finden Sie in [SNOW].

7 Datenbankobjekte

Jetzt ist es wirklich Zeit, sich mehr Gedanken über die Datenbankobjekte zu machen. Sie haben bisher einen Überblick über Exemplare, Datenbanken und Tabellenbereiche bekommen, so dass wir uns nun mit Datenbankobjekten beschäftigen können.

Bei Datenbankobjekten denkt man zuerst an Tabellen und Indizes. Wir wollen Ihnen hier, neben diesen grundlegenden Objekten, aber auch deren spezielle Ausprägungen zeigen. Außerdem behandeln wir, wie bestimmte funktionelle Abhängigkeiten auch von der Datenbank selbst übernommen werden können – Stichwort: referenzielle Integritätsbedingungen.

Dieses Kapitel behandelt die Erstellung der verschiedenen Datenbankobjekte mit den wichtigsten Optionen und zeigt ihre Verwendungsmöglichkeiten auf. Dazu benutzen wir die DDL (Data Definition Language) oder Datendefinitionssprache. Diese ist Teil des SQL-Standards und enthält die Befehle, um Datenbankobjekte zu erstellen (CREATE) oder zu löschen (DROP).

7.1 Schemata (Schema)

Ein Schema hilft dabei, Datenbankobjekte zu gruppieren. Die meisten Datenbankobjekte sind einem Schema zugeordnet und haben dadurch einen zweiteiligen Namen nach dem Muster:

`<Schema-Name>.<Objektnamen>`

also z.B. `PRODUKTION.AUFTRAG`.

Sie können sich einfach den Schema-Namen wie einen Familiennamen vorstellen. Es kann mehrere Personen mit dem Vornamen „Sonja“ geben, aber in einer Familie wird es nur eine Person mit diesem Namen geben. Personen können daher eindeutig angesprochen werden, wenn man Nachname und Vorname angibt.

Genauso funktioniert es auch in einer Datenbank. Es kann mehrere Objekte mit dem Namen `AUFTRAG` geben, aber es darf nur eine Tabelle `PRODUKTION.AUFTRAG` geben.

In der Steuerzentrale können Sie sich eine Übersicht darüber verschaffen (siehe Abbildung 7.1), welche Schemata vorhanden sind. Folgende Schemata, mit der reservierten Zeichenfolge „SYS“, werden beim Anlegen der Datenbank erstellt:

- ▶ `SYSIBM` – als Schema-Name für die Systemkatalog-Tabellen
- ▶ `SYSCAT` – als Schema-Name für die Systemkatalog-Sichten
- ▶ `SYSSTAT` – als Schema-Name für die Statistik-Systemkatalog-Sichten
- ▶ `SYSPROC` – als Schema-Name für die System-Prozeduren
- ▶ `SYSFUN` – als Schema-Name für die System-Funktionen
- ▶ `NULLID` – als Schema-Name für die Standard-DB2-Datenbankprogramme

Ein Schema-Name darf max. 30 Stellen haben.

Meistens werden Schemata implizit, also durch das Anlegen von Objekten, erstellt. Legt z.B. der Benutzer „db2admin“ ein Objekt an, wird automatisch der Benutzername als Schema-Name des Objekts verwendet, falls nicht anders (explizit) angegeben. So erstellt der Benutzer „Ivo“ durch den Befehl `create table test (c1 int)` die Tabelle IVO.TEST, während der Benutzer Jens mit dem identischen Befehl die Tabelle JENS.TEST erstellt.

Auch die explizite Erstellung von Schemata ist möglich. In Abbildung 7.1 wird dies anhand der grafischen Oberfläche gezeigt.

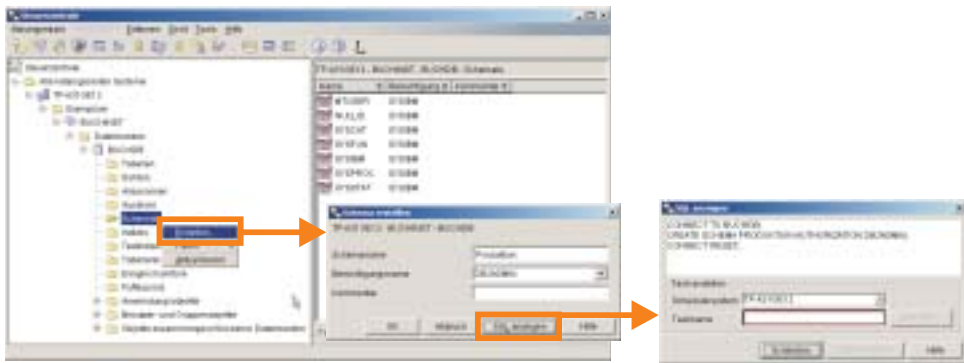


Abbildung 7.1: Schema mit der Steuerzentrale erstellen

Durch die explizite Erstellung sind Sie auch nicht mehr daran gebunden, dass das Schema den Namen eines Benutzers haben muss. Die Gruppierung Ihrer Produktionstabellen ist somit unter dem Schema PRODUKTION möglich.

Abbildung 7.1 zeigt, wie Sie über das Kontextmenü des SCHEMATA-Eintrags unter einer Datenbank auf das SCHEMA ERSTELLEN-Fenster gelangen und von dort über den SQL ANZEIGEN-Button zum generierten Befehl kommen können.

Bei der Erstellung eines neuen Schemas können Sie einen Berechtigungsnamen angeben, der festlegt, wem das Schema gehören soll. Dieser Benutzer hat dann alle Rechte im Schema und kann darin Objekte erstellen, löschen oder verändern.

Über die Befehlszeile sieht das dann so aus:

```
create schema produktion
```

Wenn das Schema dem Benutzer IVO zugeordnet werden soll, wird die AUTHORIZATION-Option verwendet:

```
create schema produktion authorization ivo
```

Ivo kann nun Tabellen im Schema PRODUKTION erstellen, wenn er weiß, wie der CREATE TABLE-Befehl lautet – oder wenn er den nächsten Abschnitt gelesen hat.

7.2 Tabellen

Eine Tabelle (table) wird immer in bzw. mit einem bestimmten Schema-Namen angelegt. Eine Tabelle ist ein logisches Datenbankobjekt mit Zeilen und definierten Spalten. Jede Spalte einer Tabelle braucht einen Namen, und ihr ist ein bestimmter Datentyp (integer, char etc.) zugeordnet.

Dieser Datentyp definiert, welche Art von Daten in einer Spalte bzw. einem Feld gespeichert werden dürfen. Zudem wird auch die physikalische Speicherstruktur durch diese Datentypen bestimmt. In DB2 stehen folgende Gruppen von Datentypen zur Verfügung:

► Numerische Felder

Diese Felder beinhalten Zahlenwerte und unterscheiden sich darin, wie groß (oder klein) die Zahlen sein können, ob Kommastellen erlaubt sind etc. Beispiele hierfür sind:

- INTEGER (INT)
- FLOAT
- REAL
- DECIMAL (DEC)

► Alphanumerische Felder

Diese Datentypen dürfen neben Buchstaben auch Zahlen und Sonderzeichen beinhalten und werden als Textfelder interpretiert. Dazu gehören:

- CHARACTER (CHAR)
- VARCHAR

► Besondere Datentypen

Für spezielle Inhalte gibt es eigene Datentypen, um entweder besonders große Inhalte abzuspeichern oder ein besonderes Format zu nutzen. Dazu zählen:

- Große Objekte (large objects)

Diese Datentypen dienen dazu, Daten zu speichern, die durch ihre Größe nicht in den anderen Datentypen gespeichert werden können. Dazu zählen u.a.:

- BLOB (Binary Large Object) für binäre Daten wie z.B. Audio- oder Bilddaten
- CLOB (Character Large Object) – dort können Sie ganze Bücher in einem Feld speichern.

- Zeit- und Datumsangaben

DATE, TIME oder TIMESTAMP sind Vertreter dieser Kategorie.

Die genauen (Größen-)Limits für die einzelnen Datentypen finden Sie im Anhang mit den SQL-Limits.

In diesen Tabellen-Strukturen werden später Ihre Daten gehalten, wobei die mögliche Größe – also die Anzahl der Zeilen – nur von dem Platz abhängt, der in der Datenbank zur Verfügung steht.

Die Speicherung der Tabelle geschieht mittels der DB2-Verwaltung auf Speicherseiten in Behältern (siehe Abbildung 7.2).

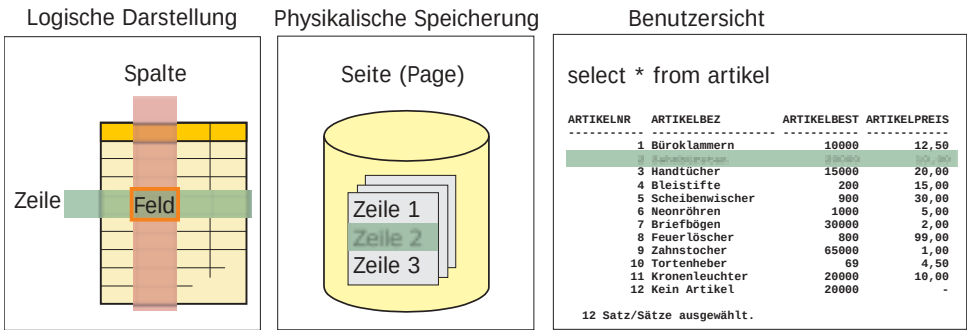


Abbildung 7.2: Tabelle

Der Benutzer greift per SQL (Structured Query Language) auf die Tabelleninhalte zu. Tabellen werden in SQL in der FROM- oder in der INTO-Klausel angegeben.

Betrachten wir die Geschichte einmal von einer ganz anderen Seite: Woher weiß ich, welche Tabellen ich mit welchen Spalten brauche, und was für einen Datentyp soll ich wählen?

Hier kommt die Datenmodellierung ins Spiel. Sie wurde schon kurz in Kapitel 1 angesprochen. Sie stellt Techniken zur Verfügung, um Ihre Geschäftsmodelle und -prozesse auf ein so genanntes ER-Modell (Entity-Relationship-Modell) abzubilden. Aus diesem Modell werden dann die Tabellen (und anderen Objekte) erzeugt. Die Bestimmung des passenden Datentyps hängt auch von der Erfahrung des Modellierers ab. Es gibt in diesem Umfeld nicht nur *ein richtiges* Modell und *eine* Implementierung, sondern viele Lösungen – jede mit individuellen Vor- und Nachteilen. Diese gilt es abzuwägen. Dieser Prozess der Datenmodellierung wird von DB2 nicht direkt unterstützt. Dafür unterstützen Modellierungswerkzeuge verschiedener Hersteller die DB2 UDB, um aus einem Modell automatisch die Tabellenstrukturen etc. in der Datenbank zu erstellen. Wir haben dabei, wie schon gesagt, die umfangreichen und nützlichen Funktionen des *AllFusion ERwin Data Modeller* der Firma Computer Associates zu schätzen gelernt.

Informationen über Ihre Tabellen sind im Datenbank-Katalog hinterlegt. Die Systemkatalog-Sicht SYSCAT.TABLES beinhaltet Informationen zu den Tabellen an sich, und in der Sicht SYSCAT.COLUMNS sind alle Spaltendefinitionen verzeichnet. Daneben gibt es noch einige weitere Sichten mit Zusatzinformationen. Eine vollständige Liste der Systemsichten finden Sie in [SQLREF].

7.2.1 Anlegen von Tabellen

Zurück zum Anlegen von Tabellen: Es steht Ihnen ein Assistent über die Steuerzentrale zur Verfügung und selbstverständlich, wie immer, ein entsprechender SQL-DDL-Befehl. Der ASSISTENT: TABELLE ERSTELLEN führt Sie Schritt für Schritt durch den Erstellungsprozess und ist sicherlich für die ersten „Gehversuche“ mit DB2 nütz-

lich. Später aber werden Sie nur noch die Variante über die Befehlszeile nutzen, da diese schneller ist und die Einbindung des CREATE TABLE-Befehls in ein Script viele Vorteile bietet. So können Sie alle Tabellen der Datenbank mit einem einzigen Script anlegen lassen. Sie haben so auch eine *Quasi*-Dokumentation, die es Ihnen zusätzlich ermöglicht, einfach und schnell kleine Fehler zu korrigieren oder Änderungen vorzunehmen, um dann das Script einfach erneut auszuführen.

Beide Techniken, GUI und Befehl, stellen wir Ihnen im Folgenden vor.

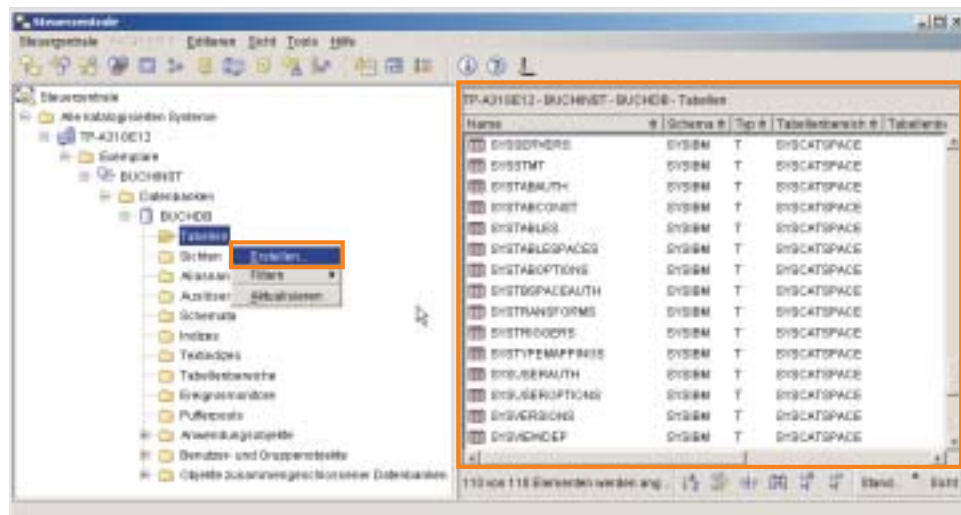


Abbildung 7.3: Assistent: Tabelle erstellen – Aufruf

Der Aufruf des Assistenten erfolgt über den ERSTELLEN...-Eintrag des Kontextmenüs auf Tabellen-Ebene. Dabei sehen Sie im Detailfenster schon die Liste Ihrer Systemtabellen – alle mit dem Schema-Namen SYSIBM –, die bei der Erstellung der Datenbank mit angelegt wurden.

Im ersten Schritt wählen Sie das gewünschte Schema aus oder überschreiben einfach den Standard-Schema-Namen und geben zusätzlich den Tabellennamen an. Der Tabellename kann bis zu 128 Zeichen lang werden. Das sollte für eine sinnvolle, sprechende Benennung genügen. Vergessen Sie nicht: Sie können Zeit sparen, wenn Sie allein durch den sprechenden Namen wissen, welche Tabelle Sie bei der Abfrage benötigen und nicht erst über eine Liste einen kryptischen Namen entschlüsseln müssen. Andererseits will so ein Name auch getippt werden – bei der SQL-Abfrage beispielsweise; da kann ein zu langer Name auch hinderlich sein. Finden Sie Ihren eigenen Kompromiss, und berücksichtigen Sie eventuelle Probleme mit Programmen von anderen Herstellern, die diese Grenze (128 Zeichen) vielleicht nicht so großzügig bemessen.



Abbildung 7.4: Assistent: Tabelle erstellen – Schritt 1 – Name

Die Spalten (im Schritt 2) werden einzeln über den HINZUFÜGEN...-Button erstellt. Auf dem SPALTEN HINZUFÜGEN-Fenster können Sie – neben dem Spaltennamen, für den Ihnen max. 30 Stellen zur Verfügung stehen – weitere Spaltenoptionen angeben (z.B. ob die Spalte leer bleiben darf oder gefüllt werden muss; siehe Checkbox DATENEINGABE ERFORDERLICH).

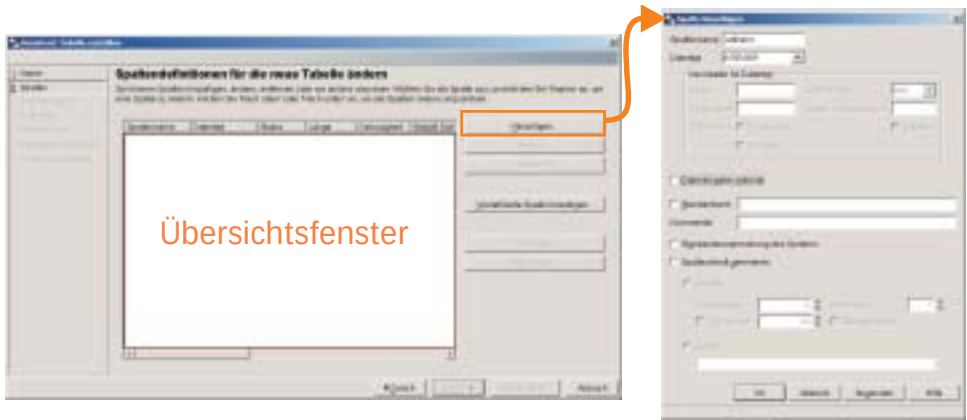


Abbildung 7.5: Assistent: Tabelle erstellen – Schritt 2 – Spalten

Haben Sie alle Spalten mit den gewünschten Datentypen und Spaltenoptionen hinzugefügt, ist es sinnvoll, diese Definitionen im Übersichtsfenster des Assistenten noch einmal gewissenhaft zu überprüfen und gegebenenfalls über den ÄNDERN...-Button zu korrigieren. Wenn Sie Spalten einmal angelegt haben, können Sie zwar zusätzliche Spalten hinzufügen, aber die bereits vorhandenen Spalten nicht mehr entfernen. Auch eine Korrektur des Datentyps oder der Spaltenoption (gerade das wird häufig übersehen) ist nicht mehr möglich. Das heißt, Sie müssen die Tabelle löschen (DROP TABLE) und neu anlegen.

In Schritt 3 geben Sie den Tabellenbereich an, in den die Tabelle abgelegt werden soll.



Abbildung 7.6: Assistent: Tabelle erstellen – Schritt 3 – Tabellenbereich

Bei DMS-Tabellenbereichen können Sie zusätzlich einen separaten (DMS-)Tabellenbereich für die Indizes und optional einen weiteren für die LOB- und LONG-Objekte angeben, um so eine noch bessere Parallelität und Leistung zu erzielen (siehe auch Abschnitt 5.8.3).

Im nächsten Schritt kann definiert werden, aus welcher bzw. welchen Spalte(n) der Schlüssel für die Tabelle bestehen bzw. zusammengesetzt werden soll. Ein Schlüssel – genauer ein Primärschlüssel (primary key) – muss einen einzelnen Tabelleneintrag eindeutig identifizieren.

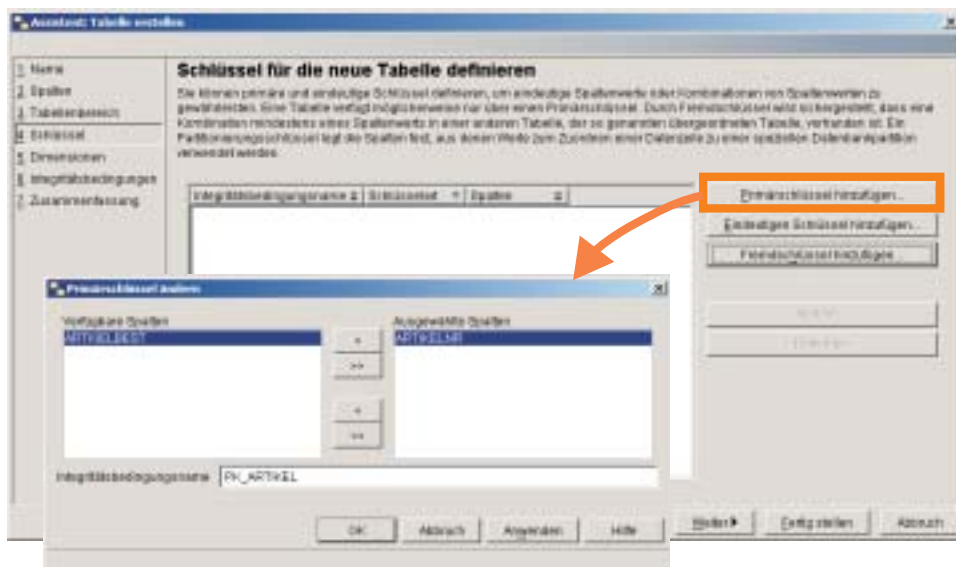


Abbildung 7.7: Assistent: Tabelle erstellen – Schritt 4 – Schlüssel

Die optionalen Schritte 5, DIMENSION, und 6, INTEGRITÄTSBEDINGUNGEN, werden in den folgenden Abschnitten separat behandelt.

Abschließend erhalten Sie auf der Zusammenfassungsseite eine grobe Übersicht über Ihre Definitionen, und über den SQL ANZEIGEN-Button können Sie den generierten CREATE TABLE-Befehl sehen.

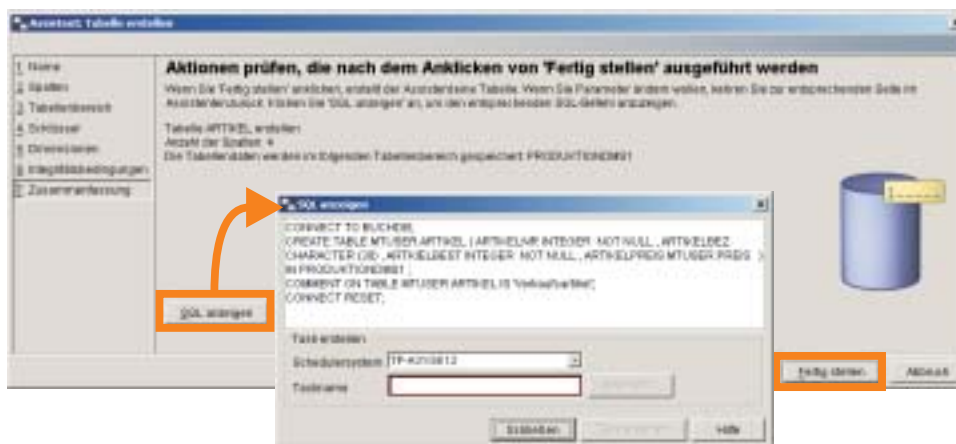


Abbildung 7.8: Assistent: Tabelle erstellen – Schritt 7 – Zusammenfassung

Dieser angezeigte Befehl hält sich an die – stark gekürzte – CREATE TABLE-Syntax aus Abbildung 7.9.

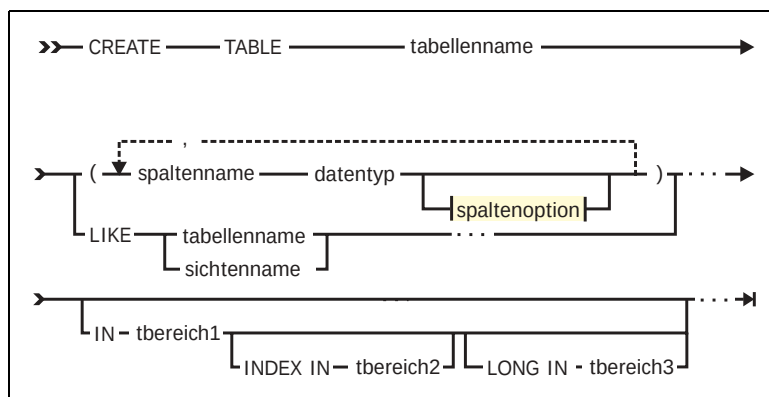


Abbildung 7.9: create table-Syntaxdiagramm

Der CREATE TABLE-Befehl bzw. seine Syntax ist sehr umfangreich. Es können viele z. T. sehr spezielle Optionen angegeben werden, und einiges kann auch im Nachhinein über einen ALTER TABLE-Befehl gesetzt werden, was oft sogar vorteilhaft ist. Wir haben uns hier auf das Wesentliche beschränkt.

Der Tabellename kann auch vollqualifiziert angegeben werden, d.h. mit Schema-Namen. Statt einer detaillierten, expliziten Spalten- und Datentyp-Definition kann auch die LIKE-Option verwendet werden, mit der eine Kopie einer anderen Tabellen- oder Sichten-Struktur erstellt wird. Das heißt, es wird die Definition, nicht aber der Inhalt kopiert.

Alternativ wird jede Spalte, aus der die Tabelle bestehen soll, einzeln mit Spaltenname, Datentyp und eventuell noch einer so genannten Spaltenoption aufgeführt. Mit Hilfe dieser Spaltenoption können Sie beispielsweise durch die Angabe von NOT NULL definieren, dass einer Spalte *immer* ein Wert zugeordnet sein muss, diese Spalte also nie leer sein darf. Auch sog. „Constraint“-Definitionen können hier gemacht werden. Darunter fällt z.B. die Angabe des Primärschlüssels.

Die IN-Option dient dazu, die Tabelle einem Tabellenbereich zuzuweisen. Falls dieser Tabellenbereich (tbereich1) vom Typ DMS ist, können auch die INDEX IN- und LONG IN-Optionen verwendet werden.

Die Option NOT LOGGED INITIALLY beim CREATE TABLE-Befehl wird in Kapitel 9, „Sicherung und Wiederherstellung einer Datenbank“, näher vorgestellt.

Mit einem ALTER TABLE-Befehl können Sie auch nach der Tabellenerstellung u.a. noch folgende Änderungen vornehmen:

- ▶ Spalten hinzufügen
- ▶ Primärschlüssel-Definitionen hinzufügen oder löschen
- ▶ Referenzielle Integrität (siehe Abschnitt 7.7 „Referenzielle Integrität (Referential Integrity – RI)“) und Prüfbedingungen (siehe Abschnitt 7.6 „Prüfung auf Integritätsbedingung (check constraint)“) definieren oder entfernen

Hinweis:

Es ist nicht möglich, Spalten umzubenennen, umzudefinieren (mit Ausnahme der Länge von VARCHAR-Feldern) oder zu entfernen!

Über den Befehl RENAME haben Sie die Möglichkeit, eine Tabelle umzubenennen. Der RENAME-Befehl hält sich an folgendes Schema:

```
rename table <alter Tabellename> to <neuer Tabellename>
```

Es gibt in DB2 noch einige besondere Tabellen, die wir im folgenden Unterkapitel behandeln wollen.

Die so genannten *typisierten Tabellen* (typed tables) beinhalten Spalten, die als *benutzerdefinierte, strukturierte Typen* durch einen CREATE TYPE-Befehl definiert wurden. Mit dieser Art von Tabellen lässt sich dann auch eine Hierarchie mit Wurzel-Tabelle und *Supertype-Subtype*-Beziehungen aufbauen. In die Praxis hat dieser eher mit der Objektorientierung verwandte Ansatz bisher kaum Einzug gehalten.

7.2.2 Temporäre Tabellen (global temporary tables)

Temporäre Tabellen – genauer „deklarierte temporäre Tabellen“ – können eingesetzt werden, um Zwischenergebnisse zu speichern und weitere Abfragen und Analysen auf diesen durchzuführen. Dabei gibt es einige Besonderheiten und auch einen eigenen DDL-Befehl (siehe Abbildung 7.10).

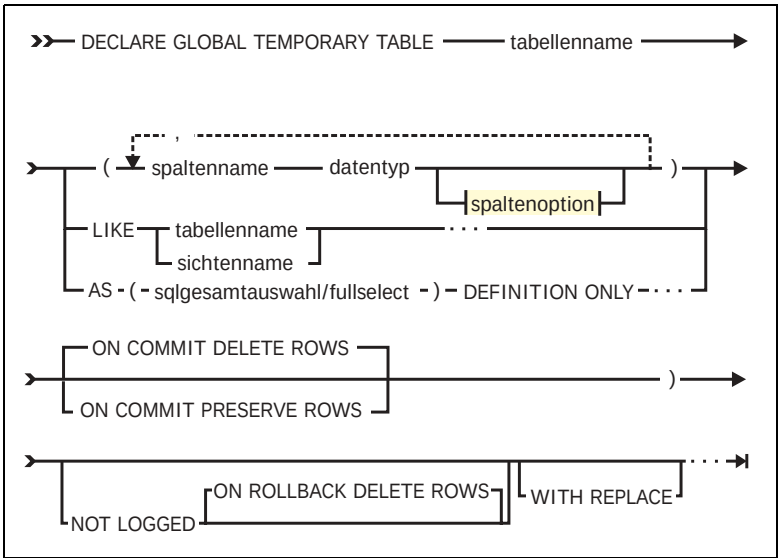


Abbildung 7.10: declare global temporary table-Syntaxdiagramm

Deklarierte temporäre Tabellen weisen folgende Besonderheiten auf:

- ▶ Ein „benutzertemporärer (USER TEMPORARY) Tabellenbereich“ (siehe Abschnitt 5.8.4) wird zwingend benötigt (inklusive des Rechts, diesen zu benutzen).
- ▶ Das Schema der temporären Tabelle ist immer SESSION.
- ▶ Die Tabelle ist nur für die aktuelle Verbindung (session) sichtbar und gültig. Das Wort „GLOBAL“ ist in diesem Zusammenhang missverständlich. Es bedeutet global für mehrere Transaktionen einer „Session“.
- ▶ Die Tabelle besteht nur für die Dauer der Verbindung (ON COMMIT PRESERVE ROWS) oder für eine Transaktion (ON COMMIT DELETE ROWS). Wird die Verbindung beendet, werden Daten und Tabelle gelöscht.
- ▶ Die Tabellendefinition ist im Datenbank-Katalog nicht eingetragen. Auch deshalb kann sie nicht von anderen Anwendungen (Verbindungen) mitbenutzt werden.
- ▶ Es besteht die Möglichkeit, die Veränderungen nicht zu protokollieren (NOT LOGGED), was die Performance erhöht, aber das Rückgängigmachen (rollback) verhindert. Hier sind zusätzlich Besonderheiten zu beachten!
- ▶ Die Datentypen für LOB- und LONG-Daten, benutzerdefinierte Datentypen (UDT) und DATALINK-Spalten werden nicht unterstützt (Fehlermeldung SQL0350).
- ▶ Es ist in DB2 UDB V8 möglich, einen Index auf eine temporäre Tabelle zu erstellen. Dieser muss ebenfalls im Schema SESSION angelegt werden. Auch dieser Index ist nicht in den Systemtabellen eingetragen.
- ▶ Die Statistiken können gesammelt werden. Dies war in DB2 V7 noch nicht möglich.

Insgesamt sind deklarierte temporäre Tabellen ein mächtiges, leistungsrelevantes Hilfsmittel, um große komplexe Abfragen zu unterstützen, deren Zwischenergebnisse nicht permanent gespeichert werden sollen. Die teilweise Umgehung der Protokollierung bringt Einschränkungen mit sich. Bei einem bewussten Einsatz sind deklarierte temporäre Tabellen aber positiv zu sehen, da dadurch Sperrungen vermieden werden und die Performance noch einmal gesteigert werden kann.

7.2.3 Gespeicherte Abfragetabelle (Materialized Query Table – MQT)

Vielleicht haben Sie schon einmal etwas von „Automatic Summary Tables (AST)“ gehört? Die deutsche Übersetzung ist so furchtbar, dass wir sie hier gar nicht erwähnen wollen. MQTs oder eben „gespeicherte Abfragetabellen“ sind eine Verallgemeinerung dieses Konzepts und sicherlich eine Funktionalität, von der man noch viel hören und lesen wird.

Die Grundidee ist, eine Tabelle zu definieren, die auf dem Ergebnis einer Abfrage von einer oder mehreren Tabellen (oder Sichten) basiert. So kann man beispielsweise eine komplexe Abfrage mit Verknüpfungen (Joins) als gespeicherte Abfrage-

tabelle – im Folgenden nur MQT genannt – abspeichern. Eine Anwendung, die Inhalte dieser komplexen Abfrage benötigt, kann jetzt direkt auf das Ergebnis zugreifen – nämlich die MQT – und muss nicht selbst den Aufwand der Verknüpfung (des Joins) betreiben. Anders als die temporären Tabellen im vorigen Abschnitt werden MQTs dauerhaft in der Datenbank gespeichert.

Der Trick ist also, vorberechnete Ergebnisse zu verwenden. Das ermöglicht, je nach Komplexität der Abfrage und nach Datenmenge, einen deutlichen Leistungsgewinn.

Doch damit nicht genug – denn so etwas kann man ja auch mit einem „normalen“ CREATE TABLE erreichen. Die Daten werden durch dieses Verfahren redundant gehalten – einmal in den Grundtabellen und dann noch mal (eventuell aggregiert) in der MQT. Normalerweise ist dies eher ein negativer Punkt für relationale Datenbanksysteme – treibt man doch erheblichen Aufwand, um mit der Normalisierung in der DB-Modellierung genau dies zu eliminieren. Mit dieser Technik kann man aber die Daten – anders als beim „normalen“ CREATE TABLE – automatisch, d.h. MAINTAINED BY SYSTEM (oder auf Benutzerwunsch, d.h. MAINTAINED BY USER) abgleichen lassen.

Der wohl interessanteste Punkt ist, dass Sie Ihre bestehenden Anwendungen nicht an diese neue Tabelle (MQT) anpassen müssen. Durch den *DB2 Optimizer*, der den Zugriffsweg auf die Daten bestimmt, wird (wie in Abbildung 7.11 gezeigt) die Anfrage automatisch auf die MQT umgeleitet. Das ist durch die so genannte „REWRITE-Funktionalität“ des Optimizers möglich. Damit kann er SQL-Befehle selbstständig umbauen und gewährleistet doch die gleiche Ergebnismenge. Sie profitieren also umgehend von Ihrer MQT.

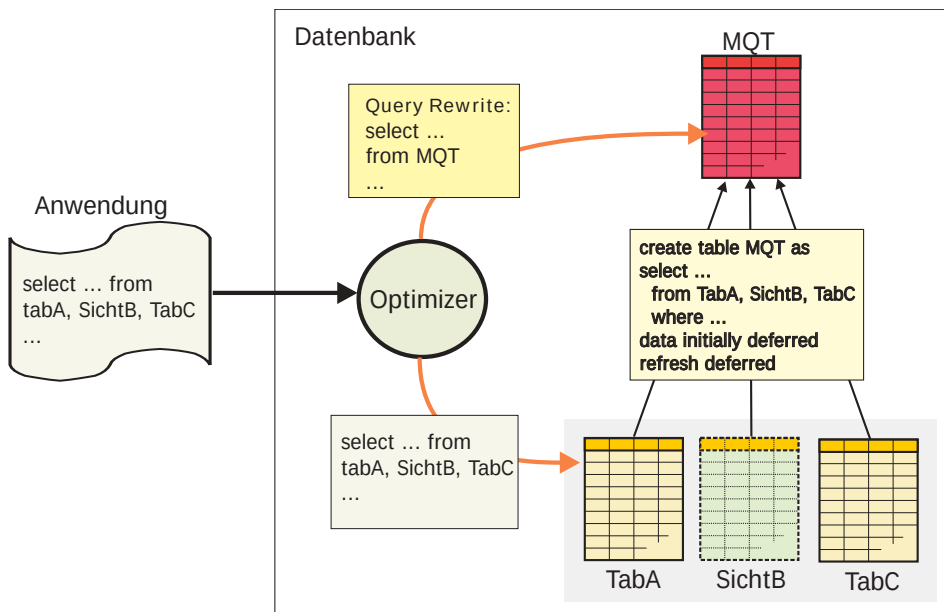


Abbildung 7.11: Gespeicherte Abfragetabelle (MQT) – Prinzip

Diese sehr mächtige Funktionalität benötigt bestimmte Optimizer-Einstellungen und wird nur bei bestimmten Optimierungsstufen (2, 5, 7 und 9) in Betracht gezogen (Details siehe Kapitel 15).

Eine MQT wird über einen CREATE TABLE-Befehl angelegt, der zusätzlich die MQT-spezifischen Definitionen beinhaltet, wie z.B. wann die Daten der MQT abgeglichen werden sollen und ob der Optimizer diese MQT berücksichtigen soll (ENABLE QUERY OPTIMIZATION) oder nicht.

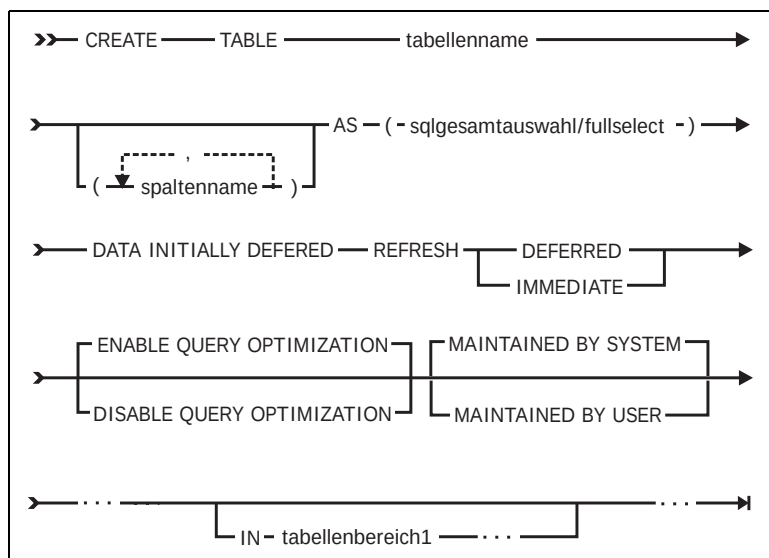


Abbildung 7.12: create table-Syntaxdiagramm – gespeicherte Abfragetabelle (MQT)

Die REFRESH IMMEDIATE-Option stellt sicher, dass die Daten der MQT immer den aktuellen Stand haben. Diese Option ist nur in Kombination mit MAINTAINED BY SYSTEM möglich.

Hinweis:

Nur in diesem Fall – also mit REFRESH IMMEDIATE und MAINTAINED BY SYSTEM – kann eine Optimierung für statisches SQL erfolgen. In allen anderen Fällen profitieren nur Anfragen mit dynamischem SQL. Ist DISABLE QUERY OPTIMIZATION gewünscht, wird der Optimizer die MQT nicht berücksichtigen, und es sind nur explizite Zugriffe auf die MQT möglich.

Wird in der SQL-Gesamtauswahl (fullselect) eine Summierung durchgeführt (Aggregationsfunktion und GROUP BY), spricht man auch von einer so genannten *Automatic Summary Table* (AST), die auch vor DB2 V8 schon verfügbar war.

Weitere Informationen und Details zu diesem Thema finden Sie in [SG246546], Kapitel 2 „DB2 UDB's materialized views“.

7.2.4 Anzeige der Tabellen

Die Tabellenübersicht über die Steuerzentrale haben Sie bestimmt selbst schon einmal angewählt oder in Bildern weiter vorn im Buch gesehen, deshalb bleibt diese Variante hier außen vor.

Der DB2-Befehl `LIST TABLES` zeigt Ihnen standardmäßig alle „Ihre“ Datenbank-Objekte an, also alle, die Ihren aktuellen Benutzernamen (mit dem Sie sich zur Datenbank verbunden haben) als Schema-Namen haben.

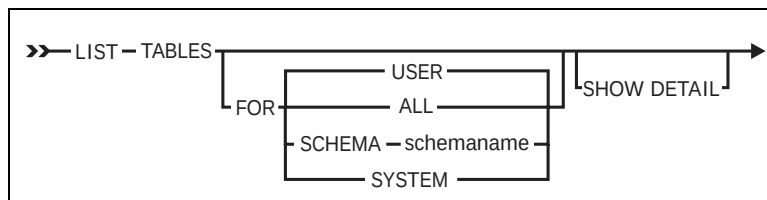


Abbildung 7.13: *list tables-Syntaxdiagramm*

Über die Option `FOR ALL` können Sie sich alle Tabellen (und andere Objekte) in der Datenbank anzeigen lassen. Die `SYSTEM`-Option listet die Systemkatalog-Tabellen und -Sichten auf, also alle mit dem Schema `SYSIBM`, `SYSCAT` oder `SYSSTAT`. Einzelne Schemata können ebenfalls ausgewählt werden.

Mit der `SHOW DETAIL`-Option können Sie sich auch Tabellennamen voll anzeigen lassen, die länger als 30 Zeichen sind und sonst der Übersichtlichkeit halber abgeschnitten werden.

Dieser Befehl stellt nur eine „vereinfachte“ Alternative zu der SQL-Abfrage der Systemkatalog-Tabellen dar, die auch noch zusätzliche Informationen zu den Tabellen bereithalten. Der dem DB2-Befehl `LIST TABLES` entsprechende SQL-Befehl lautet:

```

select tabname, tabschema, type, create_time
from syscat.tables
where tabschema = USER order by tabschema, tabname
  
```

Ein weiterer Befehl ermöglicht es Ihnen, sich einen schnellen Überblick über die Spaltendefinitionen z.B. einer Tabelle zu verschaffen. Auch dieser `DESCRIBE TABLE`-Befehl kann über eine Selektion der Systemkatalog-Sichten nachgebildet werden.

Die Syntax ermöglicht es Ihnen, nicht nur die Datenstruktur einer Tabelle auszulesen, sondern auch die Datenstruktur eines Ergebnisses eines `SELECT`-Befehls.

Sogar Indexstrukturen können Sie damit herausbekommen – doch dazu später.

Für einen SQL-`SELECT`-Befehl brauchen Sie beispielsweise die Spaltennamen der `ARTIKEL`-Tabelle, und auch die Datentypen sind interessant, um SQL-Spaltenfunktionen einsetzen zu können. Ihr Ergebnis könnte wie Abbildung 7.16 ausschauen.

```
D:\SQLLIB\BIN>db2 list tables
```

Tabelle/Sicht	Schema	Typ	Erstellungszeit
ADRESSE	MTUSER	T	2003-09-03-10.18.31.602000
ANSCHRIFT	MTUSER	A	2003-09-03-17.40.00.955000
ARTIKEL	MTUSER	T	2003-09-03-10.18.39.824001
AUFTRAG	MTUSER	T	2003-09-03-10.18.40.736000
AUFTRAGSPOSITIONEN	MTUSER	T	2003-09-03-10.18.41.296001
KUNDE	MTUSER	T	2003-09-03-10.18.35.979001
KUNDENADRESSE	MTUSER	T	2003-09-03-10.18.37.241001
V_ARTIKEL	MTUSER	V	2003-09-03-12.12.19.504001
V_ARTIKELPREIS	MTUSER	V	2003-09-03-12.07.35.586001

9 Satz/Sätze ausgewählt.

Abbildung 7.14: Beispiel list tables

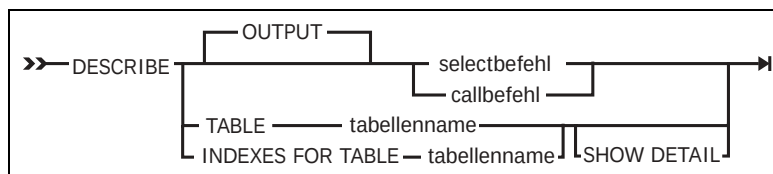


Abbildung 7.15: describe table-Syntaxdiagramm

Die entsprechende SQL-Abfrage – auf die Systemkatalog-Sicht SYSCAT.COLUMNS – würde so aussehen:

```
select colname, typeschema, typename, length, scale, nulls from syscat.columns
where tabname ='ARTIKEL'
```

Allerdings sind die Formatierungen der Ausgabe bei der SQL-Variante nicht so schön aufbereitet – oder anders gesagt: Die Aufbereitung würde wesentlich mehr SQL-Text erfordern.

```
D:\SQLLIB\BIN>db2 describe table artikel
```

Spalten- name	Schema	Name	Länge	# Komma- stellen	Null- zeichen
ARTIKELNR	SYSIBM	INTEGER	4	0	Nein
ARTIKELBEZ	SYSIBM	CHARACTER	30	0	Ja
ARTIKELBEST	SYSIBM	INTEGER	4	0	Nein
ARTIKELPREIS	MTUSER	PREIS	0	0	Ja

4 Satz/Sätze ausgewählt.

Abbildung 7.16: Beispiel describe table

7.3 Sichten (Views)

Sichten – die auch als logische Tabellen bezeichnet werden – sind individuelle Zusammenstellungen von Spalten und Zeilen von einer oder mehreren bestehenden Tabellen und/oder wiederum Sichten.

Damit ist es möglich, eine neue Abstraktionsebene zwischen der Abfrage und den Daten (Tabellen) zu erstellen, um:

- bestimmte sensible Inhalte einer Tabelle zu filtern

Die Mitarbeiter-Tabelle z.B. ist für die gesamte Personalabteilung interessant; die Gehaltsspalte der Mitarbeiter-Tabelle soll aber nur ausgewählten Personen zugänglich sein. Hier kann eine Sicht genau diese Spalte ausgrenzen, und die Benutzer greifen auf die erstellte Mitarbeiter-Sicht zu statt auf die Tabelle. Auch die angezeigten Zeilen können gefiltert werden, um beispielsweise einem Vertriebsmitarbeiter nur „seine“ Kunden und Umsätze zu zeigen und nicht die der Kollegen.

- Änderungen in den Tabellen vor den Anwendungen zu verschleiern

Strukturanpassungen in den Tabellen würden aufwändige Programmänderungen für die Anwendungen bedeuten. Mit einer Sicht kann man eventuell der Anwendung die „alte“ Struktur vorgaukeln – eine Anpassung der Anwendung ist dann erst einmal nicht mehr notwendig.

- den Anwendern einfachere Strukturen zur Verfügung zu stellen

Aufwändige Abfragen und Auswertungen erfordern häufig komplexe SQL-Befehle, um die Daten aus mehreren Tabellen und Sichten zusammenzustellen und aufzubereiten. Unerfahrene Benutzer haben damit oft Schwierigkeiten oder machen Fehler, die sehr viel Systemlast erzeugen können und andere Anwendungen „ausbremsen“. Ein erfahrener SQL-Programmierer kann diese komplexen Anforderungen sauber und in performantem SQL umsetzen und seine Ergebnisse mittels einer Sicht den anderen Benutzern zur Verfügung stellen.

Im Unterschied zu MQTs und temporären Tabellen ist bei Sichten nur die Definition, also der SQL-Befehl, gespeichert. Dieses SQL wird erst beim Zugriff auf die Sicht ausgeführt. Eine redundante Datenhaltung ist somit nicht gegeben. Im Vergleich zu einer bestehenden temporären Tabelle oder einer MQT wird der Zugriff länger brauchen, da neben dem SQL-Zugriff noch das SQL der Sicht ausgeführt werden muss. Die Definitionen der Sichten in einer Datenbank werden in den Systemkatalog-Tabellen abgelegt (z.B. in SYSCAT.VIEWS – die selbst eine Sicht auf die SYS-IBM.SYSVIEWS-Tabelle ist). Dies ist in Abbildung 7.17 bei der physikalischen Speicherung angedeutet. Für den Benutzer stellen sich Sichten wie Tabellen dar.

Der Befehl zur Erstellung einer Datenbank-Sicht benötigt neben dem Sichtenamen (eventuell mit Angabe eines Schema-Namens) eine SQL-SELECT-Abfrage, die den Sichteninhalt definiert (siehe Abbildung 7.18).

Die Tabellen, auf die sich die SQL-Gesamtauswahl (fullselect) bezieht, nennt man *Basistabellen*.

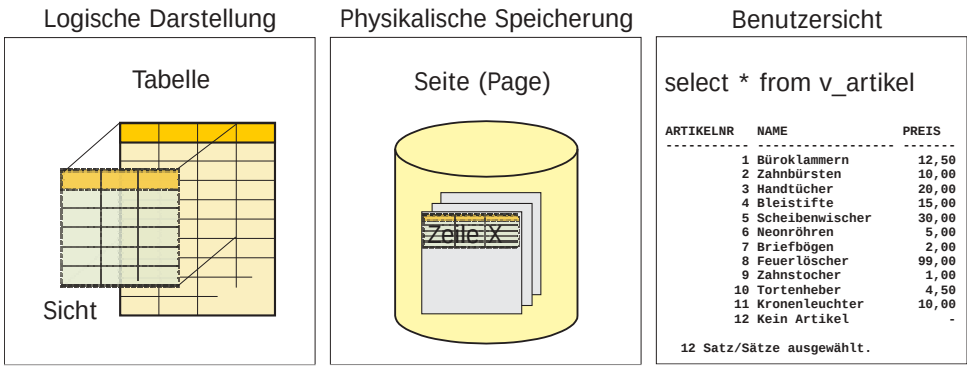


Abbildung 7.17: Sicht

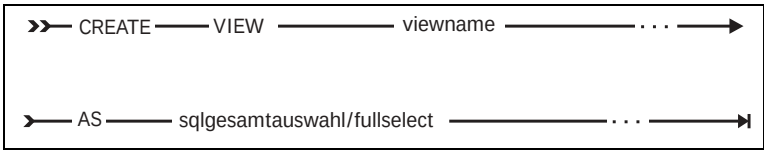


Abbildung 7.18: create view-Syntaxdiagramm

Eine Sicht kann nicht im Nachhinein angepasst oder geändert werden, ein „alter view“-Befehl existiert nicht.

Diese Abhängigkeit von anderen Objekten kann eine Sicht *unbrauchbar* machen, wenn z.B. eine Basistabelle gelöscht wird oder die Zugriffsrechte eingeschränkt werden. Ein Zugriffsversuch auf eine als unbrauchbar markierte Sicht liefert die SQL-Fehlermeldung SQL0575. Die Sicht muss gelöscht (DROP VIEW) und neu erstellt werden. Dies gilt auch, wenn eine neue Grundtabelle mit gleichem Namen erzeugt wird, da die *Unbrauchbar*-Markierung der Sicht trotzdem bestehen bleibt.

Sichten können nach folgenden Gesichtspunkten klassifiziert werden. Diese Klassifikation bezieht sich auf die möglichen Operationen, die auf die Sichten-Inhalte ausgeführt werden können und nicht auf ihre Definition:

- löschbare Sicht
- änderbare Sicht
- erweiterbare Sicht
- Sicht mit Lesezugriff

Je nach Sichten-Definition wird sie der einen oder anderen Kategorie zugeordnet. Wenn die Sicht beispielsweise auf Summierungen etc. aufbaut, können die Inhalte nicht verändert werden, da diese Veränderungen nicht an die Basistabellen weitergereicht werden können.

Der im letzten Abschnitt erwähnte LIST TABLES-Befehl kann auch für die Anzeige der definierten Sichten verwendet werden sowie für die Aliasnamen, die wir nun besprechen wollen.

7.4 Aliasnamen (Alias)

Ein Aliasname oder Alias ist ein „Spitzname“ für eine Tabelle oder eine Sicht. Wurde ein Aliasname angelegt, kann z.B. eine Tabelle sowohl über ihren eigentlichen Namen als auch über ihren Alias angesprochen werden.

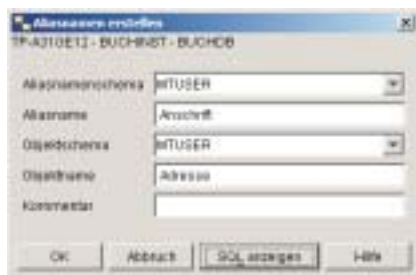


Abbildung 7.19: Alias mit der Steuerzentrale erstellen

Abbildung 7.19 zeigt die Erstellung eines Alias für eine Tabelle. Die Adresse der Kunden kann nun sowohl über

```
select * from mtuser.adresse
```

als auch über

```
select * from mtuser.anschrift
```

herausgefunden werden. Die Ergebnisse sind identisch.

Diese Aliasnamen können beispielsweise dazu eingesetzt werden, Tabellennamen (bzw. Sichtennamen) „einzudeutschen“. Werden durch erworbene Produkte englische Tabellennamen erzeugt, können diese deutsche Aliasnamen bekommen (oder umgekehrt). Eine REVENUE-Tabelle wird so schnell und einfach zur UMSATZ-Tabelle. Die Daten werden aber nur einmal gespeichert, wodurch Redundanz vermieden wird.

Der Befehl, um Aliasnamen anzulegen, lautet:

```
create alias <aliasname> for <tabelle oder sicht>
```

7.5 Indizes

Indizes kennen Sie auch aus Büchern. Sie dienen dazu, schnell den gewünschten Inhalt in einem Buch zu finden – man schaut den alphabetisch sortierten Index am Ende des Buchs durch und beginnt bei dem Anfangsbuchstaben die verfügbaren Einträge zu lesen. Ist der gesuchte Begriff dabei, kann man durch den Seitenverweis schnell die Seite mit der zugehörigen Beschreibung aufschlagen.

Genauso funktioniert das auch bei Datenbanken, wenn Sie bestimmte Daten aus Ihren Tabellen suchen bzw. abfragen wollen. Der einzige Unterschied ist, dass in der Datenbank alle Einträge im Index zu finden sind.

Der Vorteil eines *geeigneten* Index ist ein besserer bzw. schneller Zugriff auf die Daten. *Geeignet* bedeutet, dass die *richtigen* Inhalte indiziert wurden. Stellen Sie sich vor, Sie haben Ihr örtliches Telefonbuch und wollen herausfinden, wer sich hinter einer bestimmten Telefonnummer verbirgt. Die Informationen sind im Telefonbuch eingetragen. Da aber das Telefonbuch nach Namen sortiert ist, werden Sie wohl nicht umhin können – falls Sie es sich jetzt nicht anders überlegen –, das Buch Zeile für Zeile zu studieren. In der Datenbank-Welt nennt man dies eine *Tabellensuche* (tablescan). Der vorhandene alphabetische Index auf die Nachnamen ist für diese Suche *ungeeignet*.

Die *Vorteile* eines Index resultieren aus einer effizienten Datenstruktur, die ein schnelles Auffinden der Daten erlaubt und auch vom Umfang her wesentlich kleiner ist als die Tabelle selbst. Dadurch sind weniger E/A-Operationen beim Zugriff nötig. Ein Index ist auch immer sortiert und kann so in vielen Fällen aufwändige Datensortierungen vermeiden helfen – wie sie z.B. durch ORDER BY-Klauseln in SQL verursacht werden.

Allerdings muss dieser Vorteil bei der Zugriffsgeschwindigkeit auch „bezahlt“ werden. Die Speicherung der Indexstruktur kostet Speicherplatz. Kommen neue Daten hinzu, entsteht ein Pflegeaufwand für den Index, um diese neuen Daten in der Indexstruktur einzutragen.

Wie im Buch ist der Index in der Datenbank eine Struktur (ein so genannter *B+-Baum*), die es ermöglicht, in wenigen Schritten zum gesuchten Eintrag zu kommen. Dieser hat dann einen Verweis, die Satzkennung oder „record identifier“ (RID), die auf die gewünschte Zeile der Tabelle bzw. deren Speicherort auf einer Datenseite zeigt. Man spricht in diesem Zusammenhang auch von einem so genannten „*Row-Index*“.

Abbildung 7.20 zeigt die logische B*- bzw. B+-Baum-Struktur, die als Liste implementiert ist und auf Datenbankseiten in den Tabellenbereich-Behältern abgespeichert ist. Der Benutzer sieht normalerweise den Inhalt eines Indexobjekts nicht (Ausnahme: Index-Only-Zugriff). Er merkt die Auswirkungen eines Index nur an einem schnelleren Zugriff auf die Daten.

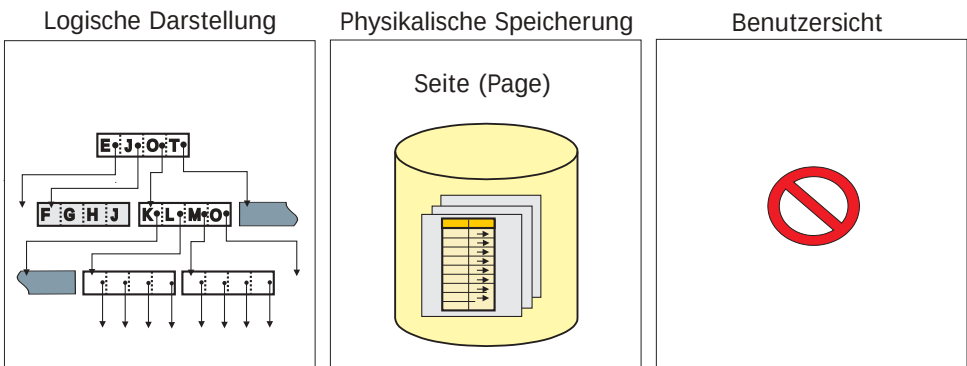


Abbildung 7.20: Index

Eine Auflistung der definierten Indizes ist natürlich auch möglich – entweder über den INDIZES-Eintrag in der Objektbaumstruktur der Steuerzentrale oder über eine SQL-Abfrage der Systemkatalog-Sicht SYSCAT.INDEXES.

Das Syntaxdiagramm in Abbildung 7.21 zeigt die wichtigsten Optionen, von denen einige (UNIQUE und CLUSTER) in einem eigenen Teilkapitel behandelt werden.

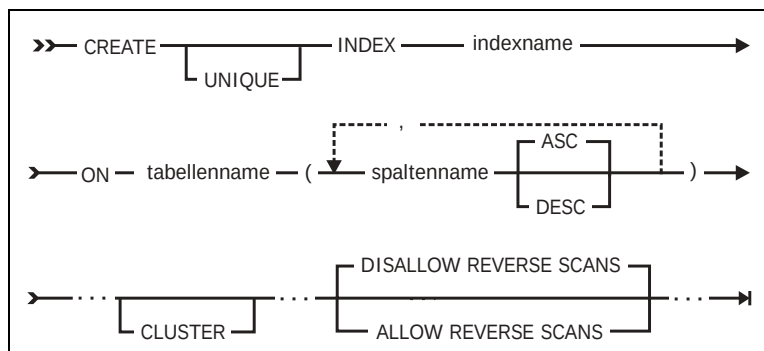


Abbildung 7.21: create index-Syntaxdiagramm

Der Indexname kann vollqualifiziert, also mit Schema-Namen, angegeben werden, ebenso wie der benötigte Tabellenname. Ein Indexname (ohne Schema-Name) darf maximal 18 Zeichen lang sein.

Nachträglich kann ein Index – wie eine Tabelle – über den RENAME-Befehl umbenannt werden:

```
rename index <alter Indexname> to <neuer Indexname>
```

Bei der Erstellung eines Index ist es vor allem wichtig, sich darüber Gedanken zu machen, was in einer Tabelle indiziert werden soll – also welche Spalte oder Spaltenkombination, um Situationen wie in unserem „Telefonbuch-Beispiel“ von oben zu vermeiden.

ASC steht für aufsteigend (ascending), und DESC (descending) bezeichnet die absteigende Reihenfolge – hier wird also die Sortierung des Index angegeben. In bestimmten Fällen, z.B. bei SQL-Abfragen mit der MIN()- und MAX()-Funktion, ist es interessant, beide Sortierreihenfolgen zur Verfügung stellen zu können. Das wäre durch zwei Indizes möglich, indem man jeden mit einer anderen Sortierung anlegt, ist aber besser durch einen so genannten bidirektionalen Index zu lösen, der mit der Option ALLOW REVERSE SCANS angelegt werden kann. Damit werden dann die Indexeinträge „beidseitig“ verknüpft, und es ist nur ein Index notwendig.

Wollen Sie einen Index über die Steuerzentrale anlegen, gehen Sie so vor:

Sie wählen den INDIZES-Eintrag aus der Objektbaumstruktur unterhalb der Datenbank-Ebene aus und klicken auf ERSTELLEN...

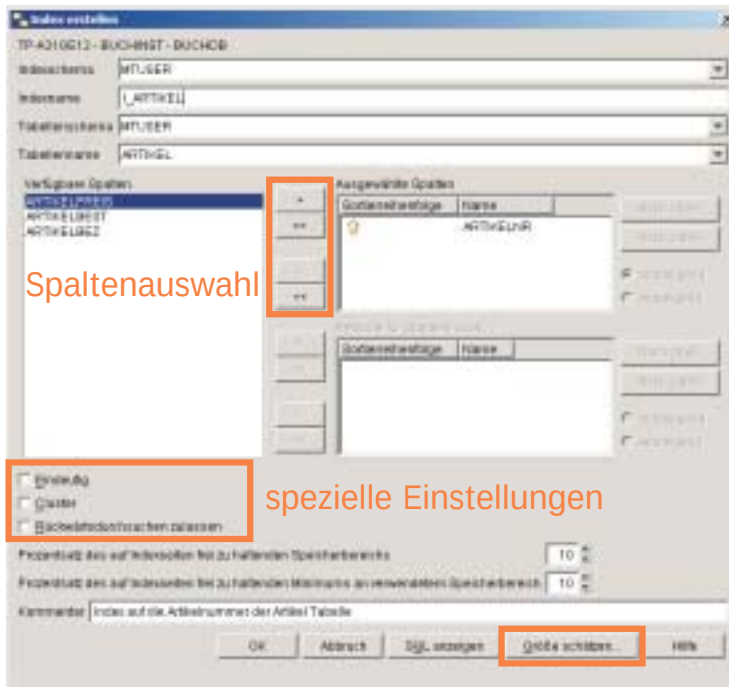


Abbildung 7.22: Index erstellen mit der Steuerzentrale

Angewählte Spalten können Sie über die Spaltenauswahl-Buttons selektieren. Die speziellen Einstellungen nehmen Sie im unteren Teil des Fensters vor. RÜCKWÄRTS-DURCHSUCHEN ZULASSEN entspricht dabei der ALLOW REVERSE SCANS-Option aus der CREATE INDEX-Syntax. Die Auswahlen EINDEUTIG und CLUSTER besprechen wir in den nächsten Teilkapiteln.

Die Einstellung PROZENTSATZ DES AUF INDEXSEITEN FREI ZU HALTENDEN SPEICHERBEREICHES entspricht der PCTFREE-Option. Diese gibt den Prozentsatz des beim Anlegen freizuhaltenen Speicherplatzes auf den Speicherseiten an. Eine Verbesserung der internen Speicherverwaltung von Indexseiten wird somit möglich, benötigt aber in etwa den gleichen Prozentsatz an zusätzlichem Speicherplatz.

Der PROZENTSATZ DES AUF INDEXSEITEN FREI ZU HALTENDEN MINIMUMS AN VERWENDETEM SPEICHERBEREICH ist eine unbrauchbare – sogar falsche – Übersetzung der eigentlichen Funktion. Hinter diesem Wert steht der Parameter MINPCTUSED, der dafür sorgen soll, dass überprüft wird, ob Index-Speicherseiten zusammengefasst werden können, um einen schnelleren Zugriff zu gewährleisten. Eine Einstellung dieses Prozentwerts auf einen Wert kleiner als 50 (%) wird empfohlen.

Tipp:

Ein besonderes Highlight – und nur über die grafische Oberfläche einzusetzen – ist die GRÖSSE SCHÄTZEN-Funktion. Damit können Sie sich den zu erwartenden Speicherverbrauch der bestehenden und der zu erstellenden Indexstrukturen sowie der Basistabelle anzeigen bzw. berechnen lassen (siehe Abbildung 7.23).

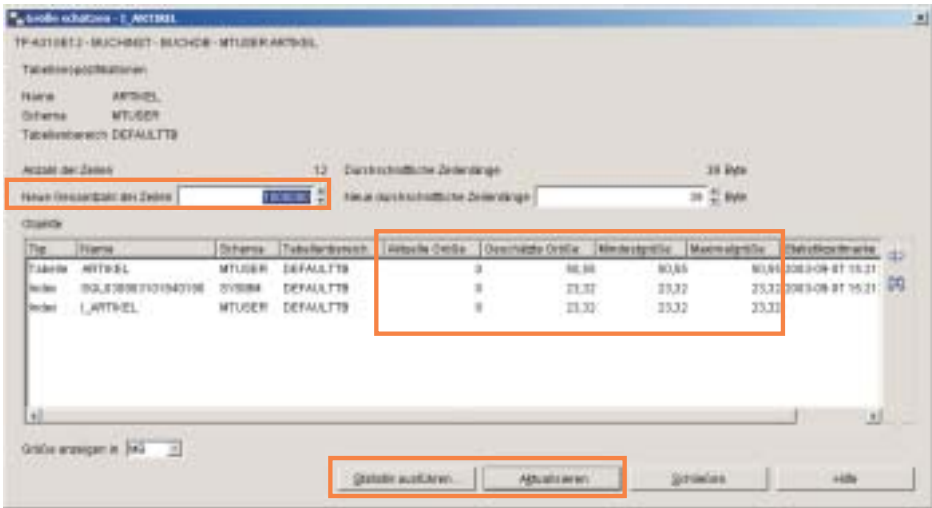


Abbildung 7.23: Größe schätzen-Fenster bei der Index-Erstellung

Sie haben die Möglichkeit, unter NEUE GESAMTZAHL DER ZEILEN die Zielgröße Ihrer Tabelle (in Zeilen) anzugeben. Wird der AKTUALISIEREN-Button angeklickt, berechnet DB2 die Größe Ihrer Objekte anhand der aktuellen bzw. angegebenen Zeilenlänge und der Zeilenanzahl. Für ein möglichst exaktes Ergebnis sollten Sie aktuelle Statistik-Informationen bereitstellen oder diese über den STATISTIK AUSFÜHREN...-Button nun einholen. Die Statistik-Informationen werden in den Systemtabellen abgelegt. Den Befehl zur Aktualisierung dieser Informationen, der hier im Hintergrund ausgeführt wird, behandeln wir in Kapitel 13.

In der Praxis ist es die Regel, dass auf wichtige Tabellen mehrere Indizes – auf unterschiedliche Spalten bzw. Spaltenkombinationen – angelegt werden, um die unterschiedlichen Zugriffe zu unterstützen. Die Bestimmung, welche und wie viele Indizes es sein sollten, hängt vor allem mit der Art der Anwendung und deren Abfrage zusammen. Hier sind Erfahrung und eine gute Kenntnis der Zugriffe wichtig. Zu viele Indizes können ähnlich schlecht sein wie zu wenige.

Weitere Optimierungsaspekte und -beispiele finden Sie in [AKUZNETSOV].

7.5.1 Eindeutiger Index (unique index)

Ein `CREATE UNIQUE INDEX`-Befehl erzeugt einen eindeutigen Index, d.h. einen Index, der die Eindeutigkeit der im `CREATE`-Befehl angegebenen Spalten gewährleistet. Nur über diesen Index haben Sie die Möglichkeit sicherzustellen, dass Datensätze nicht doppelt in einer Tabelle gespeichert werden.

Sie haben bestimmt auch schon oft Post von Ihrer Versicherung oder anderen Stellen bekommen, bei der Sie aufgefordert wurden, bei Rückfragen immer die Vorgangsnummer oder Kundennummer anzugeben. Diese Kundennummer dient dazu, Sie bzw. Ihren Vorgang zu identifizieren und zuzuordnen. Für solche Nummern muss eine Eindeutigkeit sichergestellt werden. Da es z.B. sicherlich mehrere „Christian Huber“ geben wird, die auch Kunde der gleichen Firma sein können, muss jeder Vorgang dem „richtigen“ Herrn Huber zugeordnet werden können.

Oben haben wir ja schon besprochen, dass jeder Datensatz in einer Tabelle eindeutig zu identifizieren sein muss, und zwar über den so genannten (eindeutigen) Schlüssel der Tabelle. Diese Schlüssel sind gute Kandidaten für einen `UNIQUE INDEX`.

Ein Beispiel für die Erstellung eines solchen eindeutigen Index könnte so aussehen:

```
create unique index i_artikel on artikel (artikelnr)
```

Als Sortierreihenfolge wird standardmäßig „aufsteigend“ verwendet.

7.5.2 Clustered Index

Der „Clustered Index“ wird in der Regel mit „in Gruppen zusammengefasster Index“ übersetzt. Wir denken: Das kann man machen – muss es aber nicht! Außerdem beschreibt diese „Übersetzung“ nicht annähernd das eigentliche Konzept dieses Index.

Es geht beim „Clustering“ um effizienten Datenzugriff (E/A)!

Die Idee ist es nicht nur, den Index sortiert vorliegen zu haben, sondern auch die Daten, um somit zusammengehörige bzw. aufeinander folgende Daten mit minimalem E/A-Aufwand lesen bzw. verarbeiten zu können. Während ein Index immer (logisch) sortiert ist, ist das bei den Daten nicht der Fall.

Die Daten werden ständig verändert, einige Zeilen werden gelöscht, und neue Inhalte (Zeilen) kommen hinzu. Das führt dazu, dass die Zeilen – auch wenn sie irgendwann einmal sortiert vorgelegen haben sollten – fragmentiert werden, also „durcheinander geraten“. Dieses „Durcheinander“ bezieht sich immer auf die Inhalte einer Spalte. Denken Sie an das Telefonbuch-Beispiel bei der Einführung zum Index zurück: Die Daten mögen im Telefonbuch zwar alphabetisch nach Namen geordnet sein, das heißt aber nicht, dass sie nach Telefonnummern geordnet sind.

Die Daten lassen sich nun in der Datenbank sortieren. Um dies jetzt nicht mit der sortierten Ausgabe durch einen `SQL-ORDER BY`-Befehl oder Ähnlichem zu verwechseln, müssen wir den Satz präzisieren. Es geht darum, Inhalte (Zeilen) der Datenseiten (pages) nach einem bestimmten Kriterium zu sortieren und ebenso physikalisch abzuspeichern. Das `REORG`-Tool der DB2-Datenbank übernimmt genau diese Aufgabe. Den Aufruf dieses Tools und weitere Details dazu finden Sie in Kapitel 13.

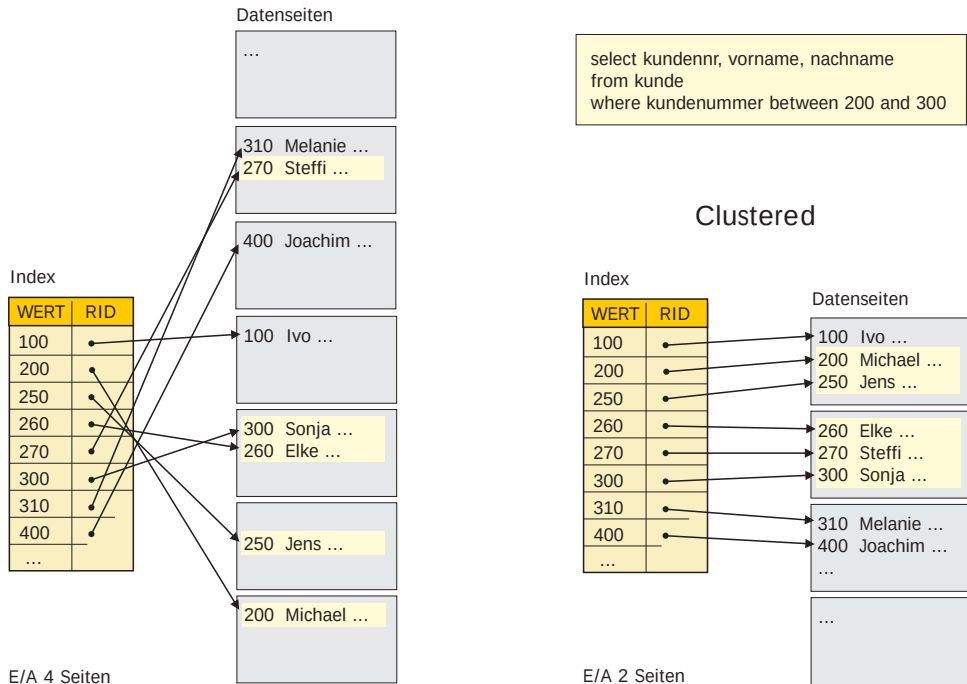


Abbildung 7.24: Clustered Index bzw. Daten in Index-Reihenfolge

Die linke Seite der Abbildung 7.24 zeigt einen Index über die KUNDENNR-Spalte einer Tabelle. Die Satzkenung (RID – Row Identifier) verweist auf den zugehörigen Dateneintrag in der Tabelle – genauer auf eine Speicherseite und die so genannte Steckplatznummer (offset), bei der die Daten zu finden sind.

Durch die SQL-Abfrage werden alle Zeilen zwischen 200 und 300 selektiert. Das entspricht im Beispiel fünf Zeilen. Diese Ergebniszeilen befinden sich auf vier verschiedenen Datenseiten, die nun jeweils mit einer E/A-Anforderung zu Buche schlagen.

Auf der rechten Seite sind die Daten wie der KUNDENNR-Index sortiert. Die fünf qualifizierten Zeilen befinden sich dadurch auf nur zwei Datenseiten und können mit zwei E/A-Anforderungen gelesen werden. Das ist eine E/A-Einsparung von 50%! Bei einer anderen Verteilung können hier durchaus noch größere Raten und somit Einsparungen möglich sein.

Die CLUSTER-Option beim CREATE INDEX-Befehl gibt an, dass Sie die Daten nach diesem Index sortiert haben wollen. Wird dieser Index im Nachhinein erstellt – also nachdem die Tabellen mit Daten gefüllt worden sind –, müssen die Daten zusätzlich durch eine Reorganisation (siehe Kapitel 13) in die entsprechende Reihenfolge gebracht werden.

Diese Funktionalität hat folgende Einschränkungen:

- Es kann nur einen CLUSTER-Index auf eine Tabelle geben, da die Daten in dieser Struktur nur in einer Reihenfolge eines Schlüssels (Spalte oder Spaltenkombination) gespeichert werden können.

- Dieses Clustering kann nicht „über die Zeit“, also dauerhaft, aufrechterhalten oder garantiert werden. Vor allem durch Löschungen und das Hinzufügen neuer Datensätze wird diese Sortierung zerstört.

Für den zweiten Punkt – also die Aufrechterhaltung der Sortierung – gibt es noch eine Möglichkeit, positiv Einfluss zu nehmen, wenn diese auch das Problem nicht verhindern kann.

Wie in Abbildung 7.25 gezeigt, kann in einer Tabelle die PCTFREE-Option gesetzt werden, die auf den einzelnen Datenseiten freien Platz lässt. Diese Option kann *nur* mit dem ALTER TABLE-Befehl gesetzt werden und benötigt eine Prozent-Angabe (PCTFREE, Percent Free Space). In unserem Beispiel sollen 30% einer Seite freigehalten werden, um „Platz an der richtigen Stelle“ (in der Sortierung) für neue Datensätze zu haben.

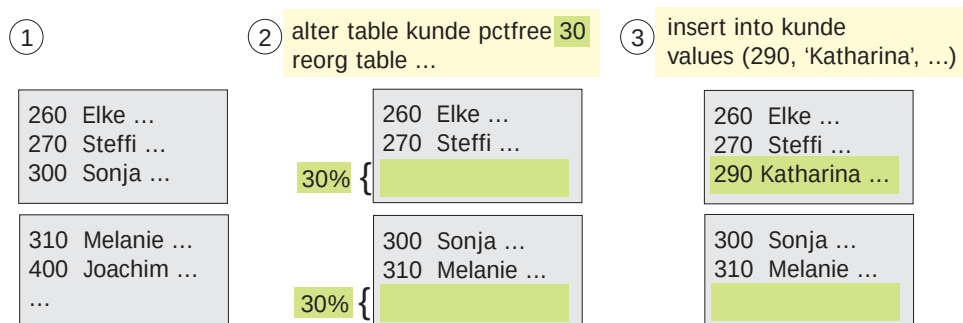


Abbildung 7.25: pctfree-Option

Die „normale“ Datenseite bei Punkt ① enthält drei Datenzeilen. Durch das Setzen der PCTFREE-Option (in Punkt ② wird die Voraussetzung für „Freiplatz“ geschaffen. Dieser „Freiplatz“ wird aber erst durch eine Reorganisation der Daten mit dem REORG-Befehl geschaffen. Insgesamt werden ca. 30% mehr Platz benötigt. In Punkt ③ kann nun der neue Datensatz mit der (Kunden-)Nummer 290 an die der Sortierung nach korrekte Stelle gespeichert werden, was ohne diese Veränderungen in Punkt ① nicht möglich gewesen wäre. Dort hätten die Daten eventuell erst „am Ende der Tabelle“ Platz gehabt, und die Sortierung bzw. das Clustering wäre verschlechtert worden.

Selbst wenn in Punkt ③ ein Datensatz mit der Nummer 265 eingefügt worden wäre, wäre er an das „Ende“ der Seite geschrieben worden. Das hätte zwar eine punktuell ungültige Sortierung ergeben, aber ein effizienter E/A wäre trotzdem möglich, da sich die zusammengehörigen Datensätze auf einer Seite befinden würden.

Dieses Clustering spielt auch eine entscheidende Rolle bei der Zugriffsweg-Bestimmung durch den DB2-Optimizer. Dieser liest die Qualität der Sortierung aus dem Systemkatalog aus, um abschätzen zu können, mit welcher Effizienz beim E/A-Zugriff gerechnet werden kann.

Hinweis:

Die Spalten `CLUSTERRATIO` oder `CLUSTERFACTOR` der Systemkatalog-Sicht `SYSCAT.INDEXES` beinhalten diese Informationen für den Optimizer, falls die Statistikinformationen gesammelt wurden (dazu mehr in Kapitel 13.1.1, „Runstats“). Je höher der Prozentwert von `CLUSTERRATIO` ist, desto besser ist die Qualität des Clusterings. Der Wert `CLUSTERFACTOR` wird alternativ zu `CLUSTERRATIO` von DB2 für große Datenmengen verwendet und auch nur, wenn sehr genaue Statistiken gefordert werden. Dann zeigt er einen Wert zwischen null und eins an. Dabei gilt, dass die Sortierung der Daten zu diesem Index umso besser ist, je größer der Wert ist. Der Unterschied zwischen diesen beiden Werten ist der Detaillierungsgrad. Durch `CLUSTERFACTOR` kann die Cluster-Eigenschaft in feineren (mehr) Stufen spezifiziert werden und ist damit genauer.

Wie erwähnt, sind in dieser Struktur die Daten ja nur zu einem Index sortiert und somit für alle anderen Indizes auf der Tabelle unsortiert. Die in den Systemtabellen abgespeicherten Indexinformationen beinhalten immer die zu diesem Index gehörenden Clusterwerte (`CLUSTERRATIO` oder `CLUSTERFACTOR`) im Bezug auf die zugehörige Tabelle. So hätte beispielsweise der Index auf die Namensspalte im Telefonbuch eine perfekte `CLUSTERRATIO` (100%), während ein Telefonnummer-Index sehr schlechte Cluster-Eigenschaften aufweisen würde.

7.5.3 Informationen zu den Indizes anzeigen lassen

In der Steuerzentrale verschaffen Sie sich mit dem Indizes-Eintrag im Objektbaum einen Überblick über die definierten Indizes.

Mehr Informationen über Indizes erhält man vor allem durch eine SQL-Abfrage der Systemkatalog-Sicht `SYSCAT.INDEXES`. Hier folgt eine (Teil-)Auflistung wichtiger Spalten und Bedeutungen dieser Sicht:

- ▶ `INDSCHEMA` und `INDNAME`
beinhalten den vollständigen Indexnamen
- ▶ `DEFINER`
Benutzer-ID des Index-Erzeugers
- ▶ `TABSCHEMA` und `TABNAME`
beinhalten den vollständigen Tabellennamen, auf den sich der Index bezieht
- ▶ `COLNAMES`
Liste der Index-Spalten bzw. der Tabellenspalten, auf die sich der Index bezieht
- ▶ `UNIQUERUL`
zeigt an, ob ein eindeutiger Index vorliegt; dabei steht D für „Duplikate erlaubt“, P für „Primärer Index“ und U für „Unique Index“

[illegible]

Abbildung 7.26: Indexansicht über die Steuerzentrale

► INDEXTYPE

REG steht in dieser Spalte für einen regulären Index, während CLUS einen Clustered Index ausweist

► CLUSTERFACTOR und CUSTERFACTOR

Diese sind in Abschnitt 7.5.2 „Clustered Index“ beschrieben

Des Weiteren steht Ihnen der Befehl `DESCRIBE` zur Verfügung. Dessen Syntax haben wir Ihnen schon bei der Anzeige der Tabellen in Abbildung 7.15 gezeigt. Sie können diesen Befehl wie folgt einsetzen, um grundlegende Informationen über die Indizes einer Tabelle zu bekommen:

```
describe indexes for table <tabellenname>
```

7.6 Prüfung auf Integritätsbedingung (Check Constraint)

Sie können in DB2 eine so genannte „Prüfung auf Integritätsbedingung“ oder „check constraint“ erstellen. Damit wird es möglich, bestimmte Integritätsprüfungen auf Spaltenebene in der Datenbank selbst vornehmen zu lassen. Für die Spalte „Geschlecht“ z.B. sollen nur die Buchstaben „M“ und „W“ gültig sein, alle anderen Eingaben sollen abgewiesen werden.

Wird trotzdem versucht, einen anders lautenden Wert einzufügen oder die Spalteninhalte zu ändern, wird ein Fehler ausgegeben:

SQL0545N Die angeforderte Operation ist nicht zulässig, da eine Zeile gegen die Prüfung auf Integritätsbedingung "MTUSER.ADRESSE.CGESCHL" verstößt.
SQLSTATE=23513

Abbildung 7.27: Fehlermeldung bei Integritätsverletzung

In der Steuerzentrale können diese Integritätsprüfungen über die ÄNDERN...-Auswahl im Kontextmenü der einzelnen Tabelle eingegeben und aufgelistet werden. Abbildung 7.28 zeigt die Registerkarte PRÜFUNG AUF INTEGRITÄTSBEDINGUNG im TABELLE ÄNDERN-Fenster.

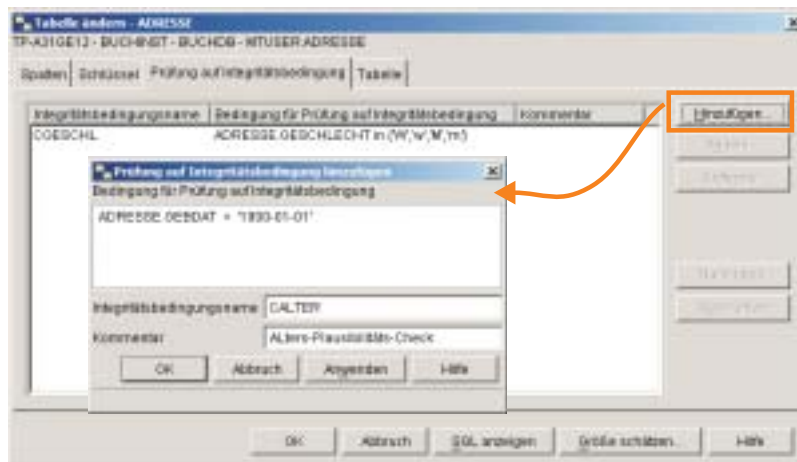


Abbildung 7.28: Prüfung auf Integritätsbedingung mit der Steuerzentrale erstellen

Dort sehen Sie alle bestehenden Regeln und können über den HINZUFÜGEN...-Button auch weitere Regeln erstellen. In diesem Beispiel sehen Sie die bestehende Prüfung mit dem Namen CGESCHL, die nur „M“, „m“, „W“ oder „w“ für die GESCHLECHT-Spalte zulässt. Eine zusätzliche Prüfung soll hinzugefügt werden, um sicherzustellen, dass das Geburtsdatum nicht vor dem Jahr 1900 liegt.

Die Syntax dieses Befehls ist in Abbildung 7.29 anhand eines ALTER TABLE-Befehls dargestellt, obwohl diese Integritätsprüfungen auch auf gleiche Weise beim CREATE TABLE-Befehl angegeben werden können.

Es empfiehlt sich, einen Namen für die Prüfbedingung anzugeben, da dieser dann im Fehlerfall, wie oben (siehe Abbildung 7.27) angezeigt, zurückgegeben wird und der Benutzer nun den Fehler besser erkennen bzw. beschreiben kann. Wird dieser Name nicht spezifiziert, wird ein 18-stelliger Name vom System generiert.

ENFORCED bedeutet, dass die Datenbank die Einhaltung dieser Vorgaben prüft und sicherstellt, während NOT ENFORCED quasi eine rein informative Definition ist, die voraussetzt, dass die Integritätsregel für diese Daten auf andere Weise umgesetzt wird und gilt.

ENABLE QUERY OPTIMIZATION zeigt dem DB2-Optimizer an, dass diese Integritätsprüfung für die Zugriffswegbestimmung mit benutzt werden kann. Das heißt, der Optimizer geht davon aus, dass alle Inhalte der Tabelle diese Regel erfüllen.

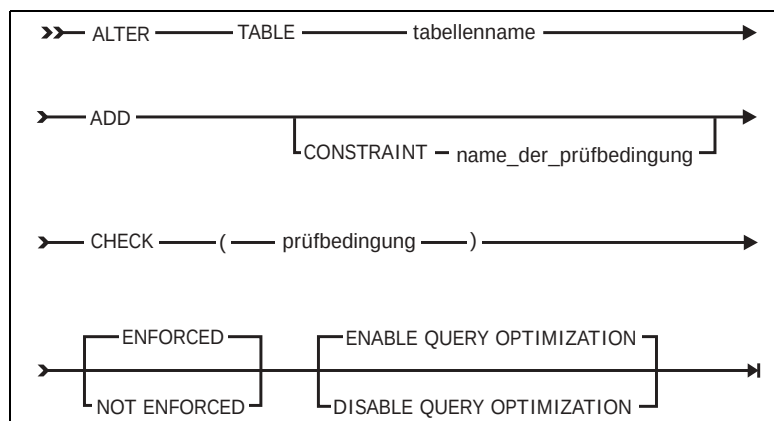


Abbildung 7.29: alter table add check-Syntaxdiagramm

Die Prüfbedingung selbst folgt den SQL-Regeln.

Vorsicht:

Eine Kombination der Optionen NOT ENFORCED und ENABLE QUERY OPTIMIZATION kann zu Problemen führen, wenn die Tabellendaten der Integritätsregel nicht genügen!

DISABLE QUERY OPTIMIZATION lässt den Optimizer diese Regel ignorieren. So aber wird der Vorteil für die Zugriffswegberechnung, der durch dieses „zusätzliche Wissen“ um Dateninhalte gegeben ist, nicht genutzt.

Mit Hilfe der Prüfung auf Integritätsbedingungen kann eine bessere Datenqualität in der Datenbank erreicht werden. In Data Warehouse-Projekten ergeben sich in der Praxis immer Schwierigkeiten durch „unsaubere“ Daten. Diese müssen dann mit viel zeitlichem und monetärem Aufwand bereinigt werden – das Fachwort dafür lautet „data cleansing“. Sinnvolle Integritätsregeln können dazu beitragen, diese Problematik zu entschärfen.

Dazu können auch die Themen in den nächsten Abschnitten beitragen.

7.7 Referenzielle Integrität (Referential Integrity – RI)

Die referenzielle Integrität bildet Abhängigkeiten zwischen zwei Tabellen ab. Bei den Tabellen spricht man in diesem Zusammenhang auch von der „übergeordneten“ und der „abhängigen“ Tabelle. Diese sind miteinander über eine Integritätsregel verknüpft. Diese Beziehung wird durch eine Primärschlüssel-Fremdschlüssel-Beziehung abgebildet, in der die eindeutige Primärschlüssel-Spalte der übergeordneten Tabelle der so genannte Fremdschlüssel in der abhängigen Tabelle wird.

In Abbildung 7.30 sehen Sie ein Beispiel für die Funktionsweise. Aus der Tabelle ABTEILUNG soll die Abteilung mit dem Abteilungscode 20 gelöscht werden – eine Umorganisation also, wie sie häufig in Firmen vorkommt und manchmal fast

Hauptbeschäftigung zu sein scheint. Die Tabelle ABTEILUNG ist in diesem Fall die übergeordnete Tabelle und hat die Spalte Abteilungscode als Primärschlüssel (primary key). Das Bild zeigt nun drei Beispiele der Verknüpfung mit der abhängigen Tabelle MITARBEITER. Der Fremdschlüssel der abhängigen Tabelle verweist immer bzw. muss auf den Primärschlüssel der übergeordneten Tabelle zeigen.

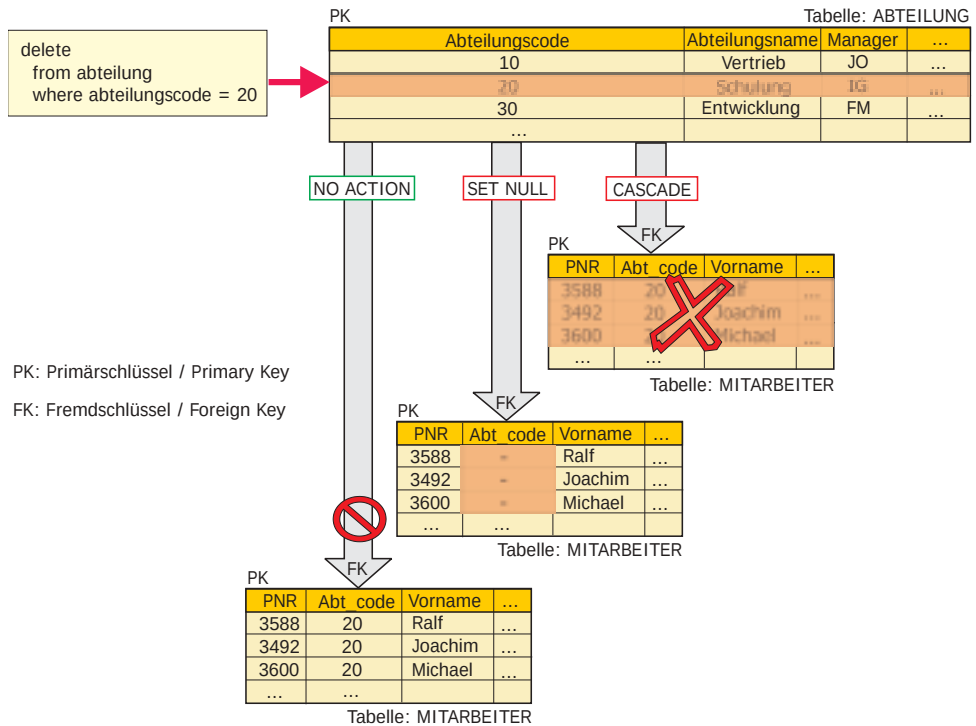


Abbildung 7.30: Referenzielle Integrität – Löschregeln

Der erste Fall zeigt eine Verknüpfung mit der NO ACTION-Regel. Dies bedeutet, dass der Abteilungseintrag 20 nicht in der übergeordneten Tabelle ABTEILUNG gelöscht werden darf, da der Schulungsabteilung noch Mitarbeiter zugeordnet sind. Die Anwendung erhält eine Fehlermeldung mit dem Verweis auf diesen RI-Namen. Hier wird sozusagen das „deutsche“ Modell abgebildet, weil der Betriebsrat so eine Umorganisation nicht zulässt, bevor die Mitarbeiter einen neuen Job und somit eine Abteilung in der Firma zugeordnet bekommen haben.

SET NULL ist die zweite Verknüpfungsmöglichkeit. Der Abteilungseintrag kann hier in der übergeordneten Tabelle gelöscht werden. In der abhängigen MITARBEITER-Tabelle werden die Abt_code-Einträge der (ehemaligen) Mitarbeiter auf „NULL“ – also nicht vorhanden – gesetzt. Dies ist ein „Chaos“-Modell, das wohl in der Realität am häufigsten dort zu finden ist, wo viel umstrukturiert wird und die Mitarbeiter häufig nicht einmal mehr ihren Abteilungsnamen und Vorgesetzten kennen. Kaum sind die „neuen“ Visitenkarten gedruckt, steht schon die nächste Reorganisation an.

Das „amerikanische“ Modell mit „hire and fire“ wird im letzten Fall – mit der CASCADE-Regel, auch mehrstufige Löschoption genannt – implementiert. Die Mitarbeiter werden automatisch zusammen mit der Abteilung gelöscht (also entlassen).

Weitere wichtige Punkte bei der referenziellen Integrität (RI) sind:

- ▶ Es kann mehrere Fremdschlüssel in einer Tabelle geben, aber nur ein Primärschlüssel kann definiert sein.
- ▶ Die Definition eines Primärschlüssels erzeugt automatisch einen eindeutigen Index auf die Primärschlüsselspalte(n), falls dieser nicht vorhanden ist.
- ▶ Die Fremdschlüsselspalte muss denselben Datentyp und die gleiche Längendefinition haben wie der Primärschlüssel der übergeordneten Tabelle, aber nicht den gleichen Spaltennamen.
- ▶ Eine Tabelle kann auch auf sich selbst verweisen und ist dann sowohl übergeordnete als auch abhängige Tabelle in einem.
- ▶ Die Werte des Primärschlüssels bestimmen die gültigen Werte für den Fremdschlüssel. In unserem Beispiel aus Abbildung 7.30 bedeutet dies, dass nur Mitarbeiter eingefügt werden dürfen, wenn ihr Abteilungscode einem gültigen Abteilungscode in der Tabelle ABTEILUNG entspricht.

Fremdschlüssel werden mit der grafischen Oberfläche erstellt, indem Sie auf der abhängigen Tabelle das Kontextmenü öffnen, ÄNDERN... auswählen und auf den SCHLÜSSEL-Reiter klicken. Dort sehen Sie alle definierten Primär- und Fremdschlüssel (vgl. Abbildung 7.31) und haben über die Buttons am rechten Fensterrand die Möglichkeit, bestehende Definitionen zu ändern bzw mit FREMDSCHLÜSSEL HINZUFÜGEN... neue Schlüssel zu erstellen.

Bei letzterer Möglichkeit öffnet sich das FREMDSCHLÜSSEL HINZUFÜGEN-Fenster, das in Abbildung 7.32 gezeigt wird.

Dort können Sie im oberen Teil die Angaben zur übergeordneten Tabelle machen und den Schema- und Tabellennamen auswählen; der vorher definierte Primärschlüssel wird automatisch angezeigt. Da Sie dieses GUI von der untergeordneten Tabelle aus aufgerufen haben, benötigen Sie nur noch die Auswahl der Fremdschlüsselspalte aus der Liste VERFÜGBARE SPALTEN.



Abbildung 7.31: Tabelle ändern – Schlüssel-Reiter in der Steuerzentrale

Hier erfolgt keine automatische Übernahme der Spalte mit gleich lautendem Namen, da diese Abhängigkeit nicht gegeben sein muss. Ist die richtige FREMD-SCHLÜSSEL-Spalte gewählt, suchen Sie sich die Integritätsregel aus und vergeben einen Namen, der sonst wieder vom System bestimmt werden würde.

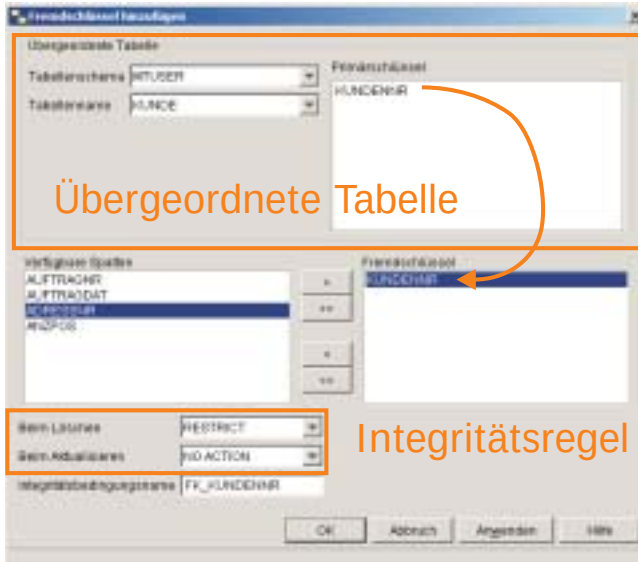


Abbildung 7.32: Fremdschlüssel hinzufügen-Fenster in der Steuerzentrale

Durch OK schließen Sie die RI-Definition ab.

Wenn Sie die Tabellen in der „richtigen“ Reihenfolge erstellen, ist es auch möglich, die RI-Definition bereits direkt im CREATE TABLE-Befehl vorzunehmen. Meistens ist es jedoch einfacher und übersichtlicher, diese Definition in einem separaten Befehl vorzunehmen. Auch dafür kann der sehr mächtige ALTER TABLE-Befehl verwendet werden. Die Syntax wurde in Abbildung 7.33 speziell nur für die RI- bzw. Fremdschlüsseldefinition aufbereitet.

Die Optionen sind denen der Integritätsprüfung sehr ähnlich. Sie sollten einen Namen für die Prüfbedingung vergeben, um Fehlermeldungen besser zuordnen zu können. Die Spalten, die Sie bei der FOREIGN KEY-Option angeben, müssen inhaltlich und strukturell (aber nicht namentlich) denen der übergeordneten Tabelle entsprechen, die in der REFERENCES-Klausel angegeben werden müssen. Die Integritätsregeln der Löschung (delete) wurden schon im Zusammenhang mit Abbildung 7.30 erläutert und gelten entsprechend für die verfügbaren Regeln beim Update.

Die Ausprägungen NO ACTION und RESTRICT haben letztlich die gleiche Löscho- bzw. Update-Einschränkung für die übergeordnete Tabelle. Allein der Zeitpunkt, wann diese Regel überprüft wird, ist unterschiedlich. Bei NO ACTION (Standardfall) wird die Regel *nach* allen anderen Integritäts-Checks überprüft, bei RESTRICT wird *vorrangig*, also als Erstes diese Regel geprüft. Nur in wenigen Ausnahmefällen ergibt sich dadurch ein Unterschied. Ein Beispiel dazu finden Sie in [SQLREF] unter der CREATE TABLE-Beschreibung.

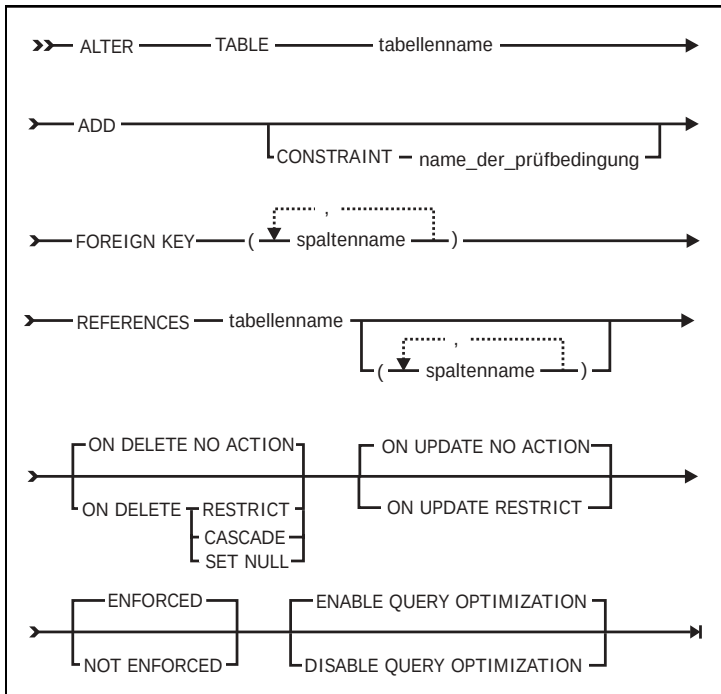


Abbildung 7.33: alter table add foreign key-Syntaxdiagramm

Seit DB2 V8 gibt es auch die Möglichkeit der so genannten „informellen referenziellen Integrität“. Diese entspricht der RI-Definition mit der Option `NOT ENFORCED`. Dabei werden die Integritätsdefinitionen zwar vorgenommen, aber nicht von DB2 überprüft.

„Warum dann der ganze Aufwand?“ werden Sie fragen. „Für den Optimizer“ lautet die Antwort.

In speziellen Fällen – und zwar in Data Warehouse-Umgebungen – will man dem Optimizer alle Informationen zur Verfügung stellen, die er für eine gute Zugriffswegberechnung braucht, kann aber andererseits nicht auf die Ladereihenfolge der Tabellen Rücksicht nehmen, da das zu viel Zeit kosten würde. Bei durch RI verknüpften Tabellen muss nämlich die übergeordnete Tabelle vor der abhängigen Tabelle gefüllt werden, um keine Integritätsverletzungen und somit Fehler zu bekommen. In diesen Fällen also ist diese Option durchaus sinnvoll, die Einhaltung der Integritätsbedingungen wird dabei durch andere Mittel (das ETL-System; ETL steht für Extract-Transform-Load) gewährleistet.

7.7.1 Kochbuch RI

Die Definition einer referenziellen Abhängigkeit erfolgt in folgenden Schritten:

1. Definition des Primärschlüssels auf der übergeordneten Tabelle

Diese kann über den `SCHLÜSSEL`-Reiter im `TABELLE ÄNDERN`-Fenster der Steuerzentrale erfolgen oder über den Befehl:

```
alter table <tab_name> add constraint <pk_name>  
primary key (spalte)
```

2. Definition des Fremdschlüssels auf der untergeordneten Tabelle (vgl. Abbildung 7.33)
3. Sind die Tabellen leer, muss sichergestellt werden, dass die übergeordnete Tabelle zuerst – also vor der abhängigen Tabelle – gefüllt wird.
4. Sind bereits Daten in den Tabellen, muss nun noch sichergestellt werden, dass diese der neuen Integritätsbedingung genügen.

Dafür steht ein SET INTEGRITY-Befehl zur Verfügung, der hier beispielhaft aufgeführt ist und der auch verwendet werden kann, um den Integritäts-Check zeitweilig abzuschalten:

```
set integrity for <tabellenname> immediate checked  
for exception in <tabellenname> use <exception_tabellenname>
```

Dieser Befehl wird in Kapitel 8 im Zusammenhang mit dem Lade-Tool noch näher beschrieben.

7.8 Auslöser (Trigger)

Der Unterschied bei Auslösern (Triggern) ist, dass es hier Integritätsverknüpfungen zu (mehreren) anderen Tabellen geben kann. INSERT-, UPDATE- oder DELETE-Operationen können einen definierten Auslöser aktivieren. Dieser „löst“ dann weitere SQL-Operationen „aus“. Somit ist es z.B. möglich, neue Tabelleneinträge zu generieren, Berechnungen automatisch anzustellen und Abläufe zu protokollieren. Bestimmt finden Sie auch für Ihre Umgebung etliche nützliche Einsatzmöglichkeiten.

Wir schauen uns die Funktionsweise einfach anhand von Abbildung 7.34 an:

- Der erste Schritt besteht darin, einen Auslöser, im Weiteren auch Trigger genannt, in der Datenbank zu definieren.

Der Trigger soll im Beispiel bei UPDATE-Operationen auslösen – und zwar nur dann, wenn eine Bedingung erfüllt ist. Die Bedingung ist hier, dass der Inhalt der Anzahl-Spalte der LAGER-Tabelle den Wert 10 unterschreitet. Ausgelöst wird dann ein INSERT-Befehl, der einen Eintrag in die Tabelle BESTELLUNG vornimmt. Die genaue Erläuterung und das Syntaxdiagramm finden Sie weiter unten in Abbildung 7.38.

- Unter Punkt ② wird eine so genannte „auslösende SQL-Operation“ gezeigt. Die Änderung, die in der LAGER-Tabelle vorgenommen wurde, erfüllt die Trigger-Kriterien, da vom Maschinenteil 47110815 nach der Änderung nur noch 9 (also < 10 Stück) im Lager verbleiben.
- Als Drittes wird der UPDATE-Befehl durchgeführt.
- Zusätzlich findet nun das ausgelöste Ereignis statt: Das Maschinenteil wird in die Bestellungstabelle eingetragen.

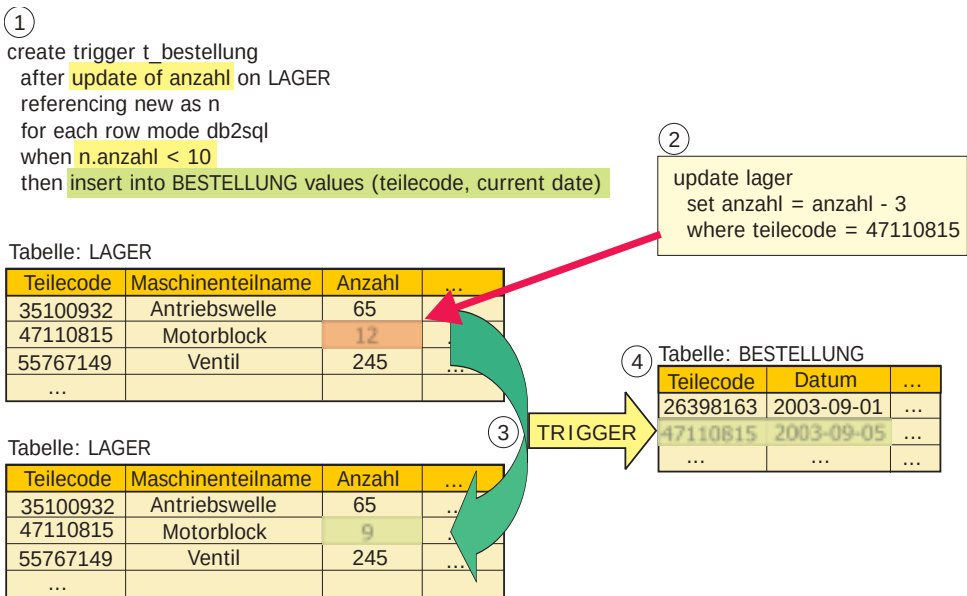


Abbildung 7.34: Auslöser (Trigger)

Durch diese Technik lassen sich auch komplexere Abhängigkeiten in der Datenbank selbst regeln. Sie vermeiden so Dateninkonsistenz und ermöglichen eine Verlagerung der Geschäftsregeln aus der Anwendung in die Datenbank.

Auch Auslöser haben einen extra Eintrag im Objektbaum der Steuerzentrale und können so grafisch basiert erstellt werden. Wählen Sie dazu ERSTELLEN... im AUSLÖSER-Kontextmenü.

Abbildung 7.35 zeigt die erste Registerkarte des AUSLÖSER ERSTELLEN-Fensters. Dort werden das AUSLÖSCHEMAS, der AUSLÖSERNAME und der ZEITPUNKT DER AKTION sowie die Operation definiert, die später den Trigger auslösen soll. Im Teilfenster OPERATION... sehen Sie, dass ein einzelner Trigger entweder auf EINFÜGEN (INSERT), SPALTEN AKTUALISIEREN (UPDATE) oder LÖSCHEN (DELETE) definiert werden kann. Wollen Sie aber bei den verschiedenen Aktionen INSERT, UPDATE und DELETE Trigger auslösen, muß der Trigger pro Aktion separat definiert werden.

Die gezeigte Auswahl entspricht dem Beispielfall aus Abbildung 7.34.

Die zweite Registerkarte, AUSGELOSTE AKTION (siehe Abbildung 7.36), ermöglicht dann die Eingabe der Aktionen per SQL.

Dabei kann auf die „alten“ und „neuen“ Tabellenwerte durch so genannte Korrelationsnamen Bezug genommen werden. In unserem Beispiel haben wir n für „neuen“ Inhalt definiert und verweisen in der ausgelösten Aktion bzw. deren Zusatzbedingung per n.anzahl damit auf den neuen Lagerbestand. Ist die Zusatzbedingung erfüllt, wird ein Eintrag in die BESTELLUNG-Tabelle veranlasst, und dort werden der zu bestellende Teilecode und das aktuelle Datum vermerkt.

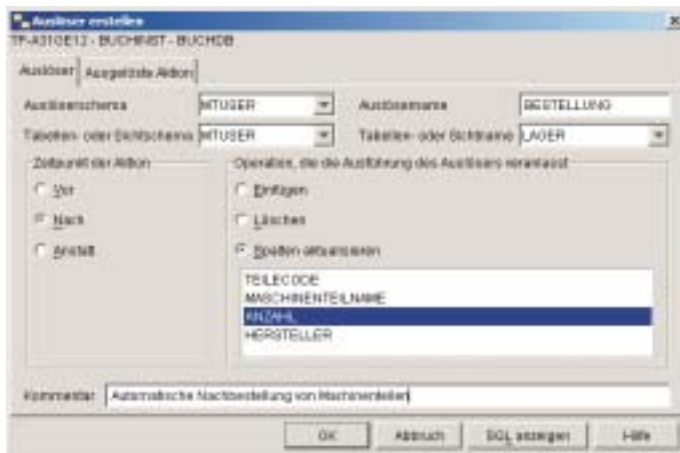


Abbildung 7.35: Fenster Trigger erstellen – Auslöser in der Steuerzentrale

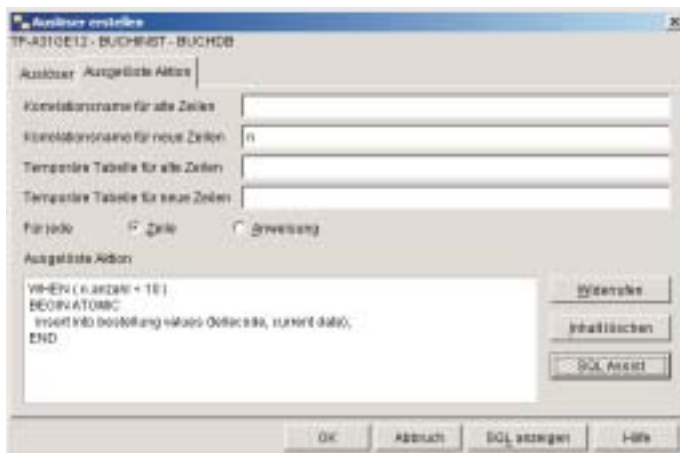


Abbildung 7.36: Fenster Trigger erstellen – Ausgelöste Aktion in der Steuerzentrale

Über diese Korrelationsnamen ist es möglich, auch einen „Vorher-nachher“-Vergleich durchzuführen oder den Trigger bei einer bestimmten prozentualen Abweichung auszulösen.

Die Konstruktion mit BEGIN ATOMIC und END lässt es zu, dass mehrere SQL-Befehle ausgeführt werden können. SQL PL – also die *SQL Procedural Language* – kann hier ebenfalls eingesetzt werden.

Über den Button SQL ANZEIGEN kommen Sie zum Fenster aus Abbildung 7.37 mit dem Ergebnis Ihrer Definitionen.

Der Befehl CREATE TRIGGER, den Sie schon in Abbildung 7.34 und Abbildung 7.37 gesehen haben, wird nach folgender Syntax erstellt.



Abbildung 7.37: Fenster Trigger erstellen – SQL anzeigen in der Steuerzentrale

Die Optionen im Überblick:

- ▶ auslösername
Schema-Name (optional) und Name des Triggers
- ▶ Auslösezeitpunkt
 - NO CASCADE BEFORE
Der Trigger wird *vor* dem eigentlichen – also dem auslösenden SQL-Befehl – ausgeführt.
 - AFTER
Der Trigger wird *nach* dem auslösenden SQL-Befehl ausgeführt.
 - INSTEAD OF
Der Trigger wird *statt* des auslösenden SQL-Befehls ausgeführt. Diese DB2 V8-Funktionalität kann nur auf Sichten definiert werden und dient u.a. der Implementierung von „änderbaren“ Sichten (updateable views).

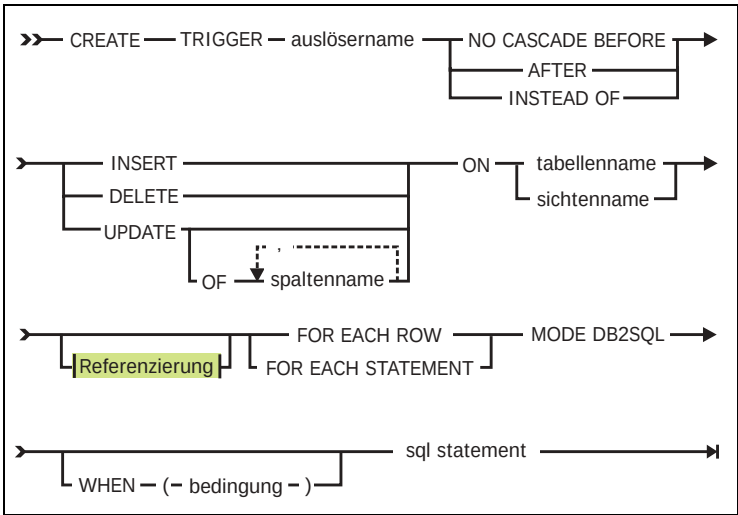


Abbildung 7.38: create trigger-Syntaxdiagramm

- ▶ **Auslöseereignis (trigger event)**
INSERT, UPDATE oder DELETE stehen hier zur Verfügung.
- ▶ **ON tabellenname**
Die auslösende Tabelle oder Sicht kann vollqualifiziert angegeben werden.
- ▶ **Referenzierung**
Mit der Referenzierung kann auf „alte“ und „neue“ Inhalte einer Spalte oder temporären Tabelle verwiesen werden. Die Angabe erfolgt nach der Syntax aus Abbildung 7.39.

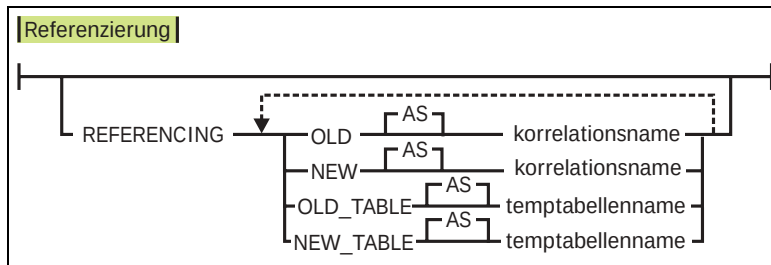


Abbildung 7.39: create trigger referencing-Syntaxdiagramm

- ▶ **FOR EACH ROW oder FOR EACH STATEMENT**
Hier wird festgelegt, ob der Trigger einmal pro Befehl ausgeführt wird oder einmal pro qualifizierter Zeile. Man bezeichnet dies auch als Auslösegranularität.
- ▶ **WHEN (bedingung)**
Eine zusätzliche Bedingung kann mit der Trigger-Auslösung verknüpft werden.
- ▶ **sqlstatement**
Das ausgelöste SQL kann auch aus mehreren Anweisungen bestehen, muss dann aber zwischen BEGIN ATOMIC- und END-Befehlen stehen.

Hinweis:

Dabei ist zu beachten, dass das Semikolon „;“, das DB2 im Befehlsfenster standardmäßig als Anweisungsende interpretiert, hier zu Problemen führen kann, da es auch als Trennzeichen bei der SQL Procedural Language verwendet wird. Sie können dieses Problem umgehen, indem Sie in der Befehlszeile bzw. im Command Center ein anderes Zeichen für das Anweisungsende definieren (siehe Abschnitt 4.2.4).

Allgemeine Punkte zu Auslösern:

- ▶ Trigger können hintereinander geschaltet werden, d.h., ein Trigger kann einen weiteren Trigger auslösen – bis zu einer theoretischen Tiefe von 16 kaskadierenden Aufrufen.

- ▶ Die ausgelöste Aktion ist Bestandteil der Transaktion, d.h., erst wenn auch das ausgelöste SQL erfolgreich durchgeführt worden ist, wird die Transaktion festgeschrieben. Schlägt der ausgelöste SQL-Befehl fehl, wird alles per ROLLBACK rückgängig gemacht.
- ▶ Trigger-Definitionen werden im Systemkatalog gespeichert und können über die Systemsicht SYSCAT.TRIGGERS abgefragt werden.

Weitere – auch komplexere Beispiele – finden Sie bei [YIP1] und [YIP2].

7.9 Benutzerdefinierter Datentyp (UDT)

Benutzerdefinierte Datentypen (UDT – User Defined Types) ermöglichen es, die Datenbank durch eigene Typdefinitionen zu erweitern.

In DB2 werden zwei Arten von UDTs unterschieden:

- ▶ benutzerdefinierter einzigartiger Datentyp
- ▶ benutzerdefinierter Strukturtyp

7.9.1 Benutzerdefinierter einzigartiger Datentyp (User Defined Distinct Type)

Diese Art basiert auf einem bestehenden Datentyp, wird aber für semantische Zwecke als separater und inkompatibler Datentyp angesehen.

Beispielsweise können Sie einen DECIMAL-Wert als Preis definieren. Dieser Preis-Datentyp kann dann nur mit anderen „Preisen“ verglichen oder aggregiert werden. Andere Beispiele könnten die Unterscheidung von Währungen oder Maßeinheiten sein – wir denken da an „Preis in US-\$“ oder „Länge in Zoll“. Dadurch verhindern Sie, dass in SQL-Abfragen Äpfel mit Birnen verglichen werden und so falsche Ergebnisse entstehen, was vorher – also durch den Basisdatentyp – möglich gewesen wäre.

Man nennt dies auch *strenge Typentrennung*, da trotz gleicher Darstellung der Daten diese nur innerhalb ihres Typs kompatibel sind.

Diese Datentypen können in der Steuerzentrale unter ANWENDUNGSOBJEKTE/BENUTZERDEFINIERT EINZIGARTIGE DATENTYPEN angelegt werden.

Als Befehl sieht das dann so aus:

```
create distinct type mtuser.zoll
as decimal (3,2) with comparisons
```

Nach dieser Definition können nun Tabellen mit Spalten dieses neuen Typs erstellt werden, beispielsweise die Tabelle Monitore:

```
create table monitore (
  ID int, Hersteller char(50), Monitortyp char(50), Größe zoll, ...)
```

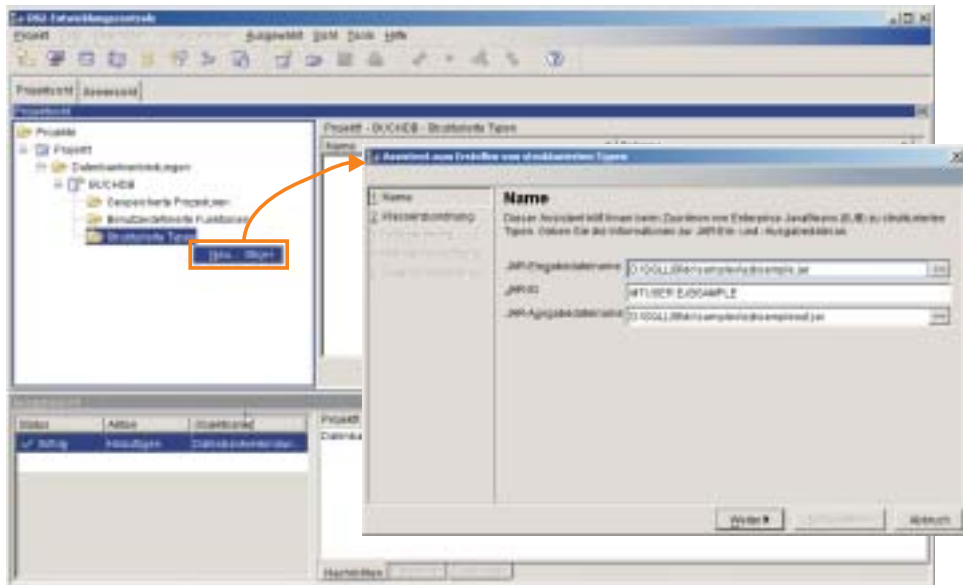



Abbildung 7.41: Strukturierten Datentyp mit der DB2-Entwicklungszentrale erstellen

Diese strukturierten Datentypen können nun in Tabellendefinitionen eingesetzt werden, z.B.:

```
create table mitarbeiter (person, bankverbindung, gehalt)
create table typedt of bankverbindung (ref is bankverbindung user generated)
```

Die Anzeige ist über die Steuerzentrale unter ANWENDUNGSOBJEKTE/BENUTZERDEFINIERTER STRUKTURIERTER DATENTYPEN möglich.

Allgemein gilt, dass benutzerdefinierte Datentypen nur dann gelöscht werden können, wenn sie nicht mehr verwendet (benutzt) werden.

7.10 Benutzerdefinierte Funktion (User Defined Function – UDF)

DB2 stellt von Haus aus viele Funktionen aus allen möglichen Bereichen in den Datenbanken zur Verfügung (z.B. dayofweek, cos, lcase etc.). Ist eine bestimmte Funktionalität, die Sie benötigen würden, nicht dabei, haben Sie die Möglichkeit, diese selbst zu definieren und die Datenbank dahingehend zu erweitern. Dabei ist es möglich, *externe UDFs* zu erstellen. Diese sind in einer externen Programmiersprache geschrieben – in Visual Basic, C/C++, Java oder SQL. Alternativ können Sie bei der Implementierung einer so genannten *Quellenfunktion* auf bestehende Funktionen zurückgreifen.

Es werden drei Typen von benutzerdefinierten Funktionen unterschieden:

► Skalar-Funktionen

Diese liefern nur einen einzelnen Wert zurück. Ein Beispiel für eine Skalarfunktion ist `dayname` oder `length` – beide sind in DB2 standardmäßig enthalten.

► Spalten-Funktionen

Diese liefern auch einen einzelnen Wert zurück, der aber ein Ergebnis aus Verknüpfungen mehrerer Werte (einer Spalte) ist. `SUM()` und `AVG()` sind Beispiele für diese Art von UDF.

Diese Art kann nur als Quellenfunktion erstellt werden – d.h., sie muss auf bestehenden Funktionen basieren. So könnte man z.B. mit `AVG(PREIS)` den Durchschnitt aus der *Preis*-Spalte berechnen lassen, die wir in Abschnitt 7.9.1 „Benutzerdefinierter einzigartiger Datentyp (user defined distinct type)“ als UDT erstellt haben.

► Tabellen-Funktionen

Auch diese UDFs werden ihrem Namen gerecht und liefern eine ganze Tabelle zurück. Im Gegensatz zu den anderen UDFs werden sie in der `FROM`-Klausel des SQL-Befehls angesprochen – zusätzlich oder als Ersatz für Tabellen. Diese Art kann nur als externe Funktion erstellt werden, muss also immer programmiert werden.

Hinweis:

Tabellen-UDFs oder Tabellen-Funktionen (table functions) sind ein überaus mächtiges Konstrukt, durch das es möglich wird, mit voller SQL-Mächtigkeit (SQL-Funktionalität) auf *nicht-relationale* Objekte zuzugreifen, also auf Objekte, die sich nicht in der Datenbank befinden. Diese Möglichkeit ist nur wenigen Nutzern bekannt, obwohl es diese Funktionalität schon lange gibt. Letztendlich kann man so auf alles mit SQL zugreifen, was man programmieren kann.

Beispielsweise könnten Sie über eine Tabellen-Funktion Inhalte einer Datei mit Ihren Tabellen verknüpfen, ohne vorher die Datei umzuformatieren und in die Datenbank zu laden. Dieses Szenario ist z.B. bei Data Warehouse-Umgebungen oft gegeben, wenn hinzugekaufte Informationen regelmäßig in einem „ungewöhnlichen“ Format bereitgestellt werden. Eine weitere Möglichkeit ist es sogar, Daten aus dem Internet durch Tabellen-Funktionen zu „tabellarisieren“ und weiterzuverarbeiten.

In DB2 werden diese Funktionen genutzt, um die Ergebnisdatei des Snapshot-Monitors aufzubereiten und über SQL abfragbar zu machen. `SNAPSHOT_TABLE()` ist ein Beispiel für Tabellen-Funktionen.

Hier gilt das gleiche Verhalten wie bei benutzerdefinierten Strukturtypen. Die benutzerdefinierten Funktionen (UDFs) können in der Steuerzentrale angezeigt, aber nicht mit ihr angelegt werden. Um UDFs zu erstellen, müssen Sie in die DB2-Entwicklungszentrale (siehe Abbildung 7.42) wechseln.

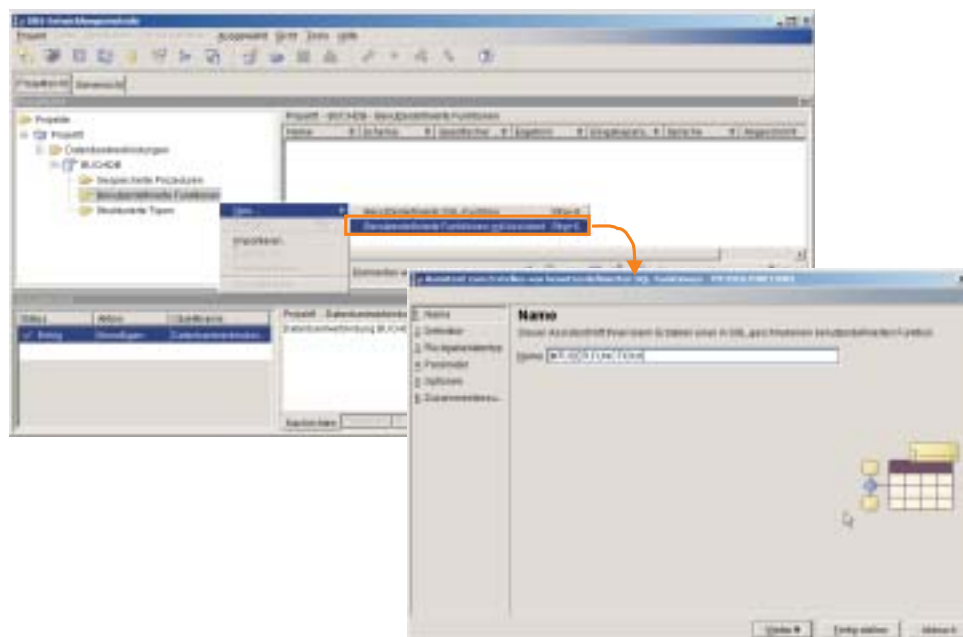


Abbildung 7.42: Benutzerdefinierte Funktionen mit der DB2-Entwicklungszentrale erstellen

Der Befehl `CREATE FUNCTION` ermöglicht es, *Quellenfunktionen* zu definieren oder *externe Funktionen* in DB2 zu registrieren.

Benutzerdefinierte Funktionen zählen (mit gespeicherten Prozeduren und Methoden) in DB2 zu den so genannten *Routinen*, da bei allen diesen Objekten die Strukturen (z.B. Parameterdarstellung) ähnlich bzw. gleich sind. Die Informationen zu diesen Objekten finden Sie im Systemkatalog unter anderem in der Sicht `SYS-CAT.ROUTINES` und `SYS-CAT.ROUTINEPARMS`.

7.11 Gespeicherte Prozeduren (Stored Procedures)

Die gespeicherten Prozeduren gehören ebenfalls – wie die benutzerdefinierten Funktionen – zu der Gruppe der *Routinen*.

Eine Liste der verfügbaren gespeicherten Prozeduren finden Sie in der Systemkatalog-Sicht `SYS-CAT.ROUTINES` oder über den Eintrag `ANWENDUNGSOBJEKTE/GESPEICHETERE PROZEDUREN` der Steuerzentrale.

Gespeicherte Prozeduren oder *Stored Procedures* sind eine Art *Unix-RPC* (Remote Procedure Call) nur für Datenbanksysteme. Dabei wird ein Teil des Anwendungsprogramms auf den DB2-Server verlagert und dort *lokal* ausgeführt. Das soll den Kommunikationsaufwand reduzieren und dadurch die Leistung, vor allem im Client-Server-Umfeld, erhöhen. Außerdem wird so die Trennung der Anwendungslogik von der Datenbanklogik möglich.

Abbildung 7.43 zeigt eine *normale* Anwendung – also ohne den Einsatz einer gespeicherten Prozedur. Jeder Verarbeitungsschritt (SQL) wird über das Netz an den

entfernten Datenbank-Server geleitet. Dort wird er ausgeführt, und dann wird eine Ergebnismeldung zurückgeschickt.

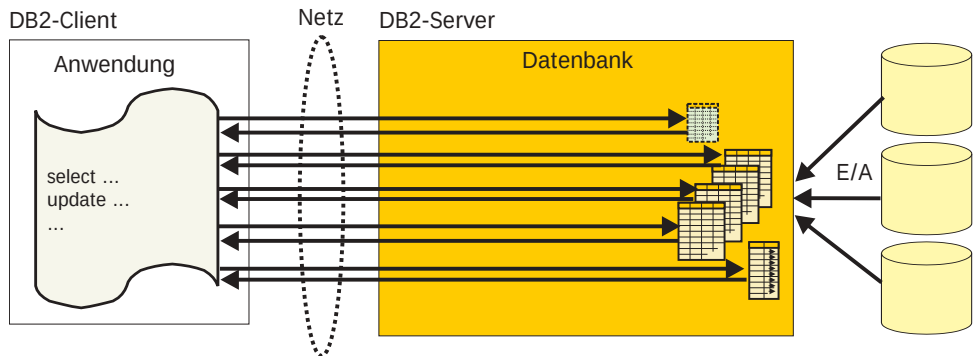


Abbildung 7.43: Funktionsweise ohne gespeicherte Prozedur

Da dies für jede Anweisung geschehen muss, entsteht ein hoher Kommunikationsaufwand, der besonders „teuer“ beim Netztransfer zu Buche schlägt.

Abbildung 7.44 zeigt die alternative Lösung mit einer gespeicherten Prozedur. Diese wird von der Anwendung über einen CALL-Befehl aufgerufen. Die Prozedur muss vom Datenbank-Server in den Speicher geladen werden. Die in ihr gespeicherten Anwendungsteile (Befehle) werden dann *lokal* ausgeführt. Letztendlich wird nur eine Ergebnismeldung an den Client zurückgeschickt.

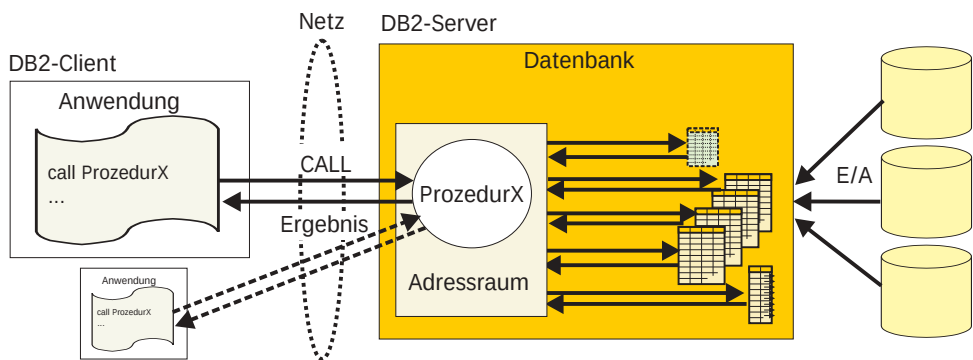


Abbildung 7.44: Funktionsweise mit gespeicherter Prozedur

Die Last am Datenbank-Server wird dadurch zwar erhöht, da zusätzliche Programmlogik und Verarbeitungsschritte ausgeführt werden müssen, aber die Netzlast wurde deutlich reduziert. Die ursprüngliche Anwendung ist so „schlanker“ geworden, und weitere Anwendungen können die gespeicherte Prozedur nun ebenfalls verwenden.

Gespeicherte Prozeduren können im abgesicherten Modus (fenced) laufen, d.h. in einem eigenen Adressraum, wie auch in Abbildung 7.44 angedeutet. Dadurch haben Fehler im Programm keine Auswirkung auf den DB2-Adressraum.

Der Modus „not fenced“ – also nicht abgesichert – dagegen bedeutet, dass die Prozedur im Adressraum des DB2 läuft. Dies bringt Leistungsvorteile, ist aber auch „gefährlich“, da Fehler unsauberer, nicht getesteter Prozeduren DB2 auf diese Weise beeinflussen können.

Gespeicherte Prozeduren können in den Sprachen SQL, C/C++, Java, OLE oder COBOL entwickelt werden.

Der CREATE PROCEDURE-Befehl dient zur Registrierung einer extern erstellten gespeicherten Prozedur in DB2. Auch hier kann uns die DB2-Entwicklungszentrale bei der Erstellung unterstützen.

Hinweis:

Bei der Erstellung einer gespeicherten Prozedur ist immer ein für die Programmiersprache passender Compiler auf dieser Maschine notwendig. Das gilt auch dann, wenn Sie die DB2-Entwicklungszentrale benutzen. Bei einer gespeicherten SQL-Prozedur ist momentan noch ein C-Compiler Voraussetzung.

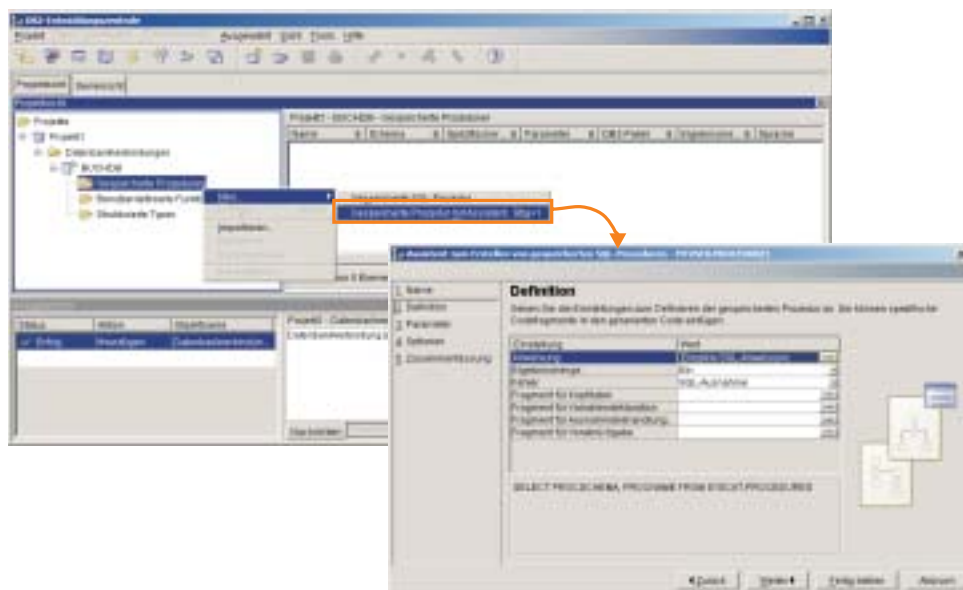


Abbildung 7.45: Gespeicherte Prozedur mit der DB2-Entwicklungszentrale erstellen

7.12 Mehrdimensionale Cluster-Tabelle (MDC)

Hinter *mehrdimensionalen Cluster-Tabellen* (Multidimensional Clustering Tables – MDC) verbirgt sich ein komplett neues Konzept.

Es geht hierbei um *Clustering*, wie der Name schon sagt. Wenn Sie an Abschnitt 7.5.2 „Clustered Index“, zurückdenken, stand dieser Begriff für effizienten E/A-

Zugriff. Effizient deshalb, weil die Daten physikalisch „zusammenstanden“ und mit wenigen E/A-Anforderungen gelesen werden konnten.

Cluster-Tabelle nennt man sie deshalb, weil diese Option beim CREATE TABLE-Befehl angegeben wird – wie gesagt, ein ganz anderes Konzept als bei Clustered Indizes.

Mehrdimensional besagt, dass die Daten nach mehreren Schlüsseln – hier auch *Dimensionen* genannt – geclustert sein können. Während es nur einen Index mit der CLUSTER-Option auf eine Tabelle geben darf und die Tabellen nur nach *einem* Index (sozusagen eindimensional) „geclustert“ sein konnten, wird diese Beschränkung durch *mehrdimensionale* Cluster-Tabellen aufgehoben.

Zusammengefasst ist MDC also eine Tabelle, die es ermöglicht, die Daten so zu speichern, dass ein Zugriff über eine oder mehrere Dimensionen durch das Clustering immer effizient durchgeführt werden kann. Das führt zu einer deutlichen Leistungssteigerung, und zusätzlich ist weniger Wartungs- und Pflegeaufwand notwendig, da diese Struktur ihre Clustering-Eigenschaft dauerhaft aufrechterhält und so z.B. keine Reorganisationen notwendig sind.

Die Erstellung einer mehrdimensionalen Cluster-Tabelle geschieht durch die ORGANIZE BY-Klausel im „normalen“ CREATE TABLE-Befehl (siehe Abbildung 7.46). Dabei werden die Dimensionen mitgegeben. Als Dimension werden die einzelnen Kriterien bezeichnet, nach denen die Daten geclustert werden bzw. über die effizient zugegriffen werden soll. Dabei kann eine einzelne Dimension auch aus mehreren Spalten zusammengesetzt sein.

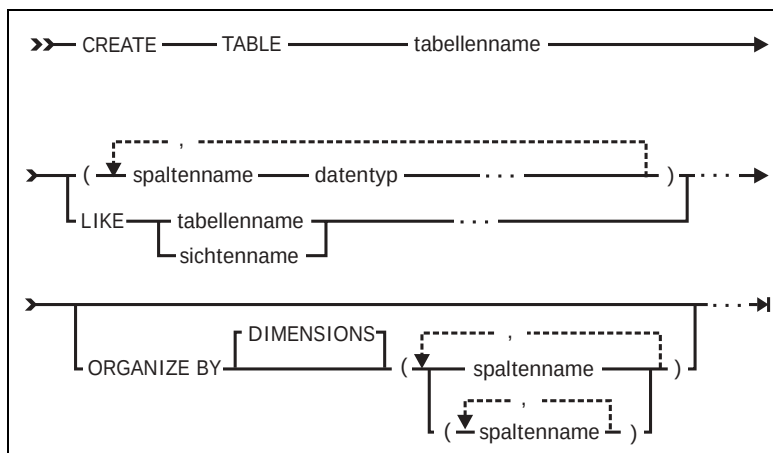


Abbildung 7.46: create table-Syntaxdiagramm für MDC-Tabellen

Zeit für ein Beispiel: Abschnitt 7.12 „Mehrdimensionale Cluster-Tabelle (MDC)“ zeigt eine Tabelle UMSATZ, in der für jedes Projekt und Geschäftsfeld der erzielte Umsatz mit zugehörigem Datum (im Bild nicht explizit gezeigt) gespeichert wird. Für eine einfachere Quartalszuordnung wurde zusätzlich die im Bild gezeigte JQuartal-Spalte erzeugt. Für den Benutzer ist bei der Abfrage dieser MDC-Tabelle kein Unterschied zu anderen Tabellen zu erkennen. Die Tabelle ist – genau wie jede

andere Tabelle auch – in Speicherbereichen (extents) bzw. in deren Seiten gespeichert. Bis hierher gibt es also keinen Unterschied – so weit, so gut.

In dem am unteren Bildrand angezeigten CREATE TABLE-Befehl erkennen Sie durch die ORGANIZE BY-Option, dass es sich um eine MDC-Tabelle handelt. Die Spalte GFeld – also Geschäftsfeld – wird als erste Dimension angegeben, die Zeit durch JQuartal als die zweite Dimension.

Es soll also effizient über die Zeit bzw. das Quartal zugegriffen werden können, genauso über das Geschäftsfeld und über die Kombination der beiden. Dazu werden in einem Extent – man spricht hier von einem „Block“ – immer nur Daten mit einer bestimmten Kombination von Dimensionen gespeichert.

Im Beispiel haben wir einige Blöcke zur besseren Referenzierbarkeit mit Nummern versehen. Block 29 z.B. beinhaltet nur Zeilen – und somit Umsätze – des Geschäftsfelds DL (Dienstleistung) vom 1. Quartal 2003 (200301). Der Block 2 – und genauso Block 13 und 25 – beinhaltet somit die Umsatzzahlen vom 1. Quartal des Geschäftsfelds Hardware (HW).

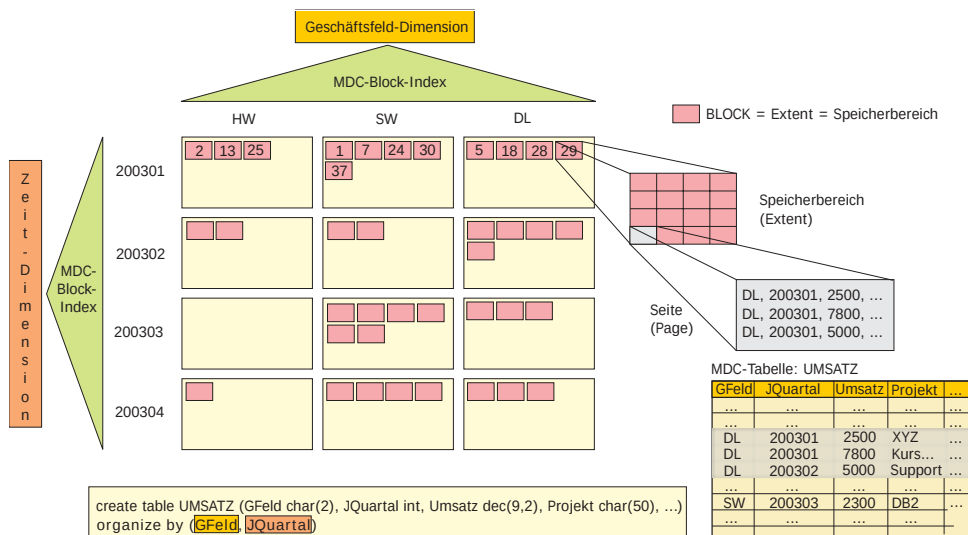


Abbildung 7.47: Mehrdimensionale Cluster-Tabelle – MDC

Da von DB2 sichergestellt wird, dass jeder Block (extent) nur eine eindeutige Kombination von Dimensionswerten beinhaltet, kann sehr schnell herausgefunden werden, welche Blöcke die gesuchten Daten enthalten. Bei einer Selektion

```
select * from UMSATZ where GFeld='SW' and JQuartal=200301
```

müssten nur die Extents 1, 7, 24, 30 und 37 (siehe Abbildung 7.47) gelesen werden.

Würde die WHERE-Klausel nur JQuartal=200301 lauten, müssten alle im Bild (oberste Reihe) mit Zahlen gekennzeichneten Blöcke gelesen werden. Es wären dann zwar

mehr Daten, aber garantiert ausschließlich die, die im SELECT gewünscht wurden. Also ein sehr effizienter Zugriff!

Abbildung 7.47 zeigt auch schon, dass, wenn ein einzelner Block für die Datenspeicherung nicht mehr ausreicht, einfach ein weiterer Block dieser Dimensionskombination zugeordnet wird. Block 2, 13 und 25 beinhalten alle dieselbe Dimensionskombination *Quartal 1 und HW-Geschäftsfeld*. Die Zahlensprünge zeigen auch an, dass Blöcke erst bei Bedarf hinzugefügt wurden.

Um nicht die Übersicht über die Blöcke zu verlieren, die zu einzelnen Dimensionskombinationen gehören, werden automatisch bei der Erstellung der MDC-Tabelle verschiedene Indizes erzeugt, die auf diese Blöcke verweisen. Diese nennt man deshalb auch *Blockindex*, weil sie auf Blöcke und nicht auf RIDs (Row Identifier) verweisen. Ansonsten sind es aber exakt dieselben Indexstrukturen, die wir schon kennen. Für jede Dimension wird so ein Blockindex angelegt; man spricht deshalb auch von einem *Dimensionsblockindex*.

Zusätzlich wird ein „zusammengesetzter Blockindex“ (composite block index) erstellt. Das ist ein Index, der alle Dimensionskombinationen aufweist, die in der Tabelle existieren. Diese Information könnte zwar durch eine Verknüpfung aller Dimensionsblockindizes ebenfalls herausgefunden werden. Ein extra Index ist aber besonders bei Einfügeoperationen für das schnelle Auffinden des „richtigen“ Blocks hilfreich. Dieser zusammengesetzte Dimensionsblockindex ist in Abbildung 7.47 nicht eingezeichnet.

Und nicht vergessen, das passiert alles automatisch – Sie müssen *nur* den ORGANIZE BY (siehe Abbildung 7.46) im CREATE TABLE-Befehl mit den entsprechenden Dimensionen angeben!

Noch einmal ein Wort zu Dimensionen: Folgende Punkte können bei der Dimensionsauswahl behilflich sein:

- ▶ Die einzelne Dimension sollte wenige verschiedene Ausprägungen haben, um letztendlich mehrere Zeilen in einem Block speichern zu können und um den Platz im Block (extent) zu nutzen und nicht zu verschwenden. Das bedeutet andererseits, dass „singuläre“ Spalten, wie z.B. Zeitstempel oder eindeutige Schlüssel, diesen Mechanismus ad absurdum führen und eine enorme Platzverschwendung zur Folge hätten.
- ▶ Typischerweise nehmen Sie Spalten oder Spaltenkombinationen als Dimension, die auch sonst in SQL-Abfragen häufig vorkommen.
- ▶ Letztendlich sind alle Spalten, die auch für einen Clustered Index in Frage kommen würden, gute Kandidaten für eine Dimension.
- ▶ Die Fremdschlüssel von Fakten-Tabellen im Data Warehouse-Umfeld sind klassische Kandidaten für Dimensionen.

Mit diesem Mechanismus *garantiert* DB2 das Clustering – etwas, was bei einem Clustered Index genau nicht der Fall war. Reorganisationen werden so nicht mehr benötigt, um die Daten korrekt anzuordnen. Der einzige Fall, in dem der Einsatz des REORG-Tools (siehe Kapitel 13) Sinn machen könnte, ist gegeben, wenn durch

größere Löschungen von Daten der Platz freigegeben werden soll. Ansonsten werden leere Blöcke von DB2 wiederverwendet.

Tipp:

Bei einer Speicherung von MDC-Tabellen in SMS-Tabellenbereichen ist der Einsatz des Tools db2empfa dringend anzuraten, da der Speicher blockweise benötigt wird und dieses Tool genau das für SMS-Tabellenbereiche aktiviert. Bei DMS-Tabellenbereichen gibt es nichts weiter zu beachten. Hinweise zum Einsatz des db2empfa-Tools finden Sie in Kapitel 5.9.5.

7.13 Pufferpools (bufferpool)

Pufferpools – den Datenbank-Cache – hatten wir schon im Zusammenhang mit den Tabellenbereichen (siehe Kapitel 5.2.3) angesprochen.

Dieser sehr leistungsfähige (Haupt-)Speicherbereich dient dazu, die Datenbankdaten zwischenspeichern, um sie für die Verarbeitungsanforderungen nicht immer aufwändig von der Platte lesen zu müssen.

Neben dem Standard-Pufferpool (IBMDEFAULTBP) können weitere angelegt werden; es muss aber immer mindestens einen Pufferpool für jede verwendete Seitengröße (pagesize) geben. Die Auswirkungen und auch eine „ideale“ Vorgehensweise beim Anlegen finden Sie in Kapitel 5.8.1 im Abschnitt „Größe der Speicherseiten“.

Für den Pufferpool gibt es einen eigenen Eintrag in der Objektbaumstruktur der Steuerzentrale unterhalb (bzw. innerhalb) der Datenbank.

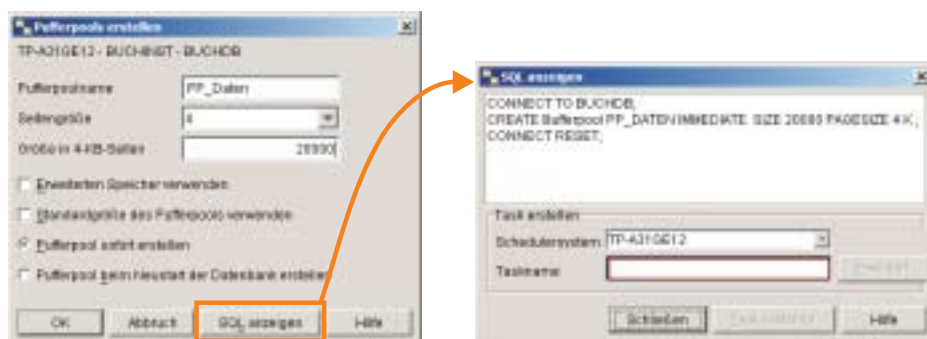


Abbildung 7.48: Pufferpool mit der Steuerzentrale erstellen

Das PUFFERPOOL ERSTELLEN-Fenster rufen Sie wie immer über das Kontextmenü auf. Als Eingabe sind nur ein Name und die Größe erforderlich. Mit DB2 V8 können Sie den Hauptspeicherbereich für den Pufferpool sofort (Standard) oder erst bei einem Neustart der Datenbank erstellen lassen. Die Option STANDARDGRÖSSE DES PUFFERPOOLS VERWENDEN bedeutet, dass an dieser Stelle keine Größenangabe erfolgt, sondern die Größe aus dem Parameter buffpage der Datenbankkonfiguration übernommen wird.

Hinweis:

Der DB-Konfigurationsparameter `buffpage` ist nur die *Vorgabe des Standardwerts* und hat *keinen* direkten Einfluss auf die Pufferpoolgrößen. Wenn Sie einen Pufferpool anpassen wollen, müssen Sie das über die Steuerzentrale oder den Befehl `ALTER BUFFERPOOL` tun und *nicht* über die Datenbankkonfiguration. Dies wird häufig verwechselt.

Das Syntaxdiagramm (siehe Abbildung 7.49) zeigt Ihnen noch einmal die wichtigsten Einstellungsmöglichkeiten. Optionen für partitionierte Umgebungen wurden nicht betrachtet.

Die Standard-Seitengröße ist 4096 Bytes. Die Seitengröße kann, im Gegensatz zur Pufferpoolgröße, später nicht mehr verändert werden. Ein Pufferpool, der in einer Größe von 100.000 angelegt wird, hat so standardmäßig 400.000 Kbyte (also ca. 400 Mbyte).

Die Seitengröße des Pufferpools muss immer mit der Seitengröße der Tabellenbereiche übereinstimmen, die diesen Pufferpool nutzen wollen.

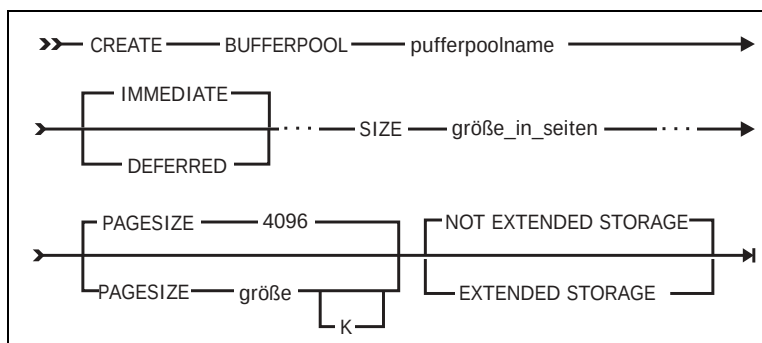


Abbildung 7.49: create bufferpool-Syntaxdiagramm

Informationen über definierte Pufferpools finden Sie – außer in der Steuerzentrale – in der Systemkatalog-Sicht `SYSCAT.BUFFERPOOLS`.

Hinweis:

Für Pufferpool-Verwaltungsstrukturen wird in DB2 ca. 1/40 des Speicherplatzes aller Pufferpools in der Datenbank benötigt. Diese Verwaltungsstrukturen werden im Datenbank-Zwischenspeicher `dbheap` abgelegt, der über die Datenbankkonfiguration angepasst werden kann. Beim Hinzufügen neuer Pufferpools oder bei der Erweiterung bestehender sollte der `dbheap`-Wert entsprechend angepasst werden.

7.13.1 Wie viele Pufferpools sollen angelegt werden?

Sie haben zwei grundlegende Optionen:

- ▶ einen großen Pufferpool
- ▶ mehrere kleinere Pufferpools

Dazu müssen wir uns erst einmal anschauen, welchen Vorteil überhaupt mehrere Pufferpools haben können.

Da grundsätzlich der verfügbare Speicherplatz limitiert ist, werden Datenseiten im Pufferpool der Datenbank, wenn der Pufferpool „voll“ ist, auf die Platten ausgelagert bzw. im Pool überschrieben. Jetzt kann es aber sein, dass Sie sicherstellen wollen, dass ganz bestimmte Daten *immer* im Hauptspeicher verbleiben. Das ist besonders bei Indexdaten der Fall, die ständig im Hauptspeicher zur Verfügung stehen sollen, um eine schnelle Ausführung der Abfragen zu gewährleisten. Die Tabellen selbst sind meist viel zu groß, um selbst komplett im Hauptspeicher gecacht zu werden. Die kleineren Indexstrukturen haben da Vorteile.

Denken Sie z.B. an die Supermarktkasse mit dem Strichcode-Scanner. Kaum ist der Artikel gescannt, wird auch schon der Preis angezeigt, und auch wenn ein Artikel einmal länger nicht verkauft worden ist, so soll sein Preis doch genauso schnell erscheinen. In diesem Fall können Sie es mit einem separaten Tabellenbereich und einem separaten Pufferpool für den Index vermeiden, dass Indexdaten aus dem Pufferpool „verdrängt“ werden.

Generelle Aussagen sind in diesem Bereich sehr schwierig. Bei transaktionsorientierten Systemen (OLTP) kann man mit Hilfe mehrerer Pufferpools einzelnen Anwendungen bzw. deren Tabellen mehr Speicher zur Verfügung stellen und sie so „priorisieren“. Dabei ist es dann aber wichtig, diese Pufferpools und ihre Auslastung (ständig) zu überwachen, um wirklich eine optimale Nutzung des Speicherplatzes sicherzustellen.

In großen Data Warehouse-Umgebungen kann durchaus ein großer Pufferpool vorteilhaft sein, da hier die verschiedenartigsten Anfragen ausgeführt werden und eigentlich alle viel Speicher benötigen, um performant zu laufen.

7.13.2 Wie groß soll ein Pufferpool werden?

Die Standardgröße von 1.000 Seiten unter Unix (und 250 Seiten bei Windows-Maschinen) ist sicherlich für alle Datenbanken – bis auf die SAMPLE-Datenbank – viel zu klein. Die Anpassungen sollten nach dem Motto „nicht kleckern, sondern klotzen“ durchgeführt werden. Bei den Speicherausstattungen heutiger Server sollten da schon 500 Mbyte als erster Ansatzpunkt möglich sein. Allerdings muss hier die Summe der Pufferpools aller Datenbanken betrachtet werden, und wenn der Server nicht als reiner Datenbank-Server gedacht ist, muss auch noch Hauptspeicher für andere Anwendungen freigehalten werden.

Mit dem ALTER BUFFERPOOL-Befehl können Sie die Größe der Pufferpools nachträglich – und über die IMMEDIATE-Option sogar „online“ – anpassen.

Hinweis:

Bei 32-Bit-Systemen gibt es eine Speicher-Obergrenze für die Pufferpools einer Datenbank, die bei AIX und Linux (Kernel 2.4 oder später) bei maximal 1,7 Gbyte, bei Solaris bei 3,35 Gbyte und bei Windows-Systemen (ohne AWE) bei 2 Gbyte liegt.

64-Bit-Systeme haben eine Begrenzung, die mit heutiger Hardware nicht erreicht wird, d.h., ihr physikalischer Speicher ist die Grenze.

7.14 Pakete (packages)

Da Sie Pakete auch in der Steuerzentrale unter ANWENDUNGSOBJEKTE/PAKETE aufgelistet bekommen, wollen wir Ihnen diese hier kurz vorstellen. Die Erstellung der Pakete wird aber erst im Anwendungsteil (Kapitel 11) behandelt.

Für die Erklärung von Paketen müssen wir etwas ausholen. In DB2 kann für Anwendungen so genanntes *statisches SQL* verwendet werden (siehe auch Kapitel 11). Dies ist dadurch gekennzeichnet, dass die Struktur des in der Anwendung abgesetzten SQL-Befehls schon bei der Programmierung feststeht. Darunter fallen z.B. die abzufragenden Tabellen und Spalten. Bestimmte Parameter des SQL-Befehls können dabei durchaus noch offen und durch eine Variable belegt sein, nur die Struktur muss feststehen. Ist das Datenbank-Programm fertig entwickelt, kann der DB2 Optimizer die im Programm enthaltenen SQL-Befehle schon analysieren und sich überlegen, wie er auf die Daten zugreifen würde.

Diese Überlegungen über die Zugriffswege werden in einem Paket in der Datenbank abgelegt – sozusagen als eine bei der Programmvorbereitung erstellte Steuerstruktur für die Ausführung von SQL-Anweisungen.

Bei der Ausführung eines (statischen SQL-)Programms werden die fertig berechneten Zugriffswege gleich aus dem Paket gelesen, und es entstehen keine Kosten für die Optimierung mehr.

Somit ergeben sich folgende Punkte für Pakete:

- ▶ Jedes Datenbankprogramm benötigt ein „zugehöriges“ Paket in der Datenbank.
- ▶ Das Paket hat standardmäßig den gleichen Namen wie die Anwendung.
- ▶ Ein Paket muss in jeder Datenbank abgelegt sein, die mit einer Anwendung lauffähig sein soll.

Über die Befehlszeile können die Pakete mit dem LIST PACKAGES-Befehl angezeigt werden. Dieser entspricht von der Syntax her dem LIST TABLES-Befehl.

Informationen über Pakete und damit über Anwendungen werden in der Datenbank u.a. in der Systemkatalog-Sicht SYSCAT.PACKAGES gespeichert.

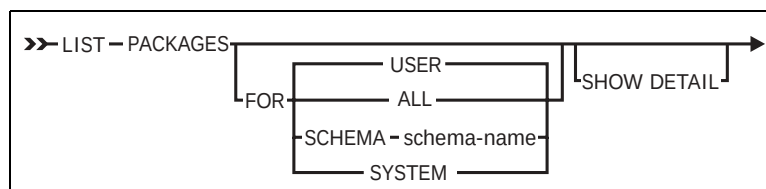


Abbildung 7.50: list packages-Syntaxdiagramm

7.15 Die Beispiel-Datenbank SAMPLE

Wenn Sie einfach ein paar Dinge ausprobieren wollen, ohne vorher lange Tabellen, Sichten, Indizes und die anderen Objekte anzulegen, können Sie in DB2 die SAMPLE-Datenbank erzeugen.

Die SAMPLE-Datenbank beinhaltet einige kleine Tabellen mit ein paar Datensätzen und ist für einfache Test- und Spielzwecke durchaus ausreichend.

Die SAMPLE-Datenbank kann über die grafische Oberfläche ERSTE SCHRITTE mit der Schaltfläche BEISPIELDATENBANK ERSTELLEN angelegt werden. Dadurch werden sowohl die Datenbank als auch die Tabellen und der Inhalt erzeugt.

Über die Befehlszeile kann die SAMPLE-Datenbank mit dem Befehl `db2samp1` generiert werden. Alternativ kann mit der Script-Datei `db2sampk.bat` aus dem Verzeichnis `/sql11b/samples/db2samp1` ebenfalls die SAMPLE-Datenbank, allerdings inklusive einiger Primärschlüsseldefinitionen, erstellt werden.

Viele Beispiele und Szenarien in den DB2-Handbüchern beziehen sich auf diese SAMPLE-Datenbank.

7.16 Tipps aus der Praxis

7.16.1 Das Tool db2look

Von einer bestehenden Datenbank können Sie sich die DDL-Befehle mit Hilfe des `db2look`-Tools generieren lassen.

Das Ergebnis wird Ihnen als DB2-Script zur Verfügung gestellt, das Sie bei Bedarf über `db2 -tf scriptname`

ausführen können. Damit haben Sie einerseits eine Art Dokumentation von Ihrer Datenbank erstellt, andererseits ist dieses Script auch praktisch, um eine Test-Datenbank mit der Struktur der Produktionsdatenbank erstellen zu lassen.

Der Aufruf über die Steuerzentrale erfolgt über den Eintrag DDL GENERIEREN ... im Kontextmenü der Datenbank.

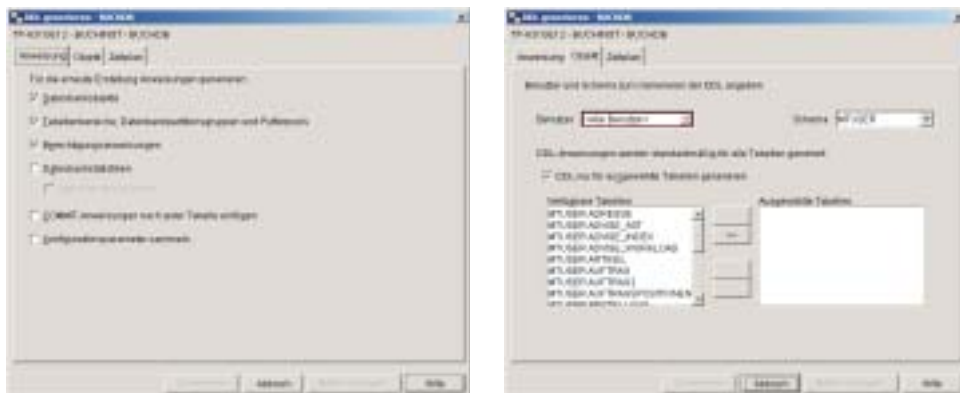


Abbildung 7.51: DDL über die Steuerzentrale generieren

Im DDL GENERIEREN-Fenster können Sie nun verschiedene Angaben machen, zu welchen Objekten die DDL erzeugt werden soll. Dazu stehen Ihnen die Registerkarten ANWEISUNG und OBJEKT zur Verfügung.

Die einzelnen Optionen werden anhand der in Abbildung 7.52 gezeigten Syntax erläutert.

db2look ist ein bei DB2 mitgeliefertes Programm. Beim Aufruf wird eine Verbindung zur angegebenen Datenbank hergestellt.

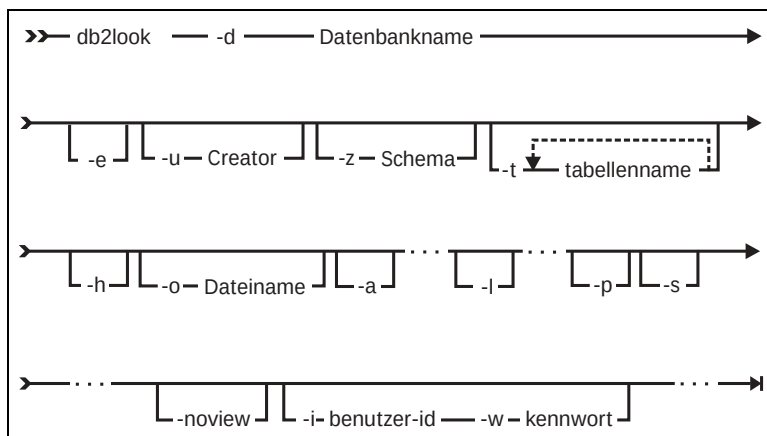


Abbildung 7.52: db2look-Syntaxdiagramm

db2look hat folgende Optionen:

- ▶ -d – gibt den Datenbanknamen an (auch OS/390-Datenbanken sind möglich).
- ▶ -e – steht für „extract DDL“.

Durch diese Option wird die DDL u.a. für folgende Objekte generiert: Tabellen, Sichten, MQTs, Aliasnamen, Indizes, Auslöser, UDTs, RI, UDFs, ...

- ▶ `-u` – Nur die DDL-Befehle der Objekte eines bestimmten „Erstellers“ werden extrahiert.
- ▶ `-z` – Nur die DDL-Befehle der Objekte eines bestimmten Schemas werden generiert.
- ▶ `-t` – Nur die DDL-Befehle der angegebenen Tabellen werden generiert.
- ▶ `-h` – zeigt die Hilfeseite auf der Befehlszeile an.
- ▶ `-o` – Hier wird der Name der Ausgabedatei angegeben (ohne Dateierweiterung).
- ▶ `-a` – Die DDL aller Objekte soll unabhängig vom Benutzer erstellt werden.
- ▶ `-l` – erzeugt die DDL für Tabellenbereiche, Partitionsgruppen und Pufferpools.
- ▶ `-p` – Das Ergebnis wird als Textdatei erstellt (.txt).
- ▶ `-s` – Das Ergebnis wird als Postscript-Datei erstellt (Zusatzsoftware notwendig).
- ▶ `-noview` – Sichten-Definitionen werden nicht generiert.
- ▶ `-i` und `-w` – Benutzername und Passwort, falls es sich um eine entfernte Datenbank handelt.

Sie sehen also, dass Ihnen über die Befehlszeile noch mehr Optionen zur Verfügung stehen als in der grafischen Oberfläche. Einige Optionen haben auch wir hier weggelassen, da sich diese auf zusammengeschlossene Systeme beziehen und hier zu weit führen würden. Die Option `-m` dient dazu, Statistikinformationen einer Datenbank in ein Script zu speichern bzw. so eventuell in eine andere Datenbank zu übertragen.

Ein Beispielaufwurf dieses Programms zur DDL-Erstellung aller Objekte aus der Datenbank BuchDB würde so aussehen:

```
db2look -d buchdb -e -o ddl_buchdb -a -l
```

7.16.2 Der Designadvisor

Ein so genannter Designadvisor soll dem Datenbankadministrator helfen, die richtigen Entscheidungen im Hinblick auf die Objekterstellung zu treffen.

Dabei ist momentan die Empfehlung auf Indizes beschränkt, d.h., Sie bekommen Hinweise, welche Indizes Vorteile für Ihre Zugriffe bringen würden. In der letzten DB2-Version hieß dieses Tool noch „Index-Advisor“. Die Umbenennung zeigt, dass hier noch einiges an Funktionalität zu erwarten ist. Später werden Sie vielleicht über diese Schnittstelle auch Empfehlungen für MQTs oder MDCs bekommen.

Die Idee ist es, Ihre so genannte „workload“, d.h. Ihre typischen Abfragen, zu analysieren und mit Hilfe des Optimizers und der vorhandenen Statistiken und Datenmengen die entsprechenden Empfehlungen zu geben.

Der Aufruf erfolgt über das Kontextmenü der Datenbank unter dem Eintrag DESIGN-ADVISOR... .



Abbildung 7.53: Designadvisor – Schritt 1 – Einführung

Nach dem Einführungsfenster (siehe Abbildung 7.53) wird in Schritt 2 (siehe Abbildung 7.54) die so genannte *Auslastung* angezeigt. Sie kann dann dort editiert werden.

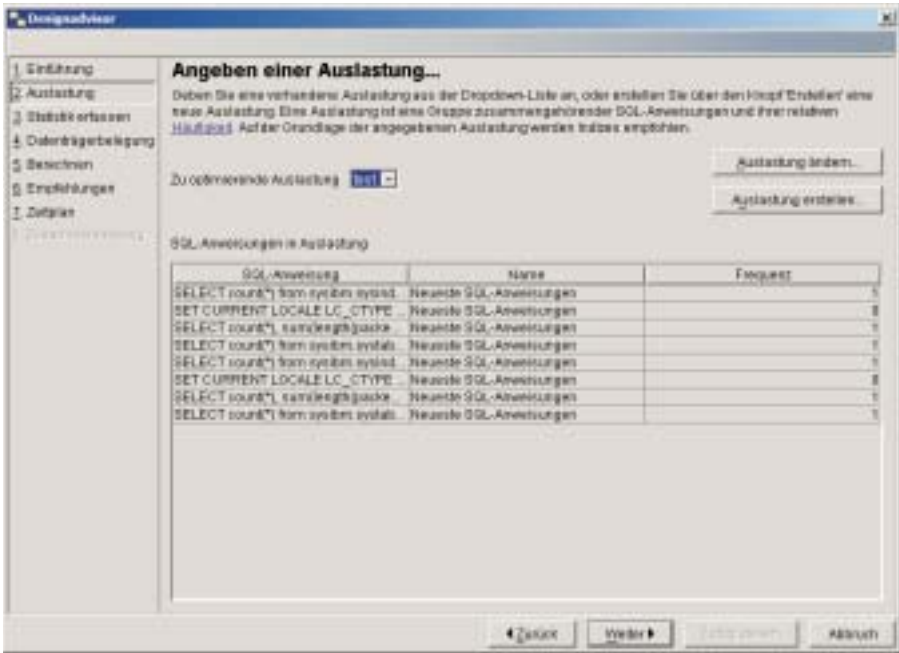


Abbildung 7.54: Designadvisor – Schritt 2 – Auslastung

Unter „Auslastung“ versteht man die Art und die Häufigkeit, in der SQL-Befehle in einem bestimmten Zeitraum ausgeführt werden.

Diese Auslastung können Sie hier manuell hinzufügen oder korrigieren bzw. durch eine Angabe der Häufigkeiten auch gewichten. Sie haben aber auch die Möglichkeit, sich die an die Datenbank gesandten SQL-Befehle über einen Monitor zusammenstellen zu lassen. Für Programme mit statischem SQL finden Sie die abgesetzten Befehle auch in der Systemkatalog-Sicht SYSCAT.STATEMENTS.

Für die Speicherung seiner Informationen legt der Designadvisor automatisch zwei Tabellen an: ADVISE_WORKLOAD und ADVISE_INDEX. Die ADVISE_WORKLOAD-Tabelle enthält die SQL-Befehle und die ADVISE_INDEX-Tabelle die Indexempfehlungen.

In den weiteren Schritten des Designadvisors haben Sie noch die Möglichkeit, die Statistik-Informationen aktualisieren zu lassen und den Ausführungszeitpunkt anzugeben. Auf der Empfehlungsseite werden Ihnen die Indizes angezeigt, die nach „Meinung“ des Designadvisors erstellt werden sollten.

Über die Befehlszeile kann diese Funktionalität mit dem db2advise-Befehl aufgerufen werden.

8 Datenbewegungen

In diesem Kapitel zeigen wir Ihnen, wie die Daten in die Datenbank eingelesen und aus ihr ausgelesen werden. Zu diesem Zweck bedienen wir uns der Standard-Tools von DB2, des *Export*-, des *Import*- und des *Load*-Tools. Das Export-Tool wird für das „Herausholen“, also das Extrahieren, der Daten aus der Datenbank verwendet. Hierbei kann die Auswahl gezielt eingeschränkt werden. Mit dem Import-Tool werden Daten in eine bestehende Datenbank eingefügt, also „importiert“. Das dritte Tool, das Load-Tool, wird auch zum Einfügen der Daten in die Datenbank verwendet. Es weist aber einige Unterschiede in der Funktionsweise und auch in der Performance auf.

Werden Daten aus einer Anwendung, z.B. über eine Dialogmaske, in die Datenbank eingebracht, also eingefügt oder verändert, so wird dies über die SQL-Befehle `INSERT INTO` oder `UPDATE` durchgeführt.

Woher kommen die Daten?

In der Praxis gibt es einige Möglichkeiten, z.B.:

- ▶ Die Daten wurden mit dem Export-Tool aus einer anderen Datenbank extrahiert.
- ▶ Spezielle Programme extrahieren Daten aus anderen Datenhaltungssystemen und stellen diese in einer Datei zur Verfügung.
- ▶ Über einen Zugriff auf entfernte oder lokale Datenbanken können Daten auch direkt über einen SQL-Befehl sowie auch über einen Load-Befehl in eine Datenbank kopiert werden.

8.1 Datenformate

Daten, die Sie „entladen“, also extrahieren, um sie später in eine andere Datenbank einzufügen, sollten ein bestimmtes DB2-Datenformat haben. DB2 nutzt dieses Format, um die zu importierenden Daten eindeutig den zugehörigen Spalten zuzuordnen zu können.

DB2 unterstützt folgende Formate für die Dienstprogramme Export, Import und Load:

- ▶ ASC – ein ASCII-Format mit fixer Länge
- ▶ DEL – ein ASCII-Format mit Zeichenbegrenzer
- ▶ WSF – ein Arbeitsblattformat für Lotus- und Symphony-Produkte
- ▶ IXF – die PC-Version des *Integrated Exchange Format* (IXF)

ASC – ein ASCII-Format mit fixer Länge

Dateien ohne einen Zeilenbegrenzer (delimiter) werden auch als „ASCII-Dateien mit fester Länge“ bezeichnet. Die Spaltenbreite muss in der Eingabedatei immer gleich groß sein und darf nicht durch ein Trennungszeichen separiert sein. In unserem Beispiel reichen die Spalte 1 von Stelle 0 bis Stelle 3, Spalte 2 von Stelle 4 bis 16 und Spalte 3 von 17 bis 20.

Beispiel:

```
012345678901234567890
0010Grodtk     0003
0011Tiefenbacher 0003
00090rhanovic   0004
```

Dieses Format wird in der Praxis selten verwendet.

DEL – ein ASCII-Format mit Zeichenbegrenzer

Bei dem am häufigsten verwendeten Format – dem DEL-Format – werden einzulesende Spalten voneinander durch ein Trennungssymbol (hier: Komma) getrennt und Zeichenketten (strings) mit extra Begrenzungszeichen (hier: doppelte Anführungszeichen) markiert.

Beispiel:

```
10,"Grodtk",3
11,"Tiefenbacher",3
9,"Orhanovic",4
```

Die Markierungszeichen können hierbei individuell vereinbart werden:

► Spaltenbegrenzung

Mit der Option COLDEL wird ein Zeichen (hier: |) gewählt, das die Spalten voneinander trennt. Als Standard wird in DB2 das Komma (,) eingesetzt. Wollen Sie dies ändern, so müssen Sie folgende Option bei den Tools angeben:

```
.. modified by coldel|
```

Dazu muss die Eingabedatei wie folgt aussehen:

```
10|"Grodtk"|3
11|"Tiefenbacher"|3
9|"Orhanovic"|4
```

► Zeichenfolgebegrenzung (character delimiter)

Mit der Option CHARDEL wird ein Zeichen (hier: \$) als Zeichenbegrenzer zugewiesen, das eine Zeichenfolge als zusammengehörig eingrenzt. Notwendig ist das bei Texten, die ein Leerzeichen bzw. Sonderzeichen enthalten, um eindeutig die Grenzen der Spalte aufzuzeigen.

Die Standardeinstellung für das Trennungszeichen ist ein doppeltes Hochkomma ("). Wollen Sie diese Standardeinstellung ändern, so tun Sie das mit folgendem Zusatz:

```
.. modified by chardel$
```

Die Datei könnte wie folgt aussehen :

```
10|$I. Grodtke$|3  
11|$M. Tiefenbacher$|3  
9|$J. Orhanovic$|4
```

► Zeilenbegrenzungen

Die Trennungszeichen für die Zeilenbegrenzung können nicht vereinbart werden, sondern werden von Standards des Betriebssystems vorgegeben. Diese sind bei Windows-Systemen 'X'0D0A' (Rücklauf+Zeilenvorschubzeichen) und bei Unix-Systemen 'X'0A' (Zeilenvorschubzeichen).

Hinweis:

Bei der Angabe der Optionen CHARDEL und COLDEL darf vor dem Markierungszeichen kein Leerzeichen stehen!

WSF – Arbeitsblattformat für Lotus- und Symphony-Produkte

Dieses Arbeitsblattformat wird von den Produkten Lotus 1-2-3 und Symphony zum Datenaustausch genutzt. Das Load-Tool unterstützt dieses Format nicht.

IXF – PC-Version des „Integrated Exchange Format“

Das IXF-Format ist das beste Format, um Daten zwischen Plattformen zu transportieren, da DB2 intern die Daten immer im Windows-Format abspeichert. Dies wird auch auf Unix-Systemen durchgeführt bzw. werden dort die Daten automatisch in dieses Format konvertiert.

Zusätzlich zu den exportierten Daten enthält eine Datei im IXF-Format die Beschreibung der Spalten- und Indexdefinitionen.

8.2 Export-Tool

Das Export-Tool gibt Ihnen die Möglichkeit, benötigte Daten aus einer DB2-Datenbank zu extrahieren. Die Daten werden über eine SQL-Abfrage zusammengestellt und in eine Ausgabedatei geschrieben. In der SQL-Abfrage können Sie alle Möglichkeiten von SQL nutzen. Sie können auf diese Weise eine Kopie der Tabelle (select * from <tabellen_name>) erstellen oder durch Selektionen und Berechnungen individuelle Zusammenstellungen erzeugen, die auch mehrere Tabellen und Sichten umfassen können.

Die Form der Ausgabedatei kann über die Datenformate (siehe Abschnitt 8.1 „Datenformate“) und weitere Optionen beeinflusst werden.

8.2.1 Export über die Steuerzentrale

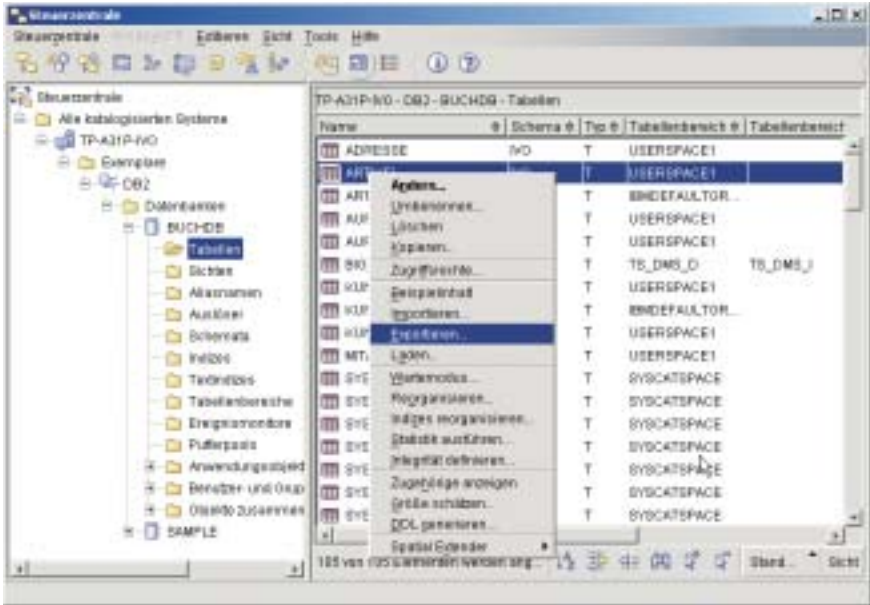


Abbildung 8.1: Aufruf des Exports aus der Steuerzentrale

Obwohl Sie die Daten über einen SQL-Befehl zusammenstellen können, wird das Fenster des Export-Tools über das Kontextmenü einer Tabelle aufgerufen.

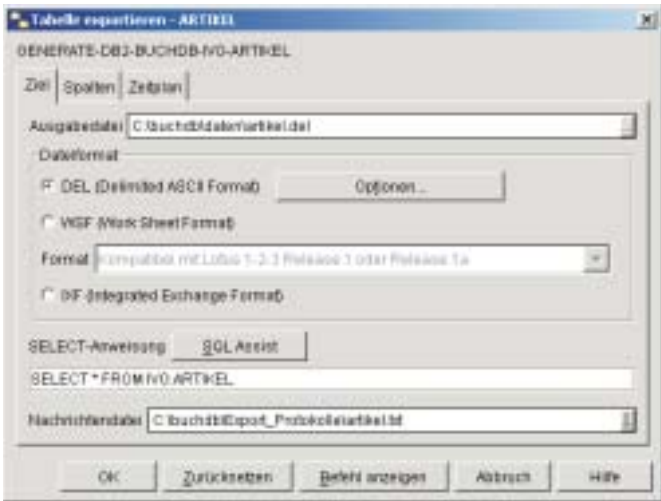


Abbildung 8.2: Tabelle exportieren – Optionen

In der ersten Auswahl (siehe Abbildung 8.2) geben Sie die Pfade der Ausgabedatei sowie der Nachrichtendatei, in der die Fehlermeldungen stehen, an. Im Feld SELECT-ANWEISUNG können Sie mit Hilfe des SQL-Assistenten einen SQL-Befehl für die Auswahl der Ausgabedaten erstellen oder diesen SQL-Befehl direkt eintippen. Standardmäßig ist ein SQL-Befehl vorbereitet, um alle Felder aus der Tabelle zu übernehmen. Beim Export in das Dateiformat DEL (Delimited ASCII Format) können über den OPTIONEN...-Button weitere Optionen angegeben werden, die aber schon mit Standardwerten vorbelegt sind.

Tipp:

Wir empfehlen, das Exportformat mit in den Namen der Exportdatei einfließen zu lassen, da DB2 keine automatischen Dateierweiterungen vergibt. Beispielsweise helfen später <tabellenname>.del oder <dateiname>_del bei der Erkennung der Datenformate.

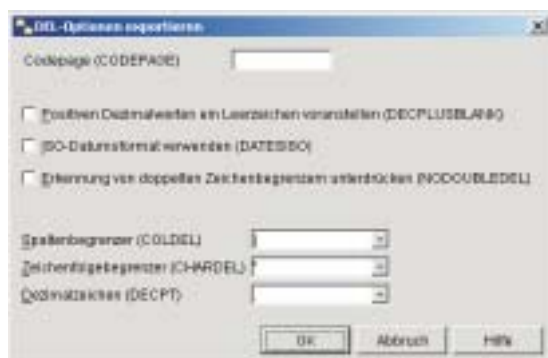


Abbildung 8.3: Exportoptionen

► CODEPAGE

Wenn Sie die exportierten Daten später in einer anderen Sprachumgebung bzw. unter einer anderen Codepage einsetzen wollen, kann dies hier schon berücksichtigt werden. DB2 konvertiert dann die Daten in die hier angegebene Codepage, z.B. in 850.

Als Standardwert wird die Codepage der Datenbank genommen. In unserem Beispiel würde die Codepage 1252 verwendet werden.

► POSITIVEN DEZIMALWERTEN EIN LEERZEICHEN VORANSTELLEN (DECPLUSBLANK)

Dies bewirkt, dass in der Ausgabedatei den Dezimalspalten jeweils ein Leerzeichen vorangestellt wird, z.B.:

4,501000, 000100.00,"mit DECPLUSBLANK"

Ohne Verwendung dieser Option hätten wir folgende Ausgabe:

4,501000,000100.00,"ohne Option"

Sobald jedoch eine Spalte einen negativen Wert hat, wird dieser wie folgt angezeigt:

```
5,5020000,000100.00,"ohne Option mit positivem Wert"
5,5020000,-000100.00,"ohne Option mit negativem Wert"
```

Wird diese Option beim Export-Befehl angegeben, so geht dies wie folgt:

```
EXPORT .... MODIFIED BY DECPLUSBLANK
```

► ISO-DATUMSFORMAT VERWENDEN (DATEISO)

Hier wird ein Datum in das ISO-Format umgewandelt. Das ISO-Format hat immer die Darstellung JJJJ-MM-TT. Ein Beispiel mit ISO- und ohne ISO-Angaben:

```
1,"Herr","Kolb ","Hans","1940-12-24","mit ISO-Option"
1,"Herr","Kolb ","Hans","19401224","ohne Angaben"
```

► SPALTENBEGRENZER (COLDEL)

Diese Auswahlliste hilft bei der Spaltenbegrenzung und definiert, welches Zeichen als Trennungssymbol für die Spaltentrennung genommen wird. Der Standardwert ist ein Komma. Dies kann nicht nur in unseren Regionen zu Problemen führen. Wird das Komma in Zahlen oder Texten verwendet, kann eventuell das nachfolgende Import- oder Load-Tool die Spalten nicht mehr eindeutig zuweisen. Es entstehen viele Fehler durch Inkompatibilitäten der Felder, z.B.:

```
1,"Jens",2034,12,"Texte und Zahlen mit , Komma im Text"
```

Hier wurden sechs Spalten erkannt!

Besser ist es, die Trennzeichen eindeutig zu definieren, also Zeichen zu verwenden, die so nicht im Text bzw. in der Import- bzw. Exportdatei vorkommen, z.B.:

```
1|"Jens"|2034,12|"Texte und Zahlen mit , Komma im Text"
```

In der Praxis kommt es nun vor, dass auch diese Sonderzeichen in Texten verwendet werden. Die einzige Lösung ist dann, einen Hex-Code von einem Sonderzeichen auszusuchen, das nicht auf den Tastaturen vorkommt, und dieses hier anzugeben, z.B.:

```
x23 oder 0x23 = #
xEf oder 0xEf = ï
```

Die Eingabeoption beim EXPORT- bzw. auch beim IMPORT-Befehl ist:

```
EXPORT .... MODIFIED BY COLDELxEf
```

► ZEICHENFOLGEBEGRENZER (CHARDEL)

Der Standard für den Zeichenfolgebegrenzer ist das doppelte Hochkomma (""). Auch hier ist darauf zu achten, dass dieses Zeichen nicht im Text vorkommt, um falsche Zuordnungen beim späteren IMPORT oder LOAD zu vermeiden.

Beim Einsatz im Befehl ist darauf zu achten, dass zwischen der Option und dem Trennungssymbol kein Zeichen – also auch kein Leerzeichen – steht.

► DEZIMALZEICHEN (DECPT)

Je nach beabsichtigter Weiterverarbeitung kann hier das Zeichen für den Dezimalpunkt gesetzt werden, das in der Ausgabedatei für die Abtrennung der Nachkommastelle verwendet wird. Wie der Name „Nachkommastelle“ schon sagt, sollte hier also ein Komma stehen. Die Standardeinstellung kommt aber aus dem Amerikanischen und ist der Punkt (.). Diese Einstellung kann oft vernachlässigt werden, da sie beim Import-/Load-Tool wieder als Option mitgegeben werden kann.



Abbildung 8.4: EXPORT-Befehl

Diese DB2-Befehle werden von DB2 generiert und ausgeführt. Sie sehen in diesem Befehl alle Optionen, die Sie eingegeben haben, und zusätzliche Befehle, die DB2 benötigt, um diesen Export durchzuführen.

8.2.2 Exportieren per DB2-Befehl

Um mit dem Export-Tool Daten aus der Datenbank herauszuholen, müssen Sie zuerst mit der Datenbank verbunden sein. Erst im zweiten Schritt geben Sie den EXPORT-Befehl ein.

1. connect to <dbname>
2. export to ...

Das folgende Syntaxdiagramm (siehe Abbildung 8.5 auf der nächsten Seite) zeigt die wichtigsten Optionen des EXPORT-Befehls auf. Hier sehen Sie, dass die Datei für die Meldungen eigentlich optional ist.

Somit lautet der einfachste EXPORT-Befehl:

```
EXPORT to mitarbeiter.del OF del SELECT * FROM MITARBEITER
```

METHOD N ist eine Option, die nur Sinn macht, wenn Sie die Daten im IXF-Format exportieren und dabei den Spaltennamen verändern möchten. Dies ist sehr sinnvoll, wenn Sie Daten aus mehreren Spalten erzeugen und für diese „sprechende“ Namen vergeben wollen, z.B.:

```
EXPORT ..METHOD N('BONUS', ...).. SELECT MoGehalt*2,... FROM ..
```

Auch das Exportieren von LONG-Objekten ist möglich, doch es ist mit einigen Einschränkungen verbunden.

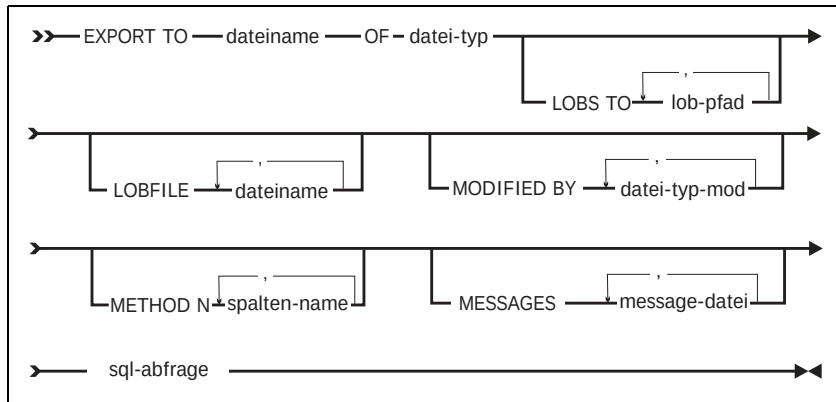


Abbildung 8.5: export-Syntaxdiagramm

Sind die LONG-Daten kleiner als 32 Kbyte, können sie mit dem normalen EXPORT-Befehl ohne jegliche Zusatzoption in die normale Exportdatei mit exportiert werden. Sind die LONG-Daten aber größer, werden sie abgeschnitten.

Sie können dies vermeiden, indem Sie die Daten in eine separate Datei ausgeben. Dazu müssen Sie folgende Optionen angeben bzw. hier noch einiges beachten.

Für die Ausgabe in extra Dateien müssen Sie MODIFIED BY LOBSINFILE als Option angeben und die LOBS TO-Option, um den oder die Pfade zu bestimmen, in die die LOBS geschrieben werden sollen. Werden mehrere Pfade mitgegeben, so werden diese in der angegebenen Reihenfolge beschrieben.

Um die Namen für die LONG-Objekte zu vergeben, steht die LOBFILE-Option zur Verfügung. Der Name, der hier eingesetzt wird, wird vom Export-Tool noch um eine dreistellige Nummer ergänzt, z.B.:

```
EXPORT TO Mitarbeiter_Bild.del OF DEL LOBS TO
E:\lobdir1\F:\lobdir2\
LOBFILE MA_Bild ..... MODIFIED BY LOBSINFILE
SELECT * FROM Mitarbeiter
```

Die Dateien der LONG-Export-Objekte würden wie folgt aussehen:

```
E:\lobdir1\MA_Bild.001
E:\lobdir1\MA_Bild.002
E:\lobdir1\MA_Bild.003
```

Das Verzeichnis F:\lobdir2\ wird erst beschrieben, wenn das erste Verzeichnis bzw. der erste Pfad voll ist.

8.3 Import-Tool

Das Import-Tool wird eingesetzt, um Daten in die Datenbank einzufügen. Ein Import bringt die Datensätze durch INSERT-Operationen in die Datenbank – also

eine Art Massen-Insert. Der Import-Befehl kann in eine bestehende Tabelle oder Sicht Daten einfügen bzw. hat Optionen, um auch bestehende Daten zu verändern. Voraussetzung ist, dass die Daten in einem bestimmten Format – ASC, DEL, WSF oder IXF – vorliegen und Ihnen dieses Format auch bekannt ist, damit Sie es bei den Input-Optionen berücksichtigen können. Wichtig an dieser Stelle ist, dass Sie die einzufügenden Daten von ihrer Struktur her kennen und Sie somit auch sicher sind, dass die „richtigen“ Spalten aus der Eingabedatei in die dafür vorgesehenen Spalten in der Zieltabelle kommen.

8.3.1 Import per DB2-Befehl

Der Import-Befehl hat folgende Optionen, die in Abbildung 8.4 auf das Wesentliche gekürzt wurden.

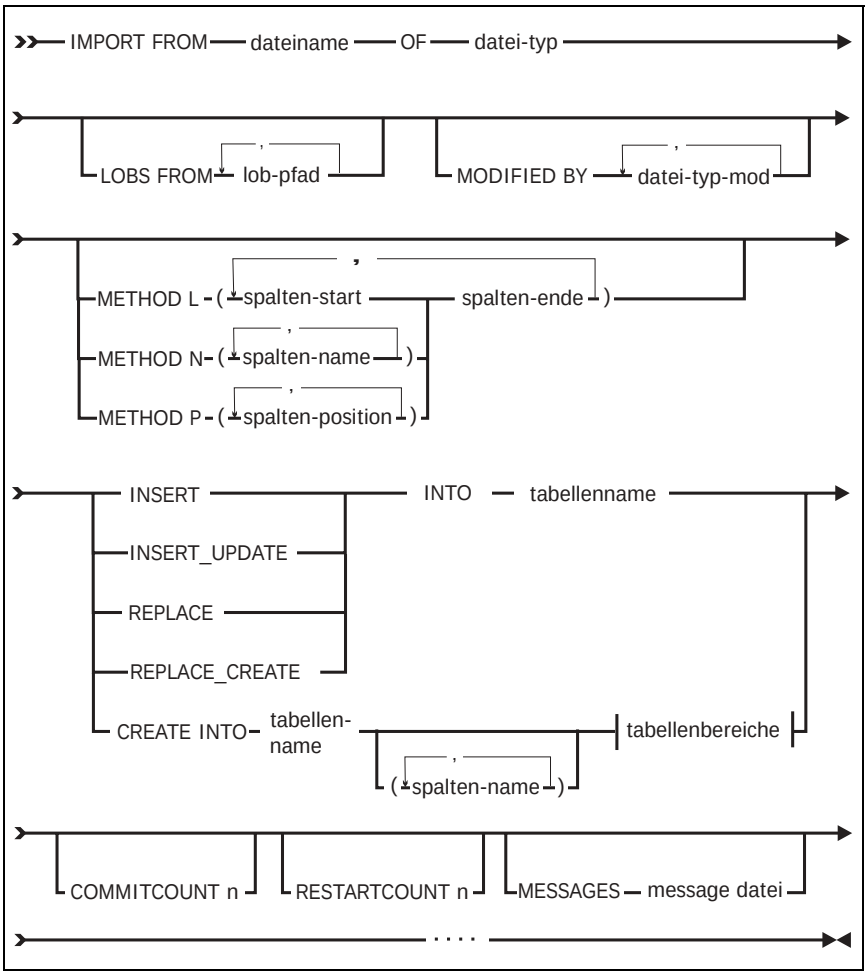


Abbildung 8.6: import-Syntaxdiagramm

Der einfachste IMPORT-Befehl benötigt nur die Input-Datei, das Format, in dem die Input-Datei vorliegt, also ASC, DEL, WSF oder IXF, und den Namen der Tabelle, in die die Daten eingefügt werden sollen.

```
IMPORT FROM inputdatei.del OF DEL INSERT INTO <tabellen-name>
```

Die weiteren Optionen dienen dem DB2-Administrator zur Anpassung an unterschiedliche Gegebenheiten, z.B. zur Anpassung an verschiedene Datenformate. Sie bieten ihm auch die Möglichkeit, Daten zuzuordnen oder die Datenbank im Falle eines Abbruchs wiederaufzusetzen.

Spaltenzuweisung abhängig vom Datenformat ändern

Sie haben die Möglichkeit, die Reihenfolge der Eingabespalten zu ändern. Dies geht über so genannte *Methoden*. Es gibt hier drei Methoden, die Sie abhängig von der Art der Eingabedatei einsetzen können.

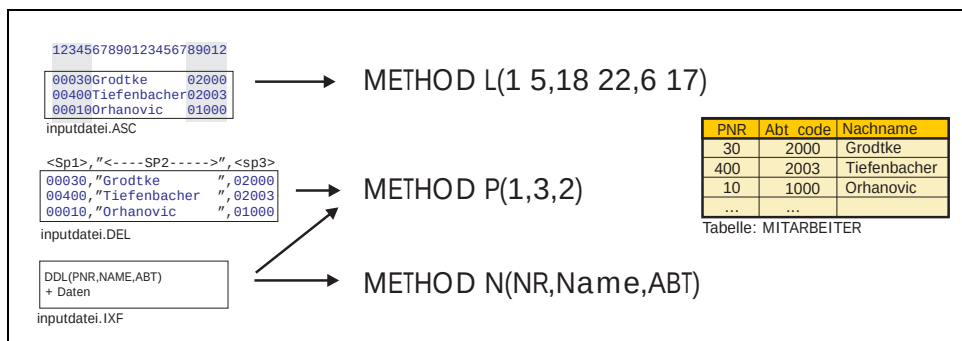


Abbildung 8.7: IMPORT-Option: METHOD

Die Option METHOD L wird beim EXPORT-Befehl eingesetzt, wenn Ihre Eingabedatei im Datenformat ASC ist, also die Eingabedaten ohne Trennungssymbole vorliegen. Mit ihr können Sie die Stellen angeben, die zu einer Spalte gehören bzw. ihr zugeordnet werden sollen. Sie beschreiben, „von welcher“ bis „zu welcher“ Stelle eine Spalte reicht; z.B. beginnt die erste Spalte (PNR-Spalte) bei Position 1 und geht bis zur Position 5, die zweite Spalte (Abt_code) geht von 18 bis 22. Über diese Methode weisen Sie also die Felder sequenziell den Spalten in der Tabelle zu.

Die Methode METHOD P kann für die Datenformate DEL und IXF eingesetzt werden. In diesen Formaten sind die Spaltenpositionen bekannt, somit kann direkt auf diese Spalten Bezug genommen werden. Um also aus der Inputdatei nur die Spalten 1, 2 und 3 in die Tabelle MITARBEITER zu importieren, muss die Reihenfolge geändert werden: METHOD P(1,3,2). Auch ein Import nur bestimmter Spalten der Eingabedatei wäre so möglich.

Die Methode METHOD N können Sie nur für das Datenformat IXF einsetzen, da nur im IXF-Format die Spalten mit Namen bekannt sind; sonst ist die Vorgehensweise wie bei METHOD P.

Importverfahren

Beim Einfügen der Daten in die Tabelle oder Sicht gibt es mehrere Verfahren, wie die Daten behandelt werden, die schon in der Tabelle vorhanden sind.

Importverfahren	Unterstützte Datenformate
INSERT INSERT_UPDATE REPLACE	IXF, ASC, DEL, WSF
REPLACE_CREATE CREATE	IXF

Tabelle 8.1: Mögliche Datenformate für die verschiedenen IMPORT-Verfahren

Diese Importverfahren funktionieren wie folgt:

- INSERT – fügt neue Sätze in die Tabelle ein, ohne bestehende Sätze zu verändern. Bei diesem Verfahren werden die Spalten, falls Sie keine Anpassung über die METHOD-Option vornehmen, in der Reihenfolge übernommen, wie sie in der Eingabedatei vorliegen.

```
IMPORT ... INSERT INTO ARTIKEL
```

Bei Verwendung einer IXF- oder DEL-Eingabedatei können Sie zusätzlich die Spaltennamen mitgeben, um so eventuelle Reihenfolgeunterschiede zwischen Tabelle und Eingabedatei abzufangen.

```
IMPORT ... INSERT INTO ARTIKEL(PNR,ABT,NAME)
```

- INSERT_UPDATE – fügt neue Sätze in eine bestehende Tabelle ein oder verändert vorhandene Datensätze. Besteht ein eindeutiger Index auf die Tabelle, werden die bestehenden Datensätze verändert, falls Daten mit dem gleichen Schlüssel importiert werden. In allen anderen Fällen werden die Daten wie bei der INSERT-Option einfach hinzugefügt.
- REPLACE – Mit dieser Option werden die bestehenden Daten in der Tabelle erst gelöscht, bevor die neuen Daten durch den IMPORT eingefügt werden. Diese Option ist nur gültig, wenn keine referenzielle Abhängigkeit zu dieser Tabelle besteht.
- REPLACE_CREATE – Diese Option wird nur unterstützt, wenn die Eingabedatei im IXF-Format vorliegt, denn nur in diesem Format sind die notwendigen Informationen gegeben, um eine neue Tabelle eventuell sogar mit dem zugehörigen Index anzulegen. Bestehen die Tabelle und der Index, so werden nur die vorhandenen Daten gelöscht und die neuen eingefügt. In diesem Fall entspricht diese Option der Option REPLACE. Existiert die Tabelle nicht, wird sie durch die Definitionen in der IXF-Datei automatisch angelegt und gefüllt (siehe auch CREATE).

- ▶ CREATE – legt die Tabelle an und füllt sie mit den Eingabedaten. Die Informationen für die DDL-Befehle können nur aus einer Inputdatei im IXF-Format ausgelesen werden.
- ▶ IN <tabellenbereichsname> – gibt an, in welchem Tabellenbereich die Tabelle neu erstellt werden soll.
- ▶ INDEX IN <tabellenbereichsname> – gibt den Tabellenbereich für den Index an.

Beide Optionen gelten in gleicher Weise auch für den Create-Fall der Option REPLACE_CREATE.

Protokollplatzbelegung und Wiederaufsetzpunkte

Das Import-Tool ist eine Anwendung, die alle Einfügungen in einer UOW (Unit Of Work), also einer Transaktion, durchführt. Dies kann bei sehr großen Imports bzw. sehr kleinen Protokolldateien zu Problemen führen, da DB2 oft alle konfigurierten Protokolldateien benötigt, um diese (IMPORT-)Transaktion zu protokollieren. Wird keine weitere Option angegeben bzw. sind die Protokolldateien zu klein angelegt, führt dies oft zu einem Abbruch dieses Importprozesses. Dies kann dadurch vermieden werden, dass die Protokolldateien an die benötigte Größe angepasst werden bzw. die Anzahl der „sekundären“ Protokolldateien an diese Situation angepasst wird. Weitere Informationen zu den Protokolldateien finden Sie in Kapitel 9.

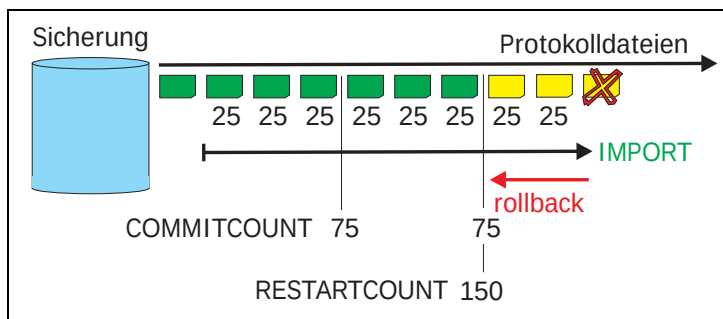


Abbildung 8.8: Import mit COMMITCOUNT und RESTARTCOUNT

Sie haben auch beim Import-Tool Optionen, die ein Überlaufen der Protokolldateien verhindern können (siehe Abbildung 8.8). Mit der COMMITCOUNT-Option geben Sie an, nach welcher Anzahl von veränderten bzw. geschriebenen Sätzen DB2 ein COMMIT, also eine Bestätigung der bisherigen Veränderungen, durchführen soll. Diese Veränderungen werden nun in der Datenbank festgeschrieben und werden im Falle eines Abbruchs nicht mehr zurückgenommen.

```
IMPORT ....COMMITCOUNT 75 ....
```

Erfolgt doch ein Abbruch, werden alle Veränderungen bis zum letzten COMMIT-Punkt zurückgenommen (undo), und Sie müssen das Import-Tool wieder aufsetzen.

Wo ist der Aufsetzpunkt? Müssen nun alle Daten neu verarbeitet werden, müssen also alle Daten von neuem gelesen werden? Das sind typische Fragen, die an dieser Stelle gestellt werden.

Doch wenn Sie eine Message-Datei beim IMPORT mit angegeben haben, z.B.

```
IMPORT .... MESSAGES import_msg.txt ....
```

dann haben Sie eine Datei, in der alle Nachrichten und Fehler festgehalten werden, und können hier nachsehen, wie viele Datensätze verarbeitet worden sind. Somit haben Sie auch gleich den Wert für den Aufsetzpunkt. Wird diese Datei nicht mit angegeben, wird die Anzeige auf den Bildschirm umgeleitet.

Der Inhalt der Messages-Datei import_msg.txt sieht wie folgt aus:

```
SQL3109N Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei
"C:\buchdb\artikel_neu.del".
SQL3221W ...COMMIT WORK beginnt. Zähler für Eingabesätze: "75".
SQL3222W ...COMMIT für alle Änderungen der Datenbank wurde erfolgreich ausgeführt.
SQL3221W ...COMMIT WORK beginnt. Zähler für Eingabesätze: "150".
SQL3222W ...COMMIT für alle Änderungen der Datenbank wurde erfolgreich ausgeführt.
SQL3005N Verarbeitung unterbrochen
SQL3110N Die Verarbeitung des Dienstprogramms ist abgeschlossen.
Es wurden "150" Zeile(n) aus der Eingabedatei gelesen.
```

Sie sehen, dass hier nach 75 Eingabesätzen ein COMMIT erfolgt und dieser auch durch die Meldung: SQL3221W bestätigt wird. Erfolgt ein Abbruch, wird die Meldung SQL3005N Verarbeitung unterbrochen hinzugefügt.

Wollen Sie nun den Import wieder aufsetzen, so ist die RESTARTCOUNT-Option wichtig. Mit dieser geben Sie die *Start*-Position für das Import-Tool an. DB2 nimmt automatisch den Wert, den Sie hier angeben, um bis zu diesem Satz zu lesen, ohne diese Daten neu zu importieren. Der nächste Satz nach dem Restart-Wert (RESTARTCOUNT) wird nun genommen, um in die Datenbank eingefügt zu werden.

Somit können Sie mit der COMMITCOUNT-Option die IMPORT-Transaktion in mehrere kürzere Transaktionen aufteilen. Im Fehlerfall müssen so maximal die in der Option angegebene Anzahl von Datensätzen bzw. Veränderungen zurückgenommen werden, während ohne diese Option *alle* Änderungen rückgängig gemacht werden müssen. Dies hat erhebliche Auswirkungen auf die Restartzeit und somit auf die Zeit, bis die Tabelle mit den neuen Inhalten wiederverwendet werden kann.

Hinweis:

Neu ab Fixpack 4: Ab DB2 UDB V8 FP4 gibt es die zusätzlichen Optionen ALLOW WRITE ACCESS und ALLOW NO ACCESS. Hiermit wird ein ONLINE-Import möglich, d.h., während des IMPORT-Vorgangs kann auf die Tabelle zugegriffen und die bestehenden Daten können verändert werden.

8.3.2 Grafische Unterstützung des Imports

Aus dem GUI heraus stehen Ihnen alle oben aufgeführten Optionen auch zur Verfügung. Im GUI haben Sie den Vorteil, dass in den Auswahl-Bildschirmen alle Optionen angezeigt werden. Zudem werden Sie im GUI zum Anlegen einer Nachrichten-

datei (Message-Datei) gezwungen, um alle Meldungen später auch nachvollziehen zu können.

Das Import-Tool wird aus der Steuerzentrale aufgerufen, indem Sie IMPORTIEREN... (siehe Abbildung 8.9) aus dem Kontextmenü der Tabelle öffnen, in die Sie importieren wollen. Dies gilt auch, wenn Sie eine neue Tabelle mit einer IXF-Eingabedatei erzeugen wollen.

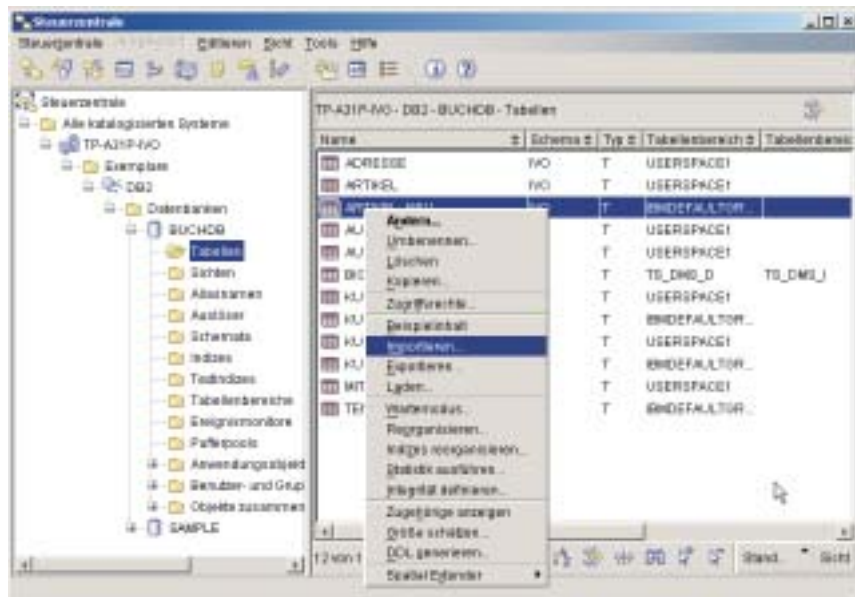


Abbildung 8.9: Steuerzentrale – Import

Das nächste Fenster (siehe Abbildung 8.10) zeigt die wichtigsten Einstellungen, die Sie durchführen müssen. Unter DATEI IMPORTIEREN geben Sie die Eingabedatei an, unter DATEITYPEN IMPORTIEREN müssen Sie den passenden Typ wählen (hier DEL). Den IMPORTMODUS, wie in Abbildung 8.10 beschrieben, finden Sie in der gleichnamigen Auswahlliste. Nun müssen Sie noch die NACHRICHTENDATEI eintragen. Diese Angaben sind zwingend. Weitere Optionen können bei Bedarf angegeben werden, z.B. die Option DATENSÄTZE VOR FESTSCHREIBEN – also den COMMITCOUNT –, der hier auf 75 gesetzt ist.

Hinweis:

Aus Performance-Gründen sollten Sie zusätzlich den Wert für COMPOUND-SQL auf 100 setzen. Diese Option beschleunigt das Import-Tool, da einige interne Meldungen nicht für jeden Datensatz erstellt werden, sondern nur für jeden hundertsten.

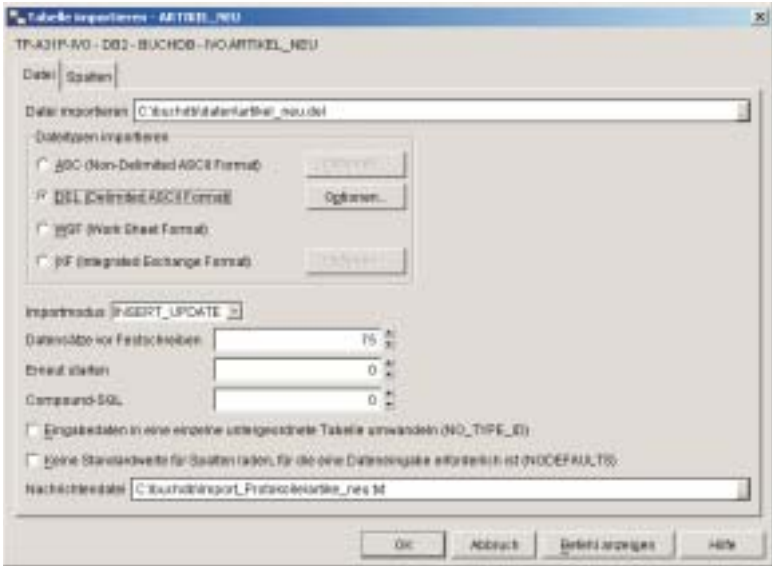


Abbildung 8.10: Import – Datentypen und Importmodus

In Abbildung 8.11 sehen Sie das Fenster, das Sie angezeigt bekommen, wenn Sie den Options-Button (OPTIONEN...) bei DATEITYPEN IMPORTIEREN auswählen.

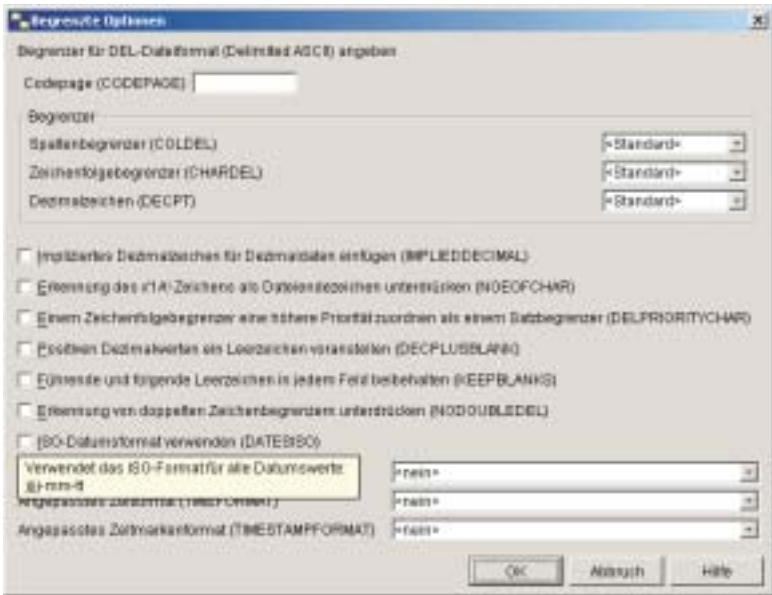


Abbildung 8.11: Import – DEL-Optionen

In diesen Optionen (siehe Abbildung 8.11) ist besonders darauf zu achten, dass die Begrenzer (COLDEL, CHARDEL, DECPT) so gesetzt sind, wie sie in der Eingabedatei vorliegen. Wurde die Datei mit den Standards erzeugt, so können Sie hier die Eingabe auch unverändert lassen.

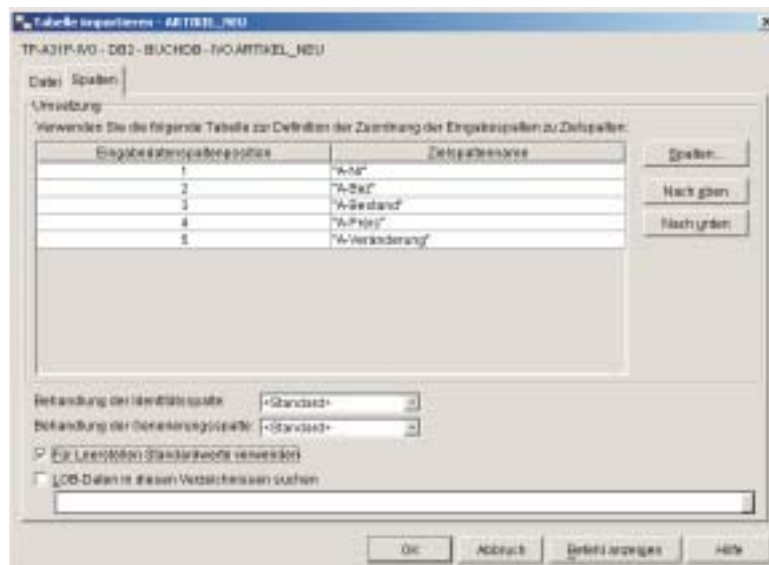


Abbildung 8.12: Import – Spaltenoptionen

In das Fenster aus Abbildung 8.12 gelangen Sie über die Auswahl des SPALTEN-Reiters. Hier können Sie, wenn notwendig, die Reihenfolge der Input-Spalten an die Reihenfolge in der Zieltabelle anpassen.

Mit BEFEHL ANZEIGEN wird Ihnen der von DB2 generierte IMPORT-Befehl gezeigt (siehe Abbildung 8.13), der so von DB2 gestartet wird, wenn Sie mit dem OK-Button, wie in Abbildung 8.12 aufgezeigt, die Eingabe bestätigen.



Abbildung 8.13: Import-Befehl

8.4 Lade-Tool

Das Lade-Tool ist die zweite Variante, um Daten in die Datenbank zu bekommen. Die erste Variante, das Import-Tool, haben wir Ihnen schon in Abschnitt 8.3 „Import-Tool“ vorgestellt.

Der Unterschied zum Import-Tool ist vor allem die Geschwindigkeit, in der die Daten in die Tabelle „geladen“ werden können. Dieser Performance-Gewinn basiert darauf, dass die Vorgehensweise, wie das Load-Tool die Daten in die Datenbank bringt, anders ist.

Der Import verarbeitet die Datensätze „einzeln“. Das heißt, für jeden Satz werden die Abhängigkeiten geprüft, z.B. auf referenzielle Integrität (RI), auf Integritätsbedingungen (check constraints) und auf abhängige Auslöser (Trigger), wenn diese vorhanden sind. Der Import protokolliert auch noch alle Aktivitäten in den Protokolldateien. Das ist viel Aufwand für jeden Satz.

Im Gegensatz dazu packt das LOAD-Tool die gelesenen Sätze auf eine Datenseite und schreibt dann diese direkt in die Tabellenbereiche, *ohne* die Prüfungen, die beim IMPORT durchgeführt werden, und *ohne* Protokolldateien zu schreiben. Die notwendigen Prüfungen und der Aufbau eines Index erfolgen erst in weiteren Schritten.

Sie sehen, dass das LOAD-Tool hier keine Informationen in die Protokolldateien schreibt. Somit sind andere Mechanismen bei diesem Tool gefragt, um Wiederherstellbarkeit im Fehlerfall gewährleisten zu können.

8.4.1 Phasen des Ladevorgangs

Der Ablauf des Ladeprozesses kann in vier Phasen aufgeteilt werden, siehe Abbildung 8.14.

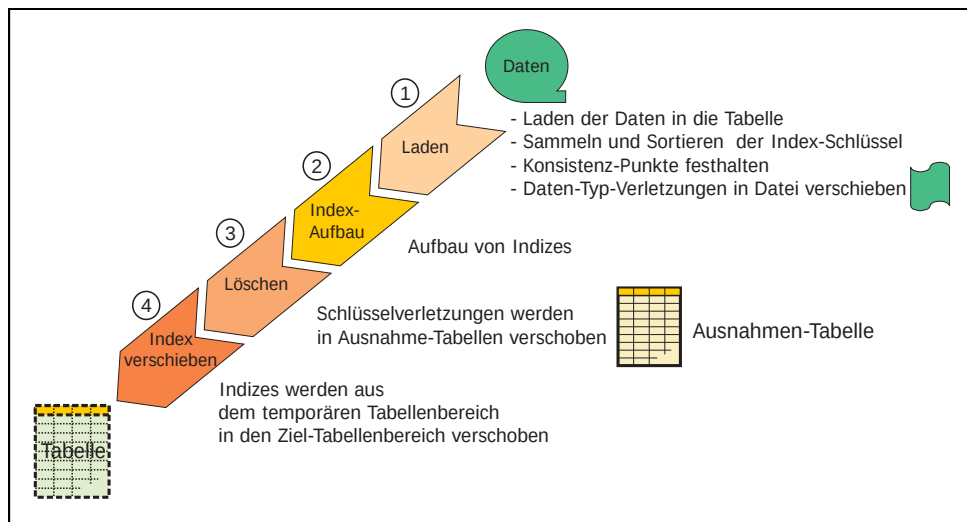


Abbildung 8.14: Die vier Phasen des LOAD-Tools

- In der ersten Phase ①, der Lade-Phase, werden die Daten in die Tabelle eingefügt. Sind Indizes für die zu ladenden Tabellen vorhanden, werden für diese Indexdaten gesammelt und sortiert, um im zweiten Schritt aufgebaut zu werden. Sie als DBA können über die Option `SAVEPOINTS` Sicherungs- bzw. Konsistenzpunkte anlegen lassen, um im Falle eines Abbruchs einen Aufsetzpunkt für den Wiederanlauf

des Load-Tools zu haben. Zudem werden diese Sicherungspunkte in einer Protokolldatei (MESSAGES) festgehalten, die jederzeit abgefragt werden kann.

- ▶ In der Aufbau-Phase ② werden die Indizes auf der Basis der gesammelten Indexwerte (aus der ersten – der Lade-Phase) aufgebaut. Es werden auch gleichzeitig die Statistiken über die Indizes erstellt, wenn Sie bei der Option INDEXES den Wert STATISTICS YES mit angegeben haben. Wird der Ladeprozess in dieser Phase ② unterbrochen, so wird bei einem Wiederaufsetzen (RESTART) diese Phase von neuem durchlaufen.
- ▶ In der dritten Phase, der Lösch-Phase ③, werden die Zeilen gelöscht, die aufgrund der Verletzung eines eindeutigen Schlüssels (Primärschlüssel bzw. eindeutiger Index) nicht in dieser Tabelle sein dürften. Die gelöschten Zeilen werden in eine Ausnahme-Tabelle übertragen, falls eine vorhanden ist, und es wird ein Eintrag der Nachrichtendatei vermerkt. Bricht der LOAD in dieser Phase ab, so wird diese Phase beim Wiederaufsetzen (RESTART) einfach wiederholt. Haben Sie jedoch als Option COPY=YES bzw. ALLOW READ ACCESS verwendet, so wird mit der Phase ②, also mit der Aufbau-Phase, fortgefahren.

Vorsicht:

Die Sätze, die aufgrund einer Verletzung eines eindeutigen Index aus der zu ladenden Tabelle gelöscht werden, werden in die Ausnahmetabelle eingefügt, falls diese vorhanden ist. Dieser Einfügevorgang wird in den Protokolldateien verzeichnet. Das heißt: Wenn Sie hier sehr viele Sätze haben, die die Eindeutigkeitsregeln verletzen, so können dadurch eventuell die Protokolldateien überlaufen.

- ▶ Die vierte Phase ④, die Index-Verschiebe-Phase, kommt nur zum Zuge, wenn Sie die Option ALLOW READ ACCESS gewählt haben, also einen so genannten *online load*, durchführen und die Option/Anweisung USE <tabellenbereichsname> angegeben haben.

„Online load“ bedeutet, dass Benutzer diejenigen Datensätze in der zu ladenden Tabelle weiterhin lesen können, die schon vor dem LOAD bestanden haben. Vor DB2 UDB V8 wurde beim Ladevorgang der ganze Tabellenbereich exklusiv gesperrt.

Die vier Phasen gehören immer zusammen. Erst wenn alle Phasen erfolgreich durchlaufen worden sind, ist auch der LOAD beendet. Wird das LOAD-Tool an irgendeiner Stelle unterbrochen, kann es wieder aufgesetzt werden, doch dazu später.

8.4.2 Vorbereitung des Ladeprozesses

Folgendes sollten Sie wissen, bevor Sie mit dem Laden von Tabellen beginnen:

- ▶ Wo liegen die Daten, die Sie laden wollen?
 - In welchem Pfad? (Wie lautet der Name der PIPE?)
 - Wie lautet der Name der Eingabedatei?
 - Sollen die Daten über die CURSOR-Funktion direkt aus einem SQL-Abfrage-Ergebnis geladen werden?

- ▶ den Namen oder den Alias der Tabelle, die Sie laden wollen
- ▶ das Datenformat der Eingabedatei
 - Es werden die Formate ASC, DEL und IXF unterstützt, wie schon beim Import erwähnt.
- ▶ Sollen die zu ladenden Daten eingefügt oder die vorhandenen Daten ersetzt werden?
 - INSERT
 - REPLACE
- ▶ Soll der Lese-Zugriff während des Ladevorgangs erlaubt oder unterbunden werden?
- ▶ In welchen Pfad soll die Nachrichtendatei geschrieben werden?
- ▶ Ist die zu ladende Tabelle mit Integritätsregeln versehen?
 - Für den Fall, dass die Eindeutigkeitsregeln verletzt werden, *sollten hier unbedingt* Ausnahme-Tabellen erstellt werden, um diese Verletzung aufnehmen zu können, auch wenn das in der Syntax eigentlich optional ist. Werden keine Ausnahme-Tabellen beim LOAD angegeben, so erhalten Sie keinerlei Informationen über diese Ausnahmen. Bei dieser Ausgabe/Auflistung wird am Ende des Ladevorgang nur Folgendes angezeigt:

Anzahl gelesener Zeilen	= 96016
Anzahl übersprungener Zeilen	= 0
Anzahl geladener Zeilen	= 96007
Anzahl zurückgewiesener Zeilen	= 9
Anzahl gelöschter Zeilen	= 5
Anzahl festgeschriebener Zeilen	= 96016
 - Die Anzahl gelöschter Load-Daten besagt, dass 5 Sätze aufgrund einer Verletzung der Eindeutigkeit gelöscht wurden. Welche das sind, können Sie nur in den Ausnahme-Tabellen sehen.
- ▶ Ist für die temporären LOAD-Daten (TEMPFILES PATH) ein Verzeichnis bzw. Pfad mit ausreichend Platz vorhanden?

Wenn Sie sich nun fragen, was Sie beim Anlegen der Ausnahme-Tabellen beachten müssen, haben wir hier einige Hinweise für Sie:

- ▶ Die Ausnahme-Tabelle muss mit der Original-Tabelle identisch sein.
- ▶ Eine sinnvolle Ergänzung ist, eine Zusatzspalte als Zeitstempel einzuführen und eventuell noch eine zweite Spalte für Fehlermeldungen.

- zwei Beispiel-SQL-Befehle, um eine Ausnahme-Tabelle schnell anzulegen:

Beispiel 1:

```
CREATE TABLE Artikel_Ausnahmen1 like ARTIKEL;  
ALTER TABLE Artikel_Ausnahmen1  
    ADD column ts          timestamp  
    ADD COLUMN msg         clob(32K);
```

Beispiel 2:

```
CREATE TABLE Artikel_Ausnahmen2 as (select Artikel.*,  
    current timestamp          as ts,  
    clob(' ',32767)           as msg  
    from Artikel)  
definition only;
```

8.4.3 Grafische Unterstützung mit dem Assistent: Laden

Der Ladevorgang kann aus der Steuerzentrale über das Assistenten-Menü oder direkt aus dem Kontextmenü des Tabellenobjekts aufgerufen werden. Sie kommen hier immer in den gleichen Lade-Assistenten, der Sie durch die Lade-Optionen führt.

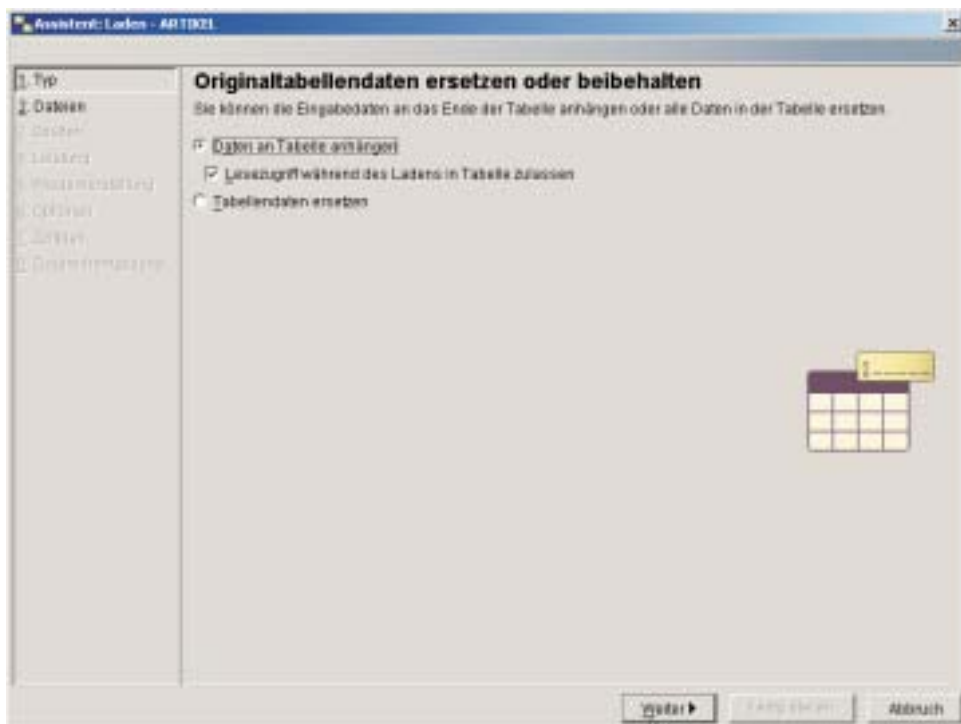


Abbildung 8.15: Assistent: Laden – Typ

Im ersten Schritt des Lade-Assistenten (siehe Abbildung 8.15) müssen Sie sich entscheiden, ob Sie die bestehenden Daten erweitern (INSERT INTO) oder überschreiben (REPLACE INTO) wollen. Die Verfahren entsprechen denen des IMPORT-Tools.

Soll ein Lesezugriff während des Ladens in die Tabelle zugelassen werden?

Wenn Sie diese Frage mit „Ja“ beantworten, so sollten Sie die zugehörige Checkbox auch kennzeichnen. DB2 verwendet dafür die Option ALLOW READ ACCESS. Wird diese Checkbox nicht markiert, gilt der Standardwert ALLOW NO ACCESS, der keinen Zugriff auf die geladene Tabelle zulässt.

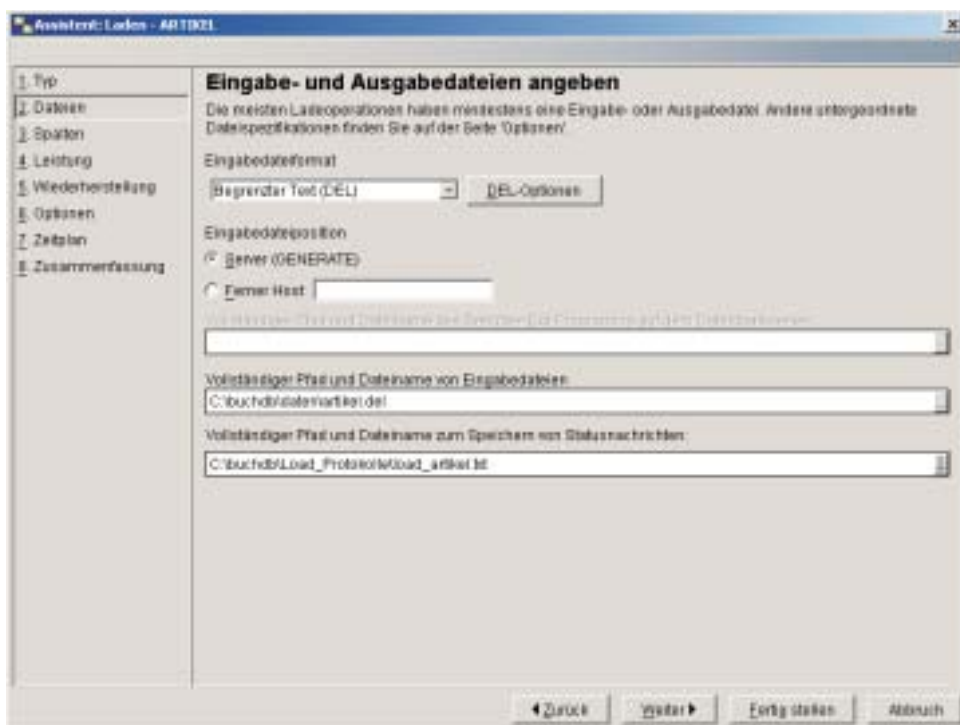


Abbildung 8.16: Assistent: Laden – Dateien

Im zweiten Schritt des Lade-Assistenten (siehe Abbildung 8.16) werden Sie nach den Eingabe-Informationen gefragt: In welchem Datenformat liegt die Eingabedatei vor, also ASC, DEL, IXF, oder werden die Daten über einen Cursor geladen, d.h. direkt aus einer SQL-Abfrage heraus? Wird der Cursor als Eingabe gewählt, so erhalten Sie ein Zusatzfenster, in dem die SQL-Abfrage ausformuliert werden kann, eventuell auch mit Hilfe eines SQL-Assistenten. Abhängig vom Datenformat können Sie hier auch den Button DEL-OPTIONEN für das DEL- bzw. ASC-Format auswählen, um die Positionsgebundenen Textoptionen zu beeinflussen, die auch schon im IMPORT-Tool (siehe Abbildung 8.7) vorgestellt wurden.

Die Option FERNER HOST und das Eingabefeld VOLLSTÄNDIGER PFAD UND DATEINAME DES BENUTZER-EXIT-PROGRAMMS AUF DEM DATENBANKSERVER (die deutsche Übersetzung

ist irreführend) entsprechen den Load-Optionen HOSTNAME und FILE_TRANSFER_CMD und sind an dieser Stelle unbrauchbar, da sie nur in einer partitionierten Umgebung von Interesse sind, wo dann aber weitere spezielle Optionen notwendig werden.

Haben Sie die Eingabedatei im DEL-Format ausgewählt, so wird das nächste Fenster angezeigt, wenn Sie auf WEITER klicken.

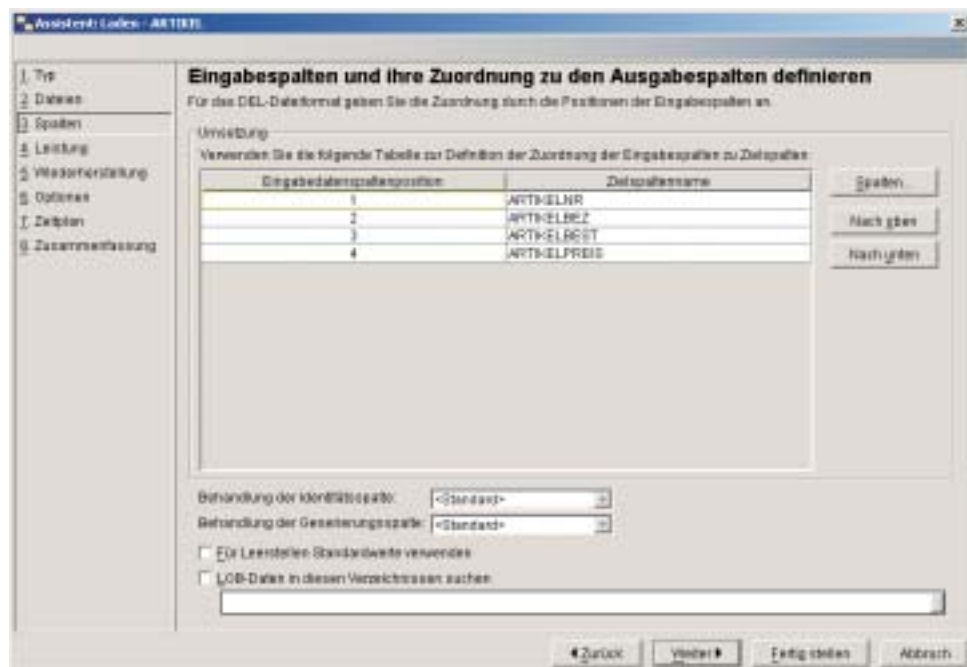


Abbildung 8.17: Assistent: Laden – Spalten

Die Ausgabe des dritten Fensters (siehe Abbildung 8.17) unterscheidet sich sehr von der Auswahl des Datenformats im zweiten Schritt (siehe Abbildung 8.16). Wurde DEL als Format gewählt, so haben Sie nun die Möglichkeit, über die Spaltenpositionen die Zuweisung zu den Zielspalten durchzuführen.

Der Schritt 4 (siehe Abbildung 8.18) dient zur Optimierung des Lade-Vorgangs. Die Überprüfung der Datentypen kann ausgeschaltet werden, wenn Sie einen Haken bei NUR MINIMALE ÜBERPRÜFUNG DER EINGABEDATEN AUSFÜHREN setzen. Dies ist nicht empfehlenswert, da Sie mit dieser Option Ihre Daten in einen inkonsistenten Zustand bringen können. Diese Option (FASTPARSE) sollte nur eingesetzt werden, wenn die Eingabedaten über ein Programm erstellt wurden und Sie sich sicher sind, dass es keine Verletzungen innerhalb der Datentypen gibt.



Abbildung 8.18: Assistent: Laden – Leistung

Wie die Indizes aufgebaut werden sollen, sollten Sie hier das DB2 entscheiden lassen. Es gibt die Möglichkeit der Teilaktualisierung und der kompletten Neu-Erstellung der Indizes. Wichtig ist bei der Erstellung von Indizes, dass DB2 genügend Speicherplatz für das Sortieren (SORTHEAP) der Indexdaten hat, um diese Aktion schnell durchzuführen. Häufig ist dieser Bereich zu klein; dann wird der Sortiervorgang auf der Festplatte im zugehörigen Tabellenbereich durchgeführt. Dies kann zum Überlaufen der Tabellenbereiche führen. Deswegen ist es sinnvoll, einen separaten temporären Tabellenbereich mitzugeben (hier TEMPSPACE1), damit DB2 den Sortiervorgang auf diesem Tabellenbereich durchführt und nicht den ursprünglichen Tabellenbereich nimmt.

Integritätsbedingungen – also Abhängigkeiten zu anderen Tabellen – werden in der Standardeinstellung „verzögert“, d.h. erst nach dem Laden der Daten mit dem zusätzlichen Befehl SET INTEGRITY überprüft und bereinigt. Wollen Sie, dass die abhängigen Tabellen sofort, also schon beim Ladeprozess, in den Überprüfung anstehend-Modus gehen, so sollten Sie den Haken bei VERSETZEN ABHÄNGIGER TABELLEN ... VERZÖGERN wegnehmen.

LADEOPERATION SOFORT STARTEN ... ist der Befehlsoption LOCK WITH FORCE gleichzusetzen und bewirkt, dass bestehende Anwendungen, die mit der zu ladenden Tabelle arbeiten, über den Befehl FORCE APPLICATION abgebrochen werden. Damit kann das Lade-Tool sofort starten und muss nicht warten, bis eine aktive Anwendung die Sperrung auf das zu ladende Objekt freigibt.

Optionen, die sich auf die Performance nach dem Laden beziehen (siehe Abbildung 8.18 OPTIONEN MIT AUSWIRKUNG AUF DIE LEISTUNG NACH DEM LADEN), ist die Freiplatzverwaltung für die Daten- und Index-Seiten, die wir schon in Kapitel 7 behandelt haben.

Die Statistikinformation, die im unteren Teil dieses Fensters modifiziert werden kann, wird nur freigegeben, wenn Sie Daten mit REPLACE laden, also komplett ersetzen. Dann haben Sie hier die Möglichkeit, die Statistiken schon beim Laden von DB2 erneuern zu lassen.

Hinweis:

Sind Sie mit dem Laden der Daten noch in der Test-Phase bzw. haben Sie beim Laden noch viele Sätze, die aufgrund von Schlüsselverletzungen aussortiert werden, so sollten Sie die zugehörigen Statistiken nachträglich mit dem RUNSTATS-Tool erzeugen lassen. Die Daten für die Statistiken werden vom LOAD in der Phase 1 erzeugt und nicht mehr aktualisiert, falls die Sätze in der Löschen-Phase (Delete-Phase) wieder gelöscht werden.

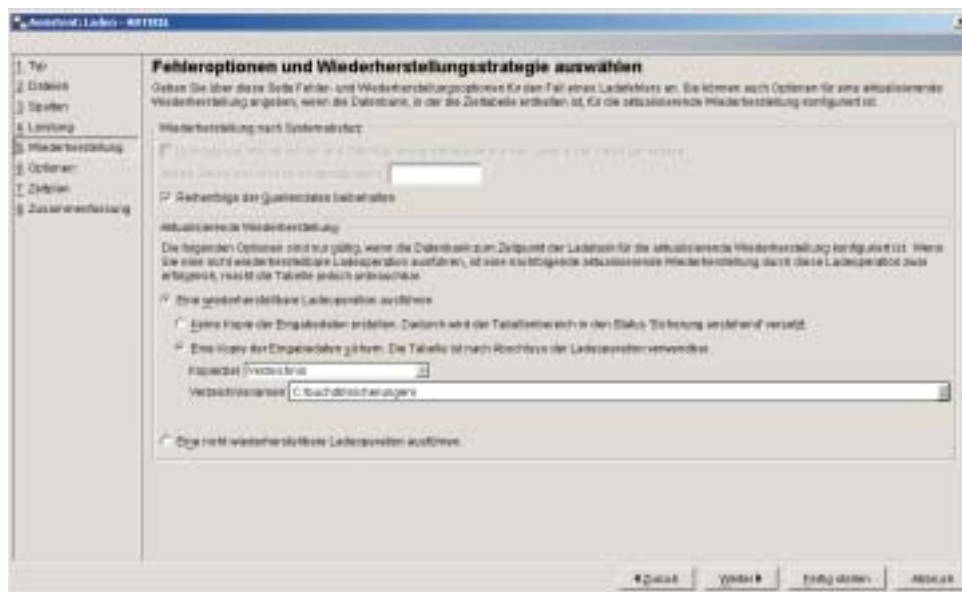


Abbildung 8.19: Assistent: Laden – Wiederherstellung

In Schritt 5 (siehe Abbildung 8.19) werden die Optionen für die Wiederherstellung gesetzt. Die Option REIHENFOLGE DER QUELLENDATEN BEIBEHALTEN ist standardmäßig eingeschaltet und sollte auch so bleiben. Dies gewährleistet, dass die Daten, wenn sie schon in „sortierter“ Reihenfolge vorliegen, auch in dieser geladen werden. Die zugehörige Befehlsoption lautet ANYORDER.

Der zweite Auswahlblock mit der Überschrift AKTUALISIERENDE WIEDERHERSTELLUNG gilt nur für den Fall, dass Sie in der Datenbank die Option LOGRETAIN=ON oder USER-EXIT=ON eingeschaltet haben. Diese Optionen besagen, dass Sie Ihre Datenbank im „aktualisierenden Wiederherstellungsmodus“ einsetzen und Protokolldateien nicht überschrieben werden (dazu erfahren Sie in Kapitel 9 mehr).

In dieser Auswahl haben Sie drei Möglichkeiten:

► KEINE KOPIE DER EINGABEDATEN ERSTELLEN

Wird diese Option ausgewählt, so zwingt DB2 Sie, nach einem Ladevorgang eine Sicherung des Tabellenbereichs durchzuführen, in dem sich die geladene Tabelle befindet. Sie erhalten dann den Status Sicherung anstehend für diesen Tabellenbereich, der erst wieder in den Normal-Status übergeht, wenn eine Sicherung durchgeführt worden ist. Erst im Normal-Zustand kann mit den neuen Daten gearbeitet werden.

► EINE KOPIE DER EINGABEDATEN SICHERN

Hierbei werden die Eingabedaten in einem Sicherungsverzeichnis abgelegt, das von Ihnen angegeben wird. Hier sollte das gleiche Verzeichnis gewählt werden wie bei den Sicherungen für die Datenbank bzw. für die Tabellenbereiche. Diese Information wird im Wiederherstellungsprotokoll festgehalten und kann jederzeit (mit dem Befehl LIST HISTORY LOAD oder im Journal-GUI) nachgelesen werden. Der Tabellenbereich bleibt bei dieser Aktion im Normal-Zustand und muss nicht zusätzlich gesichert werden.

Im Falle einer aktualisierenden Wiederherstellung holt sich DB2 diese Information aus dem Wiederherstellungsprotokoll und führt den LOAD mit Hilfe der gesicherten Eingabedatei noch einmal durch.

Hinweis:

Sie sollten die Sicherung nie aus dem Pfad herausnehmen und verlagern, da DB2 bei der aktualisierenden Wiederherstellung immer diese Information aus dem Wiederherstellungsprotokoll bezieht.

Falls dies doch passiert, ist mit folgenden Fehlermeldungen zu rechnen:

In der Steuerzentrale erhalten Sie folgenden SQL Error - SQL3785N :

'SQL3785N Die Wiederherstellung für Tabelle "IVO
.ARTIKEL_NEU" zum Zeitpunkt "20030917111536" auf Knoten "0"
schlug wegen Fehler "0" fehl. Zu diesem Fehler gibt es folgende
Zusatzinformationen: "terminated by caller".

Erklärung:

Während der Wiederherstellung trat ein schwerwiegender Fehler auf. Die Verarbeitung des Dienstprogramms wird gestoppt.

Hinweis: Bei Verwendung eines partitionierten Datenbanksservers zeigt die Knotennummer an, auf welchem Knoten der Fehler aufgetreten ist.

Andernfalls ist diese Angabe nicht relevant und kann ignoriert werden.

Benutzeraktion:

Weitere Informationen können über den Fehlercode in der Nachricht abgerufen werden. Beheben Sie den Fehler, und wiederholen Sie den Befehl.

Auszug aus der DB2-Diagnose-Datei (db2diag.log):

Terminating EDUs - load copy restore terminated by caller

2003-09-17-11.50.45.218000 Instance:DB2 Node:000
PID:1288(db2syscs.exe) TID:276 Appid:*LOCAL.DB2.00A547095031
database utilities sqludcpy Probe:0 Database:BUCHDB

Load copy restore failed.

2003-09-17-11.50.45.234000 Instance:DB2 Node:000
PID:1288(db2syscs.exe) TID:276 Appid:*LOCAL.DB2.00A547095031
database utilities sqludcpy Probe:100 Database:BUCHDB

Load copy restore terminated.

2003-09-17-11.50.45.281000 Instance:DB2 Node:000
PID:1288(db2syscs.exe) TID:276 Appid:*LOCAL.DB2.00A547095031
data protection sqlufrol Probe:1100 Database:BUCHDB

Load copy image restore terminated with rc -3785

2003-09-17-11.50.45.312000 Instance:DB2 Node:000
PID:1288(db2syscs.exe) TID:276 Appid:*LOCAL.DB2.00A547095031
recovery manager sqlpForwardRecoveryTblSpace Probe:30 Database:BUCHDB
DIA0001E An internal error occurred. Report the following error code :
"ZRC=0xFFFFF137".

2003-09-17-11.50.45.328000 Instance:DB2 Node:000
PID:1288(db2syscs.exe) TID:276 Appid:*LOCAL.DB2.00A547095031
data protection sqlufrol Probe:8180 Database:BUCHDB

ADM1611W The rollforward recovery phase has been completed.

Um diesen Fehler zu vermeiden, sollten Sie unbedingt die Sicherungsdatei der Eingabedaten des LOADs, die mit COPY = YES erstellt wurde, in dem Verzeichnis aufbewahren. Alternativ müssen Sie diese Datei vor einer aktualisierenden Wiederherstellung in den zugehörigen Pfad zurückkopieren. Der zugehörige Pfad kann aus der Wiederherstellungsprotokolldatei ausgelesen werden (LIST HISTORY LOAD).

- EINE NICHT WIEDERHERSTELLBARE LADEOPERATION AUSFÜHREN (NONRECOVERABLE) – kennzeichnet die geladene Tabelle als nicht wiederherstellbar, falls eine aktualisierende Wiederherstellung gewünscht wird. Sie haben dann nur noch die Möglichkeit, die Tabelle entweder zu löschen oder noch einmal eine Wiederherstellung durchzuführen, und zwar zu einem Zeitpunkt vor dem Lade-Vorgang.

In Schritt 6 der Optionen (siehe Abbildung 8.20) wird Ihnen noch die Möglichkeit gegeben, auf folgende DB2-Parameter Einfluss zu nehmen:

- WARNINGCOUNT – gibt an, nach welcher Anzahl von fehlerhaften Sätzen der LOAD abgebrochen werden soll. Wenn Sie normalerweise davon ausgehen, dass in der Eingabedatei keine Fehler vorhanden sind, so sollten Sie hier den Wert auf „1“ setzen. Damit werden Sie bei Fehlern sofort informiert und verbrauchen nicht unnötige Zeit und Ressourcen, die ein langer und fehlerhafter LOAD beanspruchen könnte. Sind Sie jedoch in der Test-Phase, kann ein höherer Wert (z.B. 100) sinnvoll sein, um fehlerhafte Sätze zu identifizieren.

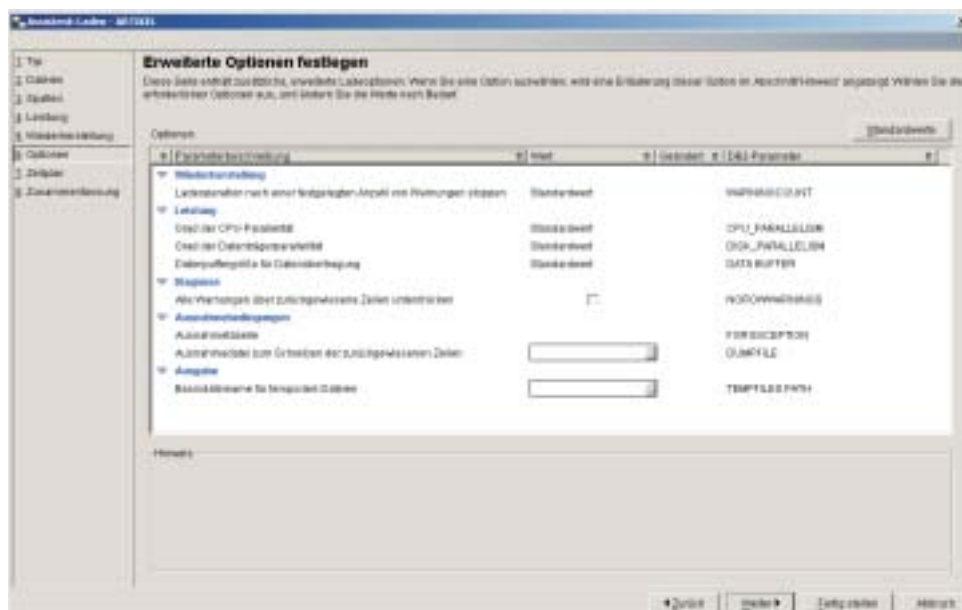


Abbildung 8.20: Assistent: Laden – Optionen

Die Leistungsparameter können beruhigt aus den Standardwerten übernommen werden, da hier die so genannten „intelligenten Standards“ eingesetzt werden und DB2 diese Werte aus der aktuellen Konfiguration ausliest.

- **BASISDATEINAME FÜR TEMPORÄRE DATEIEN (TEMPFILES PATH)** – Diese Angabe ist vor allem bei Verwendung großer Eingabedateien sinnvoll, da hiermit der Pfad für alle temporären Dateien und Kontrolldateien angegeben wird. Wird dieser Parameter nicht gesetzt, so ist der Standardpfad: ... \SQL0000\load\...

Der Zielpfad für die temporären Dateien wird wie folgt generiert:

```
\<TEMPFILE PATH>\SQL0000n.PID\SQL0000m.OID\
```

n = Tabellenbereichsnummer (syscat.tables.tablespaceid)

m = Tabellennummer (syscat.tables.tableid)

In diesem Verzeichnis stehen die Kontrolldateien (load.ct1 und load.ct2) und die Nachrichtendatei (load.msg), die über den Befehl `LOAD QUERY TABLE <tabellen_name>` abgefragt werden kann und über den Status des LOAD Auskunft gibt.

- **NOROW WARNINGS** – Hier können Informationen, die DB2 in die Nachrichtendatei schreibt, unterdrückt werden. Dies kann die Performance verbessern, da jede Fehlermeldung einen Eintrag in die Nachrichtendatei schreibt, also einen E/A durchführt. Diese Option sollte nicht eingesetzt werden, da die Nachrichten bei der Fehlerdiagnose hilfreich sind.

- **AUSNAHME-DATEI ZUM SCHREIBEN DER ZURÜCKGEWIESENEN ZEILEN (DUMPFIL)** – Hier sollten Sie eine Datei angeben, um alle aufgrund der Verletzung von Datentypen abgewiesenen Sätze aufzunehmen. Dies hilft Ihnen, vor allem bei sehr großen Eingabedateien, einen Überblick über die abgewiesenen Sätze zu bekommen. Zudem bietet sich so eine einfache Möglichkeit, diese Daten nachträglich anzupassen und als bereinigte Eingabedatei mit dem Load-Tool neu zu laden.

Die weiteren Schritte des Assistenten dienen zum Einplanen des Ladevorgangs in der Taskzentrale (Schritt 7) und zur Anzeige und Ausführung des in Schritt 8 generierten Befehls.

8.4.4 Der LOAD-Befehl

Der LOAD-Befehl hat sehr viele Optionen. Dies konnten Sie eventuell schon erahnen, als Sie die Beschreibung des Lade-Assistenten gelesen haben. Wir wollen Ihnen hier nochmals einige dieser Optionen vorstellen und manch andere Option erklären. Der einfachste Lade-Befehl besteht aus den in Abbildung 8.21 gezeigten Befehlsteilen. Einige sind optional und müssten nicht unbedingt eingegeben werden; wir aber stehen auf dem Standpunkt, dass diese Befehlsteile notwendig sind, um vernünftig arbeiten zu können.

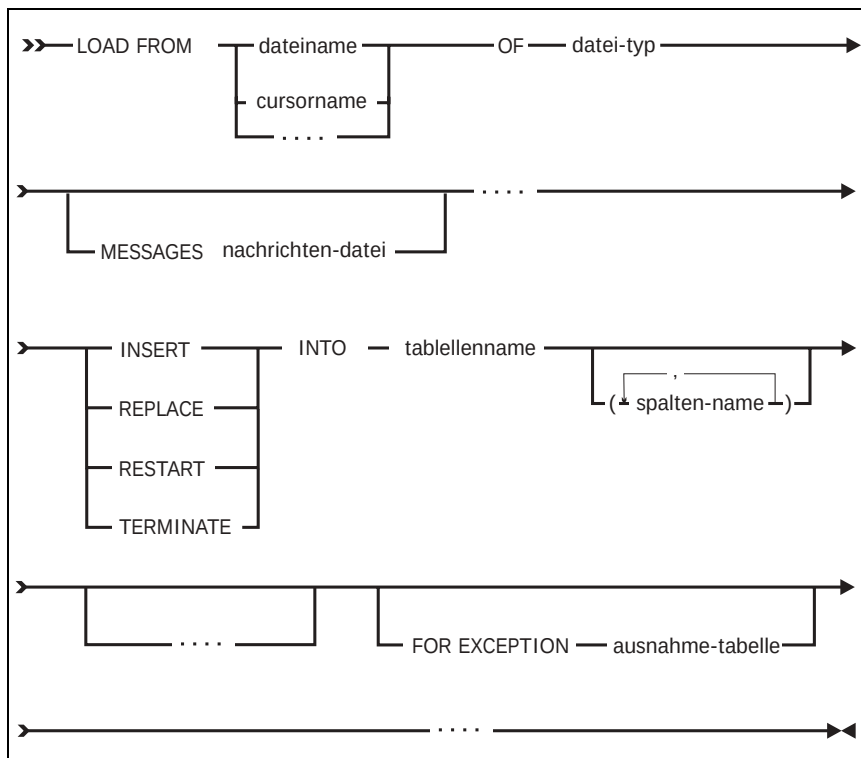


Abbildung 8.21: load-Syntaxdiagramm – wesentliche Bestandteile

Die zu ladenden Eingabedaten müssen im `LOAD FROM ...`-Befehl angegeben werden. Sie können Daten außer über eine Eingabedatei auch über eine „named pipe“ direkt aus einem anderen Programm übernehmen. Noch interessanter ist die neue Möglichkeit in DB2 UDB V8, Daten über die `CURSOR`-Funktion zu laden. Dabei werden die zu ladenden Daten direkt über eine SQL-Abfrage selektiert.

Tipp:

Sie können damit sogar Daten aus einer entfernten Datenbank selektieren und laden, wenn Sie die „Federated“-Funktionalität (Funktionalität für zusammengegeschlossene Datenbanken) in DB2 nutzen.

Die Schritte für das Laden mittels eines Cursors sind folgende:

1. Definieren Sie einen CURSOR:

```
DECLARE buchcursor CURSOR FOR SELECT spalte1, spalte2, spalte3 FROM schema.quelltab
```

2. Laden Sie die Daten direkt aus dem CURSOR:

```
LOAD FROM buchcursor OF CURSOR INSERT INTO schema.zieltab
```

Die Option `MESSAGES` sollte immer mit eingesetzt werden, auch wenn diese nur optional ist, um auftretende Fehler zu dokumentieren und in Ruhe nachvollziehen zu können. Ist diese Option nicht angegeben, werden die Informationen auf dem Bildschirm ausgegeben.

Die Option `INSERT` fügt die gültigen Eingabedaten zu den bestehenden Tabellendaten hinzu.

Die `REPLACE`-Option löscht zuerst alle Daten, was aber nur mit einem informativen Eintrag in den Protokolldateien festgehalten wird, und fügt alle neue Daten in die Tabelle ein, sofern nicht eine Datentyp-Verletzung oder eine Schlüssel-Verletzung vorhanden ist.

An dieser Stelle weisen wir nochmals auf die Option `FOR EXCEPTION` hin, die wie die `MESSAGE`-Option auch optional ist. Die Informationen, die in diese Ausnahmetabelle geschrieben werden, gehen ohne diese Option verloren bzw. sind nur als Hinweis in der Schlussübersicht zu sehen (siehe nochmals die Erklärung der Ausnahmetabellen in Abschnitt 8.4.2 „Vorbereitung des Ladeprozesses“).

8.4.5 Aktivitäten während des Ladevorgangs

Zur Laufzeit, vor allem beim Laden großer Datenbestände, stellt man sich immer die gleichen Fragen: Wie lange dauert dieser Load noch? Wie viele Sätze sind schon geladen?

Hier ist ein Befehl sehr hilfreich: der `LOAD QUERY TABLE ...`-Befehl (siehe Abbildung 8.22).

Um die oben genannten Fragen beantworten zu können, kann während des Ladevorgangs beliebig oft der Status über den `LOAD QUERY`-Befehl abgefragt werden.

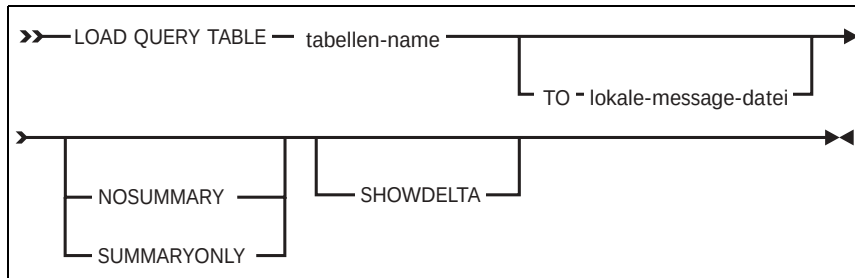


Abbildung 8.22: load query-Syntaxdiagramm

Ein Beispiel für eine Status-Abfrage:

```

C:\buchdb\Load_Protokolle> db2 load query table kunden_neu
.....
SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs.
          Zähler für Eingabesätze: "406294".
SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.
SQL3519W  Synchronisationspunkt am Beginn des Ladevorgangs.
          Zähler für Eingabesätze: "456848".
SQL3520W  Synchronisationspunkt für Ladevorgang erfolgreich.
SQL3532I  Das Dienstprogramm LOAD befindet sich derzeit in Phase "LOAD".
Anzahl gelesener Zeilen      = 508172
Anzahl übersprungener Zeilen = 0
Anzahl geladener Zeilen     = 508172
Anzahl zurückgewiesener Zeilen = 0
Anzahl gelöschter Zeilen    = 0
Anzahl festgeschriebener Zeilen = 456848
Anzahl Warnungen            = 0
  
```

Tabellenstatus:

Laden läuft

Wie Sie in der Auflistung sehen, werden hier Synchronisationspunkte angezeigt. Diese entstehen nur, wenn Sie die Option `SAVECOUNT` (hier `SAVECOUNT=500000`) im `LOAD`-Befehl verwendet haben. Durch diese Option merkt sich DB2 die Veränderungen bis zu diesem Zeitpunkt in der Datenbank, hier z.B.: Anzahl festgeschriebener Zeilen = 456848. Falls der Ladevorgang durch die `RESTART`-Option erneut aufgesetzt wird, müssen diese Datensätze nicht noch einmal geladen werden.

Hinweis:

Wird der `LOAD`-Vorgang über die Option `TERMINATE` gezielt abgebrochen, sind keinerlei neue Daten zu sehen, auch wenn vorher Synchronisationspunkte erreicht worden sind. Synchronisationspunkte entsprechen nicht einem `COMMIT` oder `COMMITCOUNT` beim `IMPORT`, da das Lade-Tool kein SQL nutzt.

Durch das mehrmalige Abfragen über den `LOAD QUERY TABLE ...`-Befehl können Sie ausrechnen, wie viele Sätze in einer Zeiteinheit (Zeit zwischen zwei Abfragen) geladen werden können, und darüber dann eine Hochrechnung auf alle Datensätze wagen.

Tipp:

Für die Berechnung der Zeitdifferenzen kann bei Windows das *timethis*-Tool verwendet werden (separat von der Microsoft-Webseite erhältlich). Bei Unix verwenden Sie den `time`-Befehl, der einfach vor den `DB2`-Befehl gesetzt wird.

Die Abfrage des `LOAD QUERY TABLE` bezieht sich nicht auf die angegebene Nachrichtendatei im `LOAD`-Befehl, sondern auf eine eigene `LOAD.msg`-Datei, die sich im `TEMPFILES PATH` befindet.

Die Abfragedaten sind nach erfolgreichem Laden nicht mehr verfügbar, außer Sie haben die Abfragen immer in einer Ausgabedatei über die Option `T0` im `LOAD QUERY`-Befehl gesichert.

Der Tabellenstatus, z.B. „Laden läuft“, kann nur über den Befehl `LOAD QUERY TABLE` angezeigt werden. Wird eine Tabelle nicht geladen, so sehen Sie den Tabellenstatus `Normal`. Eine Übersicht, ob Tabellen gerade geladen werden, gibt es nicht. Die Tabellen müssen jeweils einzeln abgefragt werden.

Folgende Statusinformationen können im Feld Tabellenstatus angezeigt werden:

► **Normal**

Die Tabelle ist im „normalen“ Zustand und wird nicht durch einen der folgenden Punkte beeinflusst.

► **Überprüfung anstehend**

Für die Tabelle gelten Integritätsbedingungen, die noch überprüft werden müssen. Mit dem Befehl `SET INTEGRITY` kann der Status „Überprüfung anstehend“ für die Tabelle aufgehoben werden. Der `LOAD` versetzt eine Tabelle in den Status „Überprüfung anstehend“, *sobald* eine Ladeoperation für eine Tabelle mit Integritätsbedingungen *beginnt*.

► **Laden läuft**

Es wird gerade eine Ladeoperation für diese Tabelle ausgeführt.

► **Laden anstehend**

Für diese Tabelle wurde ein `LOAD` durchgeführt, der aber nicht richtig beendet worden ist. Mit den Befehlen `LOAD TERMINATE`, `LOAD RESTART` oder `LOAD REPLACE` können Sie diesen Status für die Tabelle aufheben.

► **Lesezugriff**

Es wurde ein `LOAD` mit der Option `ALLOW READ ACCESS` gestartet. Die Tabellendaten sind nur für Abfragen im Lesezugriff verfügbar.

► Nicht verfügbar

Die Tabelle ist nicht verfügbar. Die Tabelle kann lediglich gelöscht oder aus einer Sicherung wiederhergestellt werden. Erkennt DB2 während der Abarbeitung der Protokolldateien bei einer aktualisierenden Wiederherstellung, dass ein Ladevorgang mit der Option NONRECOVERABLE durchgeführt wurde, so wird die so geladene Tabelle in den Status "Nicht verfügbar" versetzt.

► Für Laden nicht neu startbar

Die Tabelle befindet sich in einem teilweise geladenen Zustand, der eine LOAD RESTART-Operation nicht zulässt. Für die Tabelle gilt außerdem der Status "Laden anstehend". Mit dem Befehl LOAD TERMINATE oder LOAD REPLACE können Sie den Status "Für Laden nicht neu startbar" aufheben.

Eine Tabelle wird in diesen Status versetzt, wenn:

- eine aktualisierende Wiederherstellung nach einer fehlgeschlagenen Ladeoperation ausgeführt wird,
- eine Wiederherstellungsoperation mit einer Online-Sicherung ausgeführt wird, die erstellt wurde, als die Tabelle den Status "Laden läuft" oder "Laden anstehend" aufwies.

In beiden Fällen sind die Informationen, die für eine LOAD RESTART-Operation erforderlich sind, unzuverlässig. Der Status "Für Laden nicht neu startbar" verhindert somit, dass eine LOAD RESTART-Operation stattfinden kann.

► Unbekannt

Der Befehl LOAD QUERY TABLE ist nicht in der Lage, den Tabellenstatus zu ermitteln.

Für eine Tabelle können auch gleichzeitig mehrere Werte im Tabellenstatus sichtbar sein. Werden Daten mit Integritätsbedingungen geladen und ist die Option ALLOW READ ACCESS auch angegeben, können Sie möglicherweise folgende Statuswerte sehen:

Tabellenstatus:

Überprüfung anstehend
Laden läuft
Lesezugriff

Der Tabellenstatus ändert sich, wenn der Ladevorgang beendet ist, in:

Tabellenstatus:

Überprüfung anstehend
Lesezugriff

Mit dem Status Überprüfung anstehend wird angezeigt, dass die Integrität überprüft werden muss. Dazu wird der Befehl SET INTEGRITY ... verwendet. Ist diese Überprüfung erfolgreich gelaufen, so hat die Tabelle wieder den Zustand Normal:

Tabellenstatus:

Normal

8.4.6 Aktivitäten nach dem Laden

Bisher haben wir sehr wenig von möglichen Fehlern gesprochen, doch in der Praxis kommen Fehler aller Art vor. Somit schauen wir uns an, wie Sie vorgehen sollten, wenn typische Situationen auftreten:

- ▶ Eingabedaten haben Fehler, z.B. Datentyp-Verletzungen. Teile der Eingabedaten liegen im „falschen“ Format vor, z.B. als CHARACTER- statt als INTEGER-Daten.
- ▶ Schlüsselverletzung und Eindeutigkeitsverletzungen

Eingabedaten entsprechen nicht den Spaltendefinitionen in der Zieltabelle

Stimmen die Daten in der Eingabedatei nicht, so erhalten Sie in der Nachrichten-datei (MESSAGES ...) folgende Informationen:

SQL3120W Der Feldwert in Zeile "2" und Spalte "1" kann nicht in einen Wert mit dem Datentyp INTEGER konvertiert werden; für die Zielspalte ist jedoch die Eingabe von Daten erforderlich. Die Zeile wurde nicht geladen.

Sie erkennen an der Ausgabe die Zeile und die Spalte, in der der Fehler gefunden wurde. In der „Summary“ – also in der Zusammenfassung am Ende des Reports – erhalten Sie die Übersicht über alle geladenen und gelöschten Zeilen:

Anzahl gelesener Zeilen	=
Anzahl übersprungener Zeilen	=
Anzahl geladener Zeilen	=
Anzahl zurückgewiesener Zeilen	= 20
Anzahl gelöschter Zeilen	=
Anzahl festgeschriebener Zeilen	=

In dieser Ausgabe sehen Sie in der Anzahl zurückgewiesener Zeilen = 20 die Anzahl der Sätze, die aufgrund von Datentyp-Verletzungen nicht geladen wurden.

Was ist nun zu tun? Sie müssen die ausgewiesenen Sätze aus der Eingabedatei filtern und bereinigen. Besser ist es jedoch, wenn Sie dazu die Eingabedatei nicht noch einmal durcharbeiten müssen, da diese unter Umständen sehr groß sein kann. Darum sollten Sie im LOAD-Befehl immer mit der Option DUMPFILE arbeiten, um so eine eigene Ausgabedatei für diese abgewiesenen Sätze (Datentyp-Verletzung) zu haben.

Ein Beispiel für diese Ergänzung ist:

```
LOAD ... MODIFIED BY DUMPFILE="c:\daten\Kunden_abw.dmp" ...
```

Die Datei und auch der Pfad, den Sie hier mit angeben, wird von DB2 neu angelegt, und die Datei wird um die Partitionsnummer (000, in unserem Beispiel) ergänzt. Der Dateiname des Dumpfile lautet also Kunden_abw.dmp.000.

Tipp:

Verwenden Sie für DUMPFILE nur maximal zweistufige Namen!

Schlüsselverletzungen bzw. Verletzungen der Eindeutigkeit

Eine typische Ausgabe einer Zusammenfassung nach einem LOAD sieht so aus:

Anzahl gelesener Zeilen	= 4
Anzahl übersprungener Zeilen	= 0
Anzahl geladener Zeilen	= 4
Anzahl zurückgewiesener Zeilen	= 0
Anzahl gelöschter Zeilen	= 2
Anzahl festgeschriebener Zeilen	= 4

Wie viele Sätze wurden geladen?

Die Anzahl der geladenen Zeilen steht hier auf „4“. Wurden auch 4 Zeilen geladen? Nein! Denn bei der Anzahl gelöschter Zeilen steht der Wert „2“, und somit gibt $4 - 2 = 2$; es wurden also nur zwei Sätze geladen.

Warum ist das so? Sie erinnern sich eventuell noch an die vier Phasen des LOAD (siehe Abbildung 8.14). Die Information für die Anzahl der geladenen Sätze wird aus der Phase 1, der Load-Phase, genommen. In der Phase 3 – der Lösch-Phase – werden die Sätze gelöscht, die eine Schlüssel- oder Eindeutigkeitsverletzung aufweisen. Die Anzahl der geladenen Sätze entspricht also dann nicht mehr der tatsächlichen Anzahl.

Welche Sätze wurden gelöscht?

Schauen wir in die Nachrichten-Datei, so finden wir nur folgende Information:

```
...
SQL3500W  Die Phase "DELETE" wird gestartet (Zeit: "17/09/2003
20.05.52.231565").
SQL3509W  Vom Dienstprogramm wurde(n) "2" Zeile(n) aus der Tabelle gelöscht.
SQL3515W  Die Phase "DELETE" wurde beendet (Zeit: "17/09/2003 20.05.52.289522").
...
```

Es wird kein Hinweis auf die Daten bzw. die Zeilen gegeben, die gelöscht wurden.

Hier empfehlen wir *dringend*, immer eine Ausnahme-Tabelle für die zu ladende Tabelle mit anzugeben. Das heißt, es sollte, wie schon in Abschnitt 8.4.2 „Vorbereitung des Ladeprozesses“ aufgeführt, eine Ausnahme-Tabelle für jede Tabelle definiert werden, die mit einer Schlüsseldefinition (PK) oder einer Integritätsregel (eindeutiger Index oder Prüfung auf Integritätsbedingung) belegt ist.

Der LOAD-Befehl wird um folgende Option erweitert:

```
LOAD .... FOR EXCEPTION <tabellen-name-für-ausnahmen> ...
```

Nun werden alle Verletzungen in diese Ausnahme-Tabellen geschrieben. Sie verlieren keinen Satz, haben alle Daten in einer Tabelle und können nach einer Bereinigung die Daten mit dem SQL-Befehl INSERT INTO wieder in Ihre Zieltabelle einfügen.

Hinweis:

Arbeiten Sie mit einer „Prüfung der Integritätsbedingungen“ (check constraints), so werden Sie bei der Überprüfung der Integritätsbedingungen mit SET INTEGRITY dazu gezwungen, FOR EXCEPTION – also Ausnahme-Tabellen – anzugeben, wenn eine Verletzung gefunden wird. Bis zur Bereinigung bleibt hier die Tabelle im Status "Überprüfung anstehend".

SET INTEGRITY

Sind nun nach dem Laden alle Sätze in der Tabelle verfügbar? Dies ist nicht so einfach zu beantworten. Es hängt von den eindeutigen Schlüsseln und von der Prüfung auf Integritätsbedingungen ab. Ist eine von diesen Abhängigkeiten definiert, so müssen Sie die Integrität der Daten über den Befehl SET INTEGRITY überprüfen und wiederherstellen.

Als Benutzer einer Tabelle, also wenn Sie eine Abfrage auf die Tabelle starten, können Sie folgende Fehlermeldung erhalten; die Ursachencodes können hierbei unterschiedlich sein:

```
SQL0668N Die Operation ist für Tabelle "IVO.KUNDE" nicht zulässig. Ursachencode: "1".
SQLSTATE=57016
```

Was ist nun der Ursachencode? Eine einfache Abfrage mit db2 ? sql0668 zeigt Ihnen die Details dieses SQL-Codes:

```
SQL0668N Die Operation ist für Tabelle "<tabellenname>" nicht
        zulässig. Ursachencode: "<ursachencode>".
```

Erklärung:

Auf die Tabelle "<tabellenname>" besteht nur eingeschränkter Zugriff. Die Ursache beruht auf den folgenden Ursachencodes "<ursachencode>":

1 Die Tabelle befindet sich im Status 'Überprüfung anstehend'. Die Integrität der Tabelle wurde nicht durchgesetzt und der Inhalt der Tabelle ist möglicherweise ungültig. Diese Fehlermeldung wird möglicherweise auch dann ausgegeben, wenn eine übergeordnete oder zu Grunde liegende Tabelle, die sich nicht im Status 'Überprüfung anstehend' befindet, für eine Operation verwendet wird, und sich eine abhängige Tabelle noch im Status 'Überprüfung anstehend' befindet.

Um nun festzustellen, welche Ursache wirklich vorliegt, sollten Sie eine SQL-Abfrage aufbauen, die den Überprüfungsstatus dieser Tabelle in der Systemkatalog-Tabelle prüft. In der SYSCAT.TABLES gibt es eine STATUS-Spalte, die aufzeigt, ob eine Tabelle überhaupt geprüft werden muss (SYSCAT.TABLES.STATUS beinhaltet ein C), und eine Spalte, die die Ursache für diese notwendig gewordene Überprüfung (N = not checked) aufzeigt. Die Spalte CONST_CHECKED enthält mehrere Positionen, die am einfachsten mit folgender Abfrage zu deuten sind:

```

SELECT TABSCHEMA          AS Schema,
       TABNAME             AS Name,
       STATUS              AS Status,
       SUBSTR(CONST_CHECKED,1,1) AS Fremd_Schlüssel,
       SUBSTR(CONST_CHECKED,2,1) AS Prüfung_Integrität,
       SUBSTR(CONST_CHECKED,5,1) AS MQT,
       SUBSTR(CONST_CHECKED,6,1) AS Generierte_Spalten,
       SUBSTR(CONST_CHECKED,7,1) AS Staging_Tabellen
FROM SYSCAT.TABLES
WHERE STATE='C'

```

Die optimierte Ausgabe dieser Abfrage könnte so aussehen:

	SCHEMA	NAME	STATUS	FREMD_SCHL	PRÜFUNG_INT	MQT	GEN_SP	STAG
IVO	KUNDE	C	Y	N	Y	Y	Y	
IVO	AUFTRAG	C	N	Y	Y	Y	Y	

In dem Beispiel sehen Sie, dass in der Tabelle KUNDE und in der Tabelle AUFTRAG jeweils ein „N“ steht. Die Tabelle KUNDE enthält ein „N“ in der Spalte „Prüfung auf Integritätsbeziehungen“, und die Tabelle AUFTRAG enthält ein „N“ in der Spalte, die die „Fremdschlüssel-Beziehungen“ anzeigt. Das heißt, hier muss noch eine Prüfung erfolgen.

Um diese Prüfungen nun mit dem Befehl SET INTEGRITY... durchzuführen, sollten Sie unbedingt darauf achten, dass für die Tabellen, die im Status „C“ stehen, auch Ausnahme-Tabellen vorhanden sind, da diese bei dem Befehl SET INTEGRITY angegeben werden müssen.

Hinweis:

Wenn Sie die Ausnahme-Tabellen beim Befehl SET INTEGRITY *nicht* angeben, erhalten Sie folgende Fehlermeldung:

```
C:\buchdb\move>db2 set integrity for auftrag immediate checked
```

```
.. SQL3603N Beim Überprüfen der Daten durch die Anweisung SET INTEGRITY wurde ein
Integritätsverstoß im Zusammenhang mit der Integritätsbedingung"IVO.AUF-
TRAG.FK_KUNDENNR1" festgestellt. SQLSTATE=23514
```

Die Integrität in der Datenbank ist somit *nicht* wiederhergestellt – sie ist es erst, wenn Sie die zugehörigen Ausnahme-Tabellen mit angeben.

Die Syntax des Befehls SET INTEGRITY hat sehr viele Ausprägungen. Deswegen beschränken wir uns hier nur auf den wesentlichen Befehl, um das oben genannte Problem zu lösen. Unserer Erfahrung nach wird der folgende Befehl am häufigsten eingesetzt:

```
SET INTEGRITY FOR kunde,auftrag IMMEDIATE CHECKED FOR EXCEPTION IN kunde USE
kunde_ausnahmen, IN auftrag USE auftrag_ausnahmen;
```

Die Ausgabe nach diesem ausgeführten SQL-Befehl zeigt an, dass Daten in die Ausnahme-Tabellen übertragen wurden. Ihre Aufgabe ist es nun, diese Sätze zu prüfen, zu bereinigen und eventuell wieder in die Zieltabellen zu laden.

SQL3602W Beim Überprüfen der Daten wurden Verstöße gegen Integritätsbedingungen gefunden. Die betreffenden Daten wurden in Ausnahmetabellen übertragen. SQLSTATE=01603

Der Status der Tabellen ist nun "Normal", und auch die SQL-Abfrage über den Status bringt keine Anzeige mehr. Somit ist alles wieder in Ordnung, und Sie können mit den Tabellenobjekten weiterarbeiten.

Die SET INTEGRITY-Funktionalität lässt sich auch über die Steuerzentrale aufrufen, und zwar über das Kontextmenü der Tabelle (siehe Abbildung 8.23).

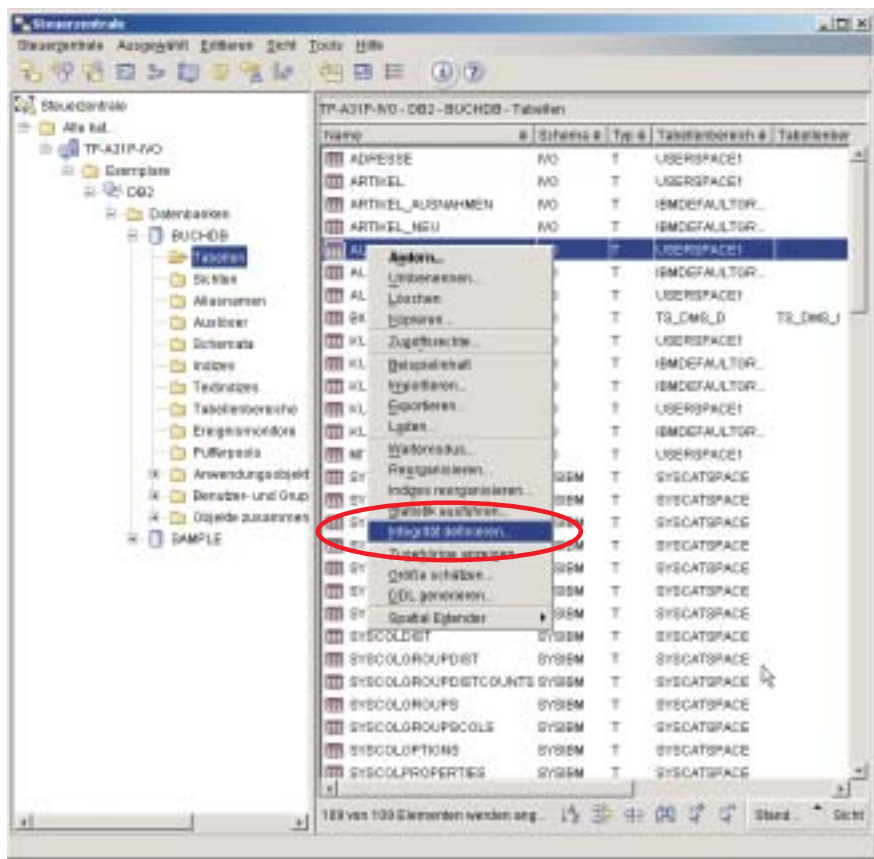


Abbildung 8.23: Integrität in der Steuerzentrale definieren

Das Menü, das nun aufgerufen wird, bietet alle Optionen und Möglichkeiten der Eingabe, die sich auf eine Tabelle beziehen.

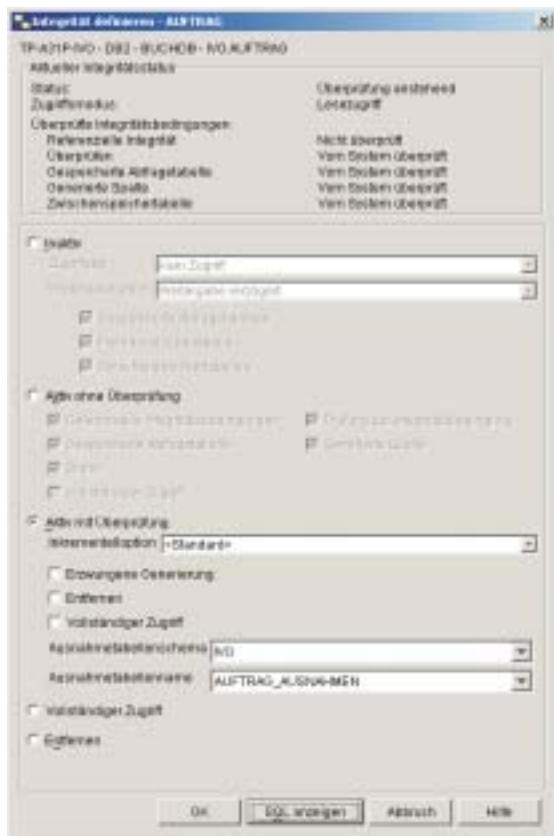


Abbildung 8.24: Optionen für die Wiederherstellung der Integrität

In Abbildung 8.24 sehen Sie im oberen Ausschnitt das Feld AKTUELLER INTEGRITÄTS-STATUS. Dies entspricht dem auf Seite 360 gezeigten SQL-Befehl, ist hier aber auf die ausgewählte Tabelle bezogen.

Backup

Wird beim LOAD die Option COPY=NO angegeben, hat der Tabellenbereich, in dem sich die zu ladende Tabelle befindet, *nach dem Start* des LOAD-Befehls den Status "Sicherung anstehend" (nur, wenn die aktualisierende Wiederherstellung aktiv ist, d.h. LOGRETAIN=ON oder USEREXIT=ON). Dieser Tabellenbereichsstatus geht erst wieder in den „Normal“-Status über, wenn eine Sicherung des Tabellenbereichs bzw. der Datenbank durchgeführt worden ist.

Der Zugriff auf die Tabelle ist weiterhin möglich, wenn beim LOAD die Option ALLOW READ ACCESS angegeben wurde; falls nicht, so ist die Option ALLOW NO ACCESS die Standardvorgabe.

Falls der Tabellenstatus „Überprüfung anstehend“ ist, muss dieser danach mit SET INTEGRITY normal bereinigt werden, wie oben schon gezeigt wurde.

8.4.7 Aktivitäten nach einem Fehler im Ladeprozess

In einigen Situationen, wie z.B. bei einem Stromausfall oder einem Plattenproblem, kann es zum Abbruch des Ladeprozesses kommen. Um zu sehen, welche Probleme der LOAD dokumentiert hat und wo der Aufsetzpunkt ist, sehen Sie am besten in der Nachrichtendatei nach. Hier sehen Sie die Phasen, die schon durchlaufen worden sind, sowie die Sicherungspunkte, falls Sie diese mit angegeben hatten.

SQL3109N Das Dienstprogramm beginnt mit dem Laden von Daten aus der Datei "C:\buchdb\daten\Kunden_neu.del".	
SQL3500W Die Phase "LOAD" wird gestartet (Zeit: "17/09/2003 22.58.29.766817").	
SQL3519W Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "0".	
SQL3520W Synchronisationspunkt für Ladevorgang erfolgreich.	
SQL3110N Die Verarbeitung durch das Dienstprogramm ist abgeschlossen. "4" Zeilen aus der Eingabedatei wurden gelesen.	
SQL3519W Synchronisationspunkt am Beginn des Ladevorgangs. Zähler für Eingabesätze: "4".	
SQL3520W Synchronisationspunkt für Ladevorgang erfolgreich.	
SQL3515W Die Phase "LOAD" wurde beendet (Zeit: "17/09/2003 22.58.30.297827").	
SQL3500W Die Phase "BUILD" wird gestartet (Zeit: "17/09/2003 22.58.30.325125").	
SQL3213I Der Indexierungsmodus ist "INCREMENTAL".	
SQL3515W Die Phase "BUILD" wurde beendet (Zeit: "17/09/2003 22.58.30.677813").	
SQL3500W Die Phase "DELETE" wird gestartet (Zeit: "17/09/2003 22.58.31.524050").	
SQL3509W Vom Dienstprogramm wurde(n) "3" Zeile(n) aus der Tabelle gelöscht.	
SQL3515W Die Phase "DELETE" wurde beendet (Zeit: "17/09/2003 22.58.31.698101").	
Anzahl gelesener Zeilen	= 4
Anzahl übersprungener Zeilen	= 0
Anzahl geladener Zeilen	= 4
Anzahl zurückgewiesener Zeilen	= 0
Anzahl gelöschter Zeilen	= 3
Anzahl festgeschriebener Zeilen	= 4

Abbildung 8.25: LOAD-Nachrichtendatei

In Abbildung 8.25 sehen Sie die LOAD-Phasen LOAD, BUILD und DELETE mit den zugehörigen Meldungen. Bricht nun der LOAD in einer dieser Phasen ab, so können Sie den LOAD-Befehl mit der Option RESTART statt mit der Option INSERT oder REPLACE nochmals starten. DB2 holt die notwendigen Informationen für den Aufsetzpunkt aus den Load-Kontrolldateien, die im TEMPFILE PATH stehen. Sie müssen nicht die Phasen und auch nicht die Synchronisationspunkte angeben, um den LOAD zu RESTARTEN.

LOAD FROM ... OF... RESTART INTO <ziel-tabelle>...

Vorsicht:

Löschen Sie *nie* Daten aus dem TEMPFILE PATH bzw. aus dem Standard-LOAD-Verzeichnis, da in diesem alle Protokolle, Kontrolldateien und auch die Aufsetzpunkte dokumentiert sind.

Was tun Sie, wenn Ihnen erst auffällt, dass sich Fehler eingeschlichen haben, wenn der LOAD schon läuft? So könnten Sie sich z.B. in der Eingabedatei vertippt haben, oder Sie haben die falsche Tabelle geladen. Diese unnötigen Einträge in die Nachrichtendatei und die vielen abgewiesenen Sätze belasten ein System unnötig. Die beste Lösung ist, den LOAD abubrechen. Ein Abbruch des LOAD wird durch die Option TERMINATE herbeigeführt. Sie ist anstelle der INSERT- oder REPLACE-Option zu setzen; alle anderen Angaben können Sie beibehalten.

```
LOAD FROM .. TERMINATE INTO <ziel-tabelle>
```

Was passiert beim TERMINATE?

Nach dem TERMINATE des LOAD haben Sie den alten Zustand wieder, falls Sie die Daten mit der Option INSERT geladen haben.

Haben Sie jedoch mit der Option REPLACE die Daten geladen, also die bestehenden Daten in der Zieltabelle komplett ersetzt, so ist die Tabelle nach dem Absetzen des LOAD mit der TERMINATE-Option *leer* – die Tabelle hat nun „0“ Sätze. Ein Index der eventuell mit aufgebaut wurde, wird als „defekt“ gekennzeichnet und beim ersten Zugriff neu aufgebaut.

Tipp:

Der Zeitpunkt für die Indexneuerstellung (INDEXREC)=ACCESS wird in der Datenbankmanager-Konfiguration gesetzt und ist standardmäßig auf folgende Werte eingestellt:

ACCESS	bei Windows-Systemen
RESTART	bei Unix-Systemen

Falls Sie die Option COPY=NO verwendet haben und somit DB2 veranlassen, eine Sicherung des Tabellenbereichs zu erzwingen (nur im Falle einer aktualisierenden Wiederherstellung, LOGRETAIN=ON oder USEREXIT=ON), so bleibt diese Anforderung auch nach einem TERMINATE-Befehl bestehen. Sie müssen also eine Sicherung durchführen, um den Status "Sicherung anstehend" auf "Normal" umzusetzen.

8.5 Tipps aus der Praxis

8.5.1 Übersicht und Vergleich von Import und Load

Um Ihnen die Entscheidung zu erleichtern, welches Tool Sie zum Füllen der Datenbank einsetzen, haben wir in Tabelle 8.2 die verschiedenen Eigenschaften der Tools gegenübergestellt.

Eigenschaft	IMPORT	LOAD
Geschwindigkeit	langsam	schnell
Administrativer Aufwand	gering	höher
Protokollierung	alles (alle Zeilen)	minimal
Unterstützte Datei-Formate	ASC, DEL, WSF, IXF	ASC, DEL, IXF
Lade-Ziel	Sichten und Tabellen, aber <i>keine</i> Unterstützung für gespeicherte Abfragetabellen (MQTs)	Bestehende Tabellen und gespeicherte Abfragetabellen (MQTs)
Optionen zur Optimierung	COMMITCOUNT und COMPOUND	Zahlreiche Optionen, z.B. FASTPARSE
Unterstützung von Auslösern (Trigger)	ja	nein
Überprüfung der Integritätsbedingungen	während des Imports	nachträglich durch SET INTEGRITY
Wiederaufsetzbarkeit im Fehlerfall	ja, durch den Einsatz COMMITCOUNT können die restlichen Zeilen der Tabelle importiert werden (RESTARTCOUNT), oder der (festgeschriebene) Inhalt wird so akzeptiert.	ja, durch den Einsatz von SAVECOUNT wird nach einem gesetzten Konsistenzpunkt automatisch wieder aufgesetzt.

Tabelle 8.2: Übersicht und Vergleich von Import und Load

8.5.2 DB2MOVE-Tool

Mit diesem Tool kann eine Datenbank oder auch nur eine große Anzahl von Tabellen von einem Server auf einen anderen Server verschoben werden. Dies kann notwendig werden, wenn Sie eine Datenbank von einem Windows-System auf ein Unix-System verlagern müssen. Zurzeit ist es nicht möglich, ein Backup von einem Windows-System auf ein Unix-System und umgekehrt zurückzuspielen. Auch bei der Verlagerung von mehreren Tabellen ist es hilfreich, wenn man sich nicht alle Scripts für die Exports und Imports/Loads selbst schreiben muss, sondern hier Tools in DB2 nutzen kann.

Das Tool hat drei Optionen: eine EXPORT-Option, mit der die Daten aus einer Datenbank exportiert werden, und zwei Optionen zum Einfügen der Daten: die IMPORT- und die LOAD-Option.

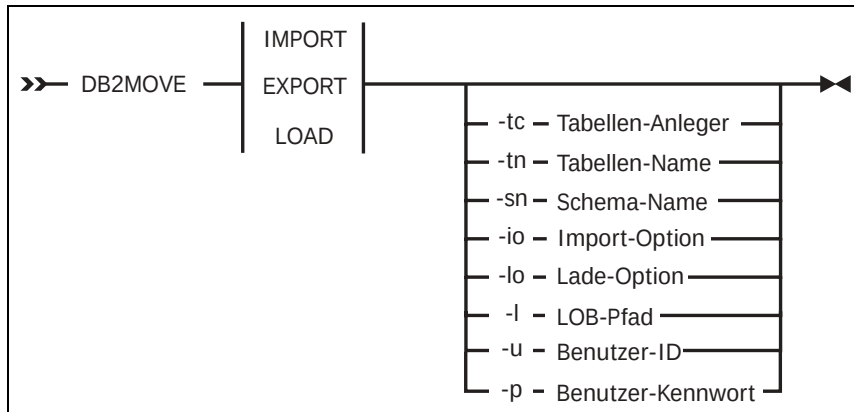


Abbildung 8.26: DB2MOVE-Syntaxdiagramm

Mit diesen Optionen können Sie folgende Auswahl treffen:

► **-tc – Tabellen-Anleger**

Wird hier kein Wert angegeben, so werden alle Tabellen-Anleger ausgewählt. Es kann auch das Platzhalterzeichen, der „*“, verwendet werden, um die Auswahl einzuschränken, z.B. **BUCH*** oder **Test***.

► **-tn – Tabellen-Name**

Der Standard ist alle Tabellen des Benutzers.

► **-sn – Schema-Name**

Wenn der Wert nicht angegeben wird, werden alle Schemata genommen. Auch hier kann mit Platzhaltern gearbeitet werden. Zum Beispiel kann für den Schema-Namen **ORHANOVIC** auch **-sn OR*N*C*** eingegeben werden.

► **-io – Import-Option**

Die Standardeinstellung ist: **REPLACE_CREATE**.

Mögliche Werte sind hier: **INSERT**, **INSERT_UPDATE**, **REPLACE**, **CREATE** und **REPLACE_CREATE**.

► **-lo – Lade-Option**

INSERT – ist die Standardeinstellung; eine weitere mögliche Option ist **REPLACE**.

► **-l – LOB-Pfade**

Der Standardpfad für die LOBs ist das aktuelle Verzeichnis.

► **-u – Benutzer-ID**

Die aktuell genutzte Benutzer-ID wird übernommen, wenn nichts anderes angegeben ist.

► **-p – Benutzer-Kennwort**

Das aktuelle Kennwort wird übernommen, wenn der Wert nicht gesetzt wird.

Wenn ein Export über das DB2MOVE-Tool durchgeführt wird, z.B. mit

```
DB2MOVE buchdb EXPORT
```

werden die Inhalte der Tabellen im IXF-Format abgelegt, und es wird eine Kontrolldatei erzeugt (db2move.lst), die die Beziehungen der Tabellendaten zu den Exportdateien aufzeigt. Diese Kontrolldatei ist für den späteren Import oder Load notwendig, damit das DB2MOVE-Tool Informationen darüber bekommt, welche Tabelle sich in welcher Datei befindet.

Der Inhalt von db2move.lst sieht so aus:

```
!"IVO      ". "ADRESSE"!tab1.ixf!tab1.msg!
!"IVO      ". "KUNDE"!tab2.ixf!tab2.msg!
!"IVO      ". "KUNDENADRESSE"!tab3.ixf!tab3.msg!
!"IVO      ". "ARTIKEL"!tab5.ixf!tab5.msg!
!"IVO      ". "AUFTRAGSPOSITIONEN"!tab6.ixf!tab6.msg!
!"IVO      ". "MITARB"!tab7.ixf!tab7.msg!
!"IVO      ". "BIG_TAB1"!tab8.ixf!tab8.msg!
!"IVO      ". "TEMPMDC"!tab9.ixf!tab9.msg!
!"IVO      ". "KUNDEN_NEU"!tab10.ixf!tab10.msg!
!"IVO      ". "KUNDE_AUSNAHMEN"!tab11.ixf!tab11.msg!
!"IVO      ". "KUNDENKONTO"!tab12.ixf!tab12.msg!
!"IVO      ". "ARTIKEL_NEU"!tab13.ixf!tab13.msg!
!"IVO      ". "ARTIKEL_AUSNAHMEN"!tab14.ixf!tab14.msg!
!"IVO      ". "AUFTRAG_AUSNAHMEN"!tab15.ixf!tab15.msg!
```

Eine Nachrichtendatei EXPORT.out enthält Informationen über Fehlermeldungen sowie die Satzanzahl, die pro Tabelle exportiert wurde.

Wie Sie hier sehen, sind die Outputdateien einfach durchnummeriert (siehe TABnnn.ixf und TABnnn.msg).

Um die Daten nun in die neue Datenbank zu bekommen, ist der einfachste Befehl:

```
db2move buchdbN import
```

Doch in der Praxis wird dieser Befehl sehr selten verwendet, da bei der einfachen Variante alle Tabellen in den Standard-Tabellenbereich (z.B. USERSPACE1) geschrieben werden und es hier keine Option gibt, um gezielt die Daten in die jeweiligen Tabellenbereiche zu laden.

Somit müssen Sie etwas mehr Vorbereitungen treffen:

- Eine neue Datenbank muss vorhanden sein.
- Legen Sie Tabellenbereiche und Tabellen mit den zugehörigen Integritätsregeln und Indizes in der neuen Datenbank an. Dies geschieht am besten mit dem DB2LOOK-Tool, das schon in *Abschnitt 7.16.1* angesprochen wurde, z.B.:

```
db2look -d BUCHDB -u IVO -e -l -x -c -o Tabellen.ddl
```

- ▶ Führen Sie das von DB2LOOK generierte DDL-Script auf der neuen Datenbank aus:
`db2 -tf Tabellen.ddl`
- ▶ Starten Sie nun das DB2MOVE-Tool aus dem Verzeichnis, in dem sich die IXF-Daten (tabnnn.ixf) und die Kontrolldatei (db2move.lst) befinden:
`DB2MOVE buchdbN LOAD`
- ▶ Zum Schluss überprüfen Sie in der LOAD.msg-Datei, ob auch alles fehlerfrei abgelaufen ist.

9 Sicherung und Wiederherstellung einer Datenbank

Eine Datenbank kann jederzeit unbrauchbar werden. Die Ursache dafür kann ein Stromausfall, ein Hardwareproblem oder ein Softwarefehler sein. Der häufigste Fall ist jedoch, dass der Benutzer einen Fehler innerhalb der Datenbank verursacht.

Um Ihre Daten vor all diesen Ausfällen zu schützen, müssen Sie ein gutes Sicherungskonzept haben. Dessen Ziel ist es, bei einem Fehler auf den zeitlich nächsten Punkt vor dem Fehler aufsetzen zu können, also so, dass möglichst keine Daten und Veränderungen an der Datenbank verloren gehen.

DB2 bietet unterschiedliche Konzepte und Möglichkeiten für die Sicherung und Wiederherstellung einer Datenbank. Dazu gehört, dass alle Veränderungen innerhalb der Datenbank protokolliert werden.

Bevor wir auf die Feinheiten der Sicherungstechniken eingehen, zeigen wir Ihnen, wie die Protokollierung innerhalb der DB2-Datenbank grundsätzlich abläuft.

9.1 Protokollierung in einer Datenbank

Jeder DB2-Datenbank sind Protokolldateien (logfiles) zugeordnet. In diesen Protokolldateien werden die Veränderungen in der Datenbank aufgezeichnet. DB2 protokolliert jede Transaktion – mit wenigen Ausnahmen, wie z.B. beim Einsatz des Load-Tools. Dieses Tool lädt Daten in die Datenbank, ohne die Veränderungen in den Protokolldateien zu dokumentieren. Das Load-Tool haben wir in Kapitel 8 schon genauer behandelt.

In Abbildung 9.1 sehen Sie, welche Prozesse innerhalb der Datenbank ablaufen bzw. welche Komponenten mit eingebunden sind, wenn Sie eine Veränderung (hier im Beispiel ist es ein SQL-Update-Befehl) ausführen. Diese Komponenten können alle einzeln durch Konfigurationsparameter verändert werden.

Um Ihnen ein Grundverständnis für die Protokollierung innerhalb der DB2-Datenbank zu vermitteln, zeigen wir Ihnen im Folgenden schrittweise, wie eine Veränderung abläuft:

1. Aus einer Anwendung heraus setzen Sie einen SQL-Befehl ab, der Daten verändert:

```
UPDATE ...SET A=B...WHERE ..  
COMMIT
```

Das ist eine vereinfachte Darstellung einer UOW (Unit Of Work), also einer Transaktion.

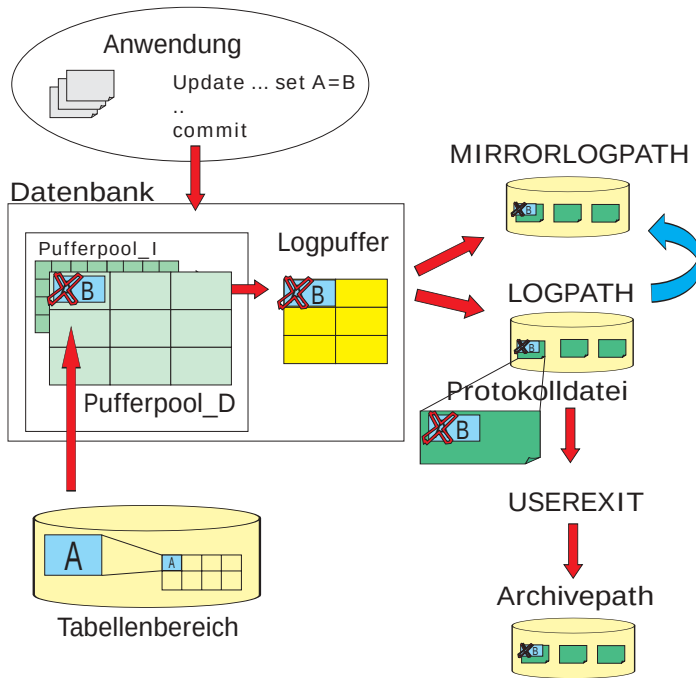


Abbildung 9.1: Die wichtigsten Komponenten für das Protokollieren

2. Der zu ändernde Datensatz wird von der Datenbank aus dem zugehörigen Tabellenbereich (table space) gelesen und in den zugehörigen Pufferpool geschrieben. In unserem Beispiel ist das der Pufferpool D. Die Seite wird jetzt intern gesperrt und mit einer exklusiven Sperrung (lock) belegt.
3. Die Veränderung – hier wird der Wert A durch B ersetzt – wird als Erstes im Logpuffer festgeschrieben. Der Ursprungszustand und die Veränderung werden festgehalten. Die geplante Änderung (hier von A nach B) wird auf der Seite vorgenommen.
4. Wenn in der Anwendung nun der COMMIT-Befehl abgesetzt wird, wird dieser COMMIT erst dann von DB2 bestätigt, wenn der Logpuffer komplett in die Protokolldateien (logfiles) geschrieben worden ist. Die Protokolldateien befinden sich im Logpath. Mit dem Datenbank-Konfigurationsparameter LOGPATH bestimmen Sie die physikalische Lage der Protokolldateien.
5. Dieser Datensatz wird nun wieder freigegeben, d.h., die gesetzte Sperrung wird entfernt. Die Datenseite im Pufferpool wird als verändert („dirty“) markiert, da diese Veränderung noch nicht auf die Platte bzw. in die Tabelle zurückgeschrieben worden ist. Bisher wurde nur die Veränderungsinformation in die Protokolldateien geschrieben.

Jetzt fragen Sie sich sicherlich: Wann werden die Daten aus dem Pufferpool wieder in die Tabellen zurückgeschrieben?

Die Antwort hat hauptsächlich mit DB2-Optimierungsgesichtspunkten zu tun. Um die E/A-Last auf die Plattensysteme zu minimieren, werden „benutzte“ („dirty“) Datenseiten versetzt – also asynchron – geschrieben.

Folgende Fälle führen dazu, dass die veränderten Seiten auf die Plattensysteme geschrieben werden:

- ▶ wenn kein Platz mehr im Pufferpool ist. In diesem Fall müssen die veränderten Seiten sogar synchron auf die zugehörigen Datenträger geschrieben werden.
- ▶ wenn Schwellenwerte in der Datenbankkonfiguration erreicht werden:
 - „Schwellenwert für geänderte Seiten“ (CHNGPGS_THRESH)
 - „Vor bedingtem Prüfpunkt zu schreibende Protokollsätze“ (SOFTMAX)
- ▶ nach einer COMMIT-Anweisung, wenn Änderungen von Tabellendaten nicht protokolliert wurden (NOT LOGGED INITIALLY).

Wenn Sie befürchten, dass Ihre Protokolldateien beschädigt werden könnten, z.B. aufgrund eines Hardwarefehlers (Plattenfehler, Kontrolleinheit), dann sollten Sie über die Möglichkeit nachdenken, die Protokolldateien zu spiegeln. Hier gibt es zwei Möglichkeiten: einmal die Hardware-Lösung und zum Zweiten die DB2-Variante. Bei der Hardware-Lösung werden die Daten vom Plattensystem gespiegelt, also automatisch doppelt gehalten. Die DB2-Variante schützt Sie zusätzlich vor dem versehentlichen Löschen der Protokolldateien durch einen „berechtigten“ Benutzer. Dazu schreibt DB2 die Protokolldateien zusätzlich in einen zweiten Pfad (MIRRORLOGPATH; siehe Abbildung 9.1).

9.1.1 Arten von Protokolldateien

Je nach Verfügbarkeit und dem jeweiligen Zustand von Protokolldateien werden diese unterschiedlich bezeichnet. In Abbildung 9.2 sehen Sie eine genauere Darstellung der möglichen Zustände und der Orte, an denen sich die Protokolldateien befinden können.

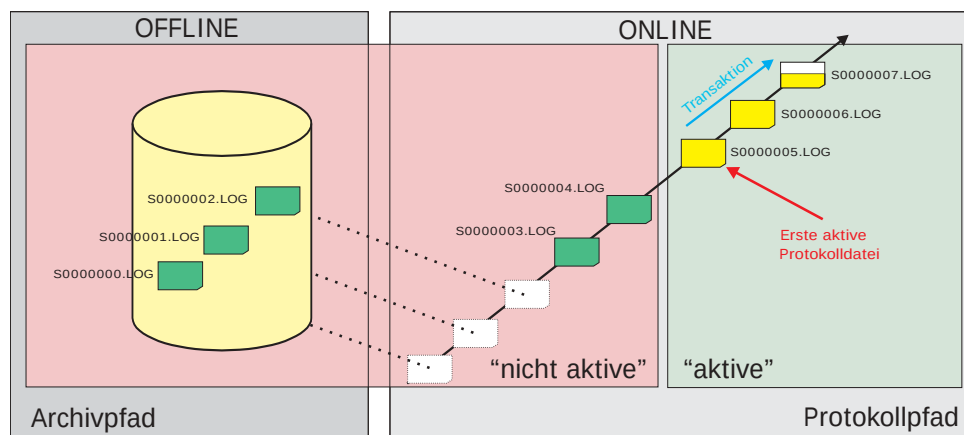


Abbildung 9.2: Arten von Protokolldateien

Die Protokolldateien werden von DB2 immer in den Protokollpfad (logpath) geschrieben. Werden diese Protokolldateien für eine laufende Transaktion verwendet, spricht man von *aktiven Protokolldateien*. Die erste Protokolldatei, die eine nicht abgeschlossene Transaktion beinhaltet, erkennen Sie an dem informativen Status Erste aktive Protokolldatei in der Datenbankkonfiguration. In der Abbildung 9.2 ist das S0000005.LOG.

Wird eine Protokolldatei nicht mehr für eine CRASH-Recovery oder ein ROLLBACK benötigt, so wird diese geschlossen, bleibt aber im Protokollpfad bestehen (S0000003.LOG - S0000004.LOG). Die Protokolldatei wird in diesem Zustand als *nicht aktiv bezeichnet*.

ONLINE-Protokolldateien sind dadurch gekennzeichnet, dass sie im Protokollpfad der Datenbank liegen. Werden die *nicht aktiven Protokolldateien* in ein anderes Verzeichnis verschoben – dies kann manuell oder über ein Programm (z.B. Benutzer-Exit) erfolgen –, sind diese Protokolldateien nicht mehr ONLINE, sondern OFFLINE.

Im Unterschied zu ONLINE-Protokolldateien müssen OFFLINE-Protokolldateien bei einer aktualisierenden Wiederherstellung erst aus einem anderen Pfad (Archivpfad) geholt werden.

9.1.2 Umlaufprotokollierung (circular logging)

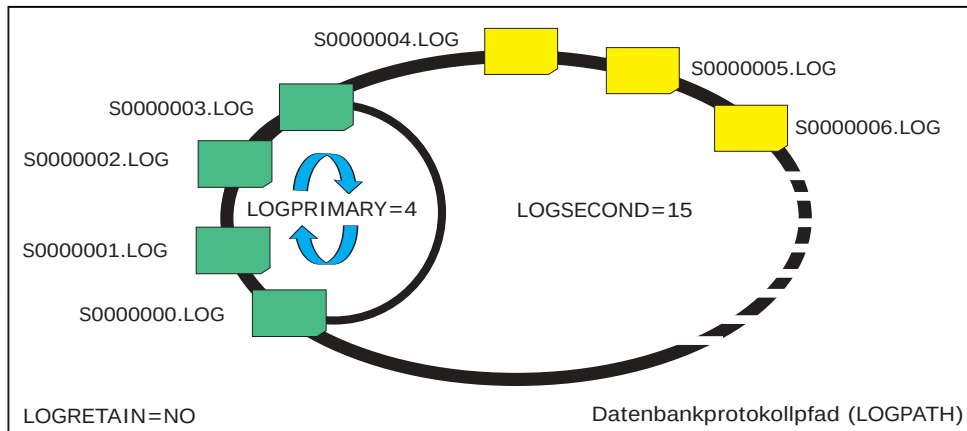


Abbildung 9.3: Umlaufprotokollierung

In Abbildung 9.3 sehen Sie die Standardeinstellung in DB2 für die Protokollierung innerhalb der Datenbank, die *Umlaufprotokollierung*. Wie der Name schon andeutet, verwendet hier die Datenbank einen Ring von Protokolldateien. Die Anzahl der Protokolldateien ist von den DB-Konfigurationsparametern LOGPRIMARY und LOGSECOND abhängig (Beispiel: LOGPRIMARY=4; S0000000.LOG bis S0000003.LOG). Diese Technik ermöglicht eine Wiederherstellung nach Systemabstürzen oder Transaktionsfehlern.

Hier werden die Protokolldateien nur so lange verwendet und behalten, wie es erforderlich ist, um die Integrität bestehender Transaktionen zu gewährleisten.

Wenn für eine aktive Transaktion die primären Protokolldateien (in Abbildung 9.3: S0000000.LOG bis S0000003.LOG) nicht ausreichen, werden so lange sekundäre Protokolldateien angelegt, bis der Wert LOGSECOND erreicht ist.

Der Unterschied zwischen primären und sekundären Protokolldateien besteht darin, dass die primären Protokolldateien aus Performance-Gründen im Voraus angelegt werden; die sekundären werden erst bei Bedarf angelegt.

Die Umlaufprotokollierung kann nicht verwendet werden, um frühere Transaktionen aktualisierend wiederherzustellen, da die Protokolldateien ständig wieder überschrieben werden. Diese Art der Protokollierung ist nur für eine CRASH- und Versionswiederherstellung geeignet (siehe Abschnitt 9.2 „Sicherungs- und Wiederherstellungskonzepte“).

9.1.3 Beibehalten von Protokolldateien (Logretain=YES)

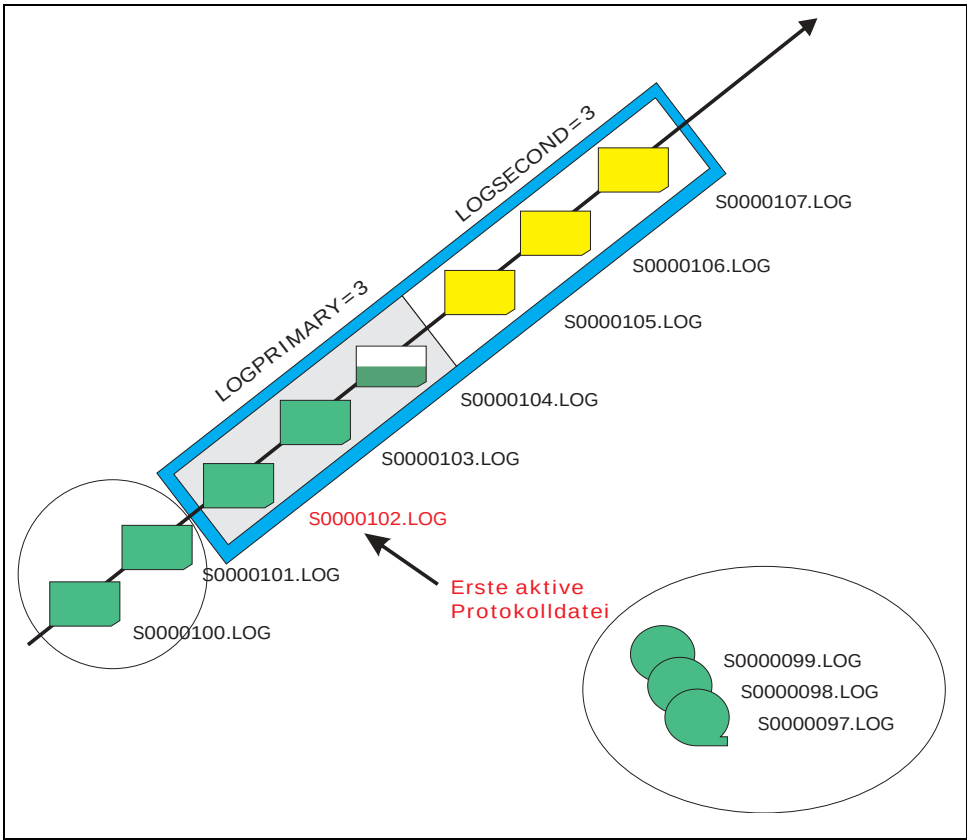


Abbildung 9.4: Archivprotokollierung

Im Gegensatz zur Umlaufprotokollierung wird bei der Archivprotokollierung keine einzige Protokolldatei überschrieben, sondern es wird immer eine neue Protokoll-

datei angelegt. Auch hier wird die Anzahl der primären Protokolldateien von dem DB-Konfigurationsparameter LOGPRIMARY gesteuert.

Die Archivprotokollierung hat mehrere Vorteile:

- Es ist möglich, eine Online-Sicherung zu erstellen.
Das heißt, zum Zeitpunkt der Sicherung sind weitere Aktivitäten (auch Veränderungen) auf der Datenbank möglich.
- Es ist möglich, nur eine Teilsicherung der Datenbank durchzuführen.
Diese umfasst Tabellenbereiche, die inkrementelle Sicherung und die Delta-Sicherung.
- Es ist eine aktualisierende Wiederherstellung möglich.
Das heißt, die Datenbank kann mit der Sicherung und mit Hilfe der Protokolldateien zu fast jedem gewünschten Zeitpunkt wiederhergestellt werden.

9.1.4 Wie stellen Sie von Umlaufprotokollierung auf Archivprotokollierung um?

Die Umstellung der Umlaufprotokollierung auf die Archivprotokollierung wird oft auch als Wechsel aus dem OFFLINE-Sicherungsmodus in den ONLINE-Sicherungsmodus bezeichnet. Nur wenn Sie die Archivprotokollierung einsetzen, sind Sie in der Lage, eine Online-Sicherung bzw. auch eine Teilsicherung der Datenbank durchzuführen.

Es gibt zwei Möglichkeiten der Umstellung: einmal die Variante über das grafische Tool von DB2, die Steuerzentrale, und die zweite Möglichkeit sind DB2-Befehle von der Kommandozeile.

Wir zeigen Ihnen erst die Variante über die Steuerzentrale, da Sie hier geführt werden und somit keine DB2-Befehle beherrschen müssen.

In der Steuerzentrale wählen Sie über den Menüpunkt TOOLS den ASSISTENTEN-Eintrag aus. Im ASSISTENTEN-Fenster (siehe Abbildung 9.5) selektieren Sie ASSISTENT: DATENBANKPROTOKOLlierung konfigurieren.



Abbildung 9.5: Assistent: Datenbankprotokollierung konfigurieren

Im nächsten Fenster (siehe Abbildung 9.6) müssen Sie nun die Auswahl ARCHIVPROTOKOLLIERUNG anklicken.

Dabei wird Ihnen im unteren Teil des Fensters aufgezeigt, welche Auswirkungen diese Änderung haben wird:

- ▶ ONLINESICHERUNGEN SIND MÖGLICH
- ▶ TABELLENBEREICHSSICHERUNGEN SIND MÖGLICH
- ▶ KANN BIS ZU EINEM BELIEBIGEN ZEITPUNKT WIEDERHERGESTELLT WERDEN
- ▶ AUTOMATISCHE VERWALTUNG VON PROTOKOLLDATEI (ist bei dieser Einstellung nicht möglich)

Der letzte Punkt ist hier nur informativ, da dies über eine weitere Option (siehe Abbildung 9.7) verändert werden kann.

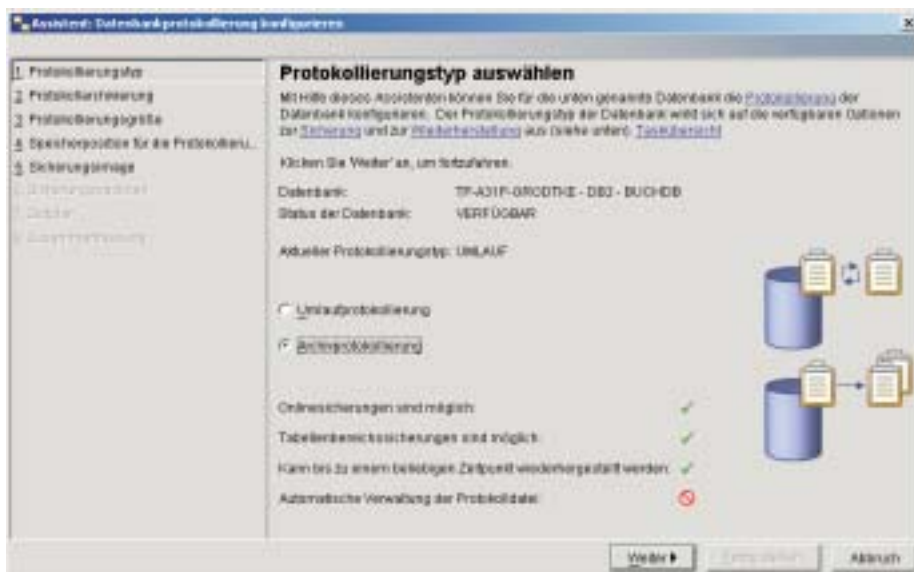


Abbildung 9.6: Assistent: Protokollierung konfigurieren – Datenbankprotokollierungstyp

Das nächste Fenster (siehe Abbildung 9.7) gibt Ihnen die Option, sich für eine automatische Protokolldatei-Archivierung zu entscheiden. Dies erfolgt mit Hilfe des Benutzerexit-Programms, das aber vorher noch konfiguriert werden muss. Es gibt zu diesem Zeitpunkt leider nicht die Möglichkeit, dieses Benutzerexit-Programm über einen Dialog anzupassen; dies muss zu einem späteren Zeitpunkt erfolgen. Weitere Details dazu finden Sie in Kapitel Benutzerexit (Userexit).

Im folgenden Fenster (siehe Abbildung 9.8) werden die Werte für die primären und die sekundären Protokolldateien gesetzt sowie die Größe der Protokolldateien (in 4-Kbyte-Seiten).



Abbildung 9.7: Assistent: Protokollierung konfigurieren – Protokollarchivierung

An dieser Stelle berechnet DB2 Ihnen gleich den Platzbedarf, den die Protokoll-dateien mit der gewählten Konfiguration im Dateisystem benötigen werden.



Abbildung 9.8: Assistent: Protokollierung konfigurieren – Protokollierungsgröße

In Schritt 4 (siehe Abbildung 9.9) können Sie den Pfad für die Protokolldateien (logpath) und für die mögliche Spiegelung (mirrorlogpath) angeben bzw. die Standardwerte für den Protokollpfad übernehmen.



Abbildung 9.9: Assistent: Protokollierung konfigurieren – Protokoll- und Spiegelungspfad

Im Schritt 5 werden Sie nach dem Pfad gefragt, in dem Sie die Sicherung Ihrer Datenbank anlegen wollen (siehe Abbildung 9.10).



Abbildung 9.10: Assistent: Protokollierung konfigurieren – Sicherungsbild

Bei den Sicherungsoptionen (siehe Abbildung 9.11) können Sie alle Vorgaben übernehmen. Sie sollten nur darauf achten, dass die Anzahl der Puffer immer doppelt so groß ist wie die Anzahl der Ausgabeeinheiten. Ab dem Fixpack 3 im DB2 UDB V8 wird dies auch vom Assistenten schon beachtet.



Abbildung 9.11: Assistent: Protokollierung konfigurieren – Sicherungsoptionen

Im Zeitplanfenster (siehe Abbildung 9.12) kann eingestellt werden, dass die Umstellung sofort ausgeführt wird oder dass die Umstellung als Script (eventuell zeitgesteuert) in der Taskzentrale angelegt wird. In der Praxis wird hier nie die Taskzentrale eingesetzt, sondern es wird immer **JETZT AUSFÜHREN...** verwendet, um die Änderungen und die Sicherung *sofort* durchzuführen.

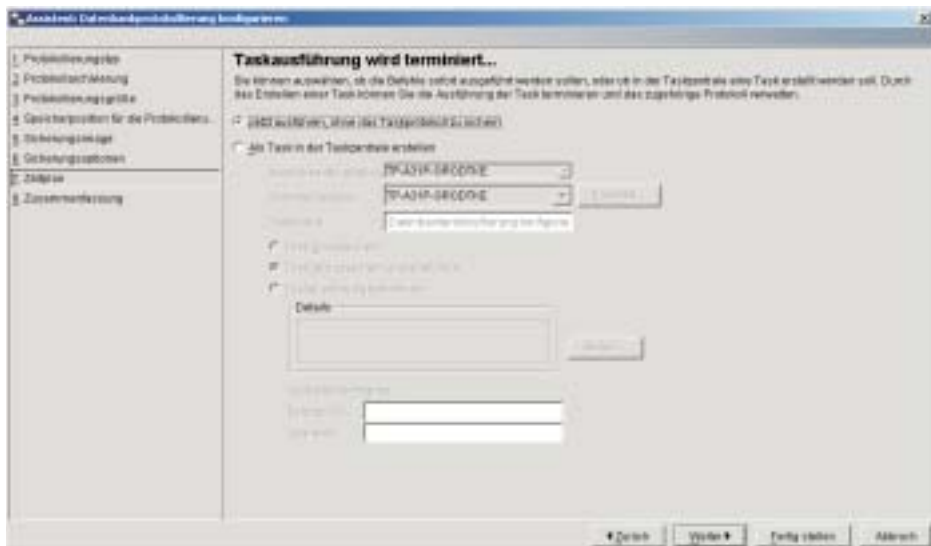


Abbildung 9.12: Assistent: Protokollierung konfigurieren – Zeitplan

Die Auflistung aller gewählten Optionen sehen Sie in dem Fenster aus Abbildung 9.13. Hier sollten Sie sich immer den Befehl anschauen, der von DB2 generiert wird, um einen Einblick in die DB2-Syntax zu bekommen.

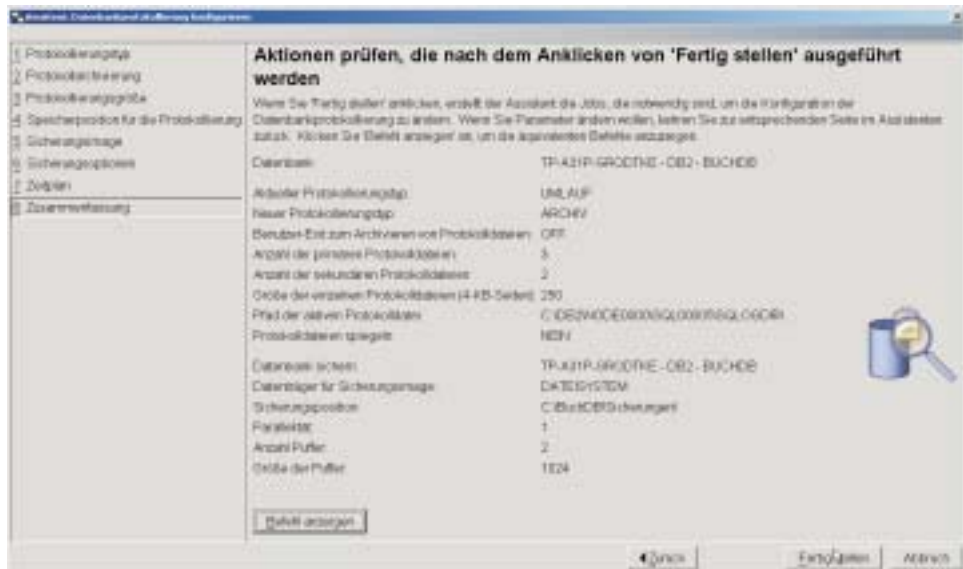


Abbildung 9.13: Assistent: Protokollierung konfigurieren – Zusammenfassung

Wenn Sie auf den Button **BEFEHL ANZEIGEN** (siehe Abbildung 9.13) klicken, dann erhalten Sie den Bildschirm aus Abbildung 9.14 mit den Befehlen, die ausgeführt werden.



Abbildung 9.14: Assistent: Protokollierung konfigurieren – Befehl anzeigen

Die Informationen aus dieser Anzeige könnten Sie auch verwenden, um diese Aktion gleich von der Kommandozeile auszuführen.

Es geht noch einfacher, wenn Sie weder die Anzahl noch die Größe der Protokolldateien ändern wollen. Dann reicht es, wenn Sie folgende DB2-Befehle absetzen:

```
connect to buchdb;
update db cfg for buchdb using logretain on
quiesce database immediate force connections;
connect reset;
backup database buchdb to C:\buchdb\sicherungen;
connect to buchdb;
unquiesce buchdb;
connect reset;
```

Näheres zum Einsatz des Befehls QUIESCE DATABASE finden Sie in Kapitel 5.9.3.

9.1.5 Welche Protokollierungsparameter gibt es?

Mit dem Befehl

- Windows: `get db cfg for buchdb | find /i "Protokoll"`
- Unix: `get db cfg for buchdb | grep -i "Protokoll"`

können Sie sich alle Parameter, die das Wort „Protokoll“ beinhalten, anzeigen lassen, also alle relevanten Protokollkonfigurationen und -informationen.

Protokollpuffergröße (4 KB)	(LOGBUFSZ)= 8
Geänderte Seiten protokollieren	(TRACKMOD)= OFF
Protokolldateigröße (4 KB)	(LOGFILSIZ)= 4
Anzahl primärer Protokolldateien	(LOGPRIMARY)= 5
Anzahl sekundärer Protokolldateien	(LOGSECOND)= 2
Datenbankprotokollpfad ändern	(NEWLOGPATH)=
Pfad zu Protokolldateien	= C:\DB2\NODE0000\
SQL00001\SQLLOGDIR\	
Überlaufprotokollpfad	(OVERFLOWLOGPATH)=
Pfad für Protokollspiegelung	(MIRRORLOGPATH)=
Erste aktive Protokolldatei	= S0000102.LOG
Blockprotokoll auf Datenträger voll	(BLK_LOG_DSK_FUL)= NO
Anz. aktiver Protokolldateien f. akt. Arbeitseinheit	(NUM_LOG_SPAN)= 0
Vor bed. Prüfpkt. Zur schreibende Protokollsätze	(SOFTMAX)= 100
Protokollspeicherung für Wiederherstellung	(LOGRETAIN)= RECOVERY
Benutzerexit für Protokollierung aktivieren	(USEREXIT)= OFF

Abbildung 9.15: Protokollierungsparameter

Die DB2-Konfigurationsparameter sind über den folgenden Befehl zu ändern:

```
update db cfg for buchdb using <LOGPARAMETER> <WERT>
```

- LOGRETAIN – Das Beibehalten von Protokollen aktivieren

– NO

gibt an, dass die Protokolle nicht aufgehoben werden sollen, sondern überschrieben werden können. Dies entspricht der Umlaufprotokollierung.

– RECOVERY

gibt an, dass die Protokolle *nicht* überschrieben werden und somit zur aktualisierenden Wiederherstellung verwendet werden können.

Ist LOGRETAIN auf "RECOVERY" oder USEREXIT auf "YES" gesetzt, werden die aktiven Protokolldateien gespeichert.

Nachdem Sie LOGRETAIN auf "RECOVERY" und/oder USEREXIT auf "YES" gesetzt haben, müssen Sie eine Gesamtsicherung der Datenbank vornehmen. Dazu werden Sie von DB2 gezwungen, das eine Markierung „Sicherung anstehend“ in der Datenbank setzt, die einen Zugriff verhindert. Das heißt, dass der Status der Datenbank erst umgeändert wird, wenn Sie eine Gesamtsicherung durchgeführt haben. Den Status bekommen Sie am Markierungsparameter Sicherung anstehend = YES (backup_pending = YES) in der Datenbankkonfiguration angezeigt.

► USEREXIT

Dieser Parameter ist vom LOGRETAIN-Wert unabhängig. Wird USEREXIT=ON gesetzt, setzt DB2 auch automatisch den Parameter LOGRETAIN auf den Wert RECOVERY. Der LOGRETAIN-Wert muss also nicht noch einmal zusätzlich gesetzt werden.

Wenn der Benutzerexit-Parameter eingeschaltet wird, werden die Protokolldateien, die zu 100% gefüllt sind, in den Archiv-Pfad kopiert, der im Benutzerexit-Programm konfiguriert ist.

Wird ein ROLLFORWARD, also eine aktualisierende Wiederherstellung durchgeführt, kann das Benutzerexit-Programm die archivierten Protokolldateien zurückholen.

► LOGPRIMARY

Legt die Anzahl der im Voraus reservierten Protokolldateien fest. Diese belegen einen fest definierten Speicherplatz, der mit dem Parameter LOGFILSIZ gesetzt wird.

Veränderungen an diesem Wert sind nicht dynamisch. Zur Aktivierung eines neuen Wertes ist eine inaktive Datenbank notwendig. Dies erreichen Sie durch einen FORCE APPLICATIONS (<Anwendungskennzeichen>, <Anwendungskennzeichen>)-Befehl, der bestehende Verbindungen zur Datenbank unterbricht.

► LOGSECOND

Legt die maximale Anzahl der zusätzlichen Protokolldateien fest, die bei Bedarf angelegt werden. Wenn also für eine Transaktion die primären Protokolldateien nicht ausreichen, dann greift DB2 auf die sekundären Protokolldatei zu bzw. legt sich eine neue Protokolldatei an. Dies erfolgt so lange, bis der Wert von LOGSECOND erreicht ist.

Die maximale Gesamtgröße des Protokolls ergibt sich aus der Summe der primären und sekundären Protokolldateien. Reicht dieser Platz nicht aus, wird eine Fehlermeldung ausgegeben, und aktive Transaktionen werden wieder auf den letzten konsistenten Punkt zurückgeführt. In einigen Fällen, wenn z.B. die Festplattenkapazität nicht mehr ausreicht, um weitere Protokolldateien anzulegen, kann dies zum Stillstand der Datenbank führen.

Beachten Sie unbedingt, dass der Wert von LOGSECOND „dynamisch“ ist (Weitergabeklasse: sofort).

In der Praxis ist dies sehr wichtig! Wenn Sie sehen, dass die Protokolldateien schneller fortgeschrieben werden, als Sie geplant hatten, und Ihre sekundären Protokolldateien fast ausgereizt sind, dann haben Sie nun (erst seit DB2 UDB V8.1)

die Möglichkeit, den Wert im laufenden Betrieb hochzusetzen. So verhindern Sie, dass Ihre Datenbank zum Stehen kommt bzw. Ihre lange laufende Transaktion abbricht.

► LOGFILSIZ

Dieser Parameter gibt die Größe der primären und sekundären Protokolldateien als Anzahl von 4-Kbyte-Seiten an.

► NEWLOGPATH

Mit diesem Parameter bereiten Sie die Datenbank darauf vor, den Standard-LOGPATH auf einen neuen Wert zu setzen. Der Pfad im NEWLOGPATH wird erst dann zum Standard-LOGPATH, wenn die Datenbank in einen konsistenten Zustand kommt. Dies ist erst der Fall, wenn die Statusvariable Datenbank ist konsistent auf YES steht bzw. alle Applikationen sich von der Datenbank getrennt haben.

► OVERFLOWLOGPATH

Werte, die hier gesetzt werden, sind sofort wirksam! Mit dem Wert geben Sie DB2 die Möglichkeit, in einem weiteren Pfad nach zugehörigen Protokolldateien zu suchen. Dies ist vor allem beim ROLLFORWARD-Befehl sinnvoll, da Sie so mögliche Archiv-Pfade mit diesem Parameter angeben können und diese auch im laufenden ROLLFORWARD-Prozess verändern können.

► MIRRORLOGPATH

Wenn Sie diesen Parameter konfigurieren, erstellt DB2 gleichzeitig eine Protokolldatei im Protokollpfad (LOGPATH) und im Pfad für die Protokollspiegelung (MIRRORLOGPATH).

► LOGBUFSZ – Protokollpuffergröße

Dieser Parameter ist notwendig, um die E/A-Last zu minimieren. Er gibt die Größe des Zwischenspeichers für die veränderten Protokollsätze in 4-Kbyte-Seiten an. Dieser Zwischenspeicher wird unter folgenden Umständen auf die Festplatte geschrieben:

- Der Protokollpuffer ist voll.
- Der Schwellenwert des Datenbankparameters MINCOMMIT wird erreicht.
- Interne Ereignisse machen das Schreiben auf die Festplatte erforderlich.

► MINCOMMIT – Anzahl von Gruppenfestschreibungen

Weitergabeklasse: sofort, d.h. dieser Parameter kann im laufenden DB2-System gesetzt werden und wird sofort wirksam.

Dieser Parameter verzögert das Schreiben von Protokollsätzen aus dem Protokollpuffer auf die Protokolldateien, bis eine Mindestanzahl (MINCOMMIT) von „commit-bereiten“ Transaktionen erreicht ist. Wenn also der Wert MINCOMMIT auf 4 steht, wird gewartet, bis vier Festschreibungen (commits) durchgeführt werden können; erst dann wird der Protokollpuffer ausgeschrieben und der Commit bestätigt. Hier gibt es ein weiteres festes Limit von einer Sekunde, das verhindert, dass commit-bereite Transaktionen zu lange auf eine Bestätigung warten müssen.

Daher werden spätestens nach einer Sekunde die Festschreibungen (commits) in die Protokolldatei geschrieben.

Diese Verzögerung kann, speziell in sehr abfrageintensiven Anwendungen (OLTP), helfen, die E/A-Last auf die Platte, auf der sich die Protokolldateien befinden, zu reduzieren.

► BLK_LOG_DSK_FUL

Weitergabeklasse: sofort

Wenn dieser Datenbank-Konfigurationsparameter auf YES gesetzt ist, verhindert er, dass ein Datenbankstillstand aufgrund einer zu kleinen Speicherkapazität (Festplatten- oder Dateisystemgröße) des Protokollpfads (LOGPATH) erfolgt, weil eine neue Protokolldatei dort nicht reserviert werden kann.

DB2 versucht alle fünf Minuten diese Protokolldatei neu anzulegen. Schlägt dies fehl, wird eine Diagnose-Information in die Diagnosedatei (db2diag.log) und Fehlerdatei (Unix: <exemplarname>.nfy, Windows: Windows Event Log) geschrieben.

Die oben genannten Parameter haben folgende Werte und Wertebereiche:

Parameter	Standardwert	Kleinster Wert	Größter Wert	Zusatzinfos
logfilsiz (4 Kb)	250	4 + 8 Kb	262.144 +8 Kb	Windows
	1.000	4 + 8 Kb	262.144 +8 Kb	Unix
logprimary	3	1	256	
logsecond	2	0	254	Weitergabeklasse: SOFORT -1, bedeutet un- endliche Anzahl
mincommit	1	1	25	Weitergabeklasse: SOFORT
softmax	100	1	100*logprimary	
logbufsz (4 Kb)	8	4	4096	32-Bit-Plattform
		4	65535	64-Bit-Plattform

Tabelle 9.1: Übersicht über die Protokollierungsparameter

Um die meisten der Parameter anzuwenden, die in diesem Abschnitt vorgestellt wurden, müssen Sie die Datenbank neu starten (siehe Kapitel 5).

Informative Parameter

Beim Auflisten der Datenbankkonfiguration gibt es auch einige nur „informative“ Parameter, wie z.B.:

Datenbank ist konsistent = NO

Bei dieser Abfrage bzw. Anzeige hat sich wohl schon so mancher Administrator den Kopf darüber zerbrochen, warum denn seine Datenbank in einem „inkonsistenten“ Zustand ist, warum also in diesem Statusfeld NO steht.

Dies lässt sich aber relativ einfach erklären. Die Datenbank ist immer in einem konsistenten (Datenbank ist konsistent = YES) Zustand, wenn:

- keine Applikationen mit der Datenbank verbunden sind
- nur „lesende“ Verbindungen zur Datenbank bestehen

Der Status der Datenbank ändert sich in NO (also „inkonsistent“), sobald Sie eine Veränderung an der Datenbank vornehmen.

Denken Sie daran, dass Veränderungen im Pufferpool geschehen und gehalten werden und dass nur die Protokolldateien sofort auf die Festplatte geschrieben werden. Das Zurückschreiben der veränderten Pufferpoolseiten erfolgt „asynchron“, also verzögert, und das heißt, die Datenbank ist zurzeit in einem „inkonsistenten“ Zustand.

Im Falle eines Crashes muss die Information über die Veränderung der Datensseiten aus den Protokolldateien gelesen werden, um einen „konsistenten“ Zustand wiederherzustellen.

Weitere selbsterklärende informative Parameter in der Datenbankkonfiguration sind:

Aktualisierende Wiederherstellung anstehend	= NO
Wiederherstellung anstehend	= NO
Status der Protokollspeicherung für Wiederherstellung	= RECOVERY
Erste aktive Protokolldatei	= S0000102.LOG
Status des Benutzerexits für Protokollierung	= NO

9.1.6 Wie und wo werden Protokolldateien angelegt?

Bei der Umlaufprotokollierung, also im Standardfall, wenn LOGRETAIN und USEREXIT auf NO gesetzt sind, werden die Protokolldateien im „Kreisverfahren“ überschrieben und wiederbenutzt.

Auch bei der Archiv-Protokollierung bleiben alle Protokolldateien im Standard-LOGPATH.

Eine Ausnahme gibt es, und zwar wenn Sie mit dem Benutzerexit-Programm (USEREXIT=ON) arbeiten (siehe Abschnitt 9.5.3 „Benutzerexit (Userexit)“). In diesem Fall werden die Protokolldateien, wenn sie intern den Status „VOLL“ erhalten, als Erstes in den ARCHIVPATH kopiert. Die schon kopierten und nicht mehr als „Aktiv“ gekennzeichneten Protokolldateien werden dann aus Optimierungsgründen wiederverwendet. Somit spart DB2 die Zeit, die Protokolldateien neu zu formatieren.

Eine weitere Besonderheit im Zusammenhang mit dem USEREXIT ist die Möglichkeit, LOGSECOND auf -1 zu setzen und dadurch die Begrenzung für eine unvollständige Transaktion von < 265 Gbyte aufzuheben.

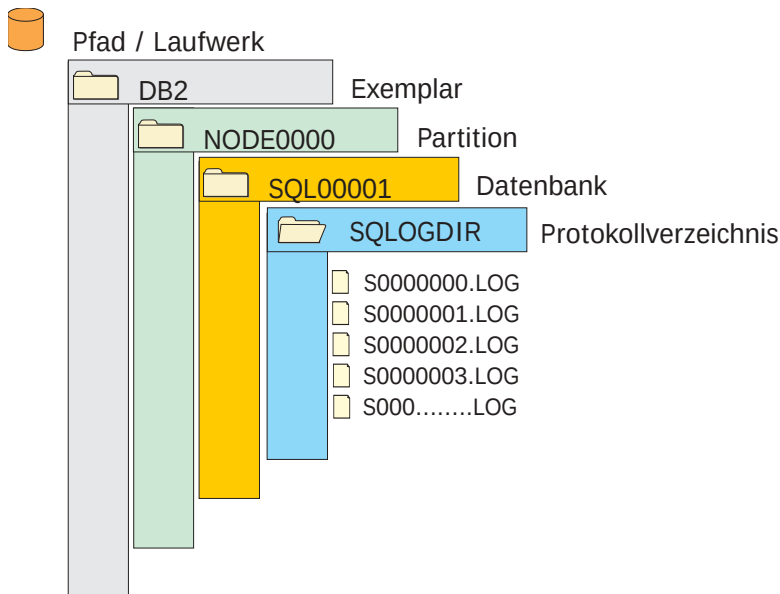


Abbildung 9.16: LOGPATH

Der Standard-LOGPATH ist bei Windows:

```
x:\<exemplarname>\NODEnnnn\SQLnnnn\SQLOGDIR
```

Bei Unix-Systemen sieht er so aus:

```
<pfad>/<exemplar-verz.>/<exemplarname>/NODEnnnn/SQLnnnn/SQLOGDIR
```

9.2 Sicherungs- und Wiederherstellungskonzepte

Ein gutes Sicherungskonzept erlaubt es Ihnen, bei einem Fehler die Datenbank in dem Zustand wiederherzustellen, den sie zu einem bestimmten Zeitpunkt hatte. Das Ziel ist es, dass möglichst keine Daten und Veränderungen an der Datenbank verloren gehen.

DB2 bietet unterschiedliche Konzepte und Möglichkeiten für die Sicherung und Wiederherstellung einer Datenbank.

9.2.1 Sicherungskonzepte

► Vollsicherung der Datenbank im Online- oder Offline-Modus

Unter einer „Offline-Vollsicherung“ ist zu verstehen, dass die gesamte Datenbank gesichert wird. Zum Zeitpunkt dieser Vollsicherung darf keine Anwendung eine Verbindung zur Datenbank haben. Das Backup-Dienstprogramm hat zum Zeitpunkt der Sicherung einen exklusiven Zugriff auf die Datenbank.

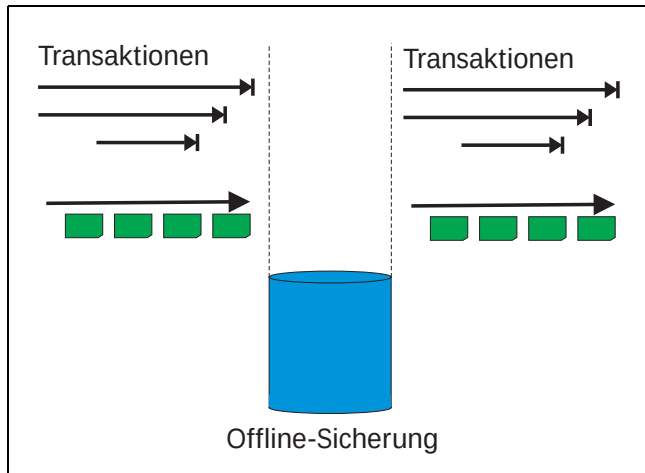


Abbildung 9.17: Offline-Sicherung

Eine „Online-Vollsicherung“ ist eine Sicherung der gesamten Datenbank, bei der alle Anwendungen ihre bestehenden Verbindungen und Aktivitäten weiterführen können.

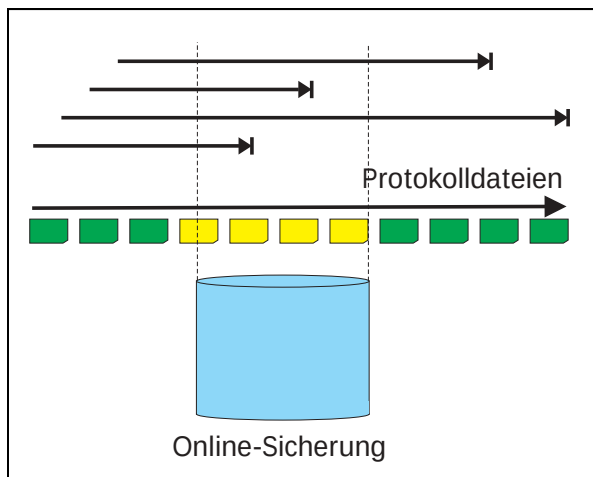


Abbildung 9.18: Online-Sicherung

Die Online-Sicherung hat folgende Besonderheit: Bei der Wiederherstellung werden die Protokolldateien benötigt, die während des Sicherungsprozesses geschrieben worden sind, um die Datenbank wieder in einen konsistenten Zustand zu bringen (siehe Abschnitt 9.4.5 „Aktualisierende Wiederherstellung (rollforward)“).

► Tabellenbereichssicherung der Datenbank im Online- oder Offline-Modus

Bei einer Tabellenbereichssicherung der Datenbank können ein oder mehrere Tabellenbereiche gesichert werden. Die kleinste Sicherungseinheit ist ein Tabellenbereich. Wollen Sie aber nur eine einzelne Tabelle sichern, dann können Sie diese Tabelle exportieren oder sie in einen einzelnen Tabellenbereich einbauen, um diesen dann separat sichern zu können. Diese Teilsicherung kann, wie schon die Vollsicherung, im Offline- bzw. Online-Modus erfolgen. Das heißt, auch hier können alle Anwendungen aktiv bleiben, während die Sicherung „online“ abläuft.

► Teilsicherung der Datenbank

Bei einer Teilsicherung der Datenbank werden nur die Veränderungen, die auf der Datenbank bzw. auf den Tabellenbereichen durchgeführt worden sind, in eine Sicherungsdatei geschrieben. Hier gibt es zwei unterschiedliche Konzepte:

- Inkrementelle Sicherung

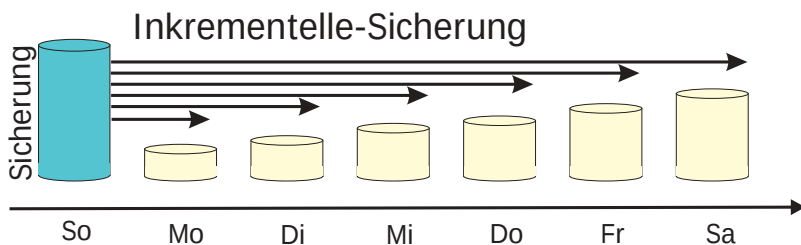


Abbildung 9.19: Inkrementelle Sicherung

Diese Art der Teilsicherung kann online und offline verwendet werden und sichert alle Veränderungen innerhalb der Datenbank bzw. der Tabellenbereiche, die seit der letzten Vollsicherung vorgenommen worden sind.

- Delta-Sicherung

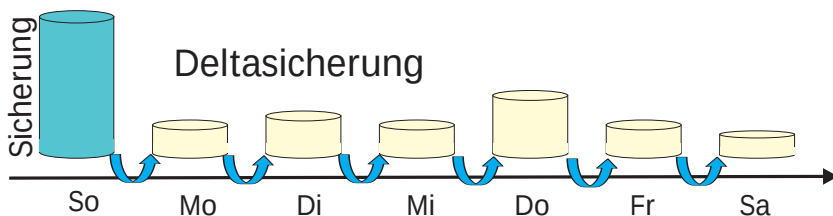


Abbildung 9.20: Deltasicherung

Im Unterschied zur inkrementellen Sicherung werden bei der Delta-Sicherung nur die Veränderungen gesichert, die seit der letzten Sicherung vorgenommen worden sind.

9.2.2 Wiederherstellungskonzepte

Es gibt drei Konzepte zur Wiederherstellung einer Datenbank:

► Versionswiederherstellung (versions recovery)

Das ist die Wiederherstellung einer Datenbank nach einer Sicherung ohne die Möglichkeit, die Daten bis zu einem definierten Punkt bzw. Zeitstempel zu aktualisieren.

Stellen Sie sich vor, Sie haben eine Vollsicherung Ihrer Datenbank erstellt (eine so genannte *Version*) und müssen nun diese zum Wiederherstellen verwenden. Dadurch bekommen Sie Ihre Daten auf dem Stand der Vollsicherung. Änderungen, die nach der Sicherung durchgeführt worden sind, sind nicht verfügbar.

► Wiederherstellung nach einem Systemabsturz (crash recovery)

Das ist die automatische Wiederherstellung eines konsistenten Zustands innerhalb einer Datenbank nach einem Stromausfall.

Dies geschieht durch das Rückgängigmachen der unvollständigen Transaktionen und durch das Nacharbeiten der festgeschriebenen Aktionen mit Hilfe der Protokolldateien.

► Eine aktualisierende Wiederherstellung (point in time recovery)

Das ist die Kombination aus der Versionswiederherstellung und der Nutzung der Protokolldateien zur zeitnahen Wiederherstellung der festgeschriebenen Transaktionen.

In zwei Schritten wird zuerst die Datenbanksicherung wieder eingespielt, und anschließend werden durch die Informationen in den Protokolldateien die durchgeführten Änderungen nachgezogen.

Welches Konzept für Sie das beste ist, können Sie mit folgenden Fragen ermitteln:

► Soll die Datenbank wiederherstellbar sein?

Diese Frage klingt sicherlich etwas komisch, ist aber begründet, da Sie erst einmal sicherstellen müssen, dass Sie Ihre Datenbank auch „wirklich“ gesichert haben. Dies beinhaltet neben einem Sicherungsbefehl an sich auch die Prüfung, dass die Sicherungsdatei korrekt abgelegt wurde.

► Haben Sie einen 7x24-Stunden-Betrieb, d.h. Ihre Datenbank ist an sieben Tagen der Woche 24 Stunden im Einsatz und es gibt keine oder nur sehr kleine Zeitfenster, um Wartungsarbeiten durchzuführen?

Ist das der Fall, so müssen Sie Ihr Sicherungs- und Wiederherstellungskonzept auf Online-Sicherungen aufbauen. Somit ist die Archiv-Protokollierung Voraussetzung.

► Wie aktuell soll das Ergebnis einer Wiederherstellung sein?

Wird diese Frage mit „sehr aktuell“ beantwortet, dann müssen Sie Ihre Datenbank in den *aktualisierenden Wiederherstellungsmodus* umschalten, also den Parameter LOGRETAIN auf ON bzw. USEREXIT auf ON setzen.

- Wie groß darf das Zeitfenster für die Wiederherstellung Ihrer Datenbank sein?
Lautet hier die Antwort „sehr klein“, dann sollten Sie häufige Sicherungen in Betracht ziehen. Eine Möglichkeit ist z.B. eine tägliche Sicherung, dann sind der Zeitaufwand und der Einsatz von Ressourcen am geringsten. Zur Wiederherstellung der Datenbank benötigen Sie die Sicherungsdatei und zusätzlich die noch notwendigen Protokolldateien von maximal 24 Stunden.
Das Sichern von großen Datenbanken führt in der Praxis nicht selten zu Zeit- und Speicherproblemen.
- Welche E/A-Subsysteme stehen zur Verfügung, um Datenbankdateien und Protokolldateien zu sichern?
Haben Sie neue Plattentechnologien im Einsatz wie etwa RAID- oder SSA-Systeme, die die Fähigkeit haben, Duplikate bzw. Spiegelungen selbstständig anzulegen? Dann können Sie mit Hilfe des Befehls SET WRITE in DB2 diese Funktionalität unterstützen und das Schreiben auf die Platteneinheit stoppen, um eine konsistente Kopie der Platteneinheiten zu ermöglichen.

9.3 Wie sichern Sie eine Datenbank?

Eine DB2-Datenbank zu sichern ist sehr einfach, weil DB2 sich um die Sicherungsinhalte kümmert und inkonsistente Zustände gar nicht zulässt.

Der einfachste Sicherungsbefehl ist:

```
backup db buchdb
```

Führen Sie diesen Befehl so aus, wird eine Sicherungsdatei der gesamten Datenbank erstellt, und zwar in das Verzeichnis, in dem Sie sich gerade befinden.

Wollen Sie das Verzeichnis bestimmen, in dem Sie die Sicherungsdatei der Datenbank ablegen möchten, müssen Sie folgenden Befehl eingeben:

```
backup db buchdb to d:\backup
```

Abbildung 9.21 zeigt das zugehörige Syntaxdiagramm.

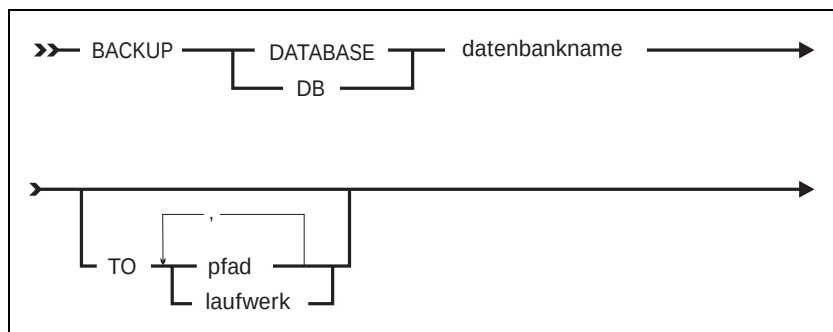


Abbildung 9.21: backup database-Syntaxdiagramm

Dabei ist es auch möglich, mehrere Ziel-Pfade bzw. -Laufwerke anzugeben, um die Sicherung parallel in mehrere Dateien zu schreiben. Diese Verteilung wird verwendet, um große Datenbanken schneller sichern zu können und eventuelle Größenprobleme bei Dateisystemen zu umgehen. Ein Beispiel sehen Sie in Abbildung 9.23.

Eine weitere Möglichkeit ist es, das Sicherungsbild an ein Speicher-Verwaltungsprogramm zu übergeben, z.B. an einen TSM-Server (Tivoli Storage Manager) oder an Sicherungsprogramme anderer Hersteller. Die Hersteller der Sicherungssoftware müssen dazu eine Bibliothek (library) zur Verfügung stellen, die Sie per `load <library-name>` in den BACKUP-Befehl einbinden können.

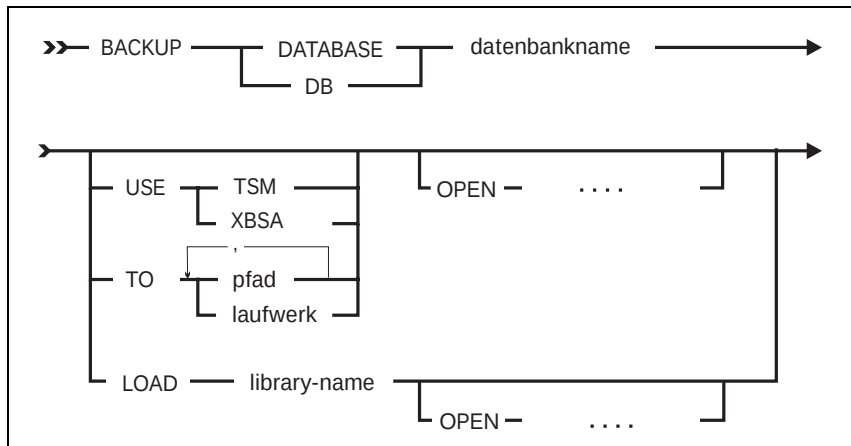


Abbildung 9.22: backup database use TSM-Syntaxdiagramm

9.3.1 Dateistruktur eines Sicherungsbildes

Beim Anlegen der Sicherungsdatei (Sicherungsbild) wird eine ganz bestimmte Namenskonvention eingehalten, die nicht veränderbar ist.

Für Windows-Betriebssysteme:

`DB_alias.Typ\Exemplar\NODEnnnn\CATnnnn\Datum\Zeit.Folgenr`

Beispiel:

`Buchdb.0\DB2\NODE0000\CAT0000\20030818\154259.001`

Für Unix-Betriebssysteme:

`DB_alias.Typ.Exemplar.NODEnnnn.CATnnnn.DatumZeit.Folgenr`

Beispiel:

`Buchdb.0.db2inst1.NODE0000.CAT0000.20030818154259.001`

Die Bedeutung der einzelnen Elemente ist:

► **DB_alias:**

bezeichnet den Aliasnamen der Datenbank und kann zwischen 1 und 8 Stellen lang sein.

► **Typ:**

Es gibt zur Zeit drei Typen:

- 0 – steht für eine Vollsicherung der ganzen Datenbank, sowohl „online“ als auch „offline“.
- 3 – steht für eine Teilsicherung auf Tabellenbereichsebene.
- 4 – steht für ein Sicherungsimagen, das mit dem Befehl `LOAD...COPY YES` erzeugt wurde.

► **Exemplar**

Hier wird der maximal achtstellige Exemplarname verwendet. Der Wert wird aus der Umgebungsvariablen `DB2INSTANCE` geholt.

► **Partitionsnummer**

- In einer nicht partitionierten Umgebung (environment) lautet diese Nummer immer `NODE0000`.
- In einem partitionierten Environment ist die Nummer `NODEnnnn` von der Einstellung in der `db2nodes.cfg` abhängig. `nnnn` repräsentiert die zugehörige Partitionsnummer innerhalb dieser Datenbank.

► **Katalogpartitionsnummer**

- In einem nicht partitionierten Environment lautet diese Nummer immer `CATN0000`.
- In einem partitionierten Environment ist die Nummer `CATNnnnn` von der Einstellung in der `db2nodes.cfg` abhängig. `nnnn` repräsentiert die zugehörige Partitionsnummer innerhalb dieser Datenbank, auf der der DB2-Katalog bei der Erstellung der partitionierten Datenbank angelegt wurde.

► **DatumZeit**

Dieser Wert gibt das Datum und die Uhrzeit des Zeitpunkts an, zu dem die Sicherung erstellt worden ist. Das Format ist `JJJJMMTThhmmss`.

► **Folgenr**

Zeigt eine dreistellige Zahl an, die als Dateierweiterung benötigt wird, wenn Sie eine Sicherung auf mehrere Ausgabemedien (z.B. Platten) gleichzeitig durchführen (siehe Abbildung 9.23).

```
C:\>db2 backup database buchdb to c:\backup,d:\backup,e:\backup
Sicherung erfolgreich. Die Zeitmarke für dieses Sicherungsimage ist:20030820183730
```

```
C:\backup\BUCHDB.0\DB2\NODE0000\CATN0000\20030820>dir
```

```
Verzeichnis von C:\backup\BUCHDB.0\DB2\NODE0000\CATN0000\20030820
```

```
20.08.2003  18:37    <DIR>          .
20.08.2003  18:37    <DIR>          ..
20.08.2003  18:38             8.392.704 183730.001
```

```
D:\backup\BUCHDB.0\DB2\NODE0000\CATN0000\20030820>dir
```

```
Verzeichnis von D:\backup\BUCHDB.0\DB2\NODE0000\CATN0000\20030820
```

```
20.08.2003  18:37    <DIR>          .
20.08.2003  18:37    <DIR>          ..
20.08.2003  18:38             12.587.008 183730.002
```

```
E:\backup\BUCHDB.0\DB2\NODE0000\CATN0000\20030820>dir
```

```
Verzeichnis von E:\backup\BUCHDB.0\DB2\NODE0000\CATN0000\20030820
```

```
20.08.2003  18:37    <DIR>          .
20.08.2003  18:37    <DIR>          ..
20.08.2003  18:38             4.214.784 183730.003
```

Abbildung 9.23: Beispiel für ein Datenbank-Backup auf mehreren Ausgabeeinheiten

9.3.2 Online- und Offline-Modus

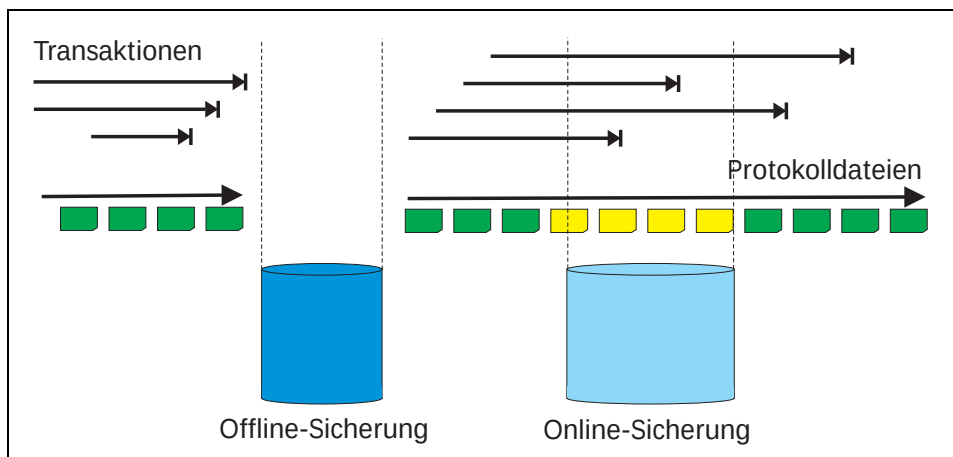


Abbildung 9.24: Online- und Offline-Sicherung

Beim Sichern der Datenbank bzw. auch beim Sichern von Tabellenbereichen oder auch beim Einsatz der inkrementellen Sicherung und Delta-Sicherung haben Sie die Möglichkeit, die Option `ONLINE` einzusetzen. Dieser „Online“-Modus bewirkt, dass beim Sichern die aktiven Anwendungen weiter ihre Transaktionen absetzen

können und somit der laufende Betrieb der Datenbank nicht behindert wird. Dieser Sicherungsmodus ist die Voraussetzung für einen 7x24h-Betrieb.

Im folgenden Syntaxdiagramm eines DB2-BACKUP-Befehls sehen Sie, dass dieser Befehl keine OFFLINE-Option enthält.

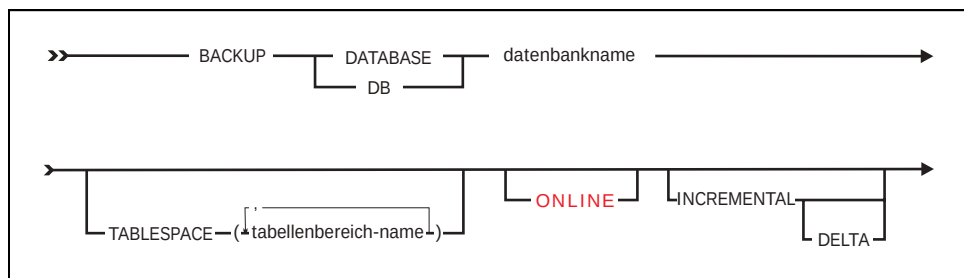


Abbildung 9.25: backup database online-Syntaxdiagramm

Wird die Option **ONLINE** nicht angegeben, gilt der Standardwert **OFFLINE**. Dann wird die Datenbank bzw. ein Teilbereich der Datenbank nur gesichert, wenn keine Applikationen mehr mit der Datenbank verbunden sind. Ist dies nicht der Fall – und bestehen noch Verbindungen zur Datenbank –, wird der Sicherungsbefehl nicht durchgeführt und Sie erhalten die Fehlermeldung aus Abbildung 9.26.

```

C:\backup>db2 backup db buchdb
SQL1035N  Die Datenbank wird bereits verwendet.  SQLSTATE=57019
  
```

Abbildung 9.26: Beispiel eines Sicherungsversuchs bei bestehender Verbindung

9.3.3 Datensicherung mit der Steuerzentrale

Die Sicherung der Datenbank kann mit der grafischen Steuerzentrale durchgeführt werden. Es gibt hier zwei Möglichkeiten, den Sicherungsassistenten aufzurufen: Die erste Möglichkeit ist die Option **TOOLS** in der Steuerzentrale. Über das Kontextmenü **ASSISTENTEN** bekommen Sie eine Auswahl von Assistenten, die Ihnen das Leben sehr vereinfachen können, wenn Sie noch nicht viel Erfahrung mit DB2 haben. In diesem Assistenten-Menü ist auch ein Sicherungsassistent vorhanden, den Sie nun auswählen können, um Ihre Datenbank zu sichern. Abbildung 9.27 zeigt Ihnen diese Auswahlmöglichkeit.

Die zweite Möglichkeit besteht darin, dass Sie direkt das Kontextmenü der Datenbank (hier „**BÜCHHDB**“) aufrufen, das Sie mit der rechten Maustaste angezeigt bekommen. Auch hier erhalten Sie die Auswahlmöglichkeit, den Sicherungsassistenten aufzurufen, nur dass die Option hier **SICHERN...** heißt. Die folgenden Fenster sind bei beiden Möglichkeiten identisch.

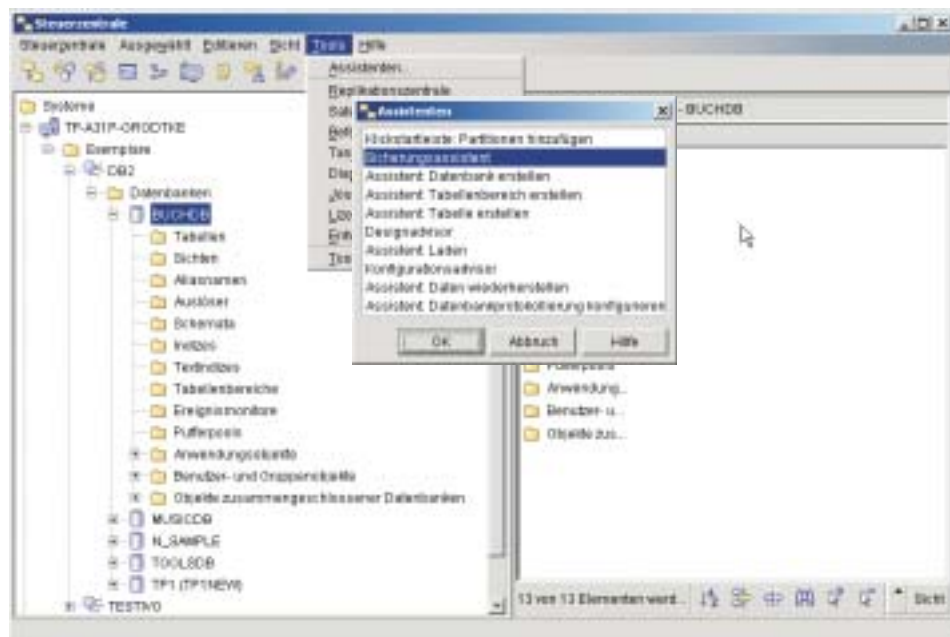


Abbildung 9.27: Assistent – Sicherungsassistent

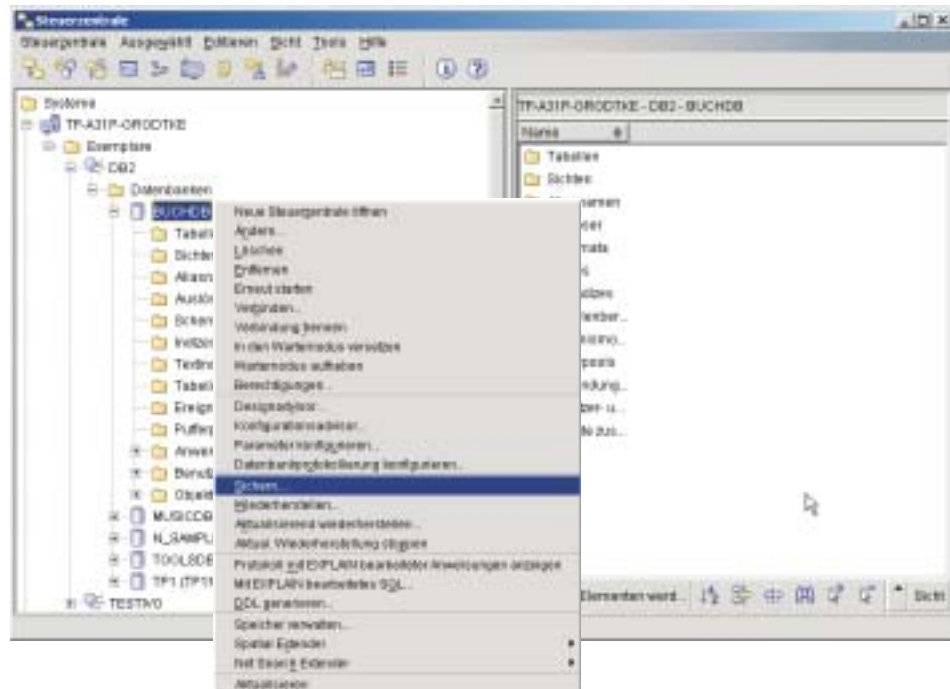


Abbildung 9.28: Auswahlenmenü – Sichern ...

Bei beiden Optionen werden Sie gleich zur Einführungsseite des Sicherungsassistenten geleitet. Hier wird Ihnen gezeigt, welche Art der Protokollierung Sie verwenden. In Abbildung 9.29 sehen Sie als Protokollierungstyp: UMLAUF, was auch die Standardeinstellung in DB2 ist. Weiter wird Ihnen hier gezeigt, ob eine „Online“- bzw. auch Tabellenbereichssicherungen möglich sind. Dies ist aber nur der Fall, wenn Sie von der Umlaufprotokollierung auf die Wiederherstellungsprotokollierung umgestellt haben.

Neben aktuellen Statusinformationen über den derzeitigen Zustand der ausgewählten Datenbank erhalten Sie auch gleichzeitig einige Links zu Hilfebildschirmen, um unbekannte Begriffe nochmals nachzulesen.



Abbildung 9.29: Sicherungsassistent – Einführung

In Abbildung 9.30 sehen Sie den anderen Fall, in dem schon auf die Wiederherstellungsprotokollierung umgestellt wurde. Mit dieser Protokolliertechnik werden auch ONLINE-Sicherungen möglich. Abhängig von Ihrer Selektion im unteren Auswahlkasten (GESAMTE DATENBANK SICHERN oder AUSGEWÄHLTE TABELLENBEREICHE SICHERN), kann sich nun die Benennung des zweiten Schritts von IMAGE in TABELLENBEREICHE ändern. Wir beschränken uns hier auf die Schritte für eine Datenbank-Komplett-sicherung.

Nachdem Sie mit Punkt 1 (siehe Abbildung 9.30) im Sicherungsassistenten ausgewählt haben, welche Sicherung Sie durchführen wollen, werden Sie zum Schritt 2 innerhalb des Sicherungsassistenten geführt.



Abbildung 9.30: Sicherungsassistent für Archivdatenbank



Abbildung 9.31: Sicherungsassistent – Image

Im zweiten Punkt (siehe Abbildung 9.31) des Sicherungsassistenten müssen Sie den Datenträgertyp und das Zielverzeichnis der Sicherungsdatei angeben. Hier gibt es mehrere Möglichkeiten. Die am häufigsten benutzten Optionen für den Typ sind:

- Dateisystem
- TSM (Tivoli Storage Manager)

Mit der Schaltfläche HINZUFÜGEN... öffnen Sie ein weiteres Menü, in dem Sie das Zielverzeichnis für die Sicherung auswählen können.

Wenn Sie die Ausgabe in mehreren Dateisystemen haben möchten, müssen Sie hier nur mehrfach HINZUFÜGEN... anklicken und jedes Mal ein anderes Zielverzeichnis auswählen. So könnten dann im Ausgabefenster folgende Informationen stehen:

Beispiel in einem Windows-System:

```
C:\buchdb\sicherungs_pfad_1
D:\buchdb\sicherungs_pfad_2
```

Beispiel für ein Unix-System:

```
/dev1/sicherungs_pfad_1/
/dev2/sicherungs_pfad_2/
```

Wenn Sie alle Sicherungsverzeichnisse angegeben haben, gehen Sie einfach mit WEITER zum nächsten Schritt (Schritt 3; siehe Abbildung 9.32).



Abbildung 9.32: Sicherungsassistent – Optionen

Im Schritt 3 können Sie jetzt einige Optionen abhängig von der Protokollierungsart verändern:

► SICHERUNGSTYP und VERFÜGBARKEIT

In unserem Beispiel wird nur eine Möglichkeit freigegeben, und zwar die für eine **GESAMTSICHERUNG – ALLE DATEN SICHERN**.

Die anderen grau dargestellten Optionen werden nur freigegeben, wenn Sie den Protokollierungsmodus auf „Wiederherstellungsprotokollierung“, also den DB2-Konfigurationsparameter **LOGRETAIN=ON** bzw. **USEREXIT=ON** gesetzt haben.

► DATENBANK IN DEN WARTEMODUS VERSETZEN

Hier wird Ihnen die Möglichkeit gegeben, die Datenbank in einen Ruhe-Modus zu versetzen, um die Datenbank **OFFLINE** zu sichern. Dieser Ruhe-Modus wird durch folgenden Befehl eingeschaltet:

```
quiesce database immediate force connection
```

Der Befehl bewirkt Folgendes:

... **force connection** bedeutet, dass alle bestehenden Applikationen beendet werden und bestehende Transaktionen zum letzten konsistenten Punkt zurückgeführt werden.

Ist Punkt 1 erfolgreich, wird die Datenbank in einen „Ruhe“-Zustand versetzt, der aber wieder explizit mit einem **UNQUIESCE DATABASE**-Befehl aufgehoben werden muss.



Abbildung 9.33: Sicherungsassistent – Leistung

Im vierten Schritt des Sicherungsassistenten (siehe Abbildung 9.33) können Sie Einfluss auf die Performance der Sicherung nehmen.

Folgende informelle Werte sind auf dem Bildschirm zu sehen:

► GESAMTZAHL SICHERUNGSZIELE: 1

(Sie wird aus der Information im Sicherungsassistenten-Schritt 2 abgeleitet. Hier wurde nur ein Ziel ausgewählt: c:\buchdb\backup.)

► GESAMTZAHL ZU SICHERNDER TABELLENBEREICHE: 2

Das ist der SYSCATSPACE und USERSPACE1, der TEMPSPACE wird nie gesichert.

Einstellbar sind:

► PARALLELITÄT: 1

Hier sollte die Anzahl der zu sichernden Tabellenbereiche stehen. In unserem Fall ist „2“ der bessere Wert, da wir zwei Tabellenbereiche sichern. Wenn die Anzahl der Tabellenbereiche nicht bekannt ist, so ist dies auch kein Problem. Sie sollten hier einfach den Wert aus dem Informationsfeld GESAMTZAHL ZU SICHERNDER TABELLENBEREICHE übernehmen.

► ANZAHL PUFFER: 2

Hier sollte der Wert immer auf mindestens 2 gesetzt sein. Noch besser ist es, wenn dieser Wert immer auf dem Doppelten des Wertes GESAMTZAHL SICHERUNGSZIELE steht. In unserem Fall ist dieser Wert 2. Wenn der Wert größer als 5 wird, bekommen Sie eine Fehlermeldung, die besagt, dass Sie nicht genügend Speicherplatz haben. Hier wird der so genannte UTILITY-Heap (Zwischenspeicher; das ist ein Wert, der den maximalen Speicherverbrauch für das BACKUP- und RESTORE-Tool vorgibt) zu klein. Die Standardeinstellung ist UTIL_HEAP_SZ=5000. Dieser Wert muss nach oben angepasst werden, wenn Sie die Anzahl der Puffer größer als 5 setzen. Der Parameter UTIL_HEAP_SZ berechnet sich wie folgt:

$$\text{UTIL\_HEAP\_SZ} = \text{Anzahl der Puffer} * \text{Puffergröße}$$

► GRÖSSE JEDES PUFFERS (4-KB-SEITEN): 1024

Dieser Wert kann problemlos übernommen werden.

Haben Sie diese Eingaben verändert, kommen Sie zum „Zeitplan“-Schritt des Sicherungsassistenten (siehe Abbildung 9.34). Sie haben die Möglichkeit, die Sicherung nun *sofort* oder über die Taskzentrale zu einem anderen Zeitpunkt zu starten.

Mit der Auswahl ALS TASK IN DER TASKZENTRALE ERSTELLEN wird Ihnen die Option gegeben, die Sicherung über die Taskzentrale einzuplanen und so diese Sicherung an einem ganz bestimmten Tag bzw. zu einer von Ihnen bestimmten Uhrzeit zu starten.

Wählen Sie aber die Option JETZT AUSFÜHREN, OHNE DAS TASKPROTOKOLL ZU SICHERN, werden Sie gleich zum nächsten Schritt weitergeleitet (siehe Abbildung 9.35).



Abbildung 9.34: Sicherungsassistent – Zeitplan



Abbildung 9.35: Sicherungsassistent – Zusammenfassung

Hier sollten Sie auf jeden Fall den BEFEHL ANZEIGEN-Button anklicken, um sich die von DB2 generierten Befehle anzeigen zu lassen (siehe Abbildung 9.36).



Abbildung 9.36: Sicherungsassistent – Befehl anzeigen

In diesem Fenster werden Ihnen die Befehle angezeigt, die von DB2 ausgeführt werden, wenn Sie den FERTIG-Button zum Ausführen der Sicherung anklicken.

9.3.4 Sicherung mit DB2-Befehlen

Wenn Sie einmal mit dem GUI bzw. mit dem Sicherungsassistenten eine Sicherung (menügeführt) Schritt für Schritt durchgeführt haben, dann ist das Sichern über eine Befehlszeile auch recht einfach, da Sie die möglichen Optionen schon kennen.

Sie können die DB2-Befehle, die Sie sich im GUI angeschaut haben, in eine Textdatei kopieren und dann in der DB2-Kommandozeile ausführen:

```
Sichern_BuchDB.txt:
connect to buchdb;
quiesce database immediate force connections;
connect reset;
backup database buchdb to C:\buchdb\sicherungen;
connect to buchdb;
unquiesce buchdb;
connect reset;
```

Dieses Script bzw. diese Befehlsfolge kann folgendermaßen über die Befehlszeile gestartet werden:

```
db2 -tvf sichern_BuchDB.txt
```

Wollen Sie eine Online-Sicherung durchführen, so müssen Sie zuerst in den Modus umzuschalten, der Ihnen eine Online-Sicherung ermöglicht. Dazu muss der DB-Parameter LOGRETAIN=ON bzw. USEREXIT=ON gesetzt werden.

Nochmals zur Erinnerung:

Wie kommen Sie vom OFFLINE-Modus in den ONLINE-Modus – oder besser gesagt, wie schalten Sie von der Umlaufprotokollierung in die Wiederherstellungsprotokollierung?

Mit DB2-Befehlen geht das so:

1. Ändern Sie den Datenbankparameter LOGRETAIN auf ON/RECOVERY.

```
update db cfg for buchdb using logretain on
```

2. Sichern Sie die Datenbank im OFFLINE-Modus.

```
connect to buchdb
quiesce database immediate force connections
connect reset
backup database buchdb to C:\buchdb\sicherungen
connect to buchdb
unquiesce buchdb
connect reset
```

Nun können Sie die Datenbank im ONLINE-Modus, und zwar sowohl als Voll- wie auch als Teilsicherung, sichern.

Sichern der Datenbank im ONLINE-Modus:

```
backup database buchdb to C:\buchdb\sicherungen online
```

Teilsicherung der Datenbank im ONLINE-Modus:

```
backup database buchdb to C:\buchdb\sicherungen delta online
```

9.3.5 Was wird gesichert?

Diese Frage stellen sich nicht nur DB2-Datenbankadministratoren, sondern auch sehr oft Administratoren, die mit Produkten anderer Datenbankhersteller arbeiten, da es hier häufig große Unterschiede zwischen den Produkten gibt.

In DB2 werden für jede Sicherung, egal welcher Art, folgende Informationen in das Sicherungsimago gespeichert:

► Media-Header :

- Zeitstempel z.B. 20030903172118
- Datenbankname z.B. buchdb
- Datenbank-Alias z.B. db2buch
- Exemplarname z.B. DB2
- Backup Mode z.B. 0=offline und 1=online
- Database Seed z.B. 3E344EDA
- DB Codeset z.B. IBM-1252
- Plattform z.B. 5

► Kontrolldateien:

- Startzeitpunkt der Sicherung
- Liste aller Tabellenbereiche
- Protokollinformationen über Sicherungen, Wiederherstellungen und Lade-Operationen (DB2RHIST.ASC)
- Informationen über Tabellenbereiche und deren Container

- Datenbankkonfiguration (SQLDBCON)
- Pufferpool-Konfiguration (SQLBP.X)
- Protokollsteuerdatei der gespiegelten Protokolle (SQLOGMIR.LFH)
- Protokolldatei der Tabellenbereichsveränderungen (DB2TSCHG.HIS)

► Sicherung aller Objektdaten

► Kontrollinformationen der Protokolldateien (SQLOGCTL.LFH)

Die exakten Informationen, die nun auf Ihrer Datenbanksicherung stehen, können Sie mit dem DB2-Tool db2ckbkp abfragen (Details finden Sie in Abschnitt 9.5.2 „DB2-Tool db2ckbkp“). Die wichtigsten Optionen sind:

- -H – ermöglicht es, nur die Media-Header-Information auszulesen
- -l – Informationen über die Kontrollinformationen der Protokolldateien anzeigen
- -o – zugehörige Pfade der Tabellenbereiche werden angezeigt

9.3.6 Liegt eine Online- oder eine Offline-Sicherung vor?

Für diese Frage gibt es drei mögliche Lösungsansätze:

- eine Abfrage über das Journal (GUI)
- eine Abfrage über den DB2-Befehl list history
- das Auslesen der Header-Information des Sicherungsimages DB2-Tool db2ckbkp

Im Journal (siehe Abbildung 9.37) wird im Feld KOMMENTAR die Art des Backups angezeigt (ONLINE oder OFFLINE). Zusätzlich hat man hier eine gute Übersicht darüber, wann eine Sicherung durchgeführt wurde. Durch die Anzeige der Start- und Endzeit kann man die Dauer des Sicherungslaufs ausrechnen.



Abbildung 9.37: Online-Backup im Journal

Wenn Sie lieber die Befehlszeile verwenden wollen, kommt das folgende Kommando in Frage:

```
list history backup all for buchdb
```

Dieser Befehl hat noch eine ganze Menge an zusätzlichen Optionen. Wir schauen uns hier nur die Optionen an, die notwendig sind, um eine Online-Sicherung zu erkennen. Am einfachsten ist es, wenn Sie die Ausgabe von diesem Befehl in eine Datei umleiten und dann einen einfachen Texteditor verwenden, um den Befehlsteil „ONLINE“ zu suchen. Nun sind Sie auch schon im richtigen Block und sehen unter dem Befehl die Startzeit und die Endzeit, zu denen das Online-Backup erfolgt ist.

Eine weitere Suchmöglichkeit ist, nach dem TYP zu suchen (siehe Abbildung 9.38). Bei einem ONLINE-Backup steht unter TYP ein „N“ für „oNline“. Für ein Offline-Backup steht unter Typ ein „F“ für „oFFline“. Weitere Werte für den Typ sind:

- „I“ – steht für „inkrementelle Offline-Sicherung“
- „0“ – steht für „inkrementelle Online-Sicherung“
- „D“ – steht für eine „Delta-Offline-Sicherung“
- „E“ – steht für eine „Delta-Online-Sicherung“

```
C:\BuchDB\Sicherungen>db2 list history backup all for buchdb

Protokolldatei für buchdb auflisten

Anzahl übereinstimmender Dateieinträge = 5
.....
.....
```

Op	Obj	Zeitmarke+Folge	Typ	Einh.	1.Protok.	Akt. Protok.	RFW-Stoppzeit
B	D	20030823185834001	F	D	S0000000.LOG	S0000000.LOG	

```
-----
Enthält 2 Tabellenbereich(e):

00001 SYSCATSPACE
00002 USERSPACE1
-----

Kommentar: DB2 BACKUP BUCHDB OFFLINE
Startzeit: 20030823185834
Endzeit: 20030823185906
-----

00004 Adresse: C:\BuchDB\Sicherungen\BUCHDB.0\DB2\NODE0000\CATN0000\20030823
```


Op	Obj	Zeitmarke+Folge	Typ	Einh.	1.Protok.	Akt. Protok.	RFW-Stoppzeit
B	D	20030823185927001	N	D	S0000000.LOG	S0000001.LOG	

```
-----
Enthält 2 Tabellenbereich(e):

00001 SYSCATSPACE
00002 USERSPACE1
-----

Kommentar: DB2 BACKUP BUCHDB ONLINE
Startzeit: 20030823185927
Endzeit: 20030823185948
-----

00005 Adresse: C:\BuchDB\Sicherungen\BUCHDB.0\DB2\NODE0000\CATN0000\20030823
```

Abbildung 9.38: Beispiel für list history backup all

Die dritte Möglichkeit wird eigentlich nur verwendet, wenn die History-Protokoll-datei aus irgendeinem Grund nicht mehr zur Verfügung steht oder wenn Sie diese Information von einem Sicherungsmedium herunterlesen müssen. In diesem Fall reicht es aus, wenn Sie die Option „H“, also nur die Header-Informationen, aus dem Sicherungsbild auslesen und nach dem „Backup Mode“ suchen. Backup Mode = 1 ist eine „ONLINE“-Sicherung, und Backup Mode = 0 ist eine „OFFLINE“-Sicherung. Sie können dies anhand der beiden Bilder (Abbildung 9.38 und Abbildung 9.39) erkennen, indem Sie sich die Zeitstempel anschauen und diese vergleichen.

```

..CATN0000\20030823> db2ckbkp -H 185927.001 > test.out
=====
MEDIA HEADER REACHED:
=====
Server Database Name          -- BUCHDB
Server Database Alias         -- BUCHDB
Client Database Alias         -- BUCHDB
Timestamp                     -- 20030823185927
..
Backup Mode                   -- 1
..
The proper image path would be:
BUCHDB.0\DB2\NODE0000\CATN0000\20030823\185927.001

-----

db2ckbkp -H 185834.001 > test.out
oder
C:\BuchDB>db2ckbkp -H ..... \CATN0000\20030823\185834.001 > test.out

=====
MEDIA HEADER REACHED:
=====
Server Database Name          -- BUCHDB
Server Database Alias         -- BUCHDB
Client Database Alias         -- BUCHDB
Timestamp                     -- 20030823185834
..
Backup Mode                   -- 0
..
The proper image path would be:
BUCHDB.0\DB2\NODE0000\CATN0000\20030823\185834.001

```

Abbildung 9.39: DB2-Befehl: db2ckbkp – H

9.3.7 Tabellenbereichssicherung

Bisher haben wir fast immer nur von einer Datenbanksicherung gesprochen, doch es ist nicht immer sinnvoll, die gesamte Datenbank zu sichern. Stellen Sie sich vor, Sie haben eine sehr große Datenbank mit vier Tabellenbereichen, und auf drei (TS1 bis TS3) von diesen vier Tabellenbereichen werden nur Abfragen ohne jegliche Veränderung gestartet. Auf dem vierten Tabellenbereich (TS4) finden sehr viele Veränderungen an den Tabellen statt, und es würde Sie viel Aufwand und Zeit kosten, diese Veränderungen zu rekonstruieren.

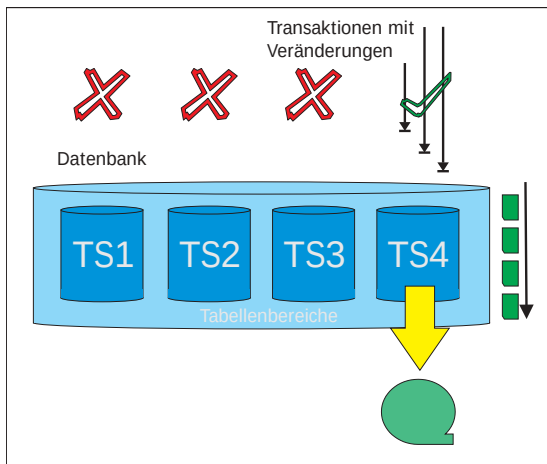


Abbildung 9.40: Tabellenbereichssicherung

Also muss eine Sicherung mit wiederherstellbarer Protokollierung her, um so zeitnah wie möglich an eine eventuelle Fehlersituation herankommen zu können.

Was ist hier die beste Strategie?

- ▶ Vollsicherung der Datenbank, also immer alle Tabellenbereiche (TS1 bis TS4) sichern?
- ▶ Teilsicherungen auf der Basis der Tabellenbereiche, also einen Tabellenbereich (TS4) komplett sichern?
- ▶ Teilsicherungen auf der Basis der Veränderungen, also eine inkrementelle Sicherung bzw. eine Delta-Sicherung, die auch auf Tabellenbereichsebene durchgeführt werden kann?

Welche die beste Strategie ist, hängt generell von den Anforderungen aus dem Umfeld und von den Rahmenbedingungen ab, in denen die Datenbank eingesetzt wird bzw. werden soll. Zu den Strategie-Überlegungen gehört nicht nur die Frage, wann und wie man die Datenbank sichert, sondern die Überlegung, welche Möglichkeiten durch das Sicherungskonzept offen stehen und welche Vorteile welches Sicherungskonzept für die Wiederherstellung bringt. Wiederherstellungskonzepte wurden in Abschnitt 9.2.2 „Wiederherstellungskonzepte“ angesprochen.

Durchführung einer Tabellenbereichssicherung

Die Tabellenbereichssicherung läuft nach dem gleichen Schema wie die Datenbanksicherung ab. Wie Sie in der Backup-Syntax sehen (siehe Abbildung 9.41), ist sie nur eine weitere Ausprägung im Backup-Befehl. Für den zu sichernden Tabellenbereich wird zusätzlich TABLESPACE, was ja „Tabellenbereich“ bedeutet, mit den zu sichernden Tabellenbereichsnamen angegeben. Bei einer Liste von Tabellenbereichen werden diese einfach durch Kommata getrennt:

```
backup db buchdb tablespace(TS1,TS2,TS3,TS4)
```

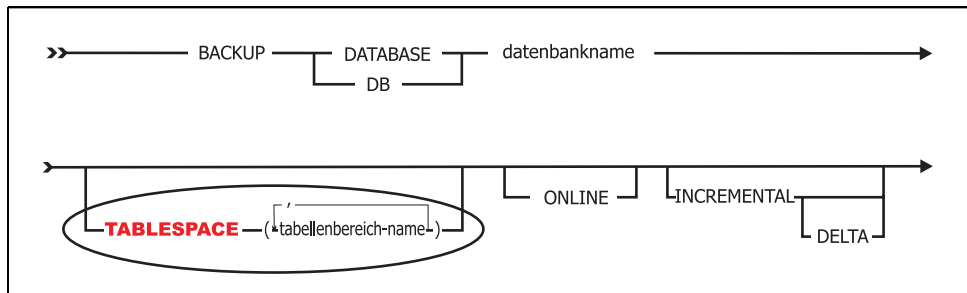


Abbildung 9.41: backup database-Syntaxdiagramm für Tabellenbereichssicherung

Alle weiteren Optionen entsprechen denen der Datenbank-Vollsicherung.

9.4 Wie führt man eine Wiederherstellung durch?

Es gibt verschiedene Verfahren, wie die Daten innerhalb einer Datenbank wiederhergestellt werden können. Welche Verfahren Sie einsetzen können, ist wiederum vom Sicherungskonzept abhängig, also davon, welche Arten von Sicherungen Sie durchgeführt haben.

Folgende Punkte sind für die Wiederherstellung zu beachten:

- ▶ Sie können eine bestehende Datenbank mit einer anderen Datenbanksicherung überschreiben, auch wenn der Datenbankname bzw. der Aliasname sowie auch der Datenbank-Ursprungspunkt unterschiedlich sind.
- ▶ Das Zurückspielen einer kompletten Datenbank kann nur im OFFLINE-Modus erfolgen.
- ▶ Tabellenbereiche können im OFFLINE- und im ONLINE-Modus zurückgespielt werden. Die Vorteile des ONLINE-Modus sind, dass die anderen Tabellenbereiche nicht betroffen sind und dass somit auf anderen Tabellenbereichen weitergearbeitet werden kann. Der wiederherzustellende Tabellenbereich ist jedoch nicht verfügbar.
- ▶ Nach einer Wiederherstellung der Datenbank geht die Datenbank automatisch in den Status Aktualisierende Wiederherstellung anstehend = DATABASE (rollforward pendig) über. Dieser Wiederherstellungsmodus ermöglicht es, die Datenbank mit den zugehörigen Protokolldateien auf den aktuellsten Stand wiederherzustellen. Dies erfolgt nur, wenn die Datenbank mit LOGRETAIN=ON konfiguriert worden ist und bei der Wiederherstellung der Datenbank die Option WITHOUT ROLLING FORWARD nicht verwendet wurde.
- ▶ Die Wiederherstellung der Datenbank besteht normalerweise, falls sie nicht unterdrückt wird, aus zwei Schritten:
 1. Wiederherstellen der Sicherung durch Zurückspielen des Sicherungsbildes
 2. Aktualisierende Wiederherstellung über die gespeicherten Protokolldateien

- ▶ Der Status Aktualisierende Wiederherstellung anstehend = DATABASE (rollforward pending) kann für eine Datenbank oder auch für einen Tabellenbereich gelten. Nur wenn dieser Status gesetzt ist, kann eine Aktualisierung über den ROLLFORWARD-Befehl vorgenommen werden. Haben mehrere Tabellenbereiche diesen Status, dann muss eine vollständige Liste dieser Tabellenbereiche beim ROLLFORWARD-Befehl angegeben werden.
- ▶ Es kann immer nur ein RESTORE, also eine Wiederherstellung eines Sicherungsimages, zu einem Zeitpunkt erfolgen.
- ▶ Es kann nur ein schon existierender Tabellenbereich wiederhergestellt werden.
- ▶ Falls Sie einen Tabellenbereich gelöscht haben, ist dieser nicht wiederherstellbar!
- ▶ Ein Tabellenbereich aus einer Datenbank kann nicht in einer fremden Datenbank wiederhergestellt werden.
- ▶ Haben Sie über Integritätsregeln verknüpfte Tabellen über mehrere Tabellenbereiche verteilt, so sollten Sie bei der Wiederherstellung alle betroffenen Tabellenbereiche in einem Schritt aktualisieren.
- ▶ Einzelne Tabellenbereiche können auch aus jeder Vollsicherung oder Teilsicherung extrahiert und wiederhergestellt werden.

9.4.1 Was ist ein Datenbank-Ursprungspunkt (database seed)?

Ein Datenbank-Ursprungspunkt ist eine eindeutige Nummer, die gesetzt wird, wenn DB2 eine Datenbank anlegt. Dieser Datenbank-Ursprungspunkt ist in jeder Sicherung enthalten und stellt sicher, dass Sie nicht einen Tabellenbereich (mit gleichem Namen) aus einer Testdatenbank in die Produktionsdatenbank einspielen können. Es gibt noch weitere Abhängigkeiten von diesem Ursprungspunkt. Diese werden in Abbildung 9.42 gezeigt.

	Wiederherstellung in eine bestehende Datenbank	Wiederherstellung in eine bestehende Datenbank	Wiederherstellung in eine neue Datenbank
Datenbank- Ursprungspunkt	identisch	unterschiedlich	wird neu erstellt
Protokollpfad	bleibt bestehen	wird aus der Sicherung übernommen	wird aus der Sicherung übernommen
Datenbank- protokolldateien	- bleibt bestehen, wenn Logretain=ON - werden gelöscht, wenn Logretain=OFF	werden gelöscht	
Datenbank- konfiguration	bleibt bestehen	wird aus der Sicherung übernommen	wird aus der Sicherung übernommen
Recovery-History Protokolldatei	bleibt bestehen	bleibt bestehen	wird aus der Sicherung übernommen

Abbildung 9.42: Auswirkungen vom Datenbank-Ursprungspunkt auf die Wiederherstellung

Als Ergänzung zu der Tabelle aus Abbildung 9.42 muss noch gesagt werden, dass bei der Wiederherstellung der Datenbank der LOGPATH, also der Datenbankprotokollpfad, angegeben werden kann. Dies geschieht über die Option NEWLOGPATH im RESTORE-Befehl. Damit wird die bestehende Definition des Protokollpfads (LOGPATH) bei der Wiederherstellung der Datenbank überschrieben.

Tipp:

Sie können den Datenbank-Ursprungspunkt (database seed) über das Tool db2chkp mit der Option „-H“ abfragen. Hiermit wird die Header-Information der Sicherung gelesen und angezeigt. Beispiel:

```
C:\BuchDB\SIChER~1\BUCHDB.0\DB2\NODE0000\CATN0000\20030824>db2chkbp -h 190657.001

=====
MEDIA HEADER REACHED:
=====
      Server Database Name      -- BUCHDB
      Server Database Alias    -- BUCHDB
      Client Database Alias    -- BUCHDB
      Timestamp                 -- 20030824190657
      Database Partition Number -- 0
      Instance                  -- DB2
      Sequence Number          -- 1
      Release ID                -- A00
      Database Seed             -- 3F468A96
      ....
      ....
```

Abbildung 9.43: Beispiel: Database Seed, mit db2chkbp angezeigt

9.4.2 Welche Daten werden zur Wiederherstellung benötigt?

Dies ist die wichtigste Frage, die zu beantworten ist, bevor Sie mit der Wiederherstellung Ihrer Datenbank beginnen. Folgende Fragen sollten detailliert beantwortet werden:

- ▶ Welche Sicherung benötige ich für die Wiederherstellung?
- ▶ Wo stehen die Sicherungen (backup-images) der Datenbank?
- ▶ Muss diese Sicherung noch aus einem Speicherverwaltungssystem (TSM) geholt werden?
- ▶ Welche Protokolldateien (logfiles) werden benötigt?
- ▶ Wo stehen die Protokolldateien für die Wiederherstellung?
- ▶ Müssen diese Dateien aus einem Speicherverwaltungssystem geholt werden?

Die erste Quelle für Antworten auf diese Fragen ist die DB2-eigene Protokollierungsinformationsdatei, das so genannte „Recovery-History-File“. Diese Datei ist im zugehörigen Datenbankverzeichnis als db2rhist.asc abgelegt. Die Protokollierungsinformationsdatei enthält alle wichtigen Informationen über:

- ▶ Sicherung und Wiederherstellung einer Datenbank
- ▶ Sicherung und Wiederherstellung eines Tabellenbereichs bzw. auch mehrerer Tabellenbereiche
- ▶ die Erstellung, Veränderung und Löschung eines Tabellenbereichs
- ▶ das Löschen, Reorganisieren und Laden von Tabellen
- ▶ das Stilllegen des Tabellenbereichs für eine Tabelle (quiesce tablespace for <tablename>)

Die Informationen aus dem „Recovery-History-File“ können über den Befehl

```
list history for all on database buchdb
```

abgefragt werden. Das Ergebnis sehen Sie in Abbildung 9.44.

Die Informationen, die Sie aus der angezeigten Protokollierungsinformationsdatei auslesen können, sind z.B.:

- ▶ die Art der Sicherung, also ob eine ONLINE- oder eine OFFLINE-Sicherung durchgeführt wurde – siehe Punkt ①

Kommentar: DB2 BACKUP BUCHDB ONLINE

- ▶ die Startzeit und die Endzeit der Sicherung – siehe Punkt ②

Startzeit: 20030901234857

Endzeit: 20030901235411

- ▶ der Ort der Sicherung – siehe Punkt ③

Adresse: C:\buchdb\BUCHDB.0\DB2\NODE0000\CATN0000\20030901

Vorsicht:

Hier ist es ganz wichtig, dass dies der Ort ist, der zur Zeit der Sicherung genutzt wurde. Jede Veränderung, die später durchgeführt wurde, wird hier nicht mehr protokolliert. Somit kann es sein, dass diese Information in dieser Protokolldatei mit dem tatsächlichen Speicherort nicht mehr übereinstimmt.

- ▶ der zugehörige Zeitstempel mit der fortlaufenden Nummer, der unter Windows auch gleichzeitig der Dateiname ist: 20030901234857.001 – siehe Punkt ④

20030901234857001

Protokolldatei für buchdb auflisten						
Anzahl übereinstimmender Dateieinträge = 3						
Op	Obj	Zeitmarke+Folge	Typ	Einh.	1.Protok.	Akt. Protok. RFW-Stoppzeit
B	D	20030901223940001	F	D	S0000000.LOG	S0000000.LOG
Enthält 2 Tabellenbereich(e):						
00001 SYSCATSPACE						
00002 USERSPACE1						
Kommentar: DB2 BACKUP BUCHDB OFFLINE						
Startzeit: 20030901223940						
Endzeit: 20030901224005						
00001 Adresse: C:\buchdb\sicherungen\BUCHDB.0\DB2\NODE0000\CATN0000\20030901						
Op	Obj	Zeitmarke+Folge	Typ	Einh.	1.Protok.	Akt. Protok. RFW-Stoppzeit
B	D	20030901233811001	N	D	S0000017.LOG	S0000018.LOG
Enthält 2 Tabellenbereich(e):						
00001 SYSCATSPACE						
00002 USERSPACE1						
Kommentar: DB2 BACKUP BUCHDB ONLINE						
Startzeit: 20030901233811						
Endzeit: 20030901233846						
00002 Adresse: C:\buchdb\sicherungen\BUCHDB.0\DB2\NODE0000\CATN0000\20030901						
Op	Obj	Zeitmarke+Folge	Typ	Einh.	1.Protok.	Akt. Protok. RFW-Stoppzeit
B	D	20030901234857001	N	D	S0000018.LOG	S0000021.LOG
Enthält 2 Tabellenbereich(e):						
00001 SYSCATSPACE						
00002 USERSPACE1						
Kommentar: DB2 BACKUP BUCHDB ONLINE						
Startzeit: 20030901234857						
Endzeit: 20030901235411						
00003 Adresse: C:\buchdb\BUCHDB.0\DB2\NODE0000\CATN0000\20030901						

Abbildung 9.44: Beispiel für list history

Welche Protokolldateien sind notwendig, um ein Online-Backup wieder in einen konsistenten Zustand zu versetzen?

Wir benötigen die Protokolldateien S0000018.LOG bis S0000020.LOG. Diese wurden zum Zeitpunkt des Online-Backups aktiv. In Abbildung 9.44 wird bei Punkt ⑤ die erste benötigte Protokolldatei unter 1.Protok. angezeigt. Der Eintrag Akt. Protok. verweist auf die nächste zu verarbeitende Protokolldatei, die nach dem Online-Backup geschrieben wird (in unserem Beispiel S0000021.LOG). Diese wird also nicht mehr für die Aktualisierung benötigt!

Das Online-Backup schließt die Protokolldatei, auf der die „Ende-Information“ vom Online-Backup geschrieben wird, so dass keine weiteren Veränderungen in dieser Protokolldatei aufgenommen werden können. Dadurch kann diese Protokolldatei archiviert werden.

Diese Informationen können auch über die Registerkarte DATENBANKPROTOKOLL im JOURNAL abgefragt werden. Bei dieser Ausgabe (siehe Abbildung 9.45) ist die Spaltenreihenfolge (Kommentarspalte) verändert worden. Die Spalten sind an unsere Abfragebedürfnisse angepasst.



Abbildung 9.45: Datenbankprotokoll im Journal

Wie Sie in Abbildung 9.45 sehen, erhalten Sie hier fast dieselben Informationen wie mit dem list history-Befehl über die Kommandozeile – allerdings fehlt die Anzeige für die benötigten Protokolldateien.

Nun wissen wir, wo die Sicherungen sind und welche Protokolldateien wir benötigen. Es fehlt nur noch die Prüfung, ob die benötigten Protokolldateien auch in unserem Protokollpfad sind oder ob diese eventuell aus Sicherungspfaden zurückkopiert werden müssen. Zusätzlich haben wir noch die Möglichkeit, beim ROLLFORWARD-Befehl, also beim Aktualisierungsschritt, einige Optionen anzugeben, um die Pfade richtig einzustellen.

9.4.3 Wiederherstellen einer Datenbank mit Hilfe des GUI

Der Startpunkt ist das JOURNAL. Hier erhalten Sie alle Informationen über die durchgeführten Sicherungen, wie schon in Abschnitt 9.4.2 „Welche Daten werden zur Wiederherstellung benötigt?“ erwähnt wurde. Aus dem JOURNAL können Sie nun eine Sicherung auswählen. Mit der rechten Maustaste erhalten Sie das Untermenü WIEDERHERSTELLEN..., wie in Abbildung 9.46 zu sehen ist.



Abbildung 9.46: Journal mit der Option zur Wiederherstellung einer Datenbank

Von hier aus werden Sie vom Assistenten für Datenwiederherstellung geführt und in zehn Schritten durch die Einstellungen geleitet. In diesen Schritten können Sie einige Punkte auswählen, müssen dies aber nicht, wenn diese Werte von DB2 vorgelegt sind.



Abbildung 9.47: Assistent: Daten wiederherstellen – Schritt 1 – Einführung

Im ersten Schritt (siehe Abbildung 9.47) treffen Sie die wichtigste Entscheidung: Sie legen fest, ob eine bestehende Datenbank überschrieben werden soll oder ob eine „neue“ Datenbank mit der Sicherung erstellt wird. Die dritte Option ist hier für die Möglichkeit, dass Sie nur die Protokollierungsinformationsdatei, also das „Recovery-History-File“, wiederherstellen möchten.

Im zweiten Schritt (siehe Abbildung 9.48) haben Sie nun die Möglichkeit, nur Teile, also hier die Tabellenbereiche, separat wiederherzustellen. Sie sehen hier eine Auflistung der Tabellenbereiche, die in diesem Sicherungsbild vorhanden sind und die auch bei Bedarf einzeln wiederhergestellt werden können.



Abbildung 9.48: Assistent: Daten wiederherstellen – Schritt 2 – Wiederherstellungsobjekte

In Schritt 3 (siehe Abbildung 9.49) wird die Sicherungsdatei ausgewählt. Diesen Schritt können Sie übergangen, wenn Sie über das JOURNAL eingestiegen sind und dort schon Ihre Auswahl getroffen haben. Diese Werte werden hier automatisch übernommen, und Sie müssen sich keine weiteren Gedanken darüber machen. Sind Sie jedoch nicht über das *Journal* eingestiegen, sondern haben Sie den Assistenten zur Datenwiederherstellung aus einem anderen Kontextmenü aufgerufen, dann müssen Sie in diesem Schritt das Sicherungsbild auswählen.

Schritt 4 (ohne Abbildung) gibt Ihnen die Auswahl zum Umdenken der Tabellenbereichsbehälter (Container). Dies kann bei fehlerhaften Platten bzw. anderen Fehlersituationen notwendig werden. Wir gehen auf dieses Detail in Abschnitt 9.4.7 „Umgeleitete Wiederherstellung (redirected restore)“ noch genauer ein.



Abbildung 9.49: Assistent: Daten wiederherstellen – Schritt 3

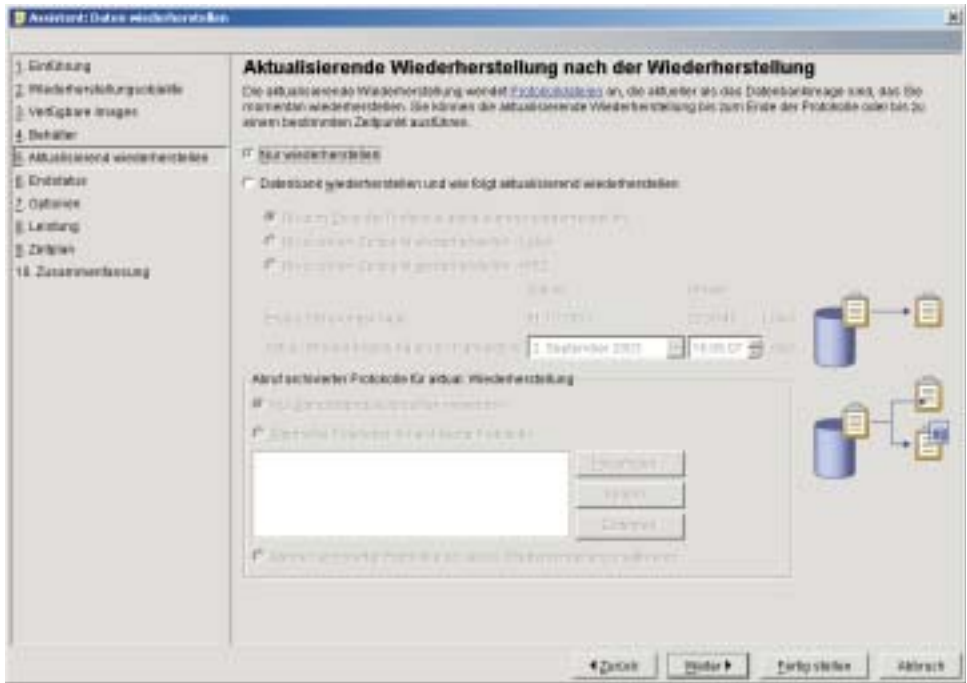


Abbildung 9.50: Assistent: Daten wiederherstellen – Schritt 5 – Aktualisierende Wiederherstellung

In Schritt 5 (siehe Abbildung 9.50) entscheiden Sie, ob Sie die Datenbank NUR WIEDERHERSTELLEN oder ob Sie die Datenbank zusätzlich aktualisieren wollen. Die zweite Option beinhaltet einen Rollforward, für den ein bestimmter Zeitpunkt gewählt werden kann bzw. der bis zum Ende der Protokolldateien durchgeführt werden kann. Details dazu finden Sie in Abschnitt 9.4.5 „Aktualisierende Wiederherstellung (rollforward)“.

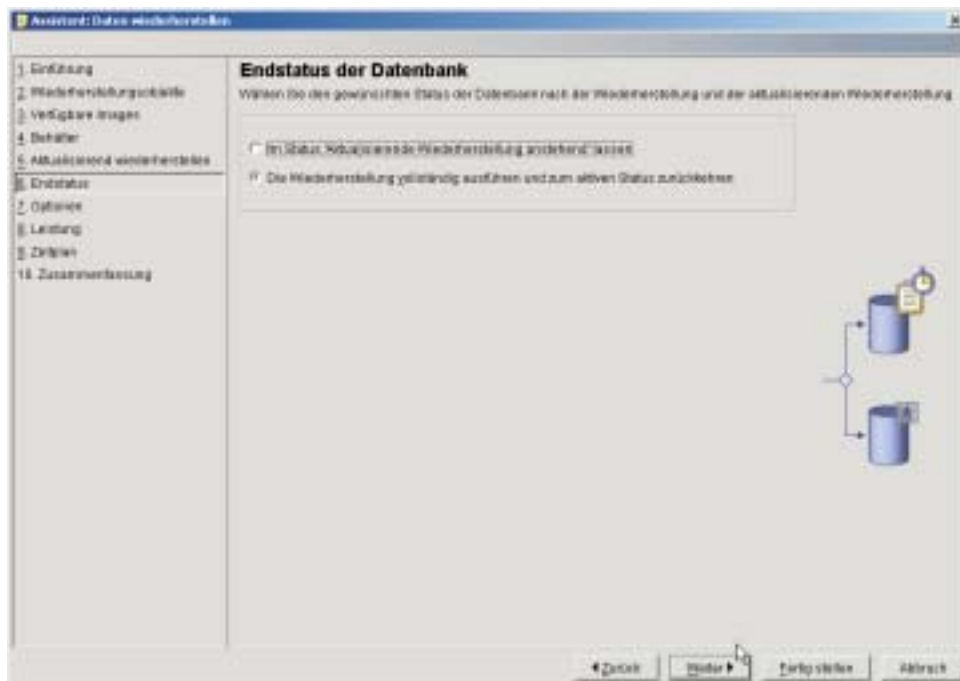


Abbildung 9.51: Assistent: Daten wiederherstellen – Schritt 6 – Endstatus

In Schritt 6 entscheiden Sie, ob Sie die Datenbank nach einem erfolgreichen Zurückspielen der Sicherung im Wiederherstellungsmodus lassen wollen. Dies ist eigentlich nur notwendig, wenn Sie danach auch eine aktualisierende Wiederherstellung der Daten über die Protokolldateien durchführen wollen. Sind Sie sich sicher, dass dieser Wiederherstellungsstand der richtige ist, so kann der zweite Punkt, DIE WIEDERHERSTELLUNG VOLLSTÄNDIG AUSFÜHREN UND ZUM AKTIVEN STATUS ZURÜCKKEHREN, ausgewählt werden.

Die Schritte 7 bis 9 können getrost übergangen werden, da sie alle mit Standardwerten belegt sind und nicht verändert werden müssen.

In Schritt 10 (siehe Abbildung 9.52) sehen wir nun die Zusammenfassung aller Eingaben, die wir hier getätigt haben bzw. die als Vorgaben übernommen wurden. In diesem Schritt ist es unserer Meinung nach immer wichtig, sich den Befehl anzeigen zu lassen, der von DB2 hier generiert wird, da solche Befehle für einen Administrator immer von Interesse sind, um sie später in Automatisierungsskripts einbauen zu können.



Abbildung 9.52: Assistent: Daten wiederherstellen – Schritt 10 – Zusammenfassung

Der Befehl wird von DB2 so ausgeführt, wie er hier im Fenster (siehe Abbildung 9.52) angezeigt wird. Er kann auch direkt so von der DB2-Kommandozeile abgesetzt werden.

9.4.4 Wiederherstellen einer Datenbank mit DB2-Befehlen

Einen DB2-Befehl zum Wiederherstellen einer Datenbank haben wir schon im Schritt 10 (siehe Abbildung 9.52) gesehen. Dieser Befehl wird so, wie er im Befehlsfenster angezeigt ist, direkt über den Assistenten zur Datenwiederherstellung ausgeführt.

Falls wir eine Datenbankwiederherstellung über die DB2-Kommandozeile durchführen, sind noch einige Punkte zu beachten.

Der einfachste Wiederherstellungsbefehl ist:

```
restore db buchdb
```

Damit dieser Befehl auch so funktioniert, müssen Sie sich in dem Verzeichnis befinden, das Sie für das Sicherungsbild angegeben haben, und es darf nur eine Sicherungsversion in diesem Verzeichnis vorhanden sein. Trifft eine Bedingung nicht mehr zu, dann müssen Sie weitere Optionen zur Einschränkung bzw. zur genaueren Auswahl der Sicherungsdatei angeben. Wenn Sie sich nicht im Pfad der Sicherung befinden, dann kann mit der Option FROM ein Pfad oder ein Laufwerk angegeben werden (siehe Abbildung 9.53).

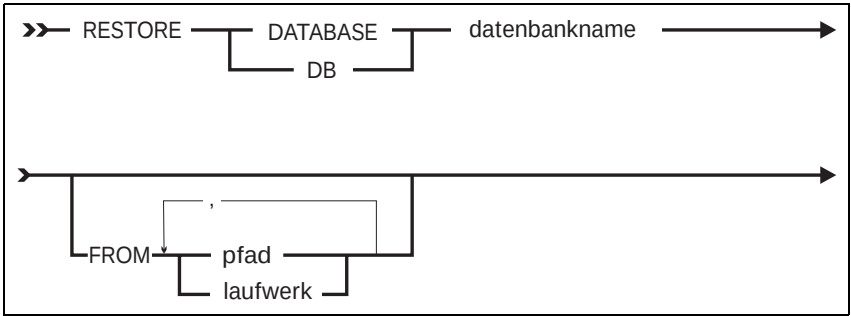


Abbildung 9.53: restore database-Syntaxdiagramm

Am einfachsten holen Sie sich diese Daten über den Befehl:

```
list history backup all for database buchdb
```

Aus dieser Auflistung kopieren Sie den zugehörigen Satz auf die DB2-Kommandozeile und ergänzen diesen um den RESTORE-Befehl:

```
restore db buchdb from C:\buchdb\sicherungen
```

Auch dieser Befehl funktioniert nur, wenn in diesem Verzeichnis nur eine einzelne Sicherungsdatei liegt. Sobald hier mehr als *ein* Sicherungsimage vorhanden ist, müssen Sie auch den Zeitstempel von dieser Sicherung mitgeben. Diesen können Sie aber auch aus der Auflistung des LIST HISTORY-Befehls herauskopieren (siehe Abbildung 9.54), damit Sie sich hier nicht vertippen.

Hinweis:

Bitte achten Sie darauf, dass hier die 001, also die letzten drei Ziffern, nicht mit angegeben werden. Diese drei Ziffern sind die Folgenummer und gehören somit nicht mehr zum Zeitstempel.

Op	Obj	Zeitmarke+Folge	Typ	Einh.	1.Protok.	Akt. Protok.	RFW-Stoppzeit
B	D	20030901233811001	N	D	S0000017.LOG	S0000018.LOG	
Enthält 2 Tabellenbereich(e):							
00001 SYSCATSPACE							
00002 USERSPACE1							
Kommentar: DB2 BACKUP BUCHDB ONLINE							
Startzeit: 20030901233811							
Endzeit: 20030901233846							
00002 Adresse: C:\buchdb\sicherungen\BUCHDB.0\DB2\NODE0000\CATN0000\20030901							

Abbildung 9.54: Auszug aus der Protokollierungsinformationsdatei mit Zeitmarke

Dies ergibt folgenden Befehl:

```
restore db buchdb from C:\buchdb\sicherungen  
taken at 20030901233811
```

Unter Umständen sehen Sie folgende Frage auf der Befehlszeile:

SQL2539W Achtung! Wiederherstellung in eine bestehende Datenbank, die mit der Datenbank des Sicherungsbildes identisch ist. Die Datenbankdateien werden gelöscht.

Wollen Sie fortfahren? (J/N)

Mit dieser Warnung stellt DB2 sicher, dass Sie sich darüber im Klaren sind, dass nun die bestehende Datenbank überschrieben wird, sobald Sie diese Frage bestätigen.

Tipp:

Eventuell müssen Sie auch in der deutschen Version „Y“ (für „YES“) anstatt „J“ für „Ja“ eingeben, obwohl es anders vorgegeben ist.

Nun werden die Daten von der Sicherungsdatei auf die Tabellenbereiche zurückgespielt. Wenn die Datenbank mit der Wiederherstellung fertig ist, erhalten Sie folgende Meldung:

DB20000I Der Befehl RESTORE DATABASE wurde erfolgreich ausgeführt.

Ist Ihre Datenbank jetzt wieder verfügbar?

Nein, denn wenn Sie nun versuchen, sich zur Datenbank zu verbinden, erhalten Sie folgende Meldung:

```
C:\buchdb>db2 connect to buchdb
```

SQL1117N Die Verbindung zur bzw. die Aktivierung der Datenbank "BUCHDB" war auf Grund einer anstehenden aktualisierenden Wiederherstellung nicht möglich.

SQLSTATE=57019

Die Datenbank steht also noch nicht zur Verfügung, da eine anstehende Aktualisierung offen ist. Diese muss in einem zweiten Schritt durchgeführt werden. Theoretisch haben wir dies schon in Abschnitt 9.1 „Protokollierung in einer Datenbank“ behandelt, und wir kommen noch mit einigen praktischen Hinweisen im folgenden Abschnitt 9.4.5 „Aktualisierende Wiederherstellung (rollforward)“ genauer darauf zurück.

Wir zeigen Ihnen hier noch eine Auswahl von Optionen aus der Vielzahl der Möglichkeiten, die der RESTORE-Befehl zur Verfügung stellt.

Die Option TAKEN AT benötigen wir, um eindeutig festzulegen, welche Sicherung wir zum Wiederherstellen unserer Datenbank benötigen.

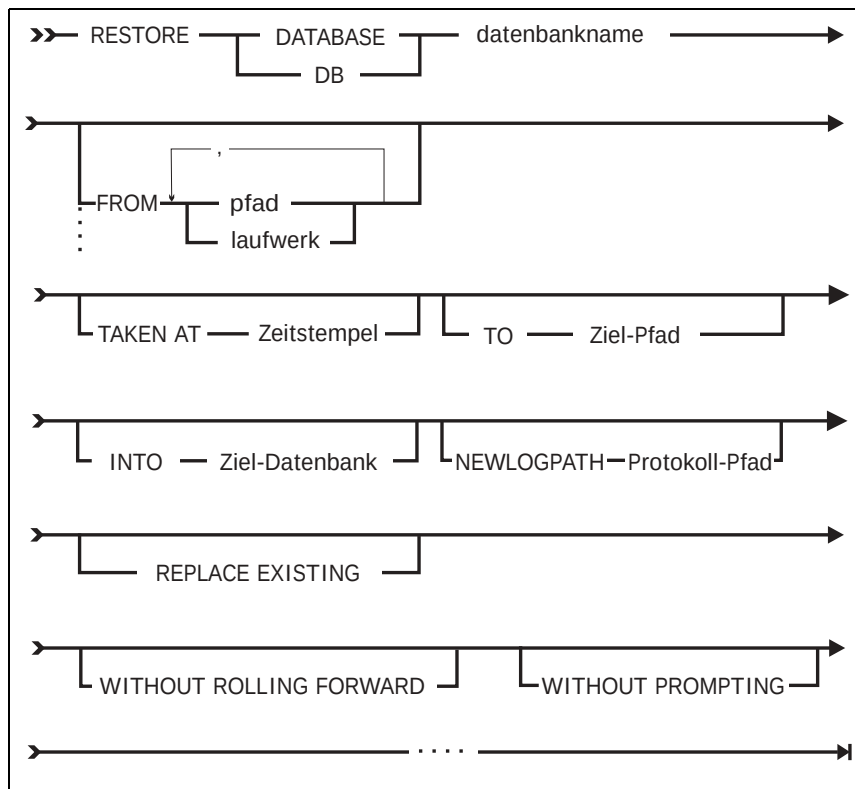


Abbildung 9.55: Restore-Befehl mit einigen Optionen

Mit der TO-Option sagen wir DB2, wohin die Daten unserer Sicherung zurückgeschrieben werden sollen, also in welchen Pfad oder in welches Verzeichnis eine neue Datenbank aus der Sicherung erstellt wird.

Mit INTO wird die Ziel-Datenbank bestimmt, in die das Sicherungsimago eingespielt werden soll.

NEWLOGPATH überschreibt die Angabe des Protokollpfads (logpath) innerhalb der Sicherung. Somit kann bei der Wiederherstellung bestimmt werden, „wohin“, also in welches Verzeichnis bzw. in welchen Pfad die Protokolldateien geschrieben werden bzw. wo sie zukünftig sein sollen.

Mit REPLACE EXISTING geben Sie an, dass Ihre bestehende Datenbank überschrieben werden soll, ohne dass DB2 nochmals einen Dialog öffnet, um nachzufragen, ob Sie wirklich sicher sind, dass Sie eine bestehende Datenbank überschreiben möchten.

WITHOUT ROLLING FORWARD ist notwendig, wenn Sie den Stand der Datenbank nach der Wiederherstellung auch als den „aktuellen“ Stand ansehen und nicht wollen, dass die Datenbank in den „aktualisierenden Wiederherstellungsmodus“ geht. Dies ist nur der Fall, wenn Sie die Datenbank mit der Archivprotokollierung betreiben, also den Parameter LOGRETAIN=ON gesetzt haben.

WITHOUT PROMPTING wird speziell für administrative Scripts benötigt, in denen kein Benutzerdialog erwünscht ist. Mit dieser Option unterdrücken Sie jegliche Nachfrage von DB2.

9.4.5 Aktualisierende Wiederherstellung (rollforward)

Die aktualisierende Wiederherstellung ist nur möglich, wenn Sie innerhalb der Datenbank den Konfigurationsparameter LOGRETAIN=ON bzw. USEREXIT=ON gesetzt haben und danach eine Offline-Sicherung angelegt wurde.

Eine aktualisierende Wiederherstellung dient dem Zweck, die Änderungen, die Sie an Ihrer Datenbank vorgenommen haben so zeitnah wie möglich wiederherzustellen. Damit sollen Ihnen im Fehlerfall so wenige Daten wie möglich verloren gehen.

Wann ist ein ROLLFORWARD notwendig ?

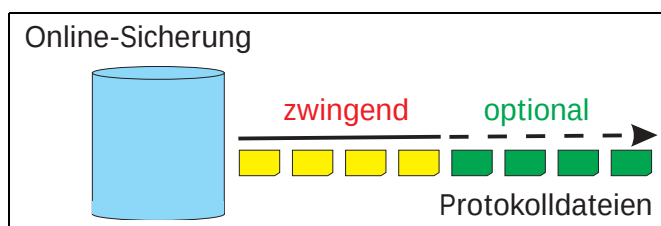


Abbildung 9.56: Online-Sicherung mit Protokolldateien

Bei einer Online-Sicherung (siehe Abbildung 9.56) können zum Zeitpunkt der Sicherung Daten im Pufferpool verändert werden. Diese Änderungen werden in den Protokolldateien festgeschrieben, aber nicht sofort in den Tabellenbereichen. Somit beinhaltet eine Online-Sicherungsdatei keine konsistente Datenbank, da der Sicherungsdatei Teile der während der Sicherung vorgenommenen Veränderungen fehlen. Erst wenn Sie die zugehörigen Protokolldateien durch den ROLLFORWARD-Befehl mit der Datenbank abgleichen und aktualisieren, erhalten Sie einen konsistenten Zustand. Deshalb zwingt DB2 Sie, bei der Wiederherstellung einer Online-Sicherung mindestens die Protokolldateien zur aktualisierenden Wiederherstellung zu verwenden, die während der Sicherung geschrieben wurden. Eine weitere aktualisierende Wiederherstellung der Datenbank zu einem Zeitpunkt nach dem Online-Backup ist über weitere vorhandene Protokolldateien möglich, aber immer von der jeweiligen Anforderung abhängig.

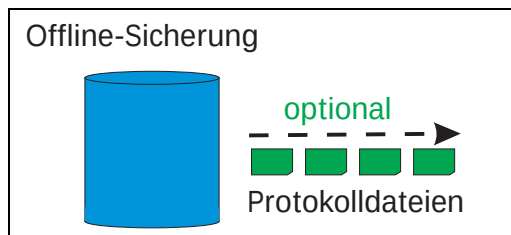


Abbildung 9.57: Offline-Sicherung mit Protokolldateien

Bei der Wiederherstellung einer Offline-Sicherung (siehe Abbildung 9.57) ist die aktualisierende Wiederherstellung optional und von den Anforderungen abhängig.

Der generelle Aufbau des ROLLFORWARD-Befehls sieht wie folgt aus:

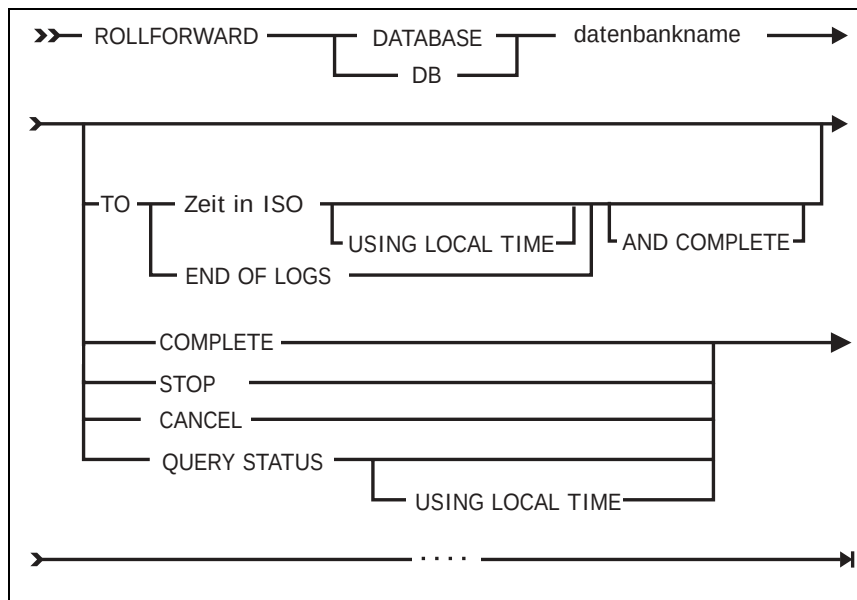


Abbildung 9.58: rollforward database-Syntaxdiagramm

In Abbildung 9.58 werden einige der wichtigsten Optionen für den ROLLFORWARD-Befehl angezeigt. (Die ausführliche Syntax ist in der DB2 UDB V8.1 Command Reference nachzulesen).

- Zeit in ISO (isotime) – Dies ist ein Zeitstempel, der in folgendem Format angegeben werden muss:

JJJ-MM-TT-SS.MM.SS.NNNNNN

(Jahr-Monat-Tag-Stunde.Minute.Sekunden.Micro-Sekunden)

Der Zeitstempel ist als isotime oder auch GMT (Greenwich Mean Time), manchmal auch als CUT (Coordinated Universal Time) oder CEST (Central European Standard Time) bezeichnet, festgelegt. Der Grund dafür ist, dass Zeitverschiebungen, wie sie durch die Sommerzeit (im Gegensatz zur „Winterzeit“) verursacht werden, sich nicht auf die Zeitstempel in den Protokolldateien auswirken dürfen. Die Zeitzone, in der wir uns befinden, weicht um eine bestimmte Anzahl von Stunden (z.B. +2) von dieser Universalzeit ab. In DB2 kann diese Zeitdifferenz über eine SQL-Abfrage herausgefunden werden:

```
select values(current timezone)
```

Sie erhalten als Ergebnis „20000“ im Format HHMMSS (Stunden, Minuten, Sekunden).

- ▶ **USING LOCAL TIME** – Damit Sie bei der wiederherstellenden Aktualisierung nicht die Zeit umrechnen müssen, haben Sie diese Option. Mit **USING LOCAL TIME** können Sie Ihre lokale Zeit im Befehl angeben. DB2 rechnet dann Ihre Eingabe intern in die ISO-Zeit um.
- ▶ **END OF LOGS** – Bei dieser Angabe werden alle Protokolldateien, die im **LOGPATH** stehen, verwendet, um die Datenbank aktualisierend wiederherzustellen.
- ▶ **END COMPLETE/STOP** – Der aktualisierende Wiederherstellungsprozess wird beendet, und die noch unvollständigen Transaktionen werden wieder auf den letzten konsistenten „Festschreibungspunkt“ zurückgerollt. Sind diese Aktivitäten erledigt, wird die Datenbank vom Status

Aktualisierende Wiederherstellung anstehend = DATABASE

(„rollforward pendig“) in den Status

Aktualisierende Wiederherstellung anstehend = NO

(„active“) versetzt, und es kann mit der Datenbank gearbeitet werden.

Hinweis:

Wird ein **ROLLFORWARD** für einen Tabellenbereich durchgeführt, wird für den jeweiligen Tabellenbereich der Status Sicherung anstehend gesetzt, und es ist eine Tabellenbereichssicherung notwendig.

- ▶ **CANCEL** – Beendet die aktive aktualisierende Wiederherstellung und setzt die Datenbank bzw. den oder die Tabellenbereiche in den Wiederherstellungsstatus („restore pendig“).

```
C:\buchdb\sicherungen>db2 rollforward db buchdb cancel
SQL4975W  Die aktualisierende Wiederherstellung wurde erfolgreich abgebrochen.
Die Datenbank oder die ausgewählten Tabellenbereiche müssen auf dem/den Knoten
"0" wiederhergestellt werden.
```

Abbildung 9.59: Beispiel: rollforward database mit cancel-Option

- ▶ **QUERY STATUS** – zeigt Statusinformationen über den Stand der Rollforward-Aktivität an:

```
Aliasname der Eingabedatenbank      = buchdb
Anzahl der Knoten, die Status zurückgaben = 1
Knotennummer                        = 0
Aktualisierende Wiederherstellung: Status = DB aktiv
```

Im Status-Feld steht der derzeitige Zustand der wiederherstellenden Aktualisierung. Hier ist es **DB aktiv**, es kann aber auch der Status **TS aktiv** und nicht anstehend sein.

```
Nächste zu lesende Protokolldatei    = S0000022.LOG
```

Hier wird die nächste Protokolldatei (**S0000022.LOG**) angezeigt, die von **DB2** gelesen werden soll.

Verarbeitete Protokolldateien = S0000017.LOG - S0000021.LOG

Hier sehen Sie die Protokolldateien (17-21), die nicht mehr für die Wiederherstellung notwendig sind. Würde noch eine Transaktion nicht vollständig wiederhergestellt sein, so wären die zugehörigen Protokolldateien hier nicht aufgeführt.

Diese Protokolldateien (17-21) können nun aus dem LOGPATH entfernt werden.

Letzte festgeschriebene Transaktion=2003-09-01-21.56.25.000000

Dieser Zeitstempel zeigt den Zeitpunkt der letzten festgeschriebenen Transaktion an, der auch in die Datenbank übernommen wird, wenn Sie den ROLLFORWARD nun mit STOP oder COMPLETE beenden.

In den nächsten Abbildungen zeigen wir Ihnen typische ROLLFORWARD-Situationen.

ROLLFORWARD wird erzwungen

Wird eine Online-Sicherung wiederhergestellt und danach sofort versucht, den ausstehenden ROLLFORWARD zu stoppen, sieht die Fehlermeldung wie in Abbildung 9.60 aus:

```
C:\buchdb> db2 rollforward db buchdb stop
SQL1276N  Der Status 'aktualisierende Wiederherstellung anstehend' der
Datenbank "BUCHDB" kann erst beendet werden, wenn die aktualisierende
Wiederherstellung einen Zeitpunkt erreicht hat, der größer bzw. gleich
"2003-09-01-21.38.45.000000" ist, da der Knoten "0" Informationen enthält, die
aktueller als der angegebene Zeitpunkt sind.
```

Abbildung 9.60: Beispiel: rollforward database mit stop-Option

In Abbildung 9.60 wurde versucht, ein ROLLFORWARD mit der STOP-Option zu beenden. Da dies aber eine Wiederherstellung einer Online-Sicherung war, gibt es einen Zeitstempel, zu dem mindestens die aktualisierende Wiederherstellung durchgeführt werden muss. Dieser Zeitstempel wird von DB2 angezeigt, so dass Sie nun diesen Zeitpunkt verwenden können, um die Wiederherstellung exakt bis zu diesem Punkt zu vervollständigen.

```
C:\buchdb>db2 rollforward db buchdb to end of logs

                                Status der aktualisierenden Wiederherstellung

Aliasname der Eingabedatenbank           = buchdb
Anzahl der Knoten, die Status zurückgaben = 1

Knotennummer                             = 0
Aktualisierende Wiederherstellung: Status = DB aktiv
Nächste zu lesende Protokolldatei        = S0000019.LOG

Verarbeitete Protokolldateien             = -
Letzte festgeschriebene Transaktion       = 2003-09-01-2
1.48.55.000000

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.
```

Abbildung 9.61: Beispiel: rollforward database query status mit fehlender Protokolldatei

In Abbildung 9.61 sehen Sie, dass die Nächste zu lesende Protokolldatei S0000019.LOG ist, aber in den Verarbeiteten Protokolldateien kein Wert steht. Hier ist also mindestens eine Transaktion immer noch offen, und es werden alle vorhergehenden Protokolldateien benötigt, um diese Transaktion(en) zurücksetzen zu können. Dies wäre notwendig, wenn die Aktualisierung zu diesem Zeitpunkt beendet wird.

```

                                Status der aktualisierenden Wiederherstellung

Aliasname der Eingabedatenbank                = buchdb
Anzahl der Knoten, die Status zurückgaben      = 1

Knotennummer                                = 0
Aktualisierende Wiederherstellung: Status      = DB aktiv
Nächste zu lesende Protokolldatei             = S0000022.LOG

Verarbeitete Protokolldateien                 = S0000017.LOG
- S0000021.LOG
Letzte festgeschriebene Transaktion           = 2003-09-01-2
1.56.25.000000

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.

```

Abbildung 9.62: Beispiel: rollforward database query status – alle Logs verarbeitet

In Abbildung 9.62 liegt der Fall schon anders: Hier wurden Transaktionen verarbeitet und festgeschrieben, so dass die zugehörigen Protokolldateien nicht mehr weiter benötigt werden (17-19). Die Nächste zu lesende Protokolldatei steht auf 22, das heißt aber nicht unbedingt, dass in der Protokolldatei 22 eine noch offene Transaktion verzeichnet ist, sondern nur, dass diese Protokolldatei die nächste ist, die gelesen wird.

```

C:\buchdb>db2 rollforward db buchdb stop

                                Status der aktualisierenden Wiederherstellung

Aliasname der Eingabedatenbank                = buchdb
Anzahl der Knoten, die Status zurückgaben      = 1

Knotennummer                                = 0
Aktualisierende Wiederherstellung: Status      = nicht anstehend
Nächste zu lesende Protokolldatei             =
Verarbeitete Protokolldateien                 = S0000017.LOG
- S0000021.LOG
Letzte festgeschriebene Transaktion           = 2003-09-01-2
1.56.25.000000

DB20000I  Der Befehl ROLLFORWARD wurde erfolgreich ausgeführt.

```

Abbildung 9.63: Beispiel: rollforward database stop

In Abbildung 9.63 wurde der STOP vom ROLLFORWARD durchgeführt. Der Status ändert sich nun auf nicht anstehend und zeigt noch einmal, welche Protokolldateien nicht mehr benötigt werden. Hier wird erneut deutlich, dass nur die Protokolldateien 17 - 21 für das ROLLFORWARD notwendig waren, nicht aber die Informationen auf dem Log 22. Das Log 22 ist nun die erste aktive Protokolldatei, die im weiteren Verlauf beschrieben wird.

9.4.6 Wann werden Protokolldateien überschrieben?

Im Wiederherstellungsfall, also wenn Sie Ihre Datenbank aus einem Sicherungsimage wiederhergestellt haben und im zweiten Schritt nun die Protokolldateien durchlaufen, müssen Sie folgende Dinge beachten:

- ▶ DB2 warnt Sie nicht, wenn Protokolldateien überschrieben werden.
- ▶ Bei der Umlaufprotokollierung werden regelmäßig Protokolldateien überschrieben, die nicht mehr für die CRASH-Recovery benötigt werden, aber das ist ja auch so gewollt.
- ▶ Vorsicht ist auch geboten, wenn Sie die Konfiguration der Datenbank verändern. Protokolldateien fangen mit der Nummerierung dann neu an, wenn:
 - die Datenbankkonfiguration auf Archiv-Protokollierung LOGRETAIN=ON umgestellt wird
 - die Datenbankkonfiguration auf Umlauf-Protokollierung LOGRETAIN=OFF zurückgestellt wird
 - die Protokollnummer auf S9999999.LOG ist und auf S0000000.LOG umschaltet
- ▶ Bevor Sie einen ROLLFORWARD-Befehl absetzen, sollten Sie eine Kopie der aktuellen Protokolldateien erstellen, die in Ihrem Protokollpfad (LOGPATH) stehen.

In Abbildung 9.64 sehen Sie, dass in der ersten Zeitachse (t1) eine Online-Sicherung erstellt wurde. Während dieser Online-Sicherung wurde weitergearbeitet, und es wurden die Protokolldateien 17-20 geschrieben. Danach ging die Verarbeitung normal weiter, und es wurden weitere Protokolldateien gefüllt (21-22).

Im weiteren Verlauf der Verarbeitung entstand eine Fehlersituation (z.B. durch einen Logikfehler in einer Anwendung), die Sie nur über eine Wiederherstellung auf einen bestimmten Zeitpunkt bereinigen können. In diesem Beispiel wird die Offline-Sicherung für die Wiederherstellung verwendet, da der Fehler vermutlich zu einem Zeitpunkt verursacht wurde, zu dem auch die Online-Sicherung lief (wodurch sie nicht für die Wiederherstellung verwendet werden kann).

Sie sehen in der Zeitachse t2 in Abbildung 9.64, dass bei dieser aktualisierenden Wiederherstellung nach dem RESTORE der Offline-Sicherung ein ROLLFORWARD notwendig ist, um zu einem bestimmten Zeitstempel zu kommen. Dieser Zeitpunkt liegt vor der vermutlichen Fehlersituation; hier am Ende der Protokolldatei 18 (S0000018.LOG).

Nach dem Beenden der aktualisierenden Wiederherstellung durch die STOP- oder COMPLETE-Option des ROLLFORWARD-Befehls wechselt der Datenbankstatus von

Aktualisierende Wiederherstellung anstehend = DATABASE

zu:

Aktualisierende Wiederherstellung anstehend = NO

Ab diesem Zeitpunkt kann wieder mit der Datenbank gearbeitet werden. Jetzt wird die „nächste“ Protokolldatei genommen, um die „neuen“ Datenveränderungen zu protokollieren. Es werden also in die bestehende (alte) Protokolldatei 19 „neue“ Protokollsätze eingefügt und somit die alten überschrieben.

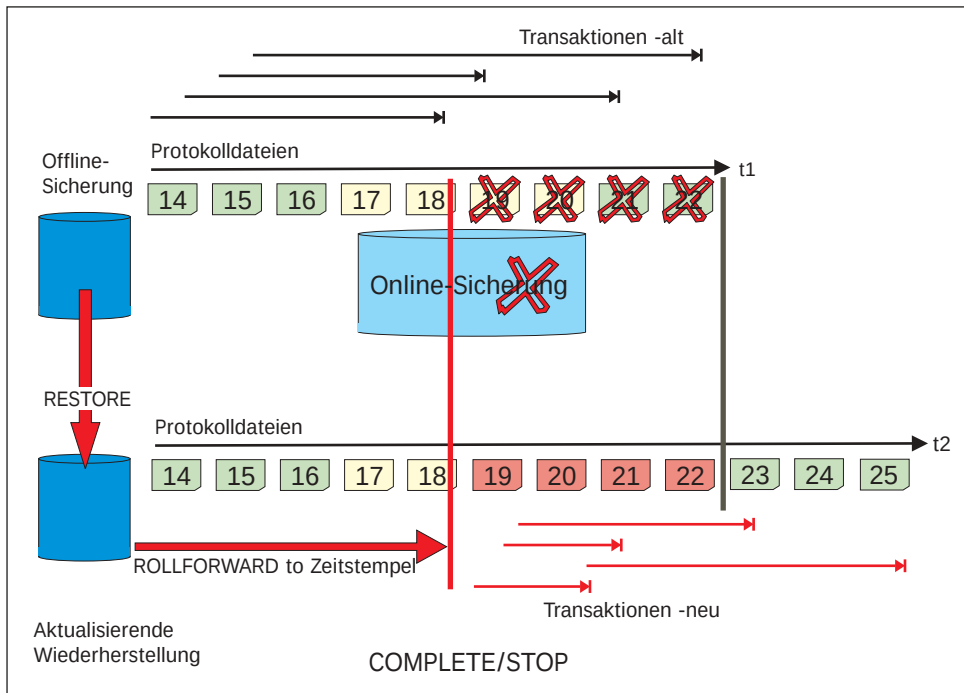


Abbildung 9.64: Wann werden Protokolldateien überschrieben?

Dies muss nicht unbedingt ein Problem sein. Wenn Sie aber später die Online-Sicherung, die Sie erstellt haben, zum Wiederherstellen verwenden möchten, dann ist diese nicht mehr zu gebrauchen, da ein Teil (19-20) der zugehörigen Protokolldateien (17-20) überschrieben worden ist. In diesem Fall ist es sehr wichtig, dass Sie alle Protokolldateien vorher gesichert haben, um später diese Online-Sicherung verwenden zu können.

Hinweis:

Diese Online-Sicherung kann nach der Wiederherstellung des Offline-Backups von Zeitachse t2 nur noch dazu dienen, Zustände wiederherzustellen, die in den gesicherten („alten“) Protokolldateien 21 und 22 sind. Warum? Weil Sie die weiteren Protokolldateien (23-25) nicht mit dem ROLLFORWARD-Befehl verarbeiten können, da diese zu einer anderen so genannten *Protokollkette* (log chain) gehören. Die Protokolldateien (23-25) gehören zu der neuen Protokollkette, die mit den neuen Protokolldateien (19-22) beginnt und die Änderungen der „neuen“ Transaktionen beinhaltet.

9.4.7 Umgeleitete Wiederherstellung (redirected restore)

Jede Sicherung beinhaltet die Information über die physikalische Lage der Tabellenbereiche, also die Information über die Tabellenbereichsbehälter (Container). Beim Wiederherstellen der Datenbank bzw. der Tabellenbereiche wird immer erst

geprüft, ob die in der Sicherung festgehaltenen Speicherort-Informationen über die Tabellenbereichsbehälter noch gültig sind. Es wird dabei überprüft, ob die Datenträger (Speicherorte) existieren und ob auch auf sie zugegriffen werden kann. Ist dies auch nur bei einem einzigen Tabellenbereichsbehälter (Container) nicht der Fall, schlägt die Wiederherstellung fehl.

In diesem Fall ist – während der Wiederherstellung – eine Umleitung, eine Löschung oder eine Änderung des Tabellenbereichsbehälters notwendig. Dazu benötigen Sie dann eine spezielle Wiederherstellungsoption: die REDIRECT-Option beim RESTORE-Befehl.

Was nicht unterstützt wird

Bei der umgeleiteten Wiederherstellung ist es nicht möglich, den Typ eines Tabellenbereichs zu ändern, also von einem SMS-Tabellenbereich in einen DMS-Tabellenbereich umzuleiten (bzw. umgekehrt).

Die Obere Grenze (HWM – High Water Mark) im DMS-Tabellenbereich kann nicht unterschritten werden (siehe auch Kapitel 5).

Was wird unterstützt?

Beim Umleiten haben Sie die Möglichkeit, die Anzahl und die Position der Tabellenbereichsbehälter (Container) für den SMS-Tabellenbereich zu ändern – etwas, was nur über diese Umleitung geht.

Für den DMS-Tabellenbereich können Sie, neben der Anzahl und Position, auch die Größe ändern. Die Größe darf aber nicht kleiner als der vom Tabellenbereich maximal benutzte Speicherbereich sein (Obere Grenze-Eintrag beim `list tablespaces`-Befehl).

Generelle Vorgehensweise

- Stellen Sie die Datenbank mit dem Befehl `RESTORE` und der Option `REDIRECT` wieder her.
 1. Definieren Sie die Tabellenbereichsbehälter mit dem `SET TABLESPACE CONTAINERS FOR ...`-Befehl um bzw. neu.
 2. Setzen Sie die Datenbankwiederherstellung mit dem Befehl `RESTORE` und der Option `CONTINUE` fort.

Praktisches Beispiel (GUI)

Im Assistenten zur Wiederherstellung (der schon in Abschnitt 9.4.3 „Wiederherstellen einer Datenbank mit Hilfe des GUI“ besprochen wurde) sind nun in Schritt 4 die entsprechenden Einstellungen durchzuführen.

In Abbildung 9.65, d.h. im Schritt 4 des Assistenten zur Datenwiederherstellung, muss die Checkbox `UMLEITUNG VON TABELLENBEREICHSBEHÄLTERN BEI DER WIEDERHERSTELLUNGSOPERATION AKTIVIEREN` ausgewählt werden. Danach erscheint ein Fenster mit einer Warnung, dass die Datenbank sofort in den Wiederherstellungsmodus geht, falls Sie das Fenster mit `JA` schließen. Danach wird das Assistenten-Fenster für weitere Eingaben freigegeben.



Abbildung 9.65: Assistent: Daten wiederherstellen – Behälter – Umleitung mit Nachricht



Abbildung 9.66: Assistent: Daten wiederherstellen – Behälter – Umleitung mit Werten

Sie können nun den Zeitstempel der Sicherung und die Tabellenbereiche auswählen, die Sie umleiten möchten. Durch HINZUFÜGEN... wird ein weiteres Fenster geöffnet, in dem Sie die neuen Parameter für den Tabellenbereichsbehälter definieren können. Wenn Sie einen bestehenden Tabellenbereichsbehälter ausgewählt haben, können Sie diesen über ÄNDERN... oder ENTFERNEN manipulieren.

Bei der Zusammenfassung, also im Schritt 10 des Assistenten, können Sie sich wieder die Befehle anzeigen lassen, die DB2 hier zusammenbaut.

Der Befehl, der die Tabellenbereichsbehälter umdefiniert, lautet wie folgt:

```
SET TABLESPACE CONTAINERS FOR < Tabellenbereichs-Nummer> USING ....
```

Das ganze Set von Befehlen sehen Sie in Abbildung 9.67.

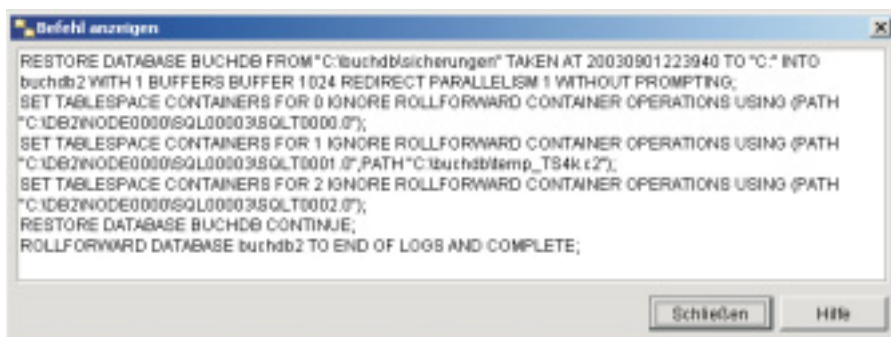


Abbildung 9.67: Befehlsfolgen für die Umleitung von Tabellenbereichsbehältern

9.5 Tipps aus der Praxis

9.5.1 Der Datenbankparameter blk_log_dsk_ful

Wenn in der Datenbank die Platte oder das Dateisystem, auf der bzw. in dem die Protokolldateien gespeichert werden (LOGPAH), nicht ausreicht, um weitere benötigte Protokolldateien zu erstellen, bricht Ihre Anwendung normalerweise ab. Sie verhindern dies, indem Sie den Datenbankkonfigurationsparameter BLK_LOG_DSK_FUL auf YES setzen.

In diesem Fall erhalten Sie eine Fehlermeldung in der Diagnosedatei db2diag.log. DB2 versucht dann alle fünf Minuten, eine neue Protokolldatei anzulegen. Die bestehenden Applikationen können weiterarbeiten. Falls diese jedoch Veränderungen an der Datenbank durchführen wollen, werden sie gestoppt. Die Anwendungen warten so lange, bis Sie das Speicherproblem im Protokollpfad gelöst haben.

9.5.2 DB2-Tool db2ckbkp

Dieses Tool prüft die Integrität einer Sicherung und stellt sicher, dass eine Sicherung zum Wiederherstellen der Datenbank verwendet werden kann. Das db2ckbkp-Tool hat auch Optionen, wie wir schon in einigen Unterpunkten (z.B. in Abschnitt 9.3.7 „Tabellenbereichssicherung“) gesehen haben, um die Metadaten aus der Sicherungsdatei auszulesen.

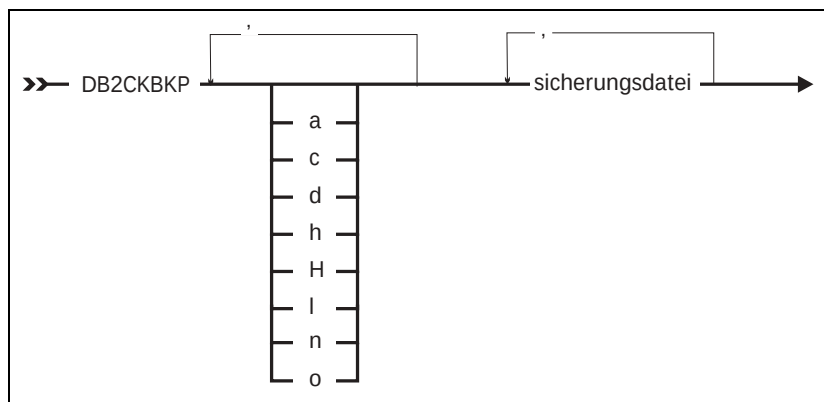


Abbildung 9.68: db2ckbkp-Syntaxdiagramm

In db2ckbkp gibt es folgende Kommando-Optionen:

- ▶ -a – listet alle verfügbaren Informationen auf.
- ▶ -c – listet das Ergebnis der Integritätsprüfung auf.
- ▶ -d – zeigt die Detailinformationen für den oder die DMS-Tabellenbereich(e) an.
- ▶ -h – listet die Media-Header-Information mit Namen und Pfadangaben auf und führt eine Integritätsprüfung durch.
- ▶ -H – zeigt die gleichen Informationen wie die „-h“-Option, aber ohne Integritätsprüfung.
- ▶ -l – listet die Header-Informationen der Protokolldatei auf.
- ▶ -n – für Sicherungen auf einer Bandeinheit.
- ▶ -o – listet Informationen über die gespeicherten Objekte auf.
- ▶ sicherungsdatei – ist der Name der Sicherungsdatei.

9.5.3 Benutzerexit (Userexit)

Die Aufgabe des Benutzerexit-Programms ist die automatische Archivierung der Protokolldateien sowie die Bereitstellung dieser Dateien im Wiederherstellungsfall.

Das Benutzerexit-Programm wird über den Datenbankparameter USEREXIT=ON eingeschaltet. Dies hat zur Folge, dass die Archiv-Protokollierung für die Datenbank

automatisch verwendet wird, unabhängig davon, ob Sie den Parameter LOGRETAIN auf ON gesetzt haben. Zusätzlich bewirkt dieser Parameter, dass ein Programm zur automatischen Archivierung der Protokolldateien verwendet wird, das mit DB2 ausgeliefert wird und vorher entsprechend angepasst werden muss.

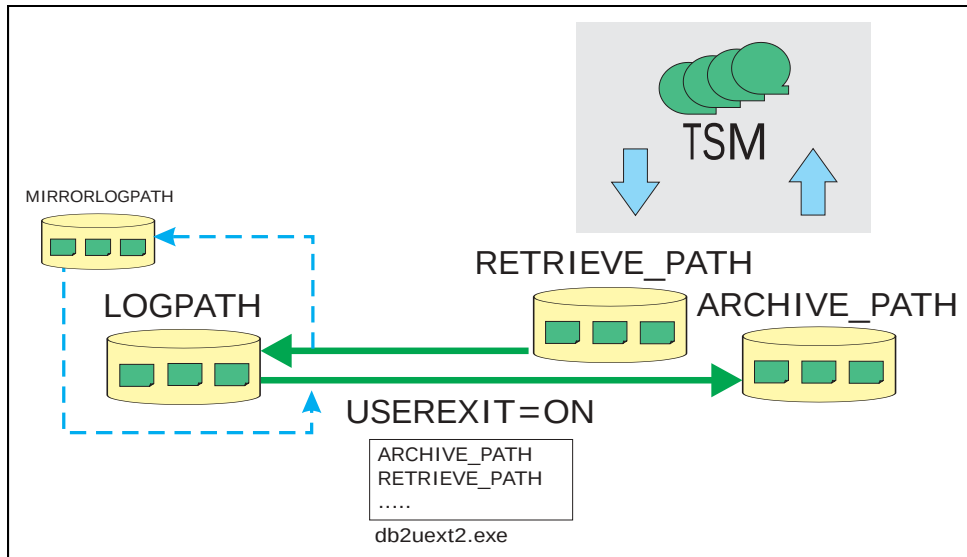


Abbildung 9.69: Überblick über die Funktionsweise des Programms Userexit

Konfiguration von Benutzerexit (Beispiel):

1. Kopieren Sie db2uext2.cdisk als db2uext2.c in ein eigenes Arbeitsverzeichnis:

```

cd c:\buchdb\work
copy c:\programme\IBM\sql1lib\sample\c\db2uext2.sqc
rename db2uext2.sqc db2uext2.c
  
```

2. Passen Sie die Pfade in dem db2uext2.c-Programm an:

```

ARCHIVE_PATH "c:\\buchdb\\archive\\"
RETRIEVE_PATH "c:\\buchdb\\archive\\"
AUDIT_ERROR_PATH "c:\\buchdb\\log\\"
  
```

3. Prüfen Sie, ob Unterverzeichnisse/Pfade schon vorhanden sind, und legen Sie diese Verzeichnisse/Pfade gegebenenfalls neu an.
4. Kompilieren und linken Sie das Benutzerexit-Programm:

```

cl -o db2uext2.exe db2uext2.c
  
```

5. Kopieren Sie das ausführbare Benutzerexit-Programm in den DB2-Pfad:

```

copy db2uext2.exe .....sql1lib\bin
  
```

Wann wird archiviert?

Die Archivierung der Protokolldateien erfolgt, sobald eine Protokolldatei vollgeschrieben worden ist. Danach erhält das Benutzerexit-Programm von DB2 den Auftrag, die Protokolldatei in den im Benutzerexit-Programm definierten `ARCHIVE_PATH` zu kopieren. Die Protokolldatei wird nach dem Kopieren *nicht* aus dem `LOGPATH` entfernt, da in dieser Protokolldatei noch Transaktionen aktiv sein könnten. Im Wiederherstellungsfall müsste DB2 auf diese Information zurückgreifen. Falls Sie Ihre Protokolldateien gespiegelt haben, also den `MIRRORLOGPATH` gesetzt haben, liest das Benutzerexit-Programm im Fehlerfall von dem noch verfügbaren Protokollpfad (`LOGPATH` oder `MIRRORLOGPATH`) die Protokolldateien.

Die Protokolldateien werden erst von DB2 wiederverwendet, wenn diese Daten nicht mehr für eine Crash-Recovery notwendig sind. Dann findet aus Leistungsgründen eine Umbenennung der Protokolldateien statt.

Diese Aussage trifft nicht zu, wenn Sie `LOGSECOND` auf „-1“ (unbegrenzter Protokollspeicher) gesetzt haben. In diesem Fall werden die Protokolldateien auch mit den noch aktiven Transaktionen überschrieben, nachdem sie vom Benutzerexit-Programm kopiert worden sind. Dies ist nur notwendig, wenn Sie Transaktionen einsetzen, bei denen das Protokollierungslimit von 256 Gbyte nicht ausreicht.

Die Archivierung kann auch manuell über den Befehl

```
ARCHIVE LOG FOR DB buchdb
```

angestoßen werden. Durch diesen Befehl werden die Protokolldateien geschlossen. Wenn die Protokolldateien zu diesem Zeitpunkt noch nicht vollständig mit Veränderungsinformationen gefüllt sind, ergibt das eine geringere Dateigröße, als in `LOGFILSIZ` definiert ist. Anschließend startet das Benutzerexit-Programm die Archivierung.

Das Abschneiden der Protokolldateien erfolgt auch nach einer Online-Sicherung, so dass nach der Fertigstellung dieser Sicherung auch die Archivierung durch das Benutzerexit-Programm ablaufen kann.

Wann werden die Protokolldateien in den LOGPATH zurückkopiert?

Wird in der `ROLLFORWARD`-Phase eine Protokolldatei nicht im Protokollpfad gefunden, dann wird das Benutzerexit-Programm aufgefordert, die angeforderte Protokolldatei in den `LOGPATH` zurückzukopieren. Diese Protokolldateien werden sofort nach der Verarbeitung aus dem `LOGPATH` gelöscht, um neuen Platz zu schaffen.

Wird die Option `LOGSECOND` auf „-1“ gesetzt, dann werden auch Protokolldateien zurückgeholt, auf denen aktive Transaktionen sind, die für die Crash-Recovery benötigt werden. Dadurch kann eine `RECOVERY` in diesem Fall länger dauern.

Weitere Hinweise

Im Unterverzeichnis `c:\programme\IBM\sqllib\sample\c\` bzw. in `$HOME/sqllib/sample/c/` gibt es noch weitere Beispiele für den Einsatz von Benutzerexit-Programmen. Zu archivierende Protokolldateien können z. B. direkt aus dem `LOGPATH` an die TSM-Speicherverwaltungssoftware (Tivoli Storage Manager) übergeben werden.

9.5.4 Protokollierung ausschalten

Es wird immer wieder gefragt, welche Möglichkeiten es gibt, um den Vorgang der Protokollierung abzuschalten. Die generelle Antwort darauf ist: Es gibt keine Möglichkeit. Das bedeutet, es gibt keinen Datenbankkonfigurationsparameter, mit dem Sie verhindern können, dass DB2 Veränderungen in seinen Protokolldateien festhält.

Wie immer gibt es auch hier Ausnahmen:

► **NOT LOGGED INITIALLY**

Beim Anlegen einer Tabelle (CREATE TABLE) gibt es die Option NOT LOGGED INITIALLY. Diese führt dazu, dass nach dem Aktivieren dieser Option mit

```
activate <tabellen-name> not logged initially
```

alle Veränderungen, die nach der Aktivierung innerhalb dieser Tabelle stattfinden, nicht in den Protokolldateien abgebildet werden. Alle Änderungen, die sich auf Katalogtabellen beziehen, werden aber weiterhin protokolliert. Dieser Zustand hat aber nur bis zur Festschreibung dieser Transaktion Gültigkeit, danach ist die Tabelle wieder im „normalen“ Zustand, und es werden wieder alle Veränderungen protokolliert.

Wichtig!

Tabellen, für die not logged initially aktiviert wurde, können bei einer aktualisierenden Wiederherstellung von DB2 nicht mehr wiederhergestellt werden. Diese Tabelle wird dann als „nicht verfügbar“ gekennzeichnet und kann nur noch gelöscht werden.

► **Deklarierte temporäre Tabellen**

Für deklarierte temporäre Tabellen findet normalerweise keine Protokollierung statt – auch nicht nach der ersten Festschreibungsanweisung.

Deklarierte temporäre Tabellen werden auch nicht im DB2-Katalog festgehalten. Somit werden hier auch keine Sperren erzeugt. Diese temporären Tabellen werden implizit gelöscht, wenn die Verbindung der Anwendung zur Datenbank beendet wird. Näheres zu temporären Tabellen finden Sie in Abschnitt 7.2.2.

► **Lade-Tool**

Beim Laden der Daten mit dem Lade-Tool (load utility) werden die Protokolldateien auch nicht geschrieben. Es werden nur Kontrolleinträge geschrieben, die aufzeigen, wann das Lade-Tool gestartet und beendet wurde.

Hinweis:

Eine Ausnahme gibt es beim Lade-Tool, wenn Sie Ausnahmetabellen einsetzen und diese befüllt werden.

9.5.5 Verändern von Kontrollinformationen (db2relocatedb)

DB2 legt viele Informationen in internen Konfigurationsdateien ab. Einige Konfigurationsdateien können Sie mit DB2-Befehlen verändern, wie z.B. die der Datenbankmanager-Konfiguration (UPDATE DBM CFG) und der Datenbankkonfiguration (UPDATE DB CFG). Wollen Sie aber z.B. die Behälter der Tabellenbereiche ändern, um z.B. die Dateinamen einheitlich zu gestalten, so müssten Sie eigentlich dazu ein RESTORE der Datenbank mit einer Umleitung durchführen.

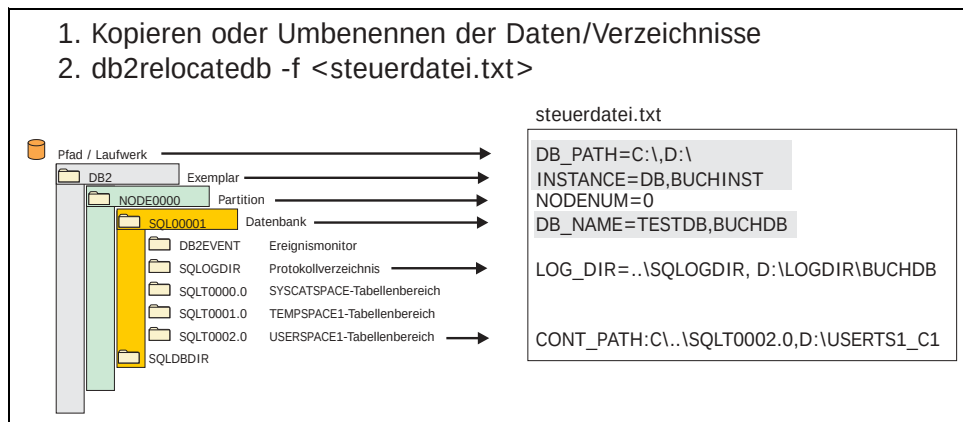


Abbildung 9.70: Beispiel: db2relocatedb

Mit dem Tool db2relocatedb ist das physikalische Verlagern von einer Datenbank oder Teilen davon um einiges einfacher.

Folgende Schritte sind durchzuführen:

1. Manuelles Umkopieren bzw. Umbenennen von Datenstrukturen – dies wird nicht vom Tool erledigt.
2. Eingabedatei erstellen mit folgenden Informationen:
 - DB_NAME=<alter DB-Name>,<neuer DB-Name>
 - DB_PATH=<alter DB-Pfad>,<neuer DB-Pfad>
 - INSTANCE=<alter Exemplar-Name>,<neuer Exemplar-Name>
 - NODENUM=<Partitionsnummer, auf der die Veränderung erfolgt>
 - LOG_DIR=<alter Protokollpfad>,<neuer Protokollpfad>
 - CONT_PATH=<alter Behältername>,<neuer Behältername>
 - CONT_PATH=<alter Behältername>,<neuer Behältername>

Die ersten drei Optionen sind „zwingend“; DB_NAME, DB_PATH und INSTANCE müssen also immer eingegeben werden, auch wenn sie nicht verändert werden sollen.

Tipp:

Sie können bei den Tabellenbereichsbehältern mit * arbeiten, damit Sie nicht jeden Behälter auflühren müssen:

CONT_PATH=<alter Behältername*>,<neuer Behältername*>

10 Zugriffsverwaltung in DB2

Die Authentifizierung auf einem DB2-Server haben wir in Kapitel 6.3 bereits besprochen. Sie stellt die eine Sicherheitsstufe dar, um überhaupt auf den Server zugreifen zu können. Eine weitere Sicherheitsstufe bildet die Rechte-Vergabe im DB2, die so genannte *Autorisierung* (authorization), mit der wir uns in diesem Kapitel beschäftigen werden. Die Autorisierung kann sich auf die Exemplar- oder Datenbank-Ebene beziehen.

10.1 Allgemeines zu DB2-Berechtigungen

Berechtigungen in der DB2 UDB beziehen sich immer auf Benutzer und Gruppen des Betriebssystems. Das heißt, es gibt keine eigenen, spezifischen DB2-Benutzer, die zusätzlich zu denen des Betriebssystems angelegt und gepflegt werden müssten.

Nur eine DB2-Gruppe bildet die Ausnahme zu dieser Regel: die Gruppe PUBLIC. Sie umfasst alle Nutzer eines Systems. Wird ein Recht an die Gruppe PUBLIC vergeben, bedeutet dies, dass jeder Nutzer im System dieses Recht besitzt.

Ein weiterer Grundsatz in DB2 ist, dass jeder, der ein Objekt (z.B. eine Tabelle) anlegt, auch alle Rechte auf „sein“ Objekt besitzt. Voraussetzung ist natürlich, dass der Benutzer zur Anlage dieses Objekts berechtigt ist.

Bei jedem Zugriff wird von der DB2 UDB automatisch eine implizite Berechtigungsprüfung durchgeführt, um diese Sicherheitseinstellungen umzusetzen.

Es werden zwei Arten von Berechtigungen unterschieden:

► Zugriffsrechte

Ein Zugriffsrecht auf Datenbankobjekte bzw. -ressourcen wird an einen Benutzer oder eine Benutzergruppe vergeben. Diese Zugriffsrechte werden in den Datenbankkatalog-Tabellen gespeichert.

► Berechtigungsstufen

Berechtigungsstufen sind eine Zusammenfassung mehrerer Zugriffsrechte auf Datenbank-Ebene und von Rechten auf Ebene des Exemplars (sie sind somit datenbankübergreifend). Einige Berechtigungsstufen werden dazu verwendet, Verwaltungsaufgaben auf Exemplarebene auszuführen. Diese Berechtigungsstufen können über die Exemplar-Konfiguration gespeichert und Betriebssystemgruppen zugeordnet werden. Die datenbankinternen Rechte sind analog zu den Zugriffsrechten im Systemkatalog gespeichert.

Beide Berechtigungsarten werden in den folgenden Abschnitten näher erläutert.

10.2 Zugriffsrechte innerhalb einer Datenbank

Die Zugriffsrechte in der Datenbank werden, wie erwähnt, in den Systemkatalog-Tabellen abgelegt. Neben der Information CREATOR, also über den Anleger bzw. Eigentümer eines Objekts, sind in den „Authority“-Tabellen (z.B. SYSCAT.TABAUTH für die Tabellenautorisierung oder SYSCAT.TBSPACEAUTH für die Tabellenbereichsautorisierung) weitere Sicherheitsinformationen gespeichert.

Wie kommt ein Benutzer eigentlich zu einer Zugriffsberechtigung?

Ein Benutzer kann Zugriffsrechte in der DB2 UDB haben, weil er:

- ▶ der Eigentümer dieses Objekts ist.

Der Eigentümer, also der Nutzer, der das Objekt angelegt hat, hat alle Rechte auf dieses Objekt. Dies wird als so genanntes CONTROL-Recht bezeichnet. Er darf also alles tun – auch dieses Objekt wieder löschen.

- ▶ das Recht explizit als individuelles Recht zugewiesen bekommen hat.

Ein berechtigter Benutzer hat explizit ein Zugriffsrecht (z.B. das Leserecht auf eine Tabelle) an diesen Benutzer oder an seine Gruppe vergeben. Die zugehörigen Befehle finden Sie in Abschnitt 10.2.3 „Erteilen und Widerrufen von Zugriffsrechten“, nachdem wir die einzelnen Rechte und Ebenen erläutert haben.

- ▶ das Recht implizit durch die Zuordnung anderer Rechte bekommen hat.

Bestimmte Rechte werden (z.B. beim Anlegen einer Datenbank) automatisch (an die Gruppe PUBLIC) vergeben.

Die folgenden Abschnitte behandeln die jeweiligen Rechte auf den unterschiedlichen Ebenen. Sie werden sehen, dass damit eine feine Abstufung der Berechtigung für einzelne Benutzer möglich ist und Sie auf diese Weise ganz genau steuern können, wer was in der Datenbank tun darf.

Hinweis:

Um eine sichere Umgebung für eine Produktionsdatenbank aufzubauen, empfiehlt es sich, die Datenbank durch den Entzug aller Standardrechte und impliziten Rechte abzuschotten. Schritt für Schritt können dann die Rechte explizit an die Benutzer vergeben werden, die diese benötigen und – laut Ihren Sicherheitsbestimmungen – haben dürfen.

10.2.1 Zugriffsrechte für Datenbanken

Bestimmte Rechte beziehen sich direkt auf die Datenbank und nicht auf eines ihrer Objekte. Dazu gehört das CONNECT-Recht, das einem authentifizierten Benutzer erlaubt, sich mit der Datenbank zu verbinden. Falls er keine weiteren Zugriffsrechte besitzt, kann er selbst mit diesem Recht nichts in der Datenbank unternehmen – Sie sehen also: Für die Sicherheit ist gesorgt.

Die Datenbank-Rechte im Überblick:

► CONNECT

Dieses Recht erlaubt es, per CONNECT eine Verbindung zur Datenbank aufzubauen. Das ist die Voraussetzung dafür, überhaupt mit einer Datenbank arbeiten zu können.

► CREATETAB

Berechtigt dazu, neue Tabellen in der Datenbank anzulegen.

► LOAD

Ermöglicht den Einsatz des LOAD-Tools, um damit Daten in eine Tabelle zu laden.

► BINDADD

Ist das Recht, neue Pakete in der Datenbank anzulegen, um so neue (statische) Datenbankprogramme mit dieser Datenbank ausführbar zu machen.

► IMPLICIT_SCHEMA

Durch dieses Recht kann der Benutzer bei der Objekterstellung ein bis dahin nicht vorhandenes Schema angeben, das dann „implizit“ erstellt wird. Dabei wird SYSIBM zum Schema-Eigner, und das Recht, Objekte in diesem Schema zu erstellen, wird an PUBLIC vergeben.

► QUIESCE_CONNECT

Ermöglicht den Zugriff auf die Datenbank, während sich diese im Warte- bzw. Wartungsmodus befindet.

► CREATE_NOT_FENCED

Durch dieses Recht kann ein Benutzer eine nicht abgeschirmte Prozedur oder UDF in der Datenbank registrieren. Dies kann, falls diese Prozedur nicht ausreichend getestet wurde und noch fehlerhaft ist, zu schwerwiegenden Fehlern führen, da sie im DB2-Adressraum ausgeführt wird.

► CREATE_EXTERNAL_ROUTINE

Eine externe, also eine selbst programmierte Prozedur kann in DB2 eingebunden und allgemein zur Verfügung gestellt werden, wenn man dieses Recht besitzt.

Bei der Erstellung der Datenbank sind auf dieser Ebene verschiedene Rechte an die Gruppe PUBLIC vergeben worden. Dazu gehören:

► CONNECT

► CREATETAB

► BINDADD

► IMPLICIT_SCHEMA

Diese implizit vergebenen Rechte müssen der Gruppe PUBLIC explizit entzogen werden. Besonders für eine Produktionsumgebung stellen diese Rechte häufig eine Verletzung der betrieblichen Sicherheitsrichtlinien dar. Diese vergebenen Standardrechte sollten also genau überprüft werden. Häufig werden diese automatisch vergebenen

Rechte vergessen. So können dann alle Nutzer irgendwelche „Test“-Tabellen in der Produktionsumgebung anlegen und so eventuell den reibungsfreien Ablauf stören.

10.2.2 Zugriffsrechte für Datenbankobjekte

Außer für Schemata und Tabellenbereiche existiert für alle anderen Datenbankobjekte das CONTROL-Recht (siehe Abbildung 10.1), also das Recht, das jeweilige Objekt vollständig zu „kontrollieren“. Dies ist gleichbedeutend mit den Rechten, die der Ersteller bzw. Eigner hat, allerdings kann dieses Recht explizit an bestimmte Benutzer oder Benutzergruppen weitergegeben werden, was für die „Eigner“-Eigenschaft nicht zutrifft.

Hinweis:

Diese hier aufgeführten Rechte können für die einzelnen Objekte vergeben werden und sind nicht als Recht auf die Objekt-Gruppe zu verstehen. Das heißt, ein CONTROL-Recht wird für die Tabelle DB2ADMIN.MITARBEITER vergeben und nicht generell für alle Tabellen einer Datenbank.

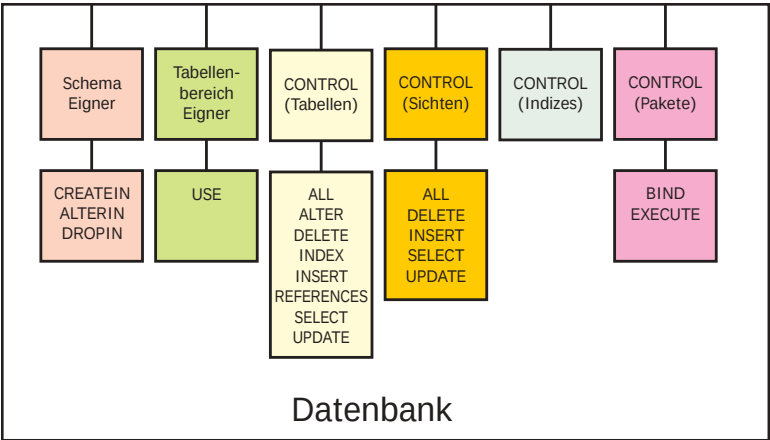


Abbildung 10.1: Objektrechte in der Datenbank

Abbildung 10.1 zeigt neben den CONTROL-Rechten auch die untergeordneten Einzelrechte, die für die jeweiligen Objekte vergeben werden können.

Schema-Rechte

Über CREATEIN, ALTERIN und DROPIN kann definiert werden, ob jemand ein Objekt in diesem Schema anlegen, verändern oder löschen darf.

Tabellenbereich-Recht

Nur die Nutzung kann bei Tabellenbereichen über das USE-Recht gesteuert werden. D.h., nur berechtigte Benutzer dürfen Objekte in einem Tabellenbereich ablegen, also speichern.

Tabellen-Rechte und Sichten-Rechte

ALL als Recht bedeutet, dass ein Benutzer alle untergeordneten Rechte auf dieses Objekt besitzt, bis auf das CONTROL-Recht. Im Gegensatz zum CONTROL-Recht berechtigt das ALL-Recht nicht dazu, das Objekt zu löschen (DROP) oder Tools (wie z.B. REORG und RUNSTATS) auszuführen.

Mit dem ALTER-Recht können Sie die Tabelle ändern und z.B. eine neue Spalte anhängen. Auf Sichten ist dieses Recht nicht verfügbar, da diese nicht geändert werden können.

INSERT, SELECT, UPDATE und DELETE sind die benötigten Rechte, um mit einem Objekt-Inhalt arbeiten zu können.

Weitere Tabellenrechte sind das INDEX- und das REFERENCES-Recht. Ersteres wird benötigt, um Indizes erstellen zu dürfen. Dabei bekommt der Nutzer dann das CONTROL-Recht auf die von ihm erstellten Indizes. Das REFERENCES-Recht berechtigt dazu, Fremdschlüssel in einer Tabelle zu löschen.

Index-Recht

Auf Indizes existiert nur das CONTROL-Recht, das der Ersteller der Indizes automatisch bekommt. Es beinhaltet die Möglichkeit, den Index sowohl zu ändern als auch zu löschen. Eine feinere Aufteilung in weitere Rechte ist hier sinnlos, da diese Verwaltungsaufgaben nie auf mehrere Personen aufgeteilt werden (was nicht heißt, dass nicht mehrere Personen dieses Recht haben können).

Paket-Rechte

Die Paket-Rechte beziehen sich auf Datenbankprogramme. Das EXECUTE-Recht gibt dem Benutzer die Möglichkeit, ein Programmpaket auszuführen. Details zu Rechten für Datenbankprogramme finden Sie in Kapitel 11. Das BIND-Recht dagegen wird benötigt, um ein bestehendes Programm neu zu binden, also den Zugriffsweg vom Optimizer neu bestimmen zu lassen. Es stellt im Gegensatz zum BINDADD-Recht, das unter den Datenbank-Rechten angeordnet ist, ein etwas „schwächeres“ Recht dar, da mit dem BIND-Recht keine neuen Programme für die Datenbank eingebunden bzw. aktiviert werden können.

10.2.3 Erteilen und Widerrufen von Zugriffsrechten

Um die gezeigten Rechte zu vergeben, benötigen Sie das entsprechende Vergabe-Recht (WITH GRANT OPTION).

Folgende Benutzer haben das Recht, eigene Berechtigungen weiterzugeben bzw. zu widerrufen:

- ▶ Mitglieder der SYSADM-Gruppe (DB2-Systemadministratoren)
- ▶ Benutzer mit dem CONTROL-Recht
- ▶ Benutzer mit dem expliziten Recht, Berechtigungen weitergeben zu dürfen (WITH GRANT OPTION)

Bei der Rechte-Vergabe können Sie zwischen der grafischen Variante und der Befehlsvariante wählen.

Beginnen wir mit dem GUI – genauer gesagt, mit der Steuerzentrale.

Rechteverwaltung mit der Steuerzentrale

Das GUI spielt hier wieder einmal seine Vorteile in Bezug auf die Übersichtlichkeit aus. Obwohl die eigentliche Rechte-Vergabe bzw. auch ein Rechte-Entzug über beide Varianten einfach zu handhaben ist, so ist doch die Übersicht über die aktuell gesetzten Rechte im GUI viel anschaulicher als durch eine Abfrage der Systemkatalog-Sichten.

Für die Rechte-Verwaltung navigieren Sie im Objektbaum der Steuerzentrale zum Eintrag **BENUTZER- UND GRUPPENOBJEKTE**, unter dem Sie dann einen Eintrag für **DATENBANKBENUTZER** und **DATENBANKGRUPPEN** finden (siehe Abbildung 10.2).

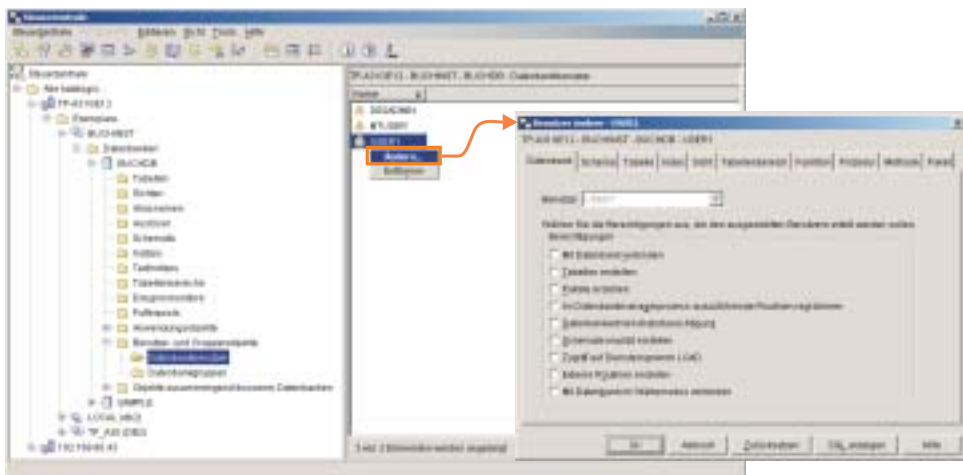


Abbildung 10.2: Berechtigungen über die Steuerzentrale ändern

Im Kontextmenü auf dieser Ebene gibt es die Option **HINZUFÜGEN...**. Mit ihr werden einem *bestehenden* Benutzer des Betriebssystems explizit Rechte in der Datenbank zugeordnet. Neue Betriebssystem-Benutzer können so nicht angelegt werden.

Um zur Anzeige der Rechte zu kommen, wählen Sie **ÄNDERN...** aus dem Kontextmenü eines bestimmten **DATENBANKBENUTZERS** oder einer **DATENBANKGRUPPE** aus, die Sie im Detail-Fenster der Steuerzentrale selektieren (siehe Abbildung 10.2). Das **BENUTZER ÄNDERN**-Fenster ist zur Verwaltung der Benutzerrechte gedacht, bietet aber auch eine gute Übersichtsmöglichkeit, wie auch Abbildung 10.3 zeigt.

Abbildung 10.2 mutet bei genauer Betrachtung vielleicht etwas merkwürdig an, da der Benutzer **USER1** nicht einmal das **CONNECT**-Recht für diese Datenbank besitzt, das hier über die Checkbox **MIT DATENBANK VERBINDEN** definiert werden kann.

Wieso wird hier also ein Benutzer aufgeführt, der gar keine Verbindungsrechte zur Datenbank hat?

Obwohl die Abbildung diesen Eindruck vermittelt, ist der Benutzer USER1 sehr wohl in der Lage, eine Datenbankverbindung aufzubauen. Und – nein – ein Fehler liegt dabei auch nicht vor!

Das liegt an seinen Gruppenrechten bzw. in diesem Fall an dem Recht der DB2-Gruppe PUBLIC, die, wie in Abschnitt 10.2.1 „Zugriffsrechte für Datenbanken“ erwähnt wurde, standardmäßig das Recht hat, Datenbankverbindungen aufzubauen. Ein zusätzliches individuelles CONNECT-Recht ist so nicht mehr nötig. Der Benutzer darf alles, was der Gruppe PUBLIC und den Gruppen, denen er selbst angehört, erlaubt ist. Zusätzlich dazu hat er noch seine individuellen Benutzer-Rechte. Sie merken: Ein guter Überblick ist hier durchaus hilfreich, um die Datenbanksicherheit nicht aus den Augen zu verlieren.

Die zahlreichen Registerkarten gruppieren die Rechte. Schauen wir uns eine Registerkarte stellvertretend etwas genauer an.

Abbildung 10.3 zeigt die Rechte auf Tabellen-Ebene, die selbst in Tabellenform angezeigt werden. Die ersten zwei Zeilen sind dabei vom Benutzer USER1 erstellte Tabellen, auf die er alle Rechte besitzt. Weil ihm diese Objekte gehören, hat er auch die Möglichkeit, Rechte auf diese Objekte weiterzugeben. Dies wird mit dem (grünen) Doppelhäkchen angezeigt.

Ein einzelnes Häkchen zeigt an, dass er das Recht auf dieses Objekt besitzt, dieses allerdings nicht weitergeben kann. Das rote Verbotssymbol bedeutet, dass dieses Recht nicht auf dieser Ebene, also explizit für diesen Benutzer, vergeben wurde. Wie auch beim CONNECT-Beispiel kann der Benutzer dieses Recht eventuell trotzdem über seine Gruppenzugehörigkeit haben.

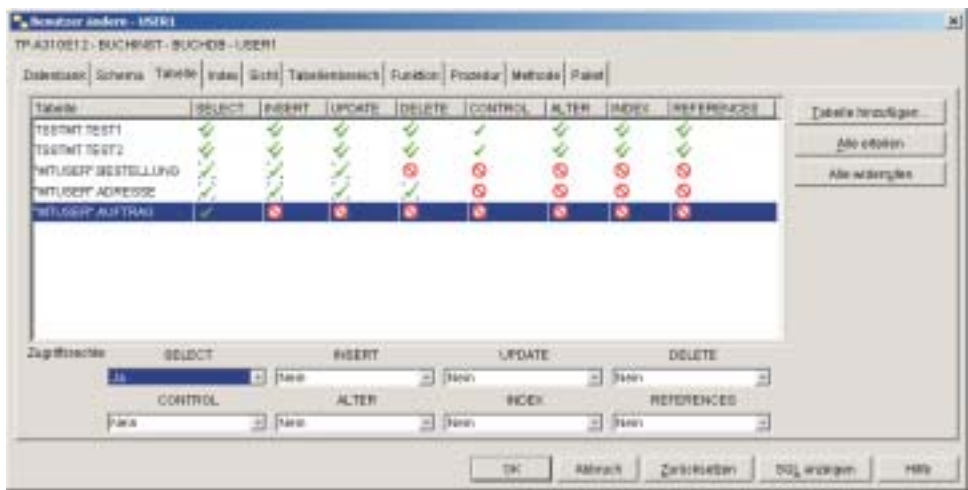


Abbildung 10.3: Tabellenrechte mit der Steuerzentrale verwalten

Die Auswahlfelder am unteren Rand der Registerkarte ermöglichen eine Anpassung der Rechte. Beachten Sie hierbei, dass neben dem selbsterklärenden JA bzw. NEIN der Eintrag ERTEILEN den Doppelhäkchen und so dem eigentlichen englischen

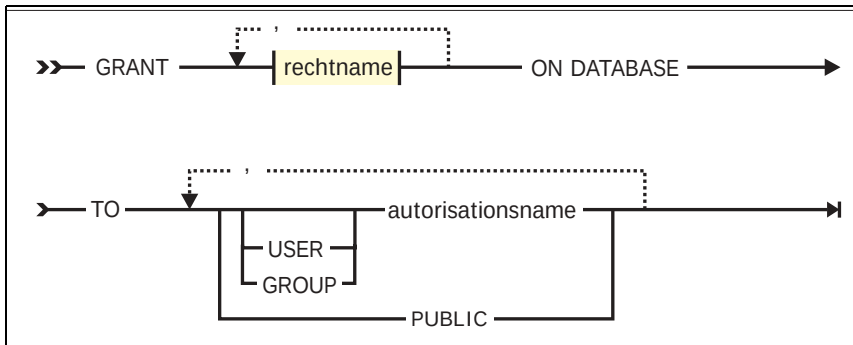


Abbildung 10.5: grant-Syntaxdiagramm für Rechte auf Datenbank-Ebene

Der Name eines einzelnen Rechts muss noch in das Feld `rechtname` eingetragen werden. Dabei sind alle in Abschnitt 10.2.1 „Zugriffsrechte für Datenbanken“ aufgeführten Rechte verwendbar. Die Zuweisung erfolgt dann an einen (Betriebssystem-) Benutzer, eine (Betriebssystem-)Benutzergruppe oder an die DB2-Gruppe `PUBLIC`. Durch die Trennung mit Kommata können auch mehrere Gruppen oder Benutzer angegeben werden.

Tipp:

Obwohl das Schlüsselwort `USER` oder `GROUP` nicht zwangsweise angegeben werden muss, empfiehlt es sich doch, dies zu tun, damit ein Recht nicht versehentlich einer Gruppe zugeordnet wird, wenn man fälschlicherweise glaubt, der `authorisationsname` sei nur ein bestimmter Benutzer.

Hinweis:

Auf der Ebene der Datenbank-Rechte kann das Recht der Rechte-Weitergabe *nicht* vergeben werden!

Gleiches gilt u.a. auch für Index-Rechte und das `CONTROL`-Recht, für die die Option `WITH GRANT OPTION` nicht wirksam ist.

Das Syntaxdiagramm für die anderen Datenbankobjekte ist vom Prinzip her ähnlich, nur etwas komplexer. Das Recht der Weitergabe von Berechtigungen versteckt sich hinter dem optionalen Zusatz `WITH GRANT OPTION`.

Das grant-Syntaxdiagramm für verschiedene Rechte in der Datenbank zeigt ein allgemein gehaltenes Syntaxdiagramm, das es erlaubt, durch die Spezifikation eines zur angegebenen Objektart gehörenden Rechts an Benutzer und Benutzergruppen zu vergeben.

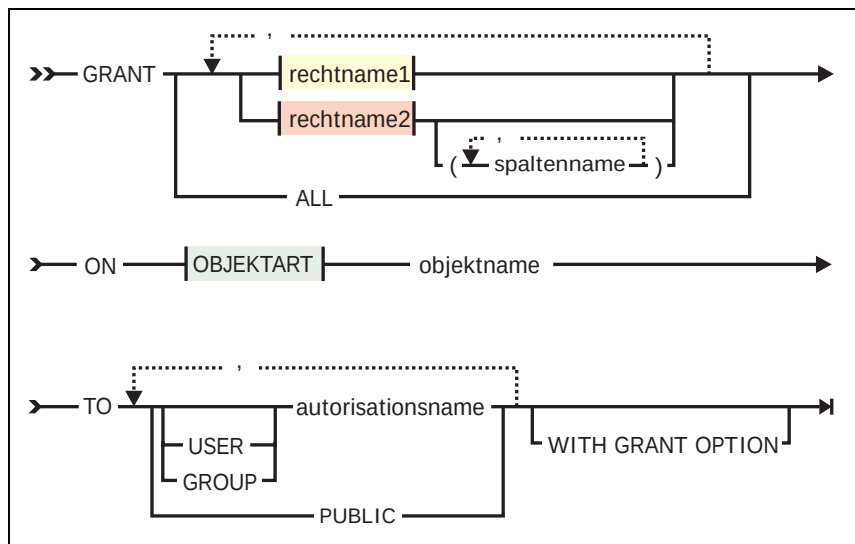


Abbildung 10.6: grant-Syntaxdiagramm für verschiedene Rechte in der Datenbank

Als OBJEKTART kommen in Frage:

► TABLE

Darunter fallen Tabellen und Sichten.

- SCHEMA
- INDEX
- PACKAGE
- TABLESPACE
- SERVER

Dies ist nur für heterogene Umfelder mit der Funktionalität für zusammen-geschlossene Datenbanken interessant (siehe auch Kapitel 6.7.2).

► Routinen, also FUNCTION, METHOD oder PROCEDURE

Der objektname ist dabei der entsprechende Name des Objekts (z.B. der Tabellen-name) – wahlweise mit Angabe der zugehörigen Schemata.

Die für eine OBJEKTART möglichen rechteamen entnehmen Sie bitte der Abbildung 10.1.

Bestimmte Rechte können sogar auf Spaltenebene vergeben werden, was der Ast mit den rechtname2 im Syntaxdiagramm (siehe Abbildung 10.6) zeigen soll. Diese Eigenschaft gibt es nur bei den Tabellen- bzw. Sichten-Rechten. Diese Eigenschaft betrifft das UPDATE- und das REFERENCE-Recht für diese Objektart.

So vergeben Sie beispielsweise das SELECT- und das UPDATE-Recht auf die PLZ- und ORT-Spalte der Tabelle Adresse an die Benutzer-ID sonja:

```
grant select, update (plz, ort) on table mtuser.adresse to user sonja
```

Hinweis:

Es ist *nicht* möglich, das UPDATE-Recht auf einzelne Spalten über die Steuerzentrale zu vergeben!

Soll ein Recht einem Benutzer oder einer Benutzergruppe entzogen werden, steht der REVOKE-Befehl zur Verfügung.

Da mit dem REVOKE-Befehl das genaue Gegenteil eines GRANT ausgelöst wird, verwundert es nicht, dass auch die Syntax bis auf das Schlüsselwort fast identisch ist. Im Gegensatz zu GRANT ... TO ... werden Rechte über REVOKE ... FROM ... entzogen. Entsprechend sieht das Syntaxdiagramm in Abbildung 10.7 aus.

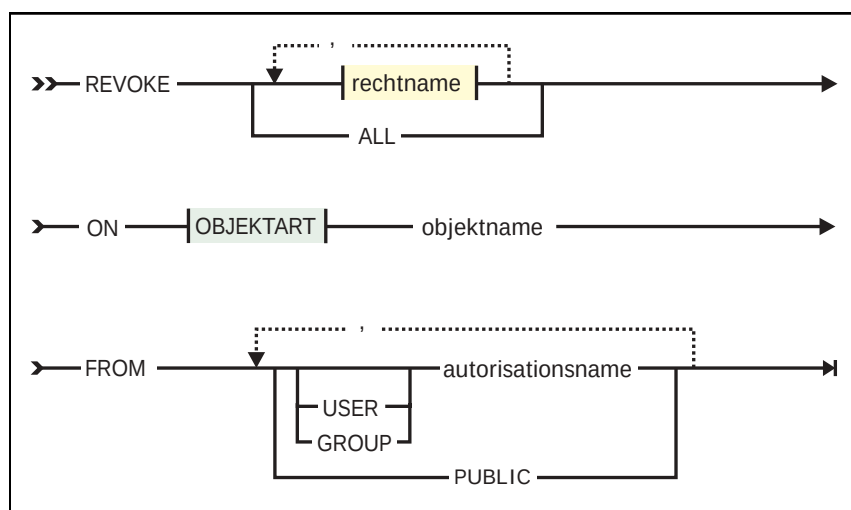


Abbildung 10.7: revoke-Syntaxdiagramm

Beispielsweise kann so der Gruppe sales das Recht entzogen werden, die Inhalte der Tabelle Auftrag zu sehen bzw. zu selektieren:

```
revoke select on table mtuser.auftrag from group sales
```

Hinweis:

Entziehen Sie einem Benutzer das Recht der Rechte-Weitergabe, hat dies keine Auswirkungen auf Benutzer und Benutzergruppen, denen dieser Benutzer das Recht weitergereicht hat. Es ist also ratsam, sehr vorsichtig mit diesem doch sehr mächtigen Recht umzugehen.

Eine Anzeige der aktuell in einer Datenbank vergebenen Rechte können Sie nur mit SQL-Abfragen der Systemkatalog-Sichten erreichen. Das ist einerseits zwar eine

sehr flexible Möglichkeit, aber andererseits doch viel aufwändiger und unübersichtlicher als über die Steuerzentrale.

Die für die Sicherheit relevanten Systemkatalog-Sichten sind:

- ▶ SYSCAT.COLAUTH – Hier sind alle Informationen über Spalten-Rechte (UPDATE und REFERENCE) enthalten.
- ▶ SYSCAT.DBAUTH – Die Rechte der Datenbank-Ebene sind hier verzeichnet.
- ▶ SYSCAT.INDEXAUTH – Eingetragene Benutzer haben das CONTROL-Recht auf einem Index.
- ▶ SYSCAT.LIBRARYAUTH – wird in der aktuellen DB2-Version nicht benutzt.
- ▶ SYSCAT.PACKAGEAUTH – Paket-Rechte, also Rechte zum Bereitstellen und Ausführen von Programmen.
- ▶ SYSCAT.PASSTHROUGH – PASSTHROUGH ist eine Funktionalität, um herstellerspezifische Funktionen an zusammengeschlossene Server weiterzureichen.
- ▶ SYSCAT.ROUTINEAUTH – Zusammenstellung der Gruppen und Benutzer mit EXECUTE-Recht auf eine Routine, also auf eine FUNCTION, METHOD oder PROCEDURE.
- ▶ SYSCAT.SEQUENCEAUTH – Hier werden die Rechte, Sequenzen zu verändern (ALTER), gespeichert.
- ▶ SYSCAT.SCHEMAAUTH – Schema-Rechte werden hier gelistet.
- ▶ SYSCAT.TABAUTH – Speicherung der Tabellen-Rechte.
- ▶ SYSCAT.TBSPACEAUTH – Nur das USE-Recht hat auf Tabellenbereichsebene Bedeutung.
- ▶ SYSCAT.XMLOBJECTAUTH – Rechte auf XML-Objekte des integrierten XML-Extenders.

So können Sie sich z.B. über folgenden Befehl Ihre Tabellen-Rechte anzeigen lassen:

```
select * from syscat.tabauth where grantee in (USER, 'PUBLIC')
```

Dabei ist unter grantee immer der berechtigte Benutzer oder die Benutzergruppe zu verstehen. USER ist eine Variable für den aktuell angemeldeten Benutzer.

In DB2 können Rechte-Pakete an Benutzer oder an Gruppen vergeben werden, um diese für spezielle Verwaltungsaufgaben mit den nötigen „Grund“-Rechten auszustatten. Dazu gibt es verschiedene Berechtigungsstufen.

10.3 Berechtigungsstufen in DB2

Eine Berechtigungsstufe umfasst mehrere Rechte, die sozusagen zu einem Paket zusammengeschlüsselt worden sind. Damit soll es ermöglicht werden, bestimmte Verwaltungsaufgaben bestimmten Personen oder Personengruppen zu übertragen.

Dies ist einerseits für Aufgaben innerhalb von Datenbanken, wie z.B. Reorganisationen etc., und andererseits datenbankübergreifend möglich. Im zweiten Fall können

Sie es so einrichten, dass z.B. ein Benutzer oder eine Gruppe zwar Datenbanksicherungen und -wiederherstellungen durchführen, aber nicht den Datenbankinhalt (die Daten) sehen darf.

Die höchste Berechtigungsstufe in DB2 UDB ist SYSADM (Systemadministrator), die DB2-Systemverwaltungsberechtigung. Diese Stufe wird in Unix-Umgebungen bei der Erstellung des Exemplars der primären Gruppe des Exemplar-Eigners übertragen. In Windows-Umgebungen liegt dieses Recht standardmäßig bei dem Benutzer, der bei der Installation angegeben wurde.

Um es diesem SYSADM zu ermöglichen, auch mal in Urlaub oder zu einer Schulung etc. zu fahren, braucht er einen – vielleicht zwei – Stellvertreter. Dies lässt sich durch die explizite Zusammenfassung dieser Personen in einer Betriebssystemgruppe erreichen. Unter Unix benutzen Sie dazu einfach die Gruppe des Exemplar-Eigners. Unter Windows legen Sie dafür eine neue Betriebssystem-Gruppe an, die Sie dann als SYSADM-Gruppe im DB2 UDB konfigurieren können.

Der Begriff „konfigurieren“ wurde hier absichtlich gewählt, da ja datenbankübergreifende Verwaltungsrechte, wie z.B. das Recht, eine Datenbank anzulegen, nicht *im* Systemkatalog einer Datenbank gespeichert sein können!

Berechtigungsstufen werden also – soweit sie datenbankübergreifende Rechte besitzen – auf der Exemplar-Ebene konfiguriert.

Bevor wir uns diesen Konfigurationsschritt anschauen, wollen wir uns erst einen Überblick über die möglichen Berechtigungsstufen verschaffen. Abbildung 10.8 zeigt alle in DB2 UDB V8 verfügbaren Berechtigungsstufen. Die umfassendste, die SYSADM-Berechtigung, steht ganz oben.

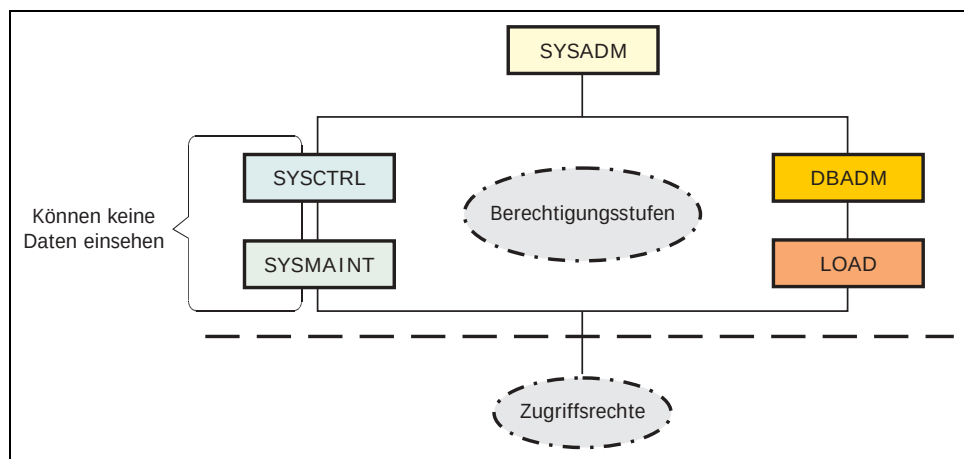


Abbildung 10.8: Berechtigungsstufen in der DB2 UDB

Die folgenden Berechtigungsstufen sind datenbankübergreifend und werden so in der Exemplar-Konfiguration festgelegt:

- ▶ **SYSADM – Systemverwaltungsberechtigung**
umfassendste Berechtigungsstufe im Exemplar, ohne jegliche Einschränkung.
- ▶ **SYSCTRL – Systemsteuerungsberechtigung**
Ein Mitglied dieser Gruppe kann Pflege- und Wartungsaufgaben im Exemplar durchführen, darf neue Datenbanken im Exemplar erstellen und Datenbanken um neue Tabellenbereiche erweitern. Legt ein Mitglied dieser Gruppe eine Datenbank an, wird es automatisch Eigner dieser Datenbank und somit auch DBADM. In seiner eigenen Datenbank kann der Benutzer dann natürlich die Daten einsehen.
- ▶ **SYSMAINT – Systempflegeberechtigung**
Diese Gruppe hat eine geringere Berechtigung wie SYSCTRL und kann auch Pflege- und Wartungsarbeiten durchführen. Dabei dürfen Mitglieder keine neuen Tabellenbereiche oder Datenbanken anlegen.

Die folgenden Berechtigungsstufen sind datenbankbezogen und werden so wie die einzelnen Zugriffsrechte im Systemkatalog gespeichert. Diese Berechtigungsstufen werden also mit GRANT vergeben und nicht konfiguriert.

- ▶ **DBADM – Datenbankadministratorberechtigung**
Diese Benutzer oder Benutzergruppen haben Berechtigungen, die auf eine einzelne Datenbank beschränkt sind, allerdings dürfen sie Daten bzw. Tabelleninhalte sehen.
Wird diese Berechtigung einem Benutzer durch den Befehl

```
grant dbadm on database to user iguser
```

übertragen, werden dem Nutzer *implizit* auch *alle* Rechte der Datenbank-Ebene, also z.B. BINDADD, CONNECT, CREATETAB und IMPLICIT_SCHEMA, zugewiesen.

Wichtig:

```
revoke dbadm from user iguser
```

entzieht *nur* die DBADM-Berechtigung auf die Datenbank, alle anderen implizit vergebenen Rechte bleiben bestehen und müssen *explizit* entzogen werden!

- ▶ **LOAD – Datenbankladeberechtigung**
Die LOAD-Berechtigung stellt auch eine Gruppierung von Rechten dar, die neben dem Recht, das Lade-Tool zu nutzen, mit vergeben werden. Deshalb ist diese Datenbankladeberechtigung hier bei den Berechtigungsstufen mit aufgeführt.

Die einzelnen Rechte dieser Berechtigungsstufen finden Sie in Abbildung 10.9 noch einmal in einer Übersicht zusammengestellt.

Funktion	SYSADM	SYSCTRL	SYSMAINT	DBADM	LOAD
Datenbank migrieren	JA				
Update DBM CFG	JA				
DBADM vergeben / entziehen	JA				
Update der db/node/dcs-Verzeichnisse	JA	JA			
Benutzer zwingen, eine Verbindung zu beenden	JA	JA			
Datenbank erzeugen/löschen	JA	JA			
Tabellenbereich erzeugen/löschen/ändern	JA	JA			
Wiederherstellung in eine neue Datenbank	JA	JA			
Update der DB-Konfiguration (DB CFG)	JA	JA	JA		
Sicherung von DB und Tabellenbereichen	JA	JA	JA		
Wiederherstellung in eine existierende Datenbank	JA	JA	JA		
Ausführen einer aktualisierenden Wiederherstellung	JA	JA	JA		
Starten/Stoppen eines DB2-Exemplars	JA	JA	JA		
Tabellenbereich wiederherstellen	JA	JA	JA		
TRACE ausführen	JA	JA	JA		
Momentaufnahme von DB/DBM machen	JA	JA	JA		
Zustand der Tabellenbereiche abfragen	JA	JA	JA	JA	JA
Update der Protokollierungsinformationsdatei	JA	JA	JA	JA	
Tabellenbereich stilllegen	JA	JA	JA	JA	JA
Reorganisieren von Tabellen	JA	JA	JA	JA	
Ausführen von RUNSTATS	JA	JA	JA	JA	JA
Protokolldateien lesen	JA			JA	
Ereignis-Monitore erstellen/löschen/aktivieren	JA			JA	

Abbildung 10.9: Einzelberechtigungen der Berechtigungsstufen

Bei den Berechtigungsstufen sollten Sie nicht vergessen, dass die Benutzer immer neben ihren individuellen Rechten die Rechte ihrer Gruppe und der Gruppe PUBLIC besitzen. Wenn Sie also wirklich einem Nutzer der SYSMAINT-Gruppe den Zugriff auf Dateninhalte verwehren wollen, müssen Sie z.B. sicherstellen, dass auch die Gruppe PUBLIC diese Rechte nicht besitzt.

Um datenbankübergreifende Berechtigungen zu vergeben bzw. die Berechtigungsgruppen in DB2 zu definieren, müssen Sie die Exemplar-Konfiguration anpassen.

Drei Parameter stehen zur Verfügung, um den DB2-Berechtigungsgruppen SYSADM, SYSCTRL und SYSMAINT Gruppen des Betriebssystems zuzuweisen:

- SYSADM_GROUP
- SYSCTRL_GROUP
- SYSMAINT_GROUP

Die Zuweisung der SYSADM-Berechtigung an die Gruppe db2adm erfolgt dann durch den Befehl:

```
update dbm cfg using sysadm_group db2adm
```

Sie sollten sehr darauf achten, dass nur zwei oder drei Kollegen dieser Betriebssystemgruppe angehören, um diese umfassenden Rechte im Auge zu behalten. Sonst wissen Sie irgendwann nicht mehr, wer für was zuständig war und was „verbrochen“ hat.

11 Programmierschnittstellen

Dieses Kapitel soll Ihnen, dem Administrator, einen Überblick über die Programmierung verschaffen. Sie erfahren, welche verschiedenen Programmiermöglichkeiten bestehen und welche Auswirkungen diese auf die Administration bzw. den Datenbankbetrieb haben.

Da wir in diesem Kapitel nicht in die Tiefe der Datenbankprogrammierung einsteigen, sondern uns vielmehr auf Konzepte und Funktionsweisen konzentrieren, werden keine besonderen Programmierkenntnisse vorausgesetzt.

11.1 Aufgabengebiete der Programmierung

Man unterscheidet in der Datenbankprogrammierung zwei Hauptaufgabengebiete:

- ▶ Programme zu erstellen, die sich mit der Abfrage und Änderung des Datenbankinhalts beschäftigen und somit SQL-Anweisungen enthalten
- ▶ Programme zu erstellen, die sich mit der Pflege und Wartung des Datenbanksystems und der Datenbank beschäftigen

Diese Trennung sollte auch in der Programmierung, wie in Abbildung 11.1 dargestellt, strikt eingehalten werden. Dies empfiehlt sich, da bei der reinen Verwendung von SQL-Befehlen im Prinzip keine datenbanksystemspezifischen Befehle vom Programm abgesetzt werden. Das setzt natürlich voraus, dass die verwendeten SQL-Kommandos dem SQL3-Standard entsprechen. Somit kann man zumindest in der Theorie und unter gegebenen Umständen auch in der Praxis davon ausgehen, dass diese Programmteile nahezu unabhängig vom Datenbanksystem sind. Dies hat zur Folge, dass sich für diese Programmteile der Aufwand für eine Portierung auf ein anderes DBMS in Grenzen hält. Hingegen werden die Programme, die sich mit der Pflege und Wartung des DBMS beschäftigen, immer datenbanksystemabhängig sein.

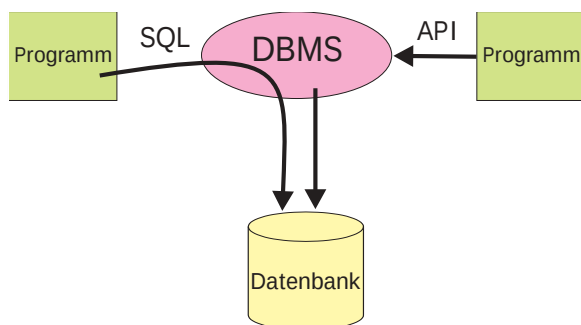


Abbildung 11.1: Unterschiedliche Hauptaufgabengebiete der Programmierung

11.1.1 Programme mit SQL-Anweisungen

Welche Befehle können Programme mit reinen SQL-Anweisungen enthalten? Es sind genau die Befehle, die im SQL-Standard definiert sind. Dazu gehören:

- ▶ DDL-Befehle (Data Definition Language) wie z.B. CREATE TABLE
- ▶ DML-Befehle (Data Manipulation Language) wie z.B. INSERT, UPDATE und DELETE
- ▶ DCL-Befehle (Data Control Language) wie z.B. GRANT und REVOKE

Diese Befehlstypen werden über die SQL-Schnittstelle des DBMS abgehandelt. Welche verschiedenen Schnittstellen die DB2 UDB hierfür zur Verfügung stellt, werden wir in Abschnitt 11.6 „Die weiteren Nicht-Embedded-SQL-Schnittstellen“ genauer betrachten.

Die Praxis zeigt uns, dass selbst bei den Befehlen, die eigentlich über den SQL-Standard genormt sind, teilweise starke Abweichungen auf den unterschiedlichen DBMS auftreten. Dies kommt speziell bei den DDL-Befehlen vor, da die unterschiedlichen DBMS zwar nahezu dieselben Datentypen unterstützen, sich hier jedoch stark in ihrer Syntax unterscheiden.

In den verschiedensten Projekten zur Umschulung und zur Portierung unterschiedlicher DBMS-Systeme nach DB2 UDB haben wir die Erfahrung gemacht, dass sich die Portierung von Sybase, Informix und SQL-Server nach DB2 UDB in dieser Hinsicht einfacher gestaltet als die Portierung von Oracle. Dies liegt daran, dass sich eine Portierung von Oracle auf andere DBMS durch die starken Abweichungen vom SQL-Standard in den DML-Befehlen als weitaus aufwändiger erweist.

Um eine größtmögliche Flexibilität hinsichtlich der verwendeten DBMS zu erhalten, sollten Sie auch in den Programmteilen, die reine SQL-Anweisungen enthalten, eine Trennung nach den Rubriken DML-, DDL- und DCL-Befehlen durchführen.

Wie Sie jedoch später noch sehen werden, ist für den Aufwand, der bei einer Datenbankportierung entsteht, die verwendete Programmier Technik weitaus ausschlaggebender.

11.1.2 Programme zur Wartung der Datenbank und des Datenbanksystems

Die Befehle zur Wartung und Pflege der Datenbank und des Datenbanksystems sind im Gegensatz zu den SQL-Befehlen nicht durch einen weltweiten Standard genormt. Somit sind diese Befehle von DBMS zu DBMS unterschiedlich.

Zu diesen Befehlen gehören die typischen Administrationsbefehle wie:

- ▶ Sichern und Wiederherstellen einer Datenbank
- ▶ Anlegen einer Datenbank
- ▶ Starten und Stoppen eines Exemplars
- ▶ Katalogisieren von Datenbanken
- ▶ Änderung der Exemplar-Konfiguration

Diese Befehle werden über die API-Schnittstelle (Application Programming Interface) des DBMS unterstützt.

Zwar wird Ihnen die eine oder andere API-Funktion inzwischen über Java-Aufrufe angeboten, jedoch sollten Sie sämtliche API-Schnittstellenfunktionen der DB2 UDB am besten über C-Bibliotheken verwenden einbinden. Diese lassen sich jederzeit in andere Programmiersprachen einbinden. Bitte beachten Sie dabei die entsprechenden Compiler-Optionen bzw. die Bedingungen für die Parameterübergabe an die C-Funktionen.

11.2 Programmiermethoden für SQL-Anweisungen

Zur Programmierung der SQL-Anweisungen stehen Ihnen innerhalb von DB2 zwei unterschiedliche Programmiermethoden zur Verfügung:

- Programmierung von statischem SQL
- Programmierung von dynamischem SQL

Statische SQL-Anweisungen sind, wie es der Name schon sagt, Anweisungen, die zum Zeitpunkt der Programmierung feststehen und sich während der Programmaufzeit nicht ändern.

Dynamische SQL-Anweisungen hingegen sind Anweisungen, die zum Zeitpunkt der Programmierung nicht feststehen und erst zur Laufzeit des Programms bekannt sind.

Die verschiedenen Business-Anforderungen des Markts haben dazu geführt, dass in der IT-Branche unterschiedliche Techniken und somit auch Programmiersprachen favorisiert werden. Abbildung 11.2 zeigt die unterschiedlichen Trends und die dabei verwendeten Programmiersprachen.

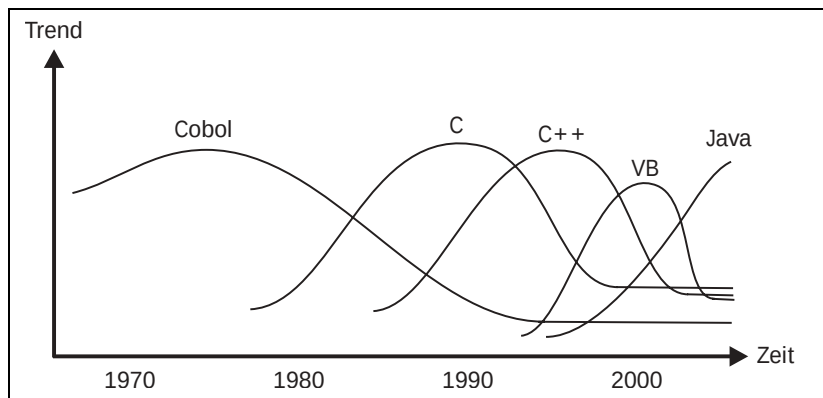


Abbildung 11.2: Trends bei Programmiersprachen

Die unterschiedlichen Anforderungen an die Datenbankschnittstelle, die sich aus der Eigenart der verwendeten Programmiersprache ergeben, führten dazu, dass auch in der Datenbankprogrammierung verschiedene Programmiertechniken entstanden sind. Dabei haben sich zwei grundsätzlich verschiedene Vorgehensweisen entwickelt:

- ▶ Embedded-SQL
- ▶ Nicht-Embedded-SQL

11.2.1 Embedded-SQL

Bei dieser Methode bindet man die Sprache SQL in eine so genannte *Wirts-* oder auch *Hostsprache* ein.

Dabei werden z.B. folgende gängigen Wirtssprachen unterstützt:

- ▶ C
- ▶ COBOL
- ▶ PLI
- ▶ FORTRAN
- ▶ JAVA (teilweise für statisches SQL)

Da die Compiler der Hostsprachen die SQL-Syntax nicht verstehen, muss der „gemischte“ Programmcode, der sowohl SQL als auch Programmcode der Wirtssprache enthält, zuvor bearbeitet werden (siehe Abbildung 11.3). Diese zuvor durchgeführte Bearbeitung nennt man *precompilieren* (vorkompilieren). Der dort verwendete Übersetzer (Compiler) wird Precompiler genannt. Im Java-Umfeld heißt er *SQLJ-Übersetzer* (Translator).

Der Precompiler wiederum übernimmt zwei Aufgaben:

- ▶ Er sucht alle SQL-Anweisungen und setzt sie in entsprechende Sprachelemente der jeweiligen Wirtssprache um. Dabei wird der SQL-Code auskommentiert und durch Programmcode der Wirtssprache ersetzt.
- ▶ Die Informationen der gefundenen SQL-Anweisungen werden in einer separaten Datei gespeichert.

Diese Dateien werden in DB2 üblicherweise als *Bind-Files* bezeichnet. Da der Precompiler von Java nicht vom jeweils verwendeten DBMS abhängig ist und die somit entstehenden Informationen DBMS-unabhängig erstellt werden, nennt man diese Dateien hier SQLJ-Profiles.

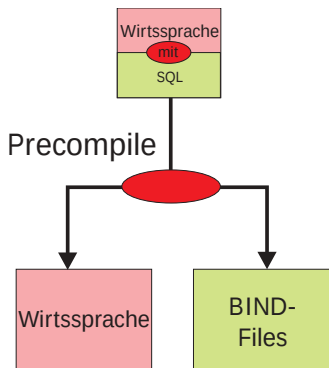


Abbildung 11.3: Embedded-SQL

11.2.2 Nicht-Embedded-SQL

Sie sehen in Abbildung 11.4, dass Ihnen, im Gegensatz zu der Embedded-SQL-Methode, hier je nach verwendeter Programmiersprache verschiedene Programmier- und Datenbankschnittstellen zur Verfügung stehen.

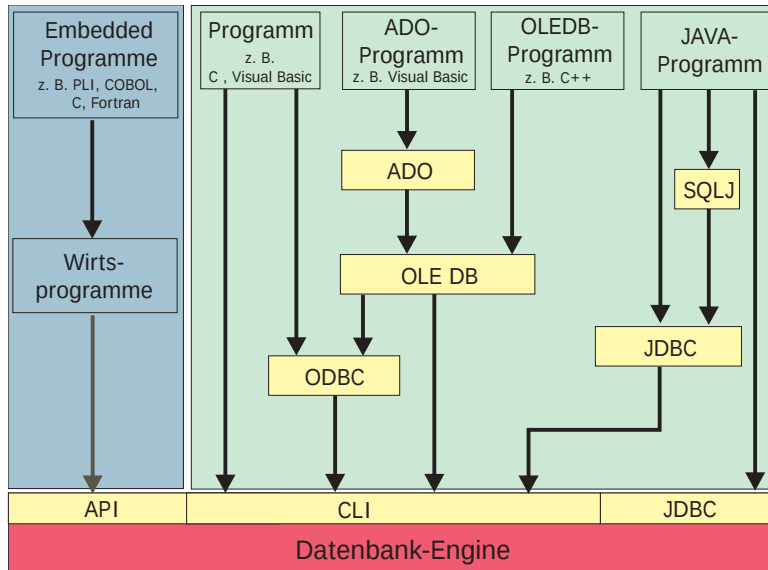


Abbildung 11.4: Die verschiedenen Datenbankschnittstellen

Bevor wir uns näher mit den Fragen beschäftigen, was genau statisches bzw. dynamisches SQL ist, welche Programmier- und Datenbankschnittstelle sich für Sie am besten eignet und welche Auswirkungen dies auf Ihr Datenbanksystem hat, betrachten wir kurz die Eigenheiten der Abfragesprache SQL.

11.3 Eigenheiten der Abfragesprache SQL

Wie schon in Kapitel 1 erwähnt wurde, handelt es sich bei der Abfragesprache SQL (Structured Query Language) um eine einfache, standardisierte und mengenorientierte Sprache zur Beschreibung einer Ergebnismenge.

Im Gegensatz zu einer reinen Programmiersprache, in der man sich auf das *Wie* konzentriert, konzentriert man sich bei SQL auf das *Was*.

Wenn wir uns dazu einmal die Abbildung 11.5 anschauen, stellen wir fest, dass bei einer normalen Programmiersprache der Programmverlauf sofort ersichtlich ist. Hingegen ist bei der etwas komplexeren und dennoch von der reinen Aufgabenstellung her sehr einfachen SQL-Anweisung nicht ersichtlich, wie und in welcher Reihenfolge die Ergebnismenge bestimmt wird. Das bedeutet: Wir wissen anhand des SQL-Befehls nicht, ob die DB2 UDB z.B. eine Zwischenmenge bilden muss, ob der Zugriff über einen Index erfolgt oder ob eine interne Sortierung erfolgt.

Programm

```

BEGIN
  VAR Integer I,A,B,X,Y;
  ...
  X:= A;
  Y:= B;
  IF (X = 1) AND (Y = 2)
  BEGIN
    Z = 3 * 7;
    writeln(" Hallo DB2 UDB V8");
  END
  ELSE
  ...
  W := Funktion();
  ...
END;

```

SQL-Befehl

```

WITH TMitarbeiter(TMitNr,TMitName,TVgNr,TVgName) AS
( SELECT A.MitNr,A.MitName,A.VgNr,B.MitName
  FROM   Mitarbeiter A, Mitarbeiter B
  WHERE  A.MitName = 'Durrer' and
         A.VgNr    = B.MitNr

  UNION ALL
  SELECT A.MitNr, A.MitName,A.VgNr,B.MitName
  FROM   Mitarbeiter A,TMitarbeiter,Mitarbeiter B
  WHERE  A.MitNr  = TVgNr and
         A.VgNr   = B.MitNr
)
select ' Der Vorgesetzte von Herr '|| substr(TMitName,1,8) ||
' mit der Personalnummer ', decimal(TMitNr,1,0),
' ist Herr '|| substr(TVgName,1,8) ||
' mit der Personalnummer' , decimal(TVgNr,1,0)
FROM TMitarbeiter;

```

Abbildung 11.5: Vergleich einer Programmiersprache mit einer SQL-Anweisung

Das „Was“ in das „Wie“ (besser gesagt in einen *Ablaufplan*) umzusetzen, ist eine der Aufgaben des Optimizers im DBMS (siehe Kapitel 15.1, „Der DB2-Optimizer“). Weiterhin muss das DBMS, bevor der Ablaufplan gebildet werden kann und der SQL-Befehl zur Ausführung kommt, z.B. prüfen, ob:

- ▶ die Syntax der SQL-Anweisung (query) korrekt ist
- ▶ die Semantik in Ordnung ist
- ▶ die angesprochenen Objekte wie z.B. Spalten, Tabellen, Funktionen vorhanden sind
- ▶ der Benutzer die Berechtigung auf die angesprochenen Objekte besitzt
- ▶ das SQL-Statement eventuell umgeschrieben werden muss (query-rewrite)
- ▶ bereits ein Zugriffsplan (package) entwickelt wurde

Außerdem muss der Optimizer prüfen, welche Systemvoraussetzungen vorhanden sind, um diese Größen zur Bildung des optimalen Ablaufplans ebenfalls heranzuziehen.

Wie Sie sehen, ist das eine Menge Arbeit, die einen gewissen Zeitaufwand kostet – Zeit, die das DBMS benötigt, bevor der Befehl tatsächlich ausgeführt und dem Benutzer die Ergebnismenge zur Verfügung gestellt werden kann.

Das ist wie im richtigen Leben. Wenn Sie ein Auto speziell nach Ihren Wünschen möchten, muss es erst gebaut werden. Kritisch wird es nur dann, wenn Ihnen erst in dem Moment, wenn Sie eigentlich losfahren möchten, einfällt, dass Sie ein speziell angefertigtes Auto benötigen. Dann können Sie nur hoffen, dass der KFZ-Hersteller sehr schnell produziert bzw. dass so ein Auto schon produziert und an Sie ausgeliefert worden ist. Je komplexer Ihre Wünsche (in unserem Fall die SQL-Anweisung), desto zeitaufwändiger ist die Produktion (Aufwendungen für das DBMS). Wenn Sie es also eilig haben, sollte Ihr Auto schon vor der Tür stehen, und Ihre SQL-Anweisung sollte komplett als auszuführendes Programm („Wie“) vorliegen.

Welche Optimierungsmöglichkeiten Sie bei der Programmierung von Datenbank-anwendungen unter der DB2 UDB besitzen und wie sich hier statisches und dyna-misches SQL auswirkt, ist das Thema der nächsten Abschnitte.

Anmerkung:

Bei dem in Abbildung 11.5 aufgeführten SQL handelt es sich eigentlich um eine ganz einfache Anforderung. Man will aus der Tabelle Mitarbeiter die Hierarchiestufen ab dem Mitarbeiter Durrer auflisten (siehe das Datenmodell BUCHDB im Vorwort, Abbildung V.3).

11.4 Statisches SQL

Im Gegensatz zu allen anderen Datenbankherstellern verfolgte IBM bei der Realisierung von relationalen Datenbanken schon von Anfang an das Konzept des statischen SQL. Das ging sogar so weit, dass in den ersten Versionen von DB2 unter MVS nur statisches SQL unterstützt wurde. Sicherlich besitzen die anderen Datenbanksysteme wie z.B. Oracle, SQL-Server und Informix Ansätze, um *quasi* statisches SQL zu unterstützen. Diese entsprechen jedoch in keiner Weise dem Ansatz, den IBM mit dem Konzept der DB2 UDB verfolgt.

Vor allen Dingen werden in den ganzen Datenbankportierungsprojekten von z.B. Oracle, Sybase, Informix und SQL-Server nach DB2 die Möglichkeiten und Vorteile des statischen SQL nicht einmal in Betracht gezogen. Diese Methode wird oft sogar verschmäht, als veraltet und als DB2-Eigenheit abgetan.

Dass dies nicht so und vielmehr eine Möglichkeit ist, Ihre Datenbankanwendung zu optimieren und dass Ihnen als Administrator ein Werkzeug in die Hand gegeben wird, mit dem Sie in Sachen Performance und Sicherheit neue Wege beschreiten können, werden wir im Folgenden zeigen.

11.4.1 Wann ist eine SQL-Anweisung statisch?

Für die genauere Betrachtung klären wir als Erstes die Frage: Wann ist eine SQL-Anweisung zum Programmierzeitpunkt bekannt bzw. nicht bekannt?

Die Antwort wollen wir Ihnen anhand des folgenden Beispiels verdeutlichen. Wir gehen von folgender SQL-Anweisung (auch Query genannt) aus:

```
select ArtikelNr, ArtikelBez, ArtikelBest, ArtikelPreis from Artikel
      where Artikelbez like 'B%' order by ArtikelBez
```

Dabei teilen wir die SQL-Anweisung wie folgt auf:

- ▶ Die SELECT-Klausel, oft auch als Spaltenliste bezeichnet, enthält die zu lesenden Attribute wie z.B. (ArtikelNr, ArtikelBez, ArtikelBest, ArtikelPreis)
- ▶ Die FROM-Klausel enthält die Namen der Datenquellen wie z.B. Tabellen, Tabellenfunktionen oder SQL-Anweisungen (from Artikel)
- ▶ Die WHERE-Klausel enthält in der where-Bedingung die entsprechenden Prädikate, in unserem Beispiel Artikelbez like 'B%'.
- ▶ Die ORDER BY-Klausel bestimmt die Sortierreihenfolge der Ergebnismenge.

Dabei ist diese Anweisung genau dann *statisch*, wenn:

- ▶ die SELECT-Klausel bekannt ist. Man spricht hierbei von einer SELECT-Klausel mit bekannter Spaltenliste
- ▶ die FROM-Klausel bekannt ist und keine Variablen enthält
- ▶ die WHERE-Klausel von ihren Prädikaten her bekannt ist; die *Ausprägung der Prädikate kann jedoch variabel sein*
- ▶ die ORDER BY-Klausel bekannt ist

Das bedeutet: In unserem Fall kann die SELECT-Anweisung maximal in der Prädikatausbildung variabel sein. Dies sieht in unserem Beispiel wie folgt aus:

```
select ArtikelNr, ArtikelBez, ArtikelBest, Artikelpreis from Artikel
      where Artikelbez like :Wert order by ArtikelBez
```

Bei **:Wert** handelt es sich um eine Programmvariable. Wir sprechen hier auch von einer Host- oder Wirtsvariablen, die erst zur Laufzeit bestimmt werden kann.

Hinweis:

Eine SQL-Anweisung ist genau dann statisch, wenn sie zum Zeitpunkt der Programmierung bis auf die Ausprägung der Prädikate komplett bekannt ist und sich zur Laufzeit nicht mehr verändert.

Da in der Praxis in den allermeisten Fällen die Überschrift für die Auflistung des SQL-Ergebnisses (auf dem Bildschirm oder Papier) bzw. die Attributnamen für die Benutzerbildschirme statisch kodiert sind, kann man davon ausgehen, dass auch 80% der Datenzugriffe einer Anwendung statisch sind.

11.4.2 Wie programmieren Sie statisches SQL?

Um Ihnen einen sehr schnellen, aber dennoch aussagekräftigen Überblick über die Möglichkeiten und Vorgehensweisen der statischen SQL-Programmierung zu geben, besprechen wir dieses Thema anhand zweier Beispiele.

In den Beispielprogrammen verwenden wir die Wirtssprachen C und Java. Da wir uns auf die reine Datenbankprogrammierung konzentrieren und somit nur die SQL-relevanten Programmteile erklären, benötigen Sie hierfür keine speziellen Java- oder C-Programmierkenntnisse.

Anhand des C-Beispiels werden wir sehr ausführlich auf die SQL-Programmierung eingehen und auch programmiersprachenübergreifende Grundlagen vermitteln. Im entsprechenden Java-Beispiel hingegen werden wir nur auf die Unterschiede eingehen.

Hinweis:

Sie finden die Source-Codes der beiden Beispiele und die entsprechenden Batch-Files zur Erstellung ablauffähiger Programme auf der beiliegenden Buch-CD.

Um die auf der CD befindlichen C-Programme mit den zur Verfügung stehenden Batch-Files übersetzen zu können, setzen wir den Microsoft Visual C++ 6.0 Compiler voraus. Sollten Sie andere C-Compiler verwenden, müssen Sie die entsprechenden Batch-Files an den jeweiligen Compiler anpassen.

Für die Erstellung des ablauffähigen Java-Programms wird nur die korrekte DB2 UDB V8-Installation vorausgesetzt. Um jedoch die entsprechende Java-Umgebung der DB2 zu nutzen, müssen Sie die Umgebungsvariable PATH richtig setzen. Dies können Sie auch durch den Aufruf des Batch-Files SETJAVA.BAT erreichen, das sich im selben Verzeichnis wie der Source-Code des Java-Programms BuchDB_S_SQLJ.SQLJ befindet.

11.4.3 Statisches SQL am Beispiel der Wirtssprache C

Wie Sie in Abbildung 11.6 sehen, haben wir das aufgeführte C-Programm in insgesamt elf Bereiche unterteilt. Nachfolgend werden wir Abschnitt für Abschnitt näher erläutern.

Allgemeines

So genannte *Delimiter* sorgen dafür, dass der Precompiler nicht die komplette Syntax der verwendeten Wirtssprache interpretieren muss, um die eigentlichen SQL-Befehle ausfindig zu machen.

- ▶ Um den Beginn einer Embedded-SQL-Anweisung zu kennzeichnen, werden in den verschiedenen Programmiersprachen folgende Delimiter verwendet:
 - EXEC SQL in C und C++
 - EXEC SQL in COBOL
 - EXEC SQL in FORTRAN
 - #sql{ in JAVA
- ▶ Auch die Endekennzeichen der Embedded-SQL-Anweisung unterscheiden sich in den einzelnen Wirtssprachen:
 - ; in C und C++
 - ENDE EXEC in COBOL
 - Kein Delimiter in FORTRAN
 - }; in Java

Beispiel :

```
EXEC SQL SELECT COUNT(*)INTO :hAnzahl FROM Adresse;
```

Abschnitt 1 (Include der SQL-Header-Files)

Damit die speziellen für die SQL-Aufrufe benötigten Programmkomponenten (wie z.B. Funktionen, Typen, Strukturen und Variablen) zur Verfügung stehen, müssen die entsprechenden Definitionen über so genannte *Include-Dateien* geladen werden.

```

/* ----- */
/* Beispiel Programm für statisches SQL
/*
/* (C) COPYRIGHT Lis.Tec GmbH
/* ----- */

#include <stdlib.h>
#include <sql.h>

EXEC SQL include sqlca ;
EXEC SQL include sqlda ;

EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 hadressnr;
    char htitel[30+1];
    char hname[30+1];
    char hvorname[30+1];
    char hgebdat[11+1];
    char hplz[6+1];
    char hort[30+1];
    long hAnzahl;
    short ititel, iname, ivorname, ige bdat, iplz, iort;
EXEC SQL END DECLARE SECTION;

EXEC SQL DECLARE Cursor1 CURSOR WITH HOLD FOR
    SELECT SubStr( titel,1,6), SubStr( name,1,15), SubStr( vorname,1,15), gebdat, plz, substr(ort,1,15)
    FROM ADRESSE
    FOR UPDATE OF titel,ort;

/* Funktion zur Bearbeitung der Indikatorvariable */
/* ----- */
/* Diese Funktion setzt die entsprechende Variable je nach Indikatorvariable auf leer */
/* ----- */
void SetCVar(char *cVar, short iVar)
{ if(iVar < (short)0)
  { cVar[0] = '-';
    cVar[1] = '\0';
  }
}

/* Hauptprogramm */
/* ----- */
/* Liest die Tabelle Adresse */
/* ----- */

void main( void )
{
    EXEC SQL CONNECT TO BUCHDB USER DB2ADMIN USING db2admin;
    if(SQLCODE == 0)
    { /* Wenn Connect erfolgreich */
        EXEC SQL
            SELECT COUNT(*) INTO :hAnzahl
            FROM Adresse;

        printf( "\n Die Tabelle beinhaltet insgesamt %d Zeilen -- FehlerCode : %d \n", hAnzahl, SQLCODE );

        EXEC SQL OPEN Cursor1;
        if(SQLCODE == 0)
        { /*
            12345 123456789012345 123456789012345 12345678901 123456 123456789012345*/
            printf( "\n Titel Vorname Nachname Geb-Datum Plz Ort iort");
            printf( "\n =====");
            while( SQLCODE == 0)
            { EXEC SQL
                { FETCH Cursor1
                    INTO :htitel :itime, :iname, :hvorname :ivorname,
                    :hgebdat :igebdat, :hplz :iplz, :hort :iort;
                    SetCVar(htitel, ititel); SetCVar(hname, iname); SetCVar(hvorname, ivorname);
                    SetCVar(hgebdat, ige bdat); SetCVar(hplz, iplz); SetCVar(hort, iort);

                    if(SQLCODE == 0)
                    { printf( "\n %s %s %s %s %s %d", htitel, hvorname, hname, hgebdat, hplz, hort, iort);
                      /* Ändern aller Titel in der erster Stelle, falls bestimmter Bedingung vorliegt */
                      /* ----- */
                      switch (htitel[1])
                      { case 'r' : if(htitel[0] == 'F') htitel[0] = 'f'; else htitel[0] = 'F'; break;
                        case 'e' : if(htitel[0] == 'H') htitel[0] = 'h'; else htitel[0] = 'H'; break;
                      }
                      if ((hvorname[0] == 'K') || (hname[0] == 'K'))
                      { if (hort[0] == '-')
                          { strcpy(hort, "Flensburg");
                            iort = 0;
                          }
                        else iort = -1;
                      }
                    }
                    EXEC SQL UPDATE Adresse
                    SET titel = :htitel, Ort = :hort :iort
                    WHERE CURRENT OF Cursor1;
                    EXEC SQL COMMIT;
                }
            } /* Ende für Schleife */
        } /* Ende wenn open Cursor erfolgreich war */
        else
            printf("\n Cursor konnte nicht geöffnet werden SQLCODE %ld \n Msg : %s", SQLCODE, sqlca.sqlerrmc);
        /* Ende wenn Connect erfolgreich */
        EXEC SQL CLOSE Cursor1;
        EXEC SQL CONNECT RESET;
    }
}

```

Abbildung 11.6: Statisches SQL-Programm mit der Wirtssprache C

Der Precompiler löst diese beiden Precompiler-Anweisungen in folgende C-Anweisungen auf:

```
/*
EXEC SQL include sqlca ;
*/
/* SQL Communication Area - SQLCA - structures and constants */
#include "sqlca.h"
struct sqlca sqlca;
/*
EXEC SQL include sqlda ;
*/
/* SQL Descriptor Area - Variable descriptor */
#include "sqlda.h"
```

Abschnitt 2 (Variablendeklaration)

Zwischen dem Programm und dem DBMS müssen verschiedene Arten von Daten ausgetauscht werden:

- ▶ Daten, die vom DBMS an das Anwendungsprogramm gesendet werden; hierzu gehören:
 - Ergebnismenge der SQL-Anweisungen
 - Kommunikationsmeldungen wie z.B. Fehlermeldungen und Warnhinweise
- ▶ Daten, die vom Anwendungsprogramm an das DBMS gesendet werden:
 - Variablen, die innerhalb einer SQL-Anweisung zur Ausprägung von Prädikaten genutzt werden (z.B. ...where Artikelbez like :Wert ...)

Um diese Daten austauschen zu können, muss ein gemeinsamer Hauptspeicherbereich (siehe Abbildung 11.18) zwischen dem Anwendungsprogramm und dem DBMS vorhanden sein.

Der Inhalt dieses Speicherbereichs wird durch die Definition und Deklaration von Variablen im Anwendungsprogramm gesteuert. Da dies im Wirtsprogramm bzw. Hostprogramm geschieht, nennt man diese Variablen auch *Host-* oder *Wirtsvariablen*. Der Precompiler sorgt dafür, dass dieser Speicherbereich auch dem DBMS bekannt gemacht wird. Er erzeugt für alle Variablendeklarationen zwischen den Anweisungen BEGIN DECLARE SECTION und END DECALRE SECTION die entsprechenden DBMS-Befehle.

Innerhalb des Wirtsprogramms können diese Host-Variablen mit ihrem normalen Namen verwendet werden. Im Gegensatz dazu muss den Wirtsvariablen innerhalb der SQL-Anweisungen ein Doppelpunkt „:“ vorangestellt werden.

Beispiel aus Abschnitt 6:

```
EXEC SQL SELECT COUNT(*) INTO :hAnzahl FROM Adresse;
```

Als Name für die Wirtsvariablen verwendet man in der Regel den Namen der anzusprechenden Tabellenspalte. Um diesen Namen besser vom eigentlichen Spaltennamen unterscheiden zu können, ist es jedoch sinnvoll, einen Präfix zu verwenden. Wir verwenden in der Praxis das Präfix *h* für die Host-Variable.

Tipp:

Achten Sie darauf, dass bei der Verwendung der Wirtssprache C die Host-Variable für den Datentyp CHAR eine Stelle länger sein muss. Dies ist notwendig, da die Programmiersprache C für CHAR-Typen den Wert „\0“ als Endekennzeichen benötigt.

Abschnitt 3 (Cursor-Deklaration)

Im Deklarationsteil (Abschnitt 1) ist pro selektierter Spalte der SQL-Anweisung eine Host-Variable deklariert. Dies ermöglicht es Ihnen, maximal eine Tabellenzeile zu verarbeiten.

Tipp:

Im DB2 UDB-Umfeld steht Ihnen das Hilfsprogramm `db2dc1gn` zur Verfügung, um die benötigten Wirtsvariablen zu deklarieren. Dieses Hilfsprogramm generiert die über die Optionen bestimmten Spaltennamen und die Deklarationen der Wirtsvariablen in eine Ausgabedatei. Diese können Sie dann entweder mit einer DB2-Precompiler-Include-Anweisung in Ihr Programm importieren oder per Drag und Drop in Ihr Programm kopieren.

In unserem Beispiel lautet der Aufruf:

```
db2dc1gn -d buchdb -a replace -t adresse -l c -u db2admin -p db2admin
```

Das Batch-File `C_dc1gn.bat`, das Sie auf der Buch-CD finden, enthält diesen Aufruf.

Wenn Sie durch die Verwendung einer Eindeutigkeit in der WHERE-Klausel sicherstellen können, dass nur eine Ergebniszeile zu erwarten ist, reicht dies aus. In dem folgenden Beispiel ist dies durch die Eindeutigkeit des Primärschlüssels `AdressNr` in der WHERE-Klausel gewährleistet:

```
SELECT titel, name, vorname, gebdat, plz, ort FROM ADRESSE WHERE AdressNr = 1
```

Da jedoch in der Regel zum Zeitpunkt der Programmierung die zu erwartende Ergebnismenge nicht kalkulierbar ist, muss die Verarbeitung variabel gehalten werden. Um diese Variabilität zu erhalten, wird das Ergebnis der SELECT-Anweisung Zeile für Zeile von dem DBMS dem verarbeitenden Programm zur Verfügung gestellt. Um eine Positionierung der gerade in der Bearbeitung befindlichen Zeile zu ermöglichen, wird die so genannte *Cursor-Technik* (Zeigertechnik) verwendet.

Wie Sie mit statischem SQL einen Cursor mit dem Namen `Cursor1` definieren, zeigt Ihnen Abschnitt 3.

Um beim Lesezugriff auf Tabellenzeilen Einfluss auf das Sperrverhalten von DB2 zu erhalten, besitzt DB drei Cursor-Typen, die durch Optionen angegeben werden können:

► Updatable Cursor (änderbarer Cursor)

Diese Option ermöglicht ein Update auf die durch den Cursor gelesene Zeile. Sie müssen dazu in der Option die zu ändernden Spalten angeben. Die DB2 UDB sperrt die jeweilige Zeile in diesem Fall mit einem so genannten *U-Lock* (*lock for*

update). Ein Datensatz kann zu ein und derselben Zeit nur mit einem *U-Lock* markiert werden.

In unserem Beispiel ermöglicht die Option `FOR UPDATE OF titel,ort` ein Update der Spalten `titel` und `ort` in der Zeile, auf der sich der Cursor (Zeiger) gerade befindet.

► **Read Only Cursor**

Ergebniszeilen dieses Cursor-Typs werden von der DB2 UDB beim Lesevorgang mit einem Sperrflag „S“ für *Shared* markiert. Tabellenzeilen, die durch ein *S-Lock* markiert werden, können von mehreren Verarbeitungsprozessen, die ebenfalls das Sperrflag *S* setzen, gleichzeitig gelesen werden.

Die Option dazu lautet `FOR FETCH ONLY` bzw. `FOR READ ONLY`.

► **Ambiguous Cursor**

Wird bei der Cursordeklaration keine Qualifizierung des Cursor-Typs angegeben, handelt es sich um einen unqualifizierten Cursor-Typ. Das bedeutet, die DB2 UDB nimmt an, dass gelesene Sätze nicht geändert werden sollen, und sperrt die gelesenen Sätze mit einem *S-Lock*.

Weitere Information über das Sperrverhalten von DB2 finden Sie in Kapitel 12.

Abschnitt 4 (Die C-Funktion SetCVar)

Spalten einer Tabelle können den Wert `NULL` enthalten. Fundamentale Datentypen einer herkömmlichen Programmiersprache können den Wert `NULL` nicht darstellen. Daher müssen wir pro Tabellenspalte, die den Wert `NULL` enthalten kann, eine zusätzliche Host-Variable aufnehmen, die wir als Indikator-Variable bezeichnen. Sie finden die Deklaration dieser Indikator-Variablen in Abschnitt 1. Der Name der Indikator-Variablen entspricht dem jeweiligen Namen der Tabellenspalte und ist mit dem Präfix *i* gekennzeichnet.

Der C-Funktion des Abschnitts 4 werden zwei Parameter übergeben. Dem ersten Parameter wird beim Aufruf der Funktion der Wert der eingelesenen Host-Variable der Spalte übergeben. Dem zweiten Parameter übergibt man die entsprechende Indikator-Variable. Dieser zweite Wert beinhaltet einen Wert kleiner null (0), wenn:

- der Wert der Host-Variable der Tabellenspalte den Wert `NULL` aufweist oder
- bei der Übertragung der Tabellenspalte in die Host-Variable ein Fehler, wie z.B. ein Codepage-Konvertierungsproblem, aufgetreten ist.

In beiden Fällen sorgt die Funktion `SetCVar` dafür, dass der erste Parameter (die Host-Variable für die Tabellenspalte) zurückgesetzt wird. Diese Funktion eignet sich nur für die Verwendung von Spalten des Datentyps `CHAR`.

Abschnitt 5 (CONNECT-Anweisung)

Dieser Abschnitt führt den SQL-Befehl `CONNECT` aus. Statt der Konstanten `db2admin` für die Parameter `USER` und `PASSWORD` können Sie auch Host-Variablen verwenden. In der alltäglichen Programmierung sollten Sie dies auch vorziehen.

Abschnitt 6 (Prüfen des SQLCODEs)

Die Variable `SQLCODE` wird durch die Include-Anweisung `EXEC SQL include sqlca` in Abschnitt 1 deklariert. Sie ist Bestandteil der Struktur `SQLCA`. Über die `SQLCA` (SQL

Communication Area) werden alle Kommunikationsmeldungen zwischen Programm und DBMS ausgetauscht. Die Inhalte der Struktur werden vom DBMS nach jedem SQL-Befehl entsprechend gesetzt.

Beinhaltet die Variable `SQLCODE` den Wert 0, bedeutet das, dass der SQL-Befehl vom DBMS fehlerfrei ausgeführt werden konnte.

Des Weiteren wird in diesem Abschnitt die Anzahl der Zeilen der Tabelle Adresse ermittelt und der Host-Variablen `hAnzahl` übergeben.

Abschnitt 7 (OPEN Cursor)

Die Anweisung `OPEN Cursor1` führt die SQL-Anweisung, die in Abschnitt 3 durch den Cursor `Cursor1` deklariert wurde, *komplett* aus. Die Ergebnismenge steht nach der Ausführung des SQL-Befehls im DBMS zur Verfügung.

Abschnitt 8 (FETCH Cursor)

Die Anweisung `FETCH Cursor1` positioniert den Zeiger zur Übergabe der Ergebnismenge an das Anwendungsprogramm.

Der erste `FETCH` ist notwendig, da nach der Anweisung `OPEN Cursor1` zwar die Ergebnismenge komplett im DBMS zur Verfügung steht, der Zeiger zur Parameterübergabe jedoch nicht definiert ist.

Jeder weitere `FETCH` in der `while`-Schleife positioniert den Zeiger (Cursor) neu und verschiebt ihn um eine weitere Ergebniszeile. Wird das Ende der zur Verfügung gestellten Ergebnismenge erreicht, wird die Variable `SQLCODE` vom DBMS auf den Wert 100 gesetzt.

Hinweis:

Bitte beachten Sie, dass wir in diesem Beispiel nicht alle vorkommenden Fehlermeldungen abfangen. Die Abfrage auf `SQLCODE == 100` in der `while`-Schleife reicht für eine professionelle Programmierung nicht aus.

Abschnitt 9 (Aufruf der C-Funktion SetCVar)

In diesem Abschnitt rufen wir für jede Host-Variable und deren Indikator-Variable die Funktion `SetCVar` auf. Dadurch wird sichergestellt, dass sich in der entsprechenden Wirtsvariable der Tabellenspalte nicht zufällig der Wert der vorhergehenden Tabellenzeile befindet. Dies wäre nämlich dann der Fall, wenn sich der Wert `NULL` für die jeweilige Spalte in der Datenbank befindet. Das führt dazu, dass das DBMS die entsprechende Host-Variable nicht setzt und sie somit den Wert vom vorhergehenden Aufruf behält.

Vorsicht:

Vergessen Sie *nie*, in einem C-, COBOL- oder FORTRAN-Programm die Indikator-Variable abzufragen. Fehlinterpretationen oder Fehler, die dabei auftreten, können eventuell verheerende Auswirkungen haben.

Abschnitt 10 (UPDATE-Anweisung)

Dieses Beispiel zeigt Ihnen, wie Sie innerhalb der Verarbeitung einer durch einen Cursor ausgelösten SELECT-Anweisung ein Update auf die gerade über den Zeiger zur Verfügung stehende Zeile ausführen können.

In unserem Beispiel ändern wir die Spalten `Ort` und `titel`.

Hinweis:

Beachten Sie, dass auch im Falle eines Updates die Indikator-Variable mit dem entsprechenden Wert gesetzt wird.

Der Wert 0 bedeutet, dass der Inhalt der Host-Variable in die Spalte übernommen wird.

Der Wert -1 bedeutet, dass unabhängig vom Wert der Wirtsvariablen der Wert NULL in die Spalte der Tabelle geschrieben wird.

Abschnitt 11 (Programm-Ende)

Die Anweisung `EXEC SQL CLOSE Cursor1` sorgt dafür, dass der Cursor geschlossen wird und das DBMS somit den Speicherbereich freigibt, der für die Ergebnismenge zur Verfügung gestellt wurde.

Der Befehl `EXEC SQL CONNECT RESET` löst die Verbindung zur Datenbank und gibt den kompletten Speicherbereich zwischen Anwendungsprogramm und DBMS frei.

Das Programm kann danach beendet werden.

11.4.4 Wie kommen Sie zum ausführbaren C-Programm?

Wie Sie in Abbildung 11.7 sehen, sind insgesamt vier Schritte notwendig, um vom Embedded-C-Programm `BuchDB_S_SQL.SQC` zum ausführbaren Programm `BuchDB_S_SQL.EXE` zu gelangen. Auch hier wollen wir anhand unseres Beispiels die genaue Vorgehensweise Schritt für Schritt erläutern.

Beachten Sie, dass zur Ausführung dieser Schritte bestimmte Voraussetzungen erfüllt sein müssen:

- ▶ Die DB2 UDB V8 ist korrekt installiert.
- ▶ Der Microsoft Visual C++ 6.00 Compiler ist korrekt installiert.
- ▶ Sie befinden sich bei der Ausführung der Befehle in einem DB2-Befehlsfenster.
- ▶ Die entsprechenden Umgebungsvariablen wie z.B. `INCLUDE`, `PATH` und `LIB` sind für die entsprechende DB2-Befehlsfenster-Session korrekt gesetzt.
- ▶ Sie haben die Datenbank `BUCHDB` mit dem auf der CD befindlichen Batch-File `CDATENB.BAT` in einem DB2-Befehlsfenster erzeugt.
- ▶ Auf Ihrem System ist der lokale Benutzer `db2admin` mit dem Passwort `db2admin` eingerichtet. Sollte dies nicht der Fall sein, müssen Sie die Aufrufe in den Batch-Files entsprechend anpassen.

Tipp:

Achten Sie bitte darauf, dass für die korrekte Ausführung der Schritte Kompilieren und Linken in den Umgebungsvariablen LIB und INCLUDE die entsprechenden DB2-Pfadangaben vor den Microsoft-Compiler-Pfadangaben stehen müssen.

Beispiele:

- INCLUDE=d:\ibm\sql1ib\include;f:\Programme\Microsoft Visual Studio\VC98\atl\include;...
- LIB=d:\ibm\sql1ib\lib;f:\Programme\Microsoft Visual Studio\VC98\mfc\lib;...

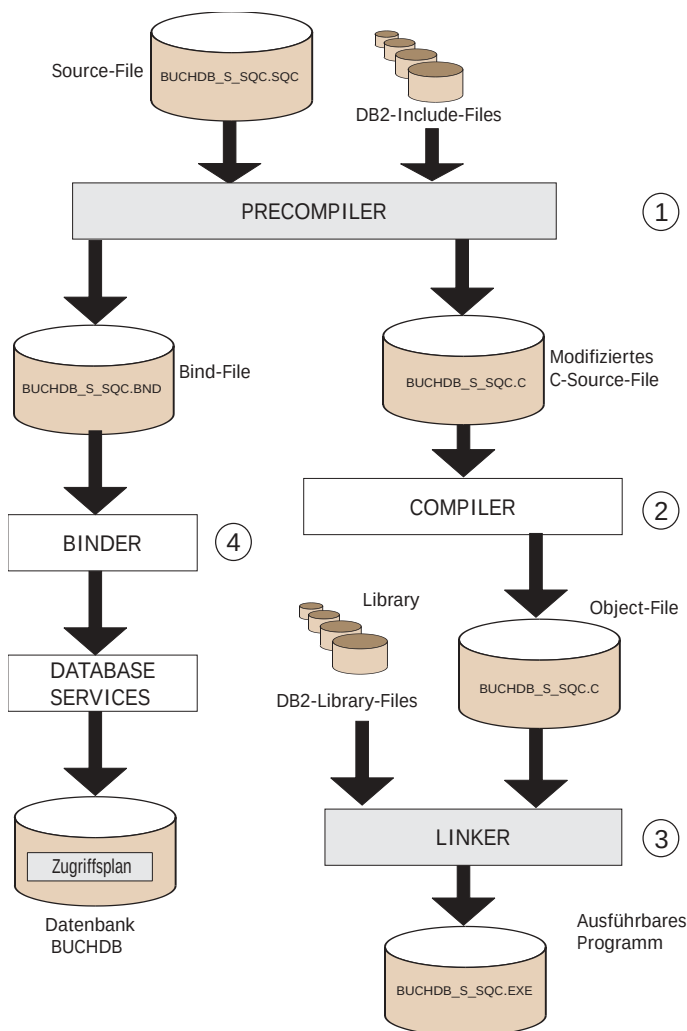


Abbildung 11.7: Vom SQLC-Source-Code zum ablauffähigen Programm

Schritt 1 – Precompile

Der DB2-Precompiler erzeugt aus dem SQL-Source-Code BuchDB_S_SQLC.SQC unter anderem die Datei BuchDB_S_SQLC.C. Dazu eliminiert er den gesamten Embedded-SQL-Code, indem er ihn auskommentiert. Statt der Embedded-SQL-Statements werden die entsprechenden DB2-C-Sprachelemente wie z.B. DB2-C-API-Funktionen und C-Compiler-Makro-Anweisungen eingesetzt. Eine weitere Aufgabe, die der Precompiler übernimmt, ist, dass er das Bind-File BuchDB_S_SQLC.BND erzeugt. In diese Datei schreibt er alle SQL-relevanten Informationen, wie z.B. SQL-Befehle, und verwendeten Host-Variablen. Dazu werden die SQL-Befehle nummeriert und – falls nicht über eine Bind-Option anders angegeben – mit dem Schema des Benutzers versehen, der den Precompile-Befehl ausführt. Dadurch muss das Anwendungsprogramm bei der Ausführung der SQL-Anweisungen nicht das komplette SQL übergeben, sondern nur das Synonym für den SQL-Befehl.

Beispiel: „Name des späteren Zugriffsplans“ + „Nummer des SQL-Statements“

Dies erfordert natürlich, dass das aus dem C-Source-Code BuchDB_S_SQLC.C ablauffähige Programm exakt mit dem Zugriffsplan übereinstimmt, der sich in der Datenbank befindet.

Um dies zu gewährleisten, schreibt der Precompiler in beide generierte Dateien BuchDB_S_SQLC.C und BuchDB_S_SQLC.BND ein so genanntes *consistency token*. Dieses *consistency token* besteht unter anderem aus dem Timestamp (Zeitstempel), der zu der Zeit gültig war, als die beiden Files erstellt wurden. Dieses Token wird in der weiteren Verarbeitung mitgeführt. Es steht also später sowohl im ablauffähigen Programm als auch in dem in DB2 erzeugten Zugriffsplan.

Stimmt beim Aufruf der SQL-Anweisung dieses *consistency token* zwischen Zugriffsplan und ablauffähigem Programm nicht überein, erzeugt die DB2 UDB zur Laufzeit den SQL-Fehler -818. In unserem Beispiel würde der Fehler so aussehen wie in Abbildung 11.8.

```
Die Tabelle beinhaltet insgesamt 0 Zeilen -- FehlerCode : -818
```

```
Cursor konnte nicht geöffnet werden SQLCODE -818
```

```
Msg : SQLRALDP~
```

Abbildung 11.8: SQL-Fehlercode -818: Consistency-Token-Fehler

Tipp:

Beachten Sie, dass der SQL-Fehler -818 erst in dem Moment auftritt, wenn die SQL-Anweisung zur Ausführung kommt. Daher kommt es in der Praxis öfter vor, dass bei komplexen Programmen mit mehreren Programm- und somit Bind-Modulen der Fehler erst Tage nach der Installation der neuen Programmversion zum Vorschein kommt.

Um unser Beispielprogramm zu precompilieren, verwenden wir folgenden Befehl:
`db2 prep BuchDB_S_SQC.SQC bindfile`

Hinweis:

Bitte beachten Sie, dass Sie vor dem Aufruf des Befehls mit einer Datenbank verbunden sein müssen (CONNECT). Wenn Sie wie wir die Option `bindfile` benutzen, kann es sich auch um eine Dummy-Datenbank handeln (d.h., es müssen keine Tabellen vorhanden sein).

Sollten Sie diese Option weglassen, wird nur ein temporäres Bind-File erzeugt, und der Precompiler versucht sofort, den BIND-Lauf mit auszuführen und somit den Zugriffsplan in der Datenbank zu erzeugen.

Um den Zugriffsplan zu erzeugen, müssen Sie jedoch mit der korrekten Datenbank verbunden sein. In unserem Fall wäre das die Datenbank BuchDB.

Einen Auszug der wichtigsten Precompiler-Optionen finden Sie in Abbildung 11.9.

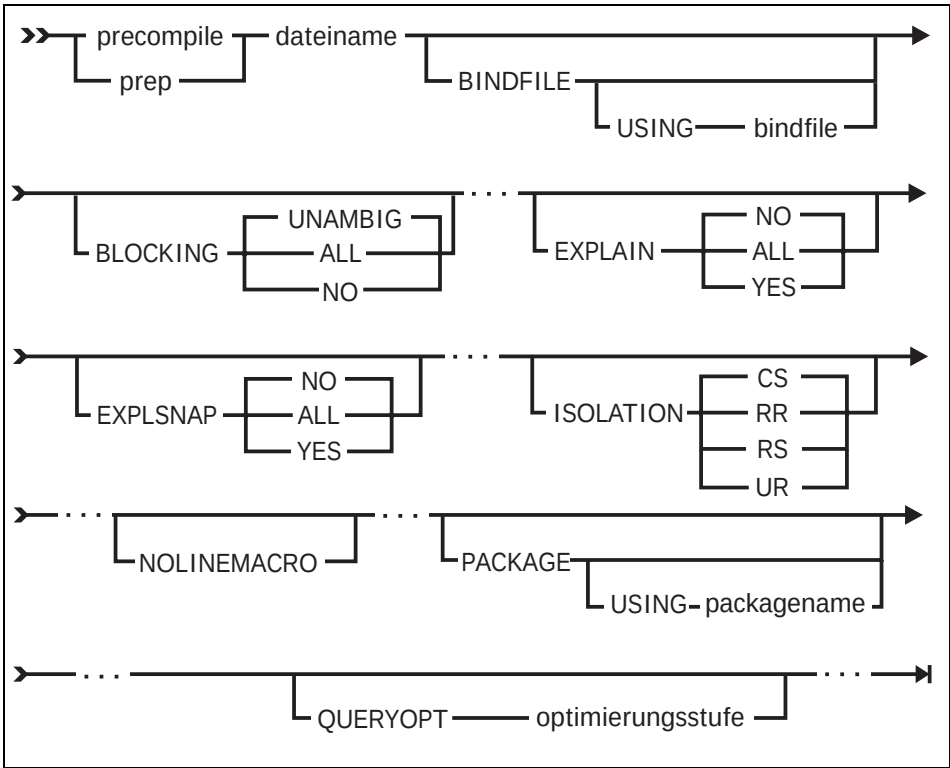


Abbildung 11.9: db2 prep-Syntaxdiagramm

Erklärung der Optionen

Da einige Optionen ebenso im Bind-Befehl zu finden und dort erklärt sind, werden wir hier nur auf die Unterschiede eingehen.

- ▶ Der Dateiname gibt den Namen des Wirtsprogramms an, das übersetzt werden soll.
- ▶ Mit der Option `BINDFILE` geben Sie an, ob Sie ein Bind-File erzeugen wollen. Wenn Sie den Parameter `USING <name des Bind-Files>` nicht angeben, erhält das Bind-File automatisch den Namen des zu übersetzenden Wirtsprogramms. Wenn Sie die Option `BINDFILE` nicht nutzen, erzeugt die DB2 UDB automatisch den Zugriffsplan in der Datenbank. Hierzu muss jedoch die komplette Datenbank vorliegen. Ebenso benötigt derjenige, der den Precompile-Befehl ausführt, alle Rechte der im Programm aufgeführten SQL-Anweisungen.
- ▶ `NOLINEMACRO` gibt Ihnen die Möglichkeit, die Line-Makro-Anweisung `#line` zu unterdrücken. Dies kann im Debugging-Fall von großem Nutzen sein.

Schritt 2 – Compile

In diesem Schritt wird ein Standard-C-Compiler eingesetzt. In unserem Beispiel haben wir den Microsoft Visual C++ 6.0 Compiler verwendet. Mit ihm übersetzen Sie das vom Precompiler erzeugte C-Programm `BuchDB_S_SQC.SQC`. Der Compiler erzeugt das Objekt-File `BuchDB_S_SQC.OBJ`.

Wir verwenden in unserem Beispiel folgenden Compiler-Befehl:

```
cl -Zi -Od -c -W2 -DWIN32 BuchDB_S_SQC.c
```

Schritt 3 – Linken

Das vom Compiler erstellte Objekt-File linken Sie mit eventuellen, zusätzlich für Ihr Programm benötigten Objekt-Dateien und Libraries. In unserem Fall sind keine weiteren Objekt-Files notwendig.

Da für ein ablauffähiges Programm die vom Precompiler hinzugefügten DB2-C-API-Funktionen vorhanden sein müssen, müssen Sie beim Link-Vorgang die DB2-Library `db2api.lib` mit einbinden, die die DB2 UDB-API-Funktionen enthält.

Der in unserem Beispiel zum Erfolg führende Befehl lautet:

```
link -debug -out:%BuchDB_S_SQC.exe BuchDB_S_SQC.obj db2api.lib
```

Schritt 4 – Binden

Als letzten Schritt können Sie den Bind-Befehl ausführen. Hierzu benötigen Sie das Bind-File und eine Verbindung zu der Datenbank, die Sie in Ihrem Programm ansprechen. Dies ist notwendig, da der Bind-Prozess zur Erstellung des Zugriffsplans entsprechende Informationen aus der Datenbank benötigt. Zu diesen Informationen gehören:

- ▶ Sind die im Programm angesprochenen Datenbankobjekte vorhanden?
- ▶ Besitzt der Programmiegnier (Binder) die Rechte, um auf die im Programm angesprochenen Datenbankobjekte zuzugreifen?
- ▶ Welche Informationen liegen in den Statistiken vor?

Hinweis:

Der Benutzer, der den Bind-Befehl für ein statisches SQL-Programm ausführt, benötigt alle Berechtigungen, die zum Ausführen der im Programm vorkommenden SQL-Befehle notwendig sind.

Im Gegensatz dazu benötigt der Benutzer, der das Programm ausführt, nur die EXECUTE-Berechtigung auf den Zugriffsplan (package).

Zum Binden des Bind-Files BuchDB_S_SQC.BND benutzen Sie den Befehl:

```
db2 bind BuchDB_S_SQC.bnd
```

Hinweis:

Wenn Sie all diese Befehle nicht einzeln eingeben möchten, finden Sie auf unserer Buch-CD das entsprechende Batch-File bcls.bat.

Bitte beachten Sie, dass wir in diesem Batch-File davon ausgehen, dass der Benutzer db2admin mit dem Kennwort db2admin vorhanden ist. Sollte dies nicht der Fall sein, müssen Sie entweder die Stapelverarbeitungsdatei ändern, oder Sie benutzen das ebenfalls auf der CD befindliche allgemeinere Batch-File BCLMSSQC.BAT und die entsprechend richtigen Aufrufparameter.

Die wichtigsten Optionen des Bind-Befehls sehen Sie in Abbildung 11.10.

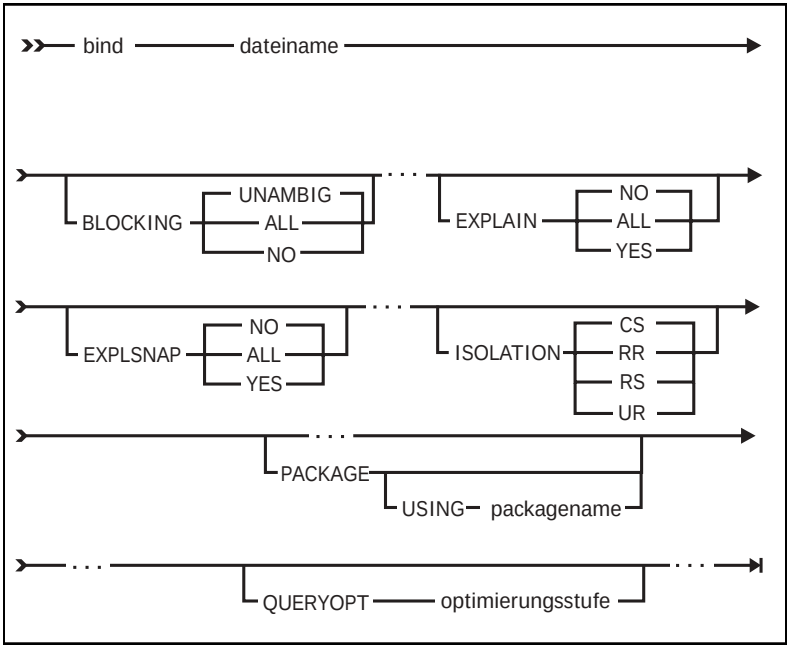


Abbildung 11.10: Syntax des Bind-Befehls

Erklärung der Optionen

- ▶ Der Dateiname gibt den Namen des zu bindenden Bind-Files an.
- ▶ Mit der Option BLOCKING nehmen Sie Einfluss darauf, ob die vom DBMS zur Verfügung gestellte Ergebnismenge zum Anwendungsprogramm hin als Block übertragen werden soll. Hier geben Sie an, für welchen Cursor-Typ geblockt werden soll.
- ▶ EXPLAIN gibt an, ob Sie Informationen für das DB2-Tool Explain zur Verfügung stellen wollen.
- ▶ EXPLSNAP gibt an, ob Sie Informationen für das DB2-Tool Visual-Explain zur Verfügung stellen wollen.
- ▶ Mit der Option ISOLATION bestimmen Sie die Isolationsstufe für dieses Programm-Modul.
- ▶ PACKAGE ermöglicht es Ihnen, den Namen des Zugriffsplans zu bestimmen.
- ▶ QUERYOPT bestimmt, mit welcher Optimierungsstufe DB2 den Zugriffsplan erstellt.

11.4.5 Statisches SQL am Beispiel der Wirtssprache Java

Wie am Beispiel des C-Programms haben wir auch hier das Programm (siehe Abbildung 11.11) in unterschiedliche Bereiche aufgeteilt. Da uns die Programmiergrundlagen (z.B. Was ist ein Cursor, was ist ein Fetch, welche Cursor-Typen gibt es und was sind Host-Variablen?) aus dem C-Beispiel bekannt sind, werden wir an dieser Stelle nur auf die Java-spezifischen SQL-Programmiereigenschaften eingehen.

Abschnitt 1 (Import der SQL-Java-Klassen)

Wie im C-Programm benötigt Java zur Unterstützung der SQL-Programmierchnittstelle gewisse Programmkomponenten bzw. -klassen. Diese müssen durch die entsprechenden Import-Anweisungen geladen werden:

```
import java.sql.*;
import sqlj.runtime.*;
import sqlj.runtime.ref.*;
```

Anders als bei der Programmiersprache C werden diese Anweisungen in Java nicht als Precompiler-Anweisungen gekennzeichnet.

```

// -----
// Beispiel Programm für statisches SQL
//
// Java Programm BUCH_S_SQLJ.SOLJ
//
// (C) COPYRIGHT Lis.Tec GmbH
// -----
import java.lang.*;
import java.text.SimpleDateFormat;
import java.sql.*;
import sqlj.runtime.*;
import sqlj.runtime.ref.*;

#sql iterator Positioniert(String, String, String, Date, String, String);

class BuchDB_S_SQLJ
{ // 1
    public static void main(String argv[])
    { // 2
        // Deklaration der Variablen
        // =====
        java.lang.String htitel = "";
        java.lang.String hname  = "";
        java.lang.String hvornamen = "";
        java.sql.Date   hgebdat = null;
        java.lang.String hplz   = "";
        java.lang.String hort   = "";
        String url          = null;
        String BenutzerID    = "db2admin";
        String Kennwort      = "db2admin";
        String server        = "localhost";
        int portNumber       = 50000;
        String alias         = "BuchDB";
        Connection con       = null;
        DefaultContext ctx = null;
        SimpleDateFormat Datum = new SimpleDateFormat("dd.MM.yyyy");
        try
        { // 3
            System.out.println();
            System.out.println(" Dieses SQLJ-Beispiel zeigt Ihnen Inhalte der Tabelle Adresse der Datenbank BUCHDB ");
            System.out.println(" ----- ");

            // Bilden der URL mit Typ4-Treiber
            // =====
            url = "jdbc:db2://" + server + ":" + portNumber + "/" + alias;
            System.out.println(" Connect zur Datenbank BuchDB mit JDBC type 4 driver." );
            try
            { Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
            }
            catch (Exception e)
            { System.out.println("\n --- Fehler beim Laden der DB2-Treiber --- \n");
              System.out.println(e);
              System.exit(1);
            }
            // connect für die Datenbank BUCHDB
            // =====
            con = DriverManager.getConnection( url, BenutzerID, Kennwort );
            con.setAutoCommit(false);

            // Setzen des Default-Context
            // =====
            ExecutionContext exeCtx = new ExecutionContext();
            ctx = new DefaultContext( con );
            DefaultContext.setDefaultContext(ctx);

            // Update-Befehl
            // =====
            htitel="herr";
            #sql [exeCtx] {UPDATE Adresse SET titel = :htitel WHERE titel = 'Herr'};
            System.out.println("\n Es wurde in " + String.valueOf(exeCtx.getUpdateCount())
                               + " Zeilen der Titel von Herr auf herr geändert\n\n");
            System.out.println("\n Titel Vorname      Nachname      Geb-Datum      Plz      Ort      iort");
            System.out.println("\n =====");

            // Bilden des Cursors
            // =====
            Positioniert ergebnis = null;

            #sql [exeCtx] ergebnis = (SELECT Substr(titel,1,6),Substr(name,1,15),
                                           Substr(vorname,1,15),gebdat,plz,substr(ort,1,15)
                                           FROM ADRESSE );
            #sql [exeCtx] {FETCH :ergebnis INTO :htitel ,:hname ,:hvornamen ,:hgebdat ,:hplz ,:hort };

            while ( !ergebnis.endFetch() )
            { // 4
                System.out.println("\n" + htitel + " " + hvornamen + " " + hname + " "
                                   + Datum.format(hgebdat) + " " + hplz + " " + hort);
                #sql {FETCH :ergebnis INTO :htitel ,:hname ,:hvornamen ,:hgebdat ,:hplz ,:hort };
            } // 4
            ergebnis.close();
            System.out.println("\n Bitte achten Sie darauf die Änderung wurden mit einem ROLLBACK zurückgenommen ");
            con.rollback();
            con.close();
        } // 3
        catch (SQLException e)
        { System.out.println(" Fehlercode : " + e.getErrorCode() + " Fehlermeldung : " + e.getMessage());
        }
    } // 2
} // 1

```

Abbildung 11.11: Statisches SQL-Programm mit der Wirtssprache Java

Tipp:

Achten Sie darauf, dass durch das Einbinden der SQL-Java-Klassen ein gewisser Konflikt durch das Importieren von Standardklassen auftreten kann.

Dies rührt daher, dass gewisse SQL-Klassen von den Java-Standardklassen abgeleitet sind und dadurch ein Zuweisungskonflikt auftreten kann.

In unserem Beispiel wäre das der Fall, wenn Sie zusätzlich zu den in Abschnitt 1 aufgeführten SQL-Klassen die Klasse `java.util.Date` importiert hätten.

Abbildung 11.12 zeigt, welche Fehlermeldungen Sie erhalten, wenn Sie diese Klasse zusätzlich importieren.

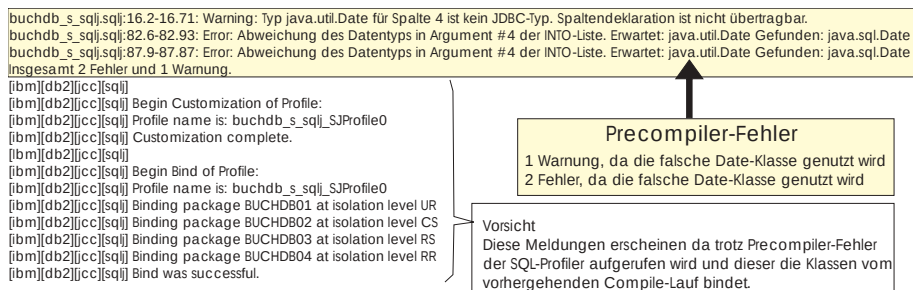


Abbildung 11.12: Precompiler-Fehler in einem Java-Programm

Abschnitt 2 (Deklaration des Iterators)

Wie Sie wissen, benötigen Sie einen Cursor, um mehr als eine Ergebniszeile einer Tabelle zu bearbeiten. Damit Ihnen in Java die entsprechende Klasse für diesen Cursor zur Verfügung steht, deklarieren Sie hier einen Iterator. Der Precompiler generiert aus dieser Iterator-Anweisung die entsprechende Klassendefinition für den Cursor.

Für die Deklaration eines Iterators (einer Cursor-Klasse) stehen Ihnen zwei Iterator-Typen zur Verfügung:

- ▶ eine Iterator-Deklaration anhand der Spaltenpositionen (*positionierter Iterator*)

Diese Methode eignet sich, wenn Sie ähnlich wie in einem Embedded-C-Programm in der FETCH-Anweisung den Embedded-SQL-Befehl `INTO :Host-Variable` nutzen möchten. Der Precompiler erzeugt hier einen so genannten *positionierten Cursor*. Er richtet sich bei der Parameterübergabe der Spalten vom DBMS zum Programm innerhalb der FETCH-Anweisung an der Position der Spalten in der SELECT-Anweisung aus.

Hinweis:

Achten Sie bei der Deklaration eines positionierten Iterators darauf, dass die Angabe der verwendeten Datentypen exakt der Reihenfolge und den Datentypen der Spalten der vom Cursor zugeordneten SELECT-Anweisung entspricht.

Wir haben in unserem Beispiel die Methode wie folgt verwendet:

```
#sql iterator Positioniert(String , String, String, Date, String, String);
```

► eine Iterator-Deklaration anhand der Spaltennamen (*benannter Iterator*)

Im Gegensatz zu dem positionierten Iterator wird hier beim Precompilieren eine Java-Cursor-Klasse definiert, mit der Sie die Spalten über Methoden ansprechen können. Die Namen der Methoden der Cursor-Klasse entsprechen exakt den Namen der Datentypen und der Reihenfolge der Spalten innerhalb der Spaltenliste der SELECT-Anweisung. Man spricht in diesem Fall von einem so genannten *benannten Cursor*. Die Groß-/Kleinschreibung spielt bei den Namen der Spalten keine Rolle.

Ein Beispiel dafür wäre die Deklaration der Anweisung:

```
#sql iterator Benannt(String titel, String name, String vorname, Date gebdat,  
String plz, String ort);
```

Tipp:

Sollten Ihre Spaltennamen zufälligerweise nicht den Konventionen gültiger Java-Namen entsprechen, müssen Sie die SELECT-Anweisung so gestalten, dass die Namen der Ergebnisspalten den Java-Konventionen entsprechen.

Beispiel:

Eine Tabelle enthält einen Spaltennamen mit dem in Java reservierten Schlüsselwort *PUBLIC*. Die SELECT-Anweisung würde wie folgt lauten:

```
SELECT PUBLIC FROM tabellenname
```

Um die Tabellenspalte in Java mit einem benannten Cursor verarbeiten zu können, muss die Anweisung wie folgt umgestaltet werden:

```
SELECT PUBLIC AS PUBLIC_SPALTE FROM tabellenname
```

Jetzt können Sie bei der Deklaration des benannten Iterators die Spalte mit ihrem Synonym *PUBLIC_SPALTE* ansprechen.

Außer den Spalten-Methoden generiert der Precompiler weitere wichtige Methoden:

► *Next*

Die Methode ermöglicht es Ihnen, den Cursor auf die nächste Position der Ergebnismenge der SELECT-Anweisung zu positionieren.

► *Close*

Dient zum Schließen des Cursors.

Abschnitt 3 (Deklaration der Host-Variablen für die SQL-Anweisungen)

Im Gegensatz zu dem DB2-SQL-C-Precompiler handelt es sich beim Java-SQL-Precompiler um einen Zwei-Phasen-Precompiler. Dies hat zur Folge, dass Sie die Wirtsvariablen nicht innerhalb einer BEGIN- und END DECLARE SECTION deklarieren müssen.

Wie im Embedded-SQL-Programm können Sie sich die Deklarationen über das DB2 UDB-Hilfsprogramm db2dc1gn erstellen lassen.

Der Aufruf hierfür lautet in unserem Beispiel:

```
db2dc1gn -d buchdb -a replace -t adresse -l java -u db2admin -p db2admin
```

Hinweis:

Sie finden diesen Aufruf auf der CD in der Stapelverarbeitungsdatei JAVA_dc1gn.bat.

Abschnitt 4 (Deklaration der Java-spezifischen Host-Variablen)

In diesem Beispiel haben wir die Angaben zur Bestimmung der url und der Datenbank-Connection variabel gehalten.

In der Angabe der url für die JDBC-Type-4-Treiber sind folgende Parameter enthalten:

- ▶ Name des Servers
Dieser Parameter beinhaltet den Namen Ihres Datenbankservers.
- ▶ Portnummer
Es handelt sich hierbei um die TCP/IP-Portnummer des Exemplars Ihres Datenbankservers, unter der die Datenbank katalogisiert ist.
- ▶ Name der Datenbank, die angesprochen werden soll

Tipp:

Diese neue Technik, die ab dem JDBC-Type-4-Treiber unterstützt wird, ermöglicht Ihnen aus einem Java-Programm heraus einen direkten Zugriff auf den Datenbankserver. Dies bedeutet, dass Sie auf Ihrem System keinen DB2-Run-Time-Client installieren und die Datenbanken nicht katalogisiert haben müssen.

Abschnitt 5 (Bilden der url und Laden des JDBC-Treibers)

In diesem Abschnitt wird die Host-Variable url gebildet und der entsprechende JDBC-Treiber geladen.

Abschnitt 6 (Exception)

Sollte beim Laden des JDBC-Treibers ein Fehler auftreten, wird eine Exception ausgelöst und dadurch dieser Programmteil aufgerufen. Es wird eine Fehlermeldung ausgegeben, und das Programm wird verlassen.

Abschnitt 7 (Connect zur Datenbank)

Hier wird die Datenbankverbindung aufgebaut und der Java-Instanz `con` zugewiesen. Damit nicht nach jedem SQL-Befehl ein automatischer COMMIT erfolgt, wird hier der Auto-Commit zurückgesetzt.

Abschnitt 8 (Context)

In der reinen JDBC-Programmierung besitzen Sie über die Methoden der entsprechenden JDBC-Klasse jederzeit Zugriff auf Ergebnisse der jeweiligen SQL-Anweisung. Diesen Bezug zwischen der Instanz der Java-Klasse und der entsprechenden SQL-Anweisung besitzen Sie bei der Programmiermethode `Embedded-SQL` nicht. Ebenso besitzen Sie in Java keinerlei Möglichkeit, direkt (ohne Klasse) auf die Ergebnisse der SQL-Anweisung zuzugreifen.

Um dieses Problem zu beheben, gibt es im `Embedded-SQL` den so genannten *Context*. Es gibt drei verschiedene Arten von Context:

► Connection-Context

Dieser ermöglicht es Ihnen, auf die Methoden der durchgeführten `Connection` zuzugreifen. Sie sind dadurch auch in der Lage, verschiedenen `Embedded-SQL`-Anweisungen verschiedene `Connections` zuzuordnen.

► Default-Context

Dieser Context ermöglicht es Ihnen, Ihre Datenbankverbindung als `Default-Connection` zu setzen.

► Execution-Context

Dieser Context ermöglicht Ihnen den Zugriff auf Methoden des ausgeführten `SQL-Statements`.

Um das schwierige Thema etwas verständlicher zu machen, schauen wir uns unser Beispiel an.

Die Anweisung `ExecutionContext exeCtx = new ExecutionContext();` erzeugt die Java-Instanz `exeCtx` der Klasse `ExecutionContext`.

Diese Java-Instanz kann dann in den `Embedded-SQL`-Anweisungen angegeben werden (siehe Abschnitt 9). Sie besitzen dadurch die Möglichkeit, über die Java-Instanz `exeCtx` auf die entsprechenden Methoden der Klasse `ExecutionContext` zuzugreifen.

Die in „Abschnitt 4 (Deklaration der Java-spezifischen Host-Variablen)“ vorhandene Anweisung `DefaultContext ctx = null` sorgt dafür, dass eine nicht-initialisierte Java-Instanz der Klasse `DefaultContext` gebildet wird. In diesem Abschnitt 7 initialisieren Sie `ctx` mit den Werten der vorhandenen Datenbankverbindung (`connection`). Durch die Anweisung `DefaultContext.setDefaultContext(ctx);` setzen Sie dann den `Default-Context` auf die Datenbankverbindung, die über `con` spezifiziert ist. Dieser `Default-Context` kommt immer dann implizit zum Tragen, wenn Sie bei Ihrer `Embedded-SQL`-Anweisung keinen expliziten `Connection-Context` angeben.

Abschnitt 9 (Update-Befehl)

Die hier abgebildete Embedded-SQL-Anweisung führt einen Update-Befehl durch.

Hier können Sie erkennen, dass Ihnen durch die Angabe des Execution-Contexts in der Embedded-SQL-Anweisung in der nachfolgenden Ausgabe über `exeCtx` unter anderem die Methode `getUpdateCount()` zur Verfügung steht. Diese Methode gibt Ihnen die Anzahl der Zeilen zurück, die von dem Update-Befehl betroffen wurden.

Hinweis:

Da die Spezifikation der Embedded-SQL-Schnittstelle durch Sun Microsystems erfolgt ist, stehen Ihnen die Informationen, welche Methoden Ihnen zur entsprechenden Klasse zur Verfügung stehen, über die Homepage von Sun Microsystems zur Verfügung (www.sun.de).

Abschnitt 10 (Deklaration des Cursors)

Durch die Anweisung `#sql iterator Positioniert(String, String, String, Date, String, String);` in „Abschnitt 2 (Deklaration des Iterators)“ steht Ihnen in diesem Abschnitt die Klasse `Positioniert` zur Verfügung.

Die Anweisung `Positioniert ergebnis = null;` instanziiert Ihnen den Cursor `ergebnis` der Klasse `Positioniert`.

Der Cursor `ergebnis` wird dann genutzt, um die SELECT-Anweisung *komplett* auszuführen. Danach steht im DBMS in der Regel die komplette Ergebnismenge zur Verfügung.

Die folgende FETCH-Anweisung dient dazu, den Cursor auf die erste Zeile zu positionieren und die Ergebnisspalten in die entsprechenden Wirtsvariablen zu übertragen:

```
#sql [exeCtx] {FETCH :ergebnis INTO :htitel ,:hname ,:hvorname ,:hgebdat ,:hplz ,:hort };
```

Abschnitt 11 (Verarbeitung und FETCH-Anweisung)

Die Schleife sorgt über die Methode `endFetch()` dafür, dass nach erfolgreicher Verarbeitung der Host-Variablen (in unserem Fall nach einer Ausgabe) der Cursor über die FETCH-Anweisung auf die nächste Stelle der Ergebnismenge positioniert wird. Ist das Ende der vom DBMS zur Verfügung gestellten Ergebnismenge erreicht, gibt die Methode `endFetch()` den Wert `false` zurück.

Hinweis:

Bitte beachten Sie, dass wir in unserem Beispiel keinerlei Fehlerfälle betrachtet haben, die beim Fetch auftreten können. Diese Betrachtungsweise reicht also für die professionelle Programmierung bei weitem nicht aus.

Abschnitt 12 (Schließen des Cursors)

Der Befehl `ergebnis.close()`; schließt den Cursor und gibt den Speicherbereich frei, der zur Bereitstellung der Ergebnismenge alloziert wurde.

Abschnitt 13 (Rollback)

Der Rollback, der mit dem Befehl `con.rollback()`; ausgeführt wird, sorgt dafür, dass die in „Abschnitt 9 (Update-Befehl)“ geänderten Zeilen nicht tatsächlich in der Datenbank festgeschrieben werden. Sie finden somit in der Datenbank die vor dem Update-Befehl ausgeführten gültigen Werte.

Abschnitt 14 (Close Connection)

Dieser letzte Abschnitt sorgt mit dem Befehl `con.close()`; dafür, dass die Datenbankverbindung zwischen dem Anwendungsprogramm und dem DBMS gelöst wird und die entsprechenden Speicherbereiche freigegeben werden.

11.4.6 Wie kommen Sie zum ausführbaren Java-Programm?

Abbildung 11.7 zeigt Ihnen drei Schritte auf, wie Sie vom SQLJ-Source-Code `BuchDB_S_SQLJ.SQLJ` zur ablauffähigen Java-Klasse `BuchDB_S_SQLJ.class` gelangen. Die einzelnen Schritte werden wir Ihnen im Folgenden erklären.

Bitte beachten Sie, dass zur Ausführung dieser Schritte bestimmte Voraussetzungen erfüllt sein müssen:

- ▶ Die DB2 UDB V8 ist korrekt installiert.
- ▶ Sie befinden sich bei der Ausführung der Befehle in einem DB2-Befehlsfenster.
- ▶ Die entsprechenden Umgebungsvariablen wie z.B. `PATH` sind für die entsprechende DB2-Befehlsfenster-Session korrekt gesetzt. Wir verwenden in unseren Beispielen das JDK der DB2 UDB V8. Achten Sie darauf, dass die DB2-JDK-Pfadangabe am besten an erster Stelle in der Umgebungsvariable `PATH` steht. Da bei uns DB2 auf `D:\IBM\SQLLIB` installiert ist, lautet der Befehl zum Setzen des Pfades in unserem Beispiel:

```
set path=D:\IBM\SQLLIB\java\jdk\bin;%path%
```

Dieser Befehl befindet sich auf der CD im Batch-File `setjava.bat`.

Bitte denken Sie daran, das File `setjava.bat` entsprechend Ihrer DB2-Installation anzupassen. Wenn Sie diese Stapelverarbeitungsdatei entsprechend verallgemeinern möchten, können Sie den dortigen Befehl durch folgenden ersetzen:

```
set path=%db2path%;%path%
```

- ▶ Sie haben die Datenbank `BUCHDB` mit dem auf der CD befindlichen Batch-File `CDATENB.BAT` in einem DB2-Befehlsfenster erzeugt.
- ▶ Auf Ihrem System ist der lokale Benutzer `db2admin` mit dem Passwort `db2admin` eingerichtet. Sollte dies nicht der Fall sein, müssen Sie den Aufruf in dem Batch-File `bc1java.bat` und die entsprechenden Wirtsvariablen im Source-Code `BuchDB_S_SQLJ.SQLJ` anpassen.

- Die Portnummer Ihres DB2-Exemplars, das die Datenbank BUCHDB katalogisiert hat, hat den Wert 5000. Sollte dies nicht der Fall sein, müssen Sie die Wirtsvariable `portNumber` in dem Source-Code `BuchDB_S_SQLJ.SQLJ` entsprechend anpassen.

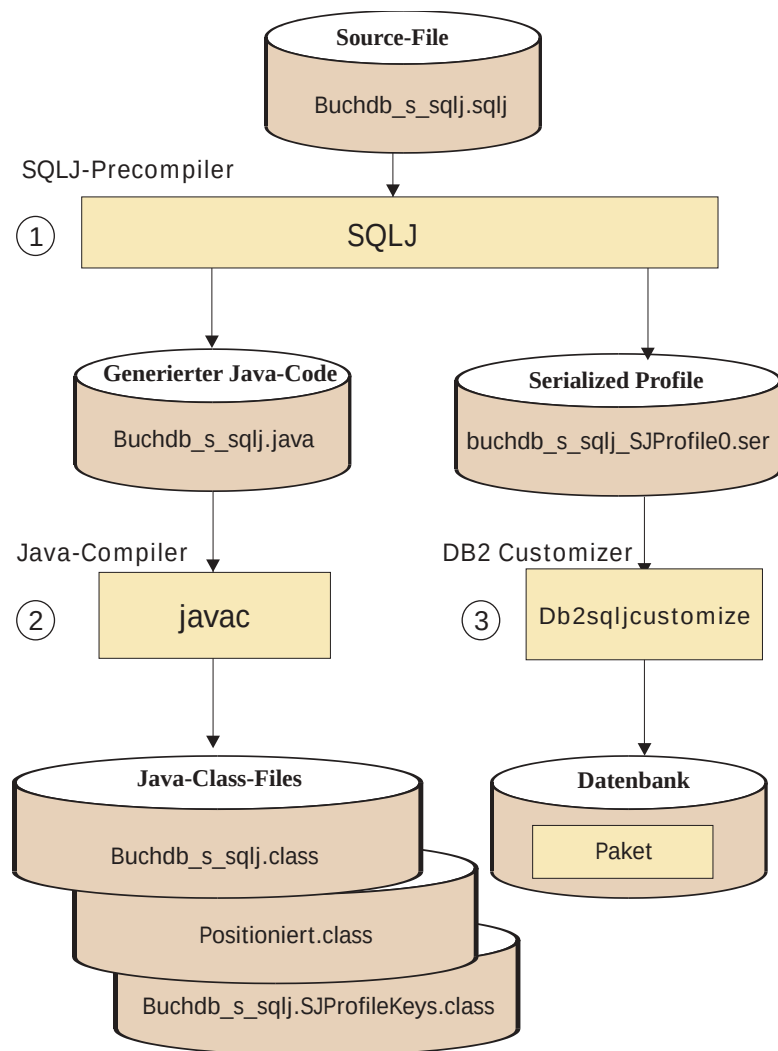


Abbildung 11.13: Vom SQLJ-Source-Code zur ablauffähigen Java-Klasse

Schritt 1 – Precompile

Der SQLJ-Precompiler ist im Gegensatz zum DB2-SQL-C-Precompiler ein Java-Standardprogramm und nicht IBM-DB2-UDB-spezifisch. Er erzeugt aus der Source-Code-Datei `BuchDB_S_SQLJ.SQLJ` die beiden Dateien `buchdb_s_sqlj.java` und `buchdb_S_SQLJ_SJProfile0.ser`. Zur Erstellung des Java-Source-Code-Files `buchdb_s_sqlj.java` werden die gesamten Embedded-SQL-Anweisungen durch JDBC-Anweisungen ersetzt.

In dem File buchdb_S_SQLJ_SJProfile0.ser werden alle SQL-relevanten Informationen wie z.B. SQL-Befehle und verwendete Host-Variablen festgehalten.

Im Gegensatz zu unserem Beispiel in der Programmiersprache C wird hier kein DB2-UDB-spezifisches Bind-File erzeugt. Es handelt sich in diesem Fall um eine DBMS-unabhängige SQL-Informationsdatei. Diese kann dann vom jeweils verwendeten DBMS spezifisch verarbeitet werden. Die DB2 UDB liefert hierzu den *DB2 SQLJ Profile Customizer*.

Der Aufruf des Java-Precompilers lautet für unser Beispielprogramm BuchDB_S_SQLJ.SQLJ wie folgt:

```
sqlj buchdb_s_sqlj.sqlj -compile=false
```

Tipp:**Die Option -compile=false beim Aufruf des Precompilers**

Diese Funktion verhindert, dass der Java-Compiler nach dem Precompile-Vorgang automatisch aufgerufen wird.

Dies ist sehr wichtig, da es vorkommen kann, dass ohne die Option der Aufruf des Programms sqlj nicht beendet wird. Dies ist genau dann der Fall, wenn beim Precompilieren Fehler auftreten. Diese werden dann nicht angezeigt, und das Programm bleibt quasi hängen. Sie warten dann vergebens auf eine Rückmeldung.

Die Groß-/Kleinschreibung. Das Programm ist zwar im Dateiverzeichnis unter der Schreibweise BUCHDB_S_SQLJ.SQLJ aufgeführt, ein Aufruf mit dieser Schreibweise führt jedoch zu dem Fehler in Abbildung 11.12.

Der Aufruf mit folgender Schreibweise führt dabei zum Erfolg:

```
sqlj buchdb_s_sqlj.sqlj -compile=false
```

```
O:\Pearson\Projdokul\Buch\CD\Programmfiles\Java>sqlj BUCHDB_S_SQLJ.SQLJ -compile=false
Error: BUCHDB_S_SQLJ.SQLJ ist kein gültiger Eingabedateiname.
Insgesamt 1 Fehler.
```

```
O:\Pearson\Projdokul\Buch\CD\Programmfiles\Java>
```

Abbildung 11.14: Precompile-Fehler durch falsche Schreibweise

Hinweis:

Im Gegensatz zu dem Precompile-Vorgang bei z.B. C müssen Sie nicht mit der Datenbank verbunden sein.

Schritt 2 – Compile

In diesem Schritt wird der Standard-Java-Compiler eingesetzt. In unserem Beispiel setzen wir das entsprechende JDK der DB2 UDB ein. Zum Kompilieren des Programms verwenden wir in unserem Beispiel folgenden Befehl:

```
javac buchdb_s_sqlj.java
```

Der Aufruf des Java-Compilers erzeugt in unserem Fall folgende drei Java-Klassen:

► `BuchDB_S_SQLJ.class`

Die Java-Klasse des eigentlichen Java-Programms

► `buchdb_s_sqlj_SJProfileKeys.class`

Eine spezielle Java-Klasse für die SQL-spezifischen Aufrufe. Diese sorgen dafür, dass die SQL-JDBC-Aufrufe nicht wie normal dynamisch, sondern statisch ablaufen, wenn das entsprechende Profil an die Datenbank gebunden ist.

► `Positioniert.class`

Spezielle Java-Klasse für die SQL-spezifischen Cursor-Aufrufe

Hinweis:

Für ein ablauffähiges Programm benötigen Sie alle vom Compiler erzeugten Java-Klassen.

Schritt 3 – Aufruf des DB2 SQLJ Profile Customizer

Der *DB2 SQLJ Profile Customizer* ist in diesen drei Arbeitsschritten das erste DB2-UDB-spezifisch eingesetzte Entwicklungs-Tool. Er generiert je nach Option aus dem vom SQLJ-Precompiler erstellten standardisierten seriellen Profil ein DB2-UDB-spezifisches Ergebnis (output):

► ein Bind-File

Das dadurch erzeugte Bind-File können Sie mit demselben Bind-Befehl wie z.B. eine C-Funktion an die Datenbank binden. Dies wird dann notwendig, wenn Ihnen z.B. auf dem DB-Zielsystem der DB2 SQLJ Profile Customizer nicht zur Verfügung steht. Der Bind-Befehl hingegen steht Ihnen in einem DB2-UDB-System jederzeit zur Verfügung.

► den DB2-UDB-Zugriffsplan in der Datenbank

Das heißt, Sie können die Bind-Funktion ohne den Umweg, ein Bind-File zu erzeugen, direkt auf das vom SQLJ-Precompiler erzeugte serielle Profil anwenden.

In unserem Beispiel erstellen wir den Zugriffsplan ohne den Umweg über das Bind-File.

Der Befehl zur Erstellung des DB2-Package (Zugriffsplan) lautet:

```
db2sqljcustomize -url jdbc:db2://%COMPUTERNAME%:50000/BUCHDB  
-user DB2ADMIN -password db2admin buchdb_s_sqlj_SJProfile0
```

Abbildung 11.15 zeigt die vollständige Befehlssyntax des DB2 SQLJ Profile Customizer.

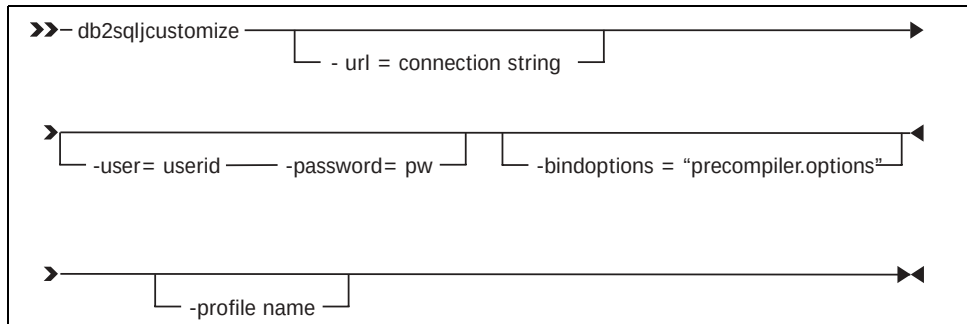


Abbildung 11.15: Syntaxdiagramm des DB2 SQLJ Profile Customizer

Hinweis:

Zur Durchführung der Schritte 1 bis 3 steht Ihnen auf der CD das Batch-File `bcljava.bat` zur Verfügung.

Aufruf des Java-Programms

Nachdem Sie die Schritte 1 bis 3 erfolgreich durchgeführt haben, können Sie die Java-Klasse `BuchDB_S_SQLJ` wie folgt ausführen:

```
java BuchDB_S_SQLJ
```

Tipp:

Bitte achten Sie auf die exakte Schreibweise. Sollten Sie dies nicht berücksichtigen, erhalten Sie die Fehlermeldung aus Abbildung 11.16.

```

Exception in thread "main" java.lang.NoClassDefFoundError: BuchDB_S_SQLj (wrong name: BuchDB_S_SQLJ)
    at java.lang.ClassLoader.defineClass0(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:694)
    at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:133)
    at java.net.URLClassLoader.defineClass(URLClassLoader.java:319)
    at java.net.URLClassLoader.access$400(URLClassLoader.java:92)
    at java.net.URLClassLoader$ClassFinder.run(URLClassLoader.java:677)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:238)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:513)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:441)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:445)
  
```

Abbildung 11.16: Java-Runtime-Fehlermeldung

11.4.7 Zusammenfassung der Abläufe

Um Ihnen eine zusammenfassende Übersicht über die verschiedenen Arbeitsschritte zu geben, die bei der Generierung von statischen SQL-Programmen auftreten können, haben wir Ihnen die Abbildung 11.17 zusammengestellt.

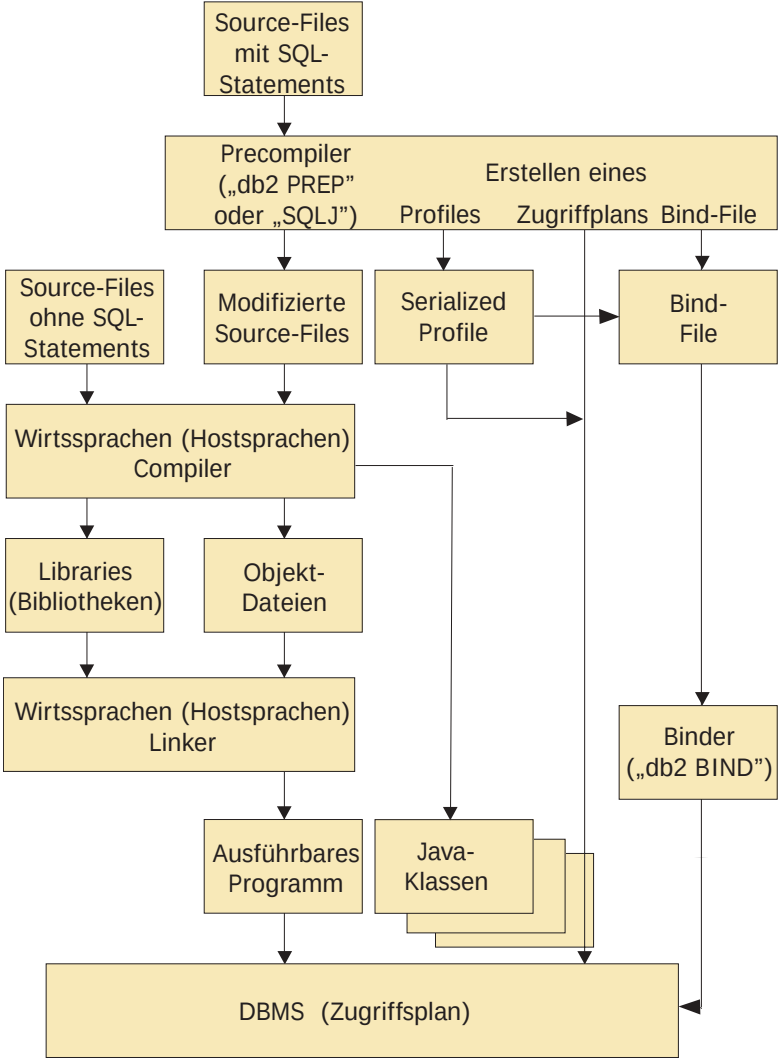


Abbildung 11.17: Übersicht über die Abläufe bei der Verwendung von statischem SQL

11.4.8 Auswirkungen der Programmierung mit statischem SQL

Wie wir schon erwähnt haben, muss das DBMS die SQL-Anweisung in einen Zugriffsplan umsetzen und ihn zuvor bearbeiten (siehe Kapitel 15.1 bzw. Abschnitt 11.3 „Eigenheiten der Abfragesprache SQL“). Inzwischen ist klar, dass diese Bear-

beitung Zeit kostet. Um jedoch diese Zeit nicht während der Programmaufzeit in Anspruch zu nehmen, was dem Anwendungsbenutzer unnötige Wartezeiten beschere würde, ist es sinnvoll, möglichst viele Arbeiten vor die eigentliche Ausführung der SQL-Anweisung zu verlagern.

Des Weiteren kommt noch eine Aufgabe hinzu, die in diesem Zusammenhang oft stark unterschätzt wird. Wie Sie in Abbildung 11.18 sehen, wird zwischen der Anwendung und dem DBMS ein Speicherbereich aufgebaut, der zum Austausch der Daten dient.

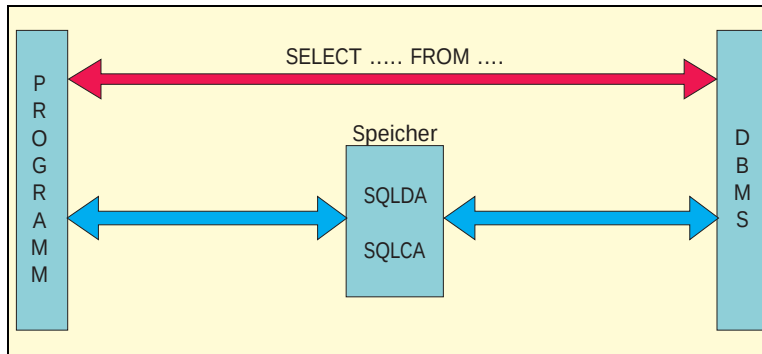


Abbildung 11.18: Austausch von Daten zwischen Anwendung und DBMS

Dieser Speicherbereich teilt sich in zwei Kategorien auf:

► **SQLCA**

Das ist die *SQL-control area*, die Kontroll-, Warn- und Fehlerhinweise beinhaltet. Es handelt sich hierbei um eine fest vordefinierte Speicherstruktur.

► **SQLDA**

Bei diesem Speicherbereich handelt es sich um eine Struktur mit einem kleinen fixen und einem größeren variablen Teil:

- Der feste Teil enthält Informationen, wie z.B. die Größe des Gesamtspeicherbereichs und die Anzahl der zu Verfügung gestellten Host-Variablen.
- Der variable Teil ist der weitaus größere Teil. Er beinhaltet die Beschreibung der zur Verfügung gestellten Host-Variablen:
 - Typ der Variable, z.B. Char oder Integer
 - Länge der Variable
 - Speicheradresse (pointer) auf den eigentlichen Speicherbereich der Variable
 - Speicheradresse (pointer) auf die Indikator-Variable der Host-Variable

Dieser Speicherbereich ist je nach Attributliste unterschiedlich groß und muss für jede SQL-Anweisung gesondert zur Verfügung gestellt werden.

Insgesamt teilen sich die Arbeiten in folgende drei Kategorien auf:

- ▶ Überprüfung des SQL-Statements (Semantik-, Syntax-, Sicherheits- und Verfügbarkeitsprüfung)
Diese Überprüfung kann immer dann ausgeführt werden, sobald alle Objekte und Attribute der Query bekannt sind.
- ▶ Optimieren und Erstellen des Zugriffsplans
- ▶ Aufbereitung des Speicherbereichs zwischen Programm und DBMS

Alle drei Aufgabenbereiche werden beim statischen SQL vor dem eigentlichen Programmablauf ausgeführt.

Dabei übernimmt der Precompiler folgende Aufgaben:

- ▶ den statischen Aufbau der SQLDA, indem er die im Wirtsprogramm deklarierten Host-Variablen heranzieht
- ▶ die Semantik- und Syntax-Prüfung der SQL-Statements
- ▶ das Zur-Verfügung-Stellen dieser Informationen für den Bind-Vorgang, entweder temporär oder in einem Bind-File.

Der Bind-Prozess ergänzt den Precompiler dann um folgende Funktionen:

- ▶ Bildung des Zugriffsplans
- ▶ Überprüfung, ob derjenige, der das Programm bindet, sämtliche Berechtigungen für die im Wirtsprogramm aufgeführten SQL-Befehle besitzt. Somit benötigt derjenige, der das ablauffähige Programm ausführt, nur noch die EXECUTE-Berechtigung auf das Package (Zugriffsplan).

Wenn wir das noch einmal zusammenfassen, ergeben sich drei hauptsächliche Vorteile in den folgenden Segmenten:

- ▶ *Sicherheit*
Während des Programmablaufs werden außer der CONNECT- und EXECUTE-Berechtigung keinerlei explizite Berechtigungen benötigt. Dies bietet einen optimalen Schutz der Daten vor unbefugtem Zugriff.

- ▶ *Performance*

Dieser Vorteil entsteht durch folgende Punkte:

- Der Zugriffsplan steht fertig in der Datenbank zur Verfügung.
- Der komplette Speicherbereich für den Datenaustausch zwischen DBMS und Programm muss nicht für jede Ausführung eines SQL-Befehls jedes Mal neu aufgebaut werden.
- Der Datenaustausch zwischen dem Anwendungsprogramm und dem DBMS wird drastisch minimiert. Dies hat zwei Gründe:
 - Da das SQL-Statement schon im Zugriffsplan der Datenbank vorliegt, muss das Anwendungsprogramm zur Laufzeit nicht das komplette SQL an das DBMS übergeben. Es übergibt nur den Namen des Pakets und die Nummer der SQL-Anweisung.

- Es ist keinerlei Datenaustausch zwischen dem Anwendungsprogramm und dem DBMS zur Bestimmung der in einer SELECT-Anweisung befindlichen Spalten notwendig. Bei dynamischem SQL ist dies erforderlich, da die Anzahl und Datentypen der Spalten erst zur Laufzeit bekannt sind. Bei dynamischem SQL wird also zusätzlicher Datentransfer zwischen DBMS und Anwendungsprogramm erzeugt.

► *Administration*

Durch die Nutzung von statischem SQL liegt der Ablaufplan einer jeden statischen SQL-Anweisung schon vor dem Programmablauf vor. Somit können alle diese SQL-Statements noch vor dem Auftreten eines Problems auf performance-kritische Aspekte hin untersucht werden, wodurch sich viele Mängel vorbeugend beheben lassen.

In der Praxis stoßen wir immer wieder auf die Meinung, dass dies alles keine großen Auswirkungen auf das Gesamtsystem habe und vernachlässigt werden könne. Leider lehrte uns die Praxis ebenfalls unzählige Male das Gegenteil. Sicherlich kann man nicht sagen, dass statisches SQL immer da, wo es möglich ist, auch erforderlich ist. Die Regeln, die sich für uns im Laufe der Jahre herausgebildet haben, lauten:

- Batch-Prozesse zum Laden von großen Datenmengen, in denen auch komplexe SELECT-Anweisungen vorkommen, sollten wenn möglich mit statischem SQL realisiert werden.
- In den meisten *Stored Procedures* ist kein dynamisches SQL notwendig und sollte somit nicht verwendet werden.
- Sollten Sie bei einer Umstellung von einem anderen DBMS auf DB2 auf *Stored Procedures* stoßen, ist zu prüfen, ob diese mit dem bisher genutzten DBMS nicht statisch ausgeführt wurden. Wenn ja, müssen Sie, um keinen Performance-Nachteil zu erleiden, diese weiterhin mit statischem SQL realisieren. Ob in dem speziellen Fall dann überhaupt noch eine *Stored Procedure* sinnvoll ist, ist ebenfalls zu prüfen.
- Welche Anforderungen werden in puncto Sicherheit an Ihr Datensystem gestellt? Kann diese Sicherheit durch die Nutzung von statischem SQL besser, einfacher oder überhaupt realisiert werden?
- Mit welchen Datenmengen und welcher Anzahl von Benutzern müssen Sie für die jeweilige Anwendung rechnen? Sind eventuelle Performance-Engpässe zu erwarten, spielt die Ausnutzung jeder kleinsten Möglichkeit zur Performance-Steigerung eine Rolle?
- Die Nutzung von statischem SQL führt bei den meisten Anwendungsentwicklern oft zu einem wesentlich bewussteren Nutzen des Datenbanksystems. Diesen Punkt können wir aus eigener Erfahrung nur unterstreichen.

Sollten Sie erst in der Produktion den Engpass in puncto Performance und Sicherheit entdecken, sind Sie in Ihren Möglichkeiten meist stark eingeschränkt. Ganz sicher können wir Ihnen an dieser Stelle jedoch sagen, dass die meisten Performance-Probleme nicht im Datenbanksystem, sondern in den Anwendungen zu suchen sind. Hier verbergen sich Steigerungsraten, die mit der reinen Administration undenkbar sind.

11.5 Dynamisches SQL

Bis zu der DB2-Version 4.x war es in der Betriebssystemwelt unter MVS und mit SQL/DS unter VSE von Seiten der Datenbankadministration aus oft sogar verboten, in Programmen dynamische SQL-Anweisungen zu verwenden. Auch in den ersten dezentralen Versionen von DB2 unter OS/2 wurden wir in unseren Projekten angehalten, kein dynamisches SQL zu verwenden.

Warum dies so war, ist aus dem vorhergehenden Abschnitt 11.4 „Statisches SQL“ deutlich geworden. Dennoch ist es heutzutage selbstverständlich, dynamisches SQL zu verwenden. Weiterhin gibt es Gründe, die dynamisches SQL rechtfertigen und sogar notwendig machen. Somit wollen wir als Nächstes die Frage klären, wann statisches SQL nicht ausreicht.

11.5.1 Wann muss eine SQL-Anweisung dynamisch sein?

Eine SQL-Anweisung muss dann und nur dann aus Sicht des DBMS dynamisch sein, wenn folgende Gründe vorliegen:

- ▶ Die WHERE-Bedingung liegt erst zur Laufzeit vor.
Dies ist vor allem dann notwendig, wenn der Endbenutzer während der Laufzeit bestimmen kann, nach welchen Kriterien er suchen möchte. Somit kann die WHERE-Bedingung erst zur Laufzeit generiert werden.
- ▶ Die Sortierreihenfolge kann erst zur Laufzeit bestimmt werden.
- ▶ Zum Zeitpunkt der Programmierung kann die FROM-Klausel nicht bestimmt werden.
- ▶ Die komplette SQL-Anweisung kann erst zur Laufzeit generiert werden.

Wenn also einer der oben genannten Gründe vorliegt, *müssen* Sie dynamisches SQL verwenden. Wenn nicht, *können* Sie statisches SQL verwenden.

11.5.2 Welche Arten von dynamischen SQL-Anweisungen gibt es?

Man unterscheidet verschiedene Arten von dynamischem SQL. Ihr Einsatz richtet sich danach, welche der zuvor erwähnten Anforderungen vorliegt:

- ▶ *Dynamisches SQL mit unbekannter Spaltenliste.* Dies liegt immer dann vor, wenn:
 - die komplette Query erst zur Laufzeit bekannt ist. Das ist jedoch sehr selten der Fall, außer Sie schreiben gerade ein Abfragewerkzeug bzw. Query-Tool wie z.B. QMF, SPUFI, Access oder Advanced Query Tool.
 - die Attribute der SELECT-Anweisung erst zur Laufzeit der Anwendung bekannt sind. Auch dieser Fall tritt unserer Meinung nach selten auf. Ein sinnvolles Beispiel aus der Praxis fällt uns hierzu nicht ein, außer der Benutzer kann seine Spaltenliste zur Laufzeit gestalten.

- ▶ *Dynamisches SQL mit bekannter Spaltenliste.* Diese Form trifft genau dann zu, wenn die Spaltenliste zum Programmierzeitpunkt bekannt ist, jedoch folgende Bedingungen erst zur Laufzeit bekannt sind:
 - die ORDER BY-Klausel
 - die WHERE-Bedingung
 - die GROUP-BY-Klausel bzw. die HAVING-Klausel
 - die FROM-KlauselDiese Art des dynamischen SQL hat in der Praxis durchaus eine Berechtigung.
- ▶ Die so genannte *Non-Select*-, also eine *Nicht-Select-Anweisung*. Sie liegt dann vor, wenn die SQL-Anweisung keine SELECT-Anweisung beinhaltet, sondern es sich z.B. um einen INSERT-, UPDATE- oder DELETE-Befehl handelt.

11.5.3 Der Grund für die Unterteilung in unterschiedliche Arten

Warum unterteilt man die dynamischen SQL-Statements in diese unterschiedlichen Arten? Wie wir schon erwähnt haben, muss

- ▶ das DBMS die SQL-Anweisung in einen Zugriffsplan umsetzen
- ▶ vom Anwendungsprogramm der entsprechende Speicherbereich zur Entgegennahme des SQL-Ergebnisses zur Verfügung gestellt werden

Die Non-Select-Anweisung

Bei einer *Nicht-Select-Anweisung* entfallen eigentlich beide Aufgaben. Da eine *Non-Select-Anweisung* keine Ergebnismengen zurückliefert, besteht weder die Notwendigkeit eines Ablaufplans noch die Notwendigkeit, einen Speicherbereich zur Aufnahme der Ergebnismenge zur Verfügung zu stellen. Es handelt sich also um eine performance-unkritische Anweisung.

Select-Anweisung mit bekannter Spaltenliste

Da alle zu lesenden Spalten bekannt sind, ist der Speicherbereich einer vom DBMS übertragenen Spalte zum Programmierzeitpunkt bekannt. Er muss deshalb nicht dynamisch, also zur Laufzeit, aufgebaut werden. Alle benötigten Wirtsvariablen können somit im Programm deklariert und dadurch vom Precompiler verarbeitet werden. Lediglich der Zugriffsplan muss während der Programmlaufzeit vom DBMS bestimmt werden.

Select-Anweisung mit unbekannter Spaltenliste

Hier fallen beide zuvor erwähnten Aufgaben an. Da hier während der Programmlaufzeit am meisten Ressourcen benötigt werden, sollte diese Methode möglichst vermieden werden. Wie wir jedoch schon gesehen haben, wird sie eigentlich nur für Datenbank-Tools benötigt.

11.5.4 Embedded-dynamisches SQL am Beispiel der Wirtssprache C

Wie Sie dynamische SQL-Anweisungen der Kategorie *Non-Select* und *Select mit bekannter Spaltenliste* programmieren können und was Sie dabei beachten müssen, werden wir Ihnen anhand eines Programms mit der Wirtssprache C demonstrieren. Da in Java momentan keine Embedded-Programmiermöglichkeit für dynamisches SQL besteht, können wir für dieses Beispiel die Wirtssprache Java nicht nutzen. Sicherlich steht für dynamisches SQL in Java die JDBC-Schnittstelle zur Verfügung. Diese sieht jedoch von der Verarbeitungslogik für das DBMS immer die Methode des *Select mit unbekannter Spaltenliste* vor.

Das Beispiel (siehe Abbildung 11.19) ist zum besseren Verständnis in zwölf unterschiedliche Abschnitte unterteilt. Wir werden es Schritt für Schritt erläutern.

Abschnitt 1 (Include der SQL-Header-Files)

Include der DB2-Header-Files `sqlca` und `sqllda` für die Deklaration der SQL-spezifischen Deklarationen.

Abschnitt 2 (Variablendeklaration)

Deklaration der Wirtsvariablen. Zusätzlich zu dem statischen SQL-Programm in C haben wir hier zwei weitere Wirtsvariablen deklariert:

- ▶ Die `Select_Anweisung` ist eine Variable, um die dynamische SELECT-Anweisung aufzunehmen.
- ▶ Die Variable `Update_Anweisung` dient zur Aufnahme der dynamischen UPDATE-Anweisung.

Abschnitt 3 (C-Funktion SetCVar)

Ebenso wie beim statischen SQL benötigen wir die Funktion `SetCVar`, um die Indikator-Variablen nach Null-Values abzufragen und die entsprechenden Host-Variablen zu setzen.

Abschnitt 4 (Connect-Anweisung)

Dieser Abschnitt führt den statischen SQL-Befehl `CONNECT` aus. Statt der Konstante `db2admin` für die Parameter `USER` und `PASSWORT` können Sie auch Host-Variablen verwenden. In der alltäglichen Programmierung sollten Sie dies auch vorziehen.

Abschnitt 5 (Prüfen des SQLCODEs)

Hier wird geprüft, ob die Datenbankverbindung durch den `CONNECT`-Befehl fehlerfrei ausgeführt werden konnte. Des Weiteren wird in diesem Abschnitt die Anzahl der Zeilen der Tabelle `Adress` ermittelt und der Host-Variablen `hAnzahl` übergeben.

Abschnitt 6 (Wertzuweisung für die Select_Anweisung)

Zur Laufzeit wird der C-Variablen `Select_Anweisung` ein Wert dynamisch zugewiesen. In diesem Beispiel liegt zwar zum Programmierzeitpunkt der Wert für die SQL-Anweisung vor, dennoch demonstriert diese Anweisung, wie eine SELECT-Anweisung zur Laufzeit dynamisch generiert werden kann.

```

/* ----- */
/* Beispiel Programm für dynamisches SQL
/*
/* (C) COPYRIGHT Lis.Tec GmbH
/* ----- */

#include <stdlib.h>
#include <sql.h>

EXEC SQL include sqlca ;
EXEC SQL include sqllda ;

EXEC SQL BEGIN DECLARE SECTION;
    sqlint32 hadressnr;
    char htitel[30+1];
    char hname[30+1];
    char hvorname[30+1];
    char hgebdat[11+1];
    char hplz[6+1];
    char hort[30+1];
    long hAnzahl;
    short ititel, iname, ivorname, igebdat, iplz, iort;
    char Select_Anweisung[200];
    char Update_Anweisung[100];
EXEC SQL END DECLARE SECTION;

/* Funktion zur Bearbeitung der Indikatorvariable */
/* ----- */
/* Diese Funktion setzt die entsprechende Variable je nach Indikatorvariable auf leer */
/* ----- */
void SetCVar(char *cVar, short iVar)
{ if(iVar < (short)0)
  { cVar[0] = '\0';
    cVar[1] = '\0';
  }
}

/* Hauptprogramm */
/* ----- */
/* Liest die Tabelle Adresse */
/* ----- */

void main( void )
{
    EXEC SQL CONNECT TO BUCHDB USER DB2ADMIN USING db2admin;
    if(SQLCODE == 0) /* Wenn Connect erfolgreich */
    { EXEC SQL
        SELECT COUNT(*) INTO :hAnzahl
        FROM Adresse;
        printf("\n Die Tabelle beinhaltet insgesamt %d Zeilen -- FehlerCode : %d \n", hAnzahl, SQLCODE );

        /* Bilden des Strings Select_Anweisung und Update_Anweisung */
        /* ----- */
        strcpy(Select_Anweisung, "SELECT SubStr(titel,1,6),SubStr(name,1,15),
        Substr(vorname,1,15),gebdat,plz,substr(ort,1,15)");
        strcat(Select_Anweisung, " FROM ADRESSE FOR UPDATE OF titel,ort");

        strcpy(Update_Anweisung, "UPDATE Adresse SET titel = ?, Ort = ? WHERE CURRENT OF DYN_CURSOR");

        EXEC SQL PREPARE STATEMENT FROM :Select_Anweisung;
        EXEC SQL PREPARE ANWEISUNG FROM :Update_Anweisung;

        if(SQLCODE != 0)
        { printf("\n Fehler beim Prepare aufgetreten ");
          exit(1);
        }

        EXEC SQL DECLARE DYN_CURSOR CURSOR WITH HOLD FOR STATEMENT;
        EXEC SQL OPEN DYN_CURSOR;

        if(SQLCODE == 0)
        { /* 12345 123456789012345 123456789012345 12345678901 123456 123456789012345 */
          printf(" \n Titel Vorname Nachname Geb-Datum PLZ Ort");
          printf(" \n =====");
          while( SQLCODE == 0)
          { EXEC SQL
              FETCH DYN_CURSOR
              INTO :htitel :ititel, :hname :iname, :hvorname :ivorname, :hgebdat :igebdat, :hplz :iplz, :hort :iort;
              SetCVar(htitel, ititel); SetCVar(hname, iname); SetCVar(hvorname, ivorname);
              SetCVar(hgebdat, igebdat); SetCVar(hplz, iplz); SetCVar(hort, iort);

              if(SQLCODE == 0)
              { printf(" \n %s %s %s %s %s %s %d", htitel, hvorname, hname, hgebdat, hplz, hort, iort);
                /* Ändern aller Titel in der erster Stelle, falls bestimmter Bedingung vorliegt */
                /* ----- */
                switch (htitel[1])
                { case 'r' : if(htitel[0] == 'F') htitel[0] = 'f'; else htitel[0] = 'F'; break;
                  case 'e' : if(htitel[0] == 'H') htitel[0] = 'h'; else htitel[0] = 'H'; break;
                }
                if ((hvorname[0] == 'K') || (hname[0] == 'K'))
                { if (hort[0] == '\0')
                  { strcpy(hort, "Flensburg");
                    iort = 0;
                  }
                  else iort = -1;
                }
                EXEC SQL EXECUTE ANWEISUNG USING :htitel, :hort :iort;
                EXEC SQL COMMIT;
              }
              /* Ende für Schleife */
          }
          /* Ende wenn open Cursor erfolgreich war */
          else printf("\n Cursor konnte nicht geöffnet werden SQLCODE %ld \n Msg : %s", SQLCODE, sqlca.sqlerrmc);
          /* Ende wenn Connect erfolgreich war */
        }
        EXEC SQL CLOSE DYN_CURSOR;
        EXEC SQL CONNECT RESET;
    }
}

```

Abbildung 11.19: Beispielprogramm für dynamisches SQL

Abschnitt 7 (Wertzuweisung für die Update_Anweisung)

Zur Laufzeit wird der Variablen `Update_Anweisung` ihr Wert dynamisch zugewiesen. Auch in diesem Beispiel steht zum Zeitpunkt der Programmierung der Wert für die UPDATE-Anweisung fest, dennoch demonstriert diese Anweisung, wie eine UPDATE-Anweisung zur Laufzeit dynamisch generiert werden kann. Bitte beachten Sie hier die so genannten *Parametermarker*. Diese werden innerhalb des Strings durch das Zeichen „?“ gekennzeichnet. *Parametermarker* sind immer dann notwendig, wenn zum Zeitpunkt der Zuweisung an die Variable, die die SQL-Anweisung ausdrückt, unbekannte Größen vorhanden sind, die sich während der Ausführung ändern können. In unserem Beispiel sind die Werte für die Wirtsvariablen `htitel` und `hort` zum Zeitpunkt der Zuweisung der UPDATE-Anweisung nicht bekannt. Die Parametermarker sind also nichts anderes als Platzhalter für Host-Variablen, die sich während der Ausführung ändern können. Die Verwendung der Parametermarker hat den Vorteil, dass die eigentliche SQL-Anweisung nicht ständig neu generiert werden muss. Dies hat zur Folge, dass nach der ersten Bekanntmachung der SQL-Anweisung an das DBMS diese Anweisung während der Lebensdauer der Variable quasi statisch ist. Somit muss die PREPARE-Anweisung in Abschnitt 8 nur einmal ausgeführt werden.

Abschnitt 8 (Prepare)

Diese SQL-Precompiler-Anweisung sorgt dafür, dass zur Laufzeit an dieser Stelle ein dynamischer Prepare- und Bind-Vorgang ausgeführt wird. Die Anweisung `EXEC SQL PREPARE` entspricht quasi den beiden Anweisungen `PREPARE` und `BIND`, die beim statischen SQL vor dem Programmablauf stattfinden. `STATEMENT` und `ANWEISUNG` sind so genannte Labels, die für den weiteren Verlauf benötigt werden. Sie stellen die Verbindung zu der entsprechend dynamisch gebildeten SQL-Anweisung her.

Abschnitt 9 (Deklaration und Öffnen des Cursors)

In diesem Abschnitt wird der Cursor `DYN_CURSOR` für das dynamische `SELECT`-Statement deklariert, das über das Label `STATEMENT` adressiert wird. Die Programmanweisung dazu lautet:

```
EXEC SQL DECLARE DYN_CURSOR CURSOR WITH HOLD FOR STATEMENT;
```

Die folgende Anweisung `EXEC SQL OPEN DYN_CURSOR;` öffnet den Cursor und führt die `SELECT`-Anweisung komplett aus. Danach steht die Ergebnismenge über das DBMS der Anwendung zur Verfügung.

Abschnitt 10 (Fetch Cursor)

Die Anweisung `DYN_CURSOR` positioniert den Zeiger zur Übergabe der Ergebnismenge an das Anwendungsprogramm.

Der erste `FETCH` ist notwendig, da nach der Anweisung `OPEN DYN_CURSOR` zwar die Ergebnismenge komplett im DBMS zur Verfügung steht, der Zeiger zur Parameterübergabe jedoch nicht definiert ist.

Jeder weitere `FETCH` in der `while`-Schleife positioniert den Zeiger (Cursor) neu und verschiebt ihn um eine weitere Ergebniszeile. Wird das Ende der zur Verfügung

gestellten Ergebnismenge erreicht, wird die Variable `SQLCODE` vom DBMS auf den Wert 100 gesetzt.

Hinweis:

Bitte beachten Sie, dass wir in diesem Beispiel nicht alle vorkommenden Fehlermeldungen abfangen. Die Abfrage `SQLCODE == 100` in der `while`-Schleife reicht für eine professionelle Programmierung nicht aus.

Abschnitt 11 (Ausführen der Update-Anweisung)

Dieser Abschnitt führt die in Abschnitt 8 dynamisch gebildete Update-Anweisung aus. Der Befehl dazu lautet:

```
EXEC SQL EXECUTE ANWEISUNG USING :htitel,:hort :iort;
```

Die Anweisung `USING :htitel,:hort :iort` sorgt dafür, dass die Werte für die Parametermarker aus „Abschnitt 7 (Wertzuweisung für die Update_Anweisung)“ mit den Wirtsvariablen `:htitel,:hort :iort` gesetzt werden.

Hinweis:

Bitte beachten Sie, dass zwar im Abschnitt 7 für die Host-Variable `hort` nur ein Parametermarker angegeben wurde, dass hier aber zwei Variablen, und zwar die Host- und die Indikator-Variable für den Ort, mitgegeben werden.

Abschnitt 12 (Programm-Ende)

Die Anweisung `EXEC SQL CLOSE DYN_CURSOR;` sorgt dafür, dass der Cursor geschlossen wird und das DBMS somit den Speicherbereich freigibt, in dem die Ergebnismenge zur Verfügung gestellt wurde.

Der Befehl `EXEC SQL CONNECT RESET` löst die Verbindung zur Datenbank und gibt den kompletten Speicherbereich zwischen dem Anwendungsprogramm und dem DBMS frei.

Das Programm kann danach beendet werden.

11.5.5 Wie kommen Sie zum ausführbaren C-Programm?

Der Vorgang zum Erstellen eines ablauffähigen Programms aus einem dynamischen Embedded-SQL-Programm unterscheidet sich in keiner Weise von der Vorgehensweise bei dem statischen Embedded-SQL-Programm. Auch hier müssen Sie folgende Schritte ausführen:

1. Precompile des SQL-Source-Codes
2. Compile aller C-Programme
3. Linken aller Objekt-Files zu einem ablauffähigen Programm
4. Binden der durch den Prepare-Lauf generierten Bind-Files

Hinweis:

Zur Ausführung dieser Schritte für das Beispielprogramm BUCHDB_D_SQC.SQC steht Ihnen auf der Buch-CD das Batch-File bcld.bat zur Verfügung.

11.5.6 Auswirkung der Programmiermethode Embedded-dynamisches SQL

Die Unterschiede zwischen dem *dynamischen SQL* und dem *Ebedded-statischen SQL* haben wir Ihnen in Abschnitt 11.4.8 „Auswirkungen der Programmierung mit statischem SQL“ aufgezeigt. An dieser Stelle werden wir erläutern, welche Vorteile Ihnen das *Embedded-dynamische SQL* gegenüber den Nicht-Embedded-Methoden bietet.

Im Gegensatz zu den *Nicht-Embedded-Schnittstellen* nutzt die Programmiermethode *Embedded-dynamisches SQL* die Eigenschaften des Precompilers. Somit können alle Embedded-SQL-Informationen, die schon zum Programmierzeitpunkt vorliegen, von ihm verarbeitet und in einem Bind-File zur Verfügung gestellt werden. Diese Informationen können dann vom Bind-Prozess dem DBMS vor dem Programmablauf übergeben werden. Da die Programmierart *dynamisches SQL mit unbekannter Spaltenliste* in der herkömmlichen Datenbankprogrammierung so gut wie nie benötigt wird, ist es sinnvoll, alle zum Programmierzeitpunkt bekannten Größen dem DBMS vor dem eigentlichen Ablauf des Anwendungsprogramms zur Verfügung zu stellen. Dies gilt auch dann, wenn es sich nur um die verwendete Spaltenliste handelt und damit der dynamische Aufbau des Speicherbereichs während der Programmlaufzeit vermieden werden kann. Sie sollten nicht vergessen, dass die Übermittlung des SQL-Statements zur Laufzeit zusätzlichen Zeitaufwand mit sich bringt, der durch drei Bereiche zustande kommt:

- ▶ Dynamischer Aufbau des Speicherbereichs zwischen Programm und DBMS
- ▶ Zusätzlicher Datenaustausch zwischen Anwendungsprogramm und DBMS zur Übermittlung der in der SELECT-Anweisung vorkommenden Spalten und deren Datentypen
- ▶ Das komplette SQL-Statement muss vom Anwendungsprogramm zur Laufzeit an das DBMS übermittelt werden. Somit kann das Anwendungsprogramm, nicht, wie in der Embedded-Programmiermethode, nur den Namen des Zugriffspfad und die Nummer der SQL-Anweisung übergeben. Bei komplexen SQL-Befehlen kann dies schon eine nicht unbeträchtliche Menge an Daten sein.

Da diese Aufgaben bei jedem Sendevorgang des dynamischen SQL von der Anwendungsapplikation an das DBMS entstehen, kann das in einer Systemumgebung mit vielen Benutzern und vielen SQL-Befehlen zu einer nicht unerheblichen Mehrbelastung in Sachen Netzwerkverkehr führen. Sollten sich dann noch der Datenbankserver z.B. unter dem Betriebssystem OS/400 oder zOS und Ihr Anwendungsprogramm auf einer dezentralen Maschine befinden, werden Sie sicherlich in der Lage sein, diese Nachteile live zu erleben. Hinzu kommt, dass zwischen den Betriebssystemwelten zOS bzw. OS/400 und der Welt der dezentralen Betriebssysteme ein zusätzlicher Netzwerkprotokoll-Overhead entsteht.

11.6 Die weiteren Nicht-Embedded-SQL-Schnittstellen

Für die Verwendung von *Nicht-Embedded-SQL-Schnittstellen* stehen Ihnen (wie in Abbildung 11.20 gezeigt) mehrere SQL-Schnittstellen zur Verfügung. Die Abbildung 11.20 zeigt Ihnen auch, welche Schnittstelle Sie, je nach der von Ihnen verwendeten Programmiersprache, einsetzen können bzw. müssen. Im Gegensatz zur Programmiermethode *Embedded-SQL* wird bei dieser Programmierart kein Precompiler benötigt. Das kommt daher, dass Ihnen die jeweiligen *Nicht-Embedded-SQL-Schnittstellen* die entsprechenden SQL-Funktionalitäten über Funktionen bzw. Klassen, die an die Bedürfnisse der jeweiligen Programmiersprache angepasst sind, zur Verfügung stellen. Die Verwendung dieser Schnittstellen soll eine Unabhängigkeit vom jeweils verwendeten DBMS ermöglichen. Dies ist jedoch, je nach verwendetem DBMS und den in den Datenbankapplikationen verwendeten Datenbankobjekten (wie z.B. Stored Procedures, Triggern und UDFs) oft sowieso nicht gegeben.

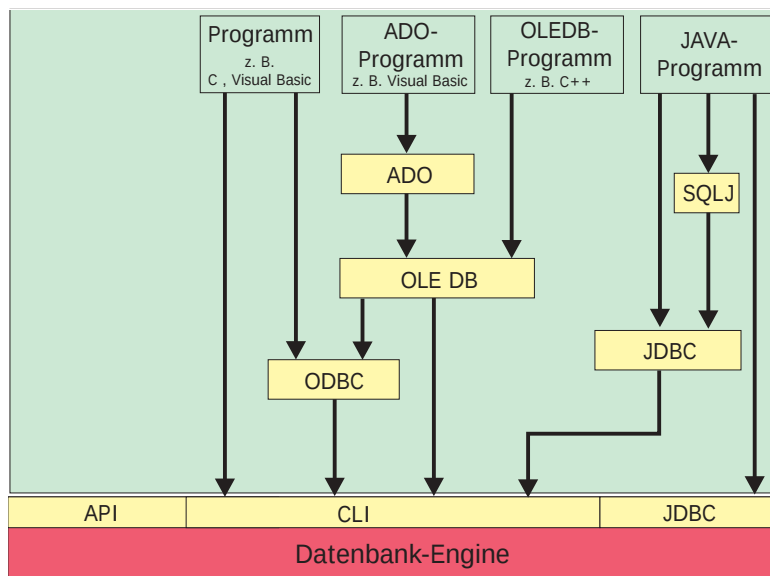


Abbildung 11.20: Nicht-Embedded-Datenbankprogramm-Schnittstellen

Was sich hinter den Abkürzungen der *Nicht-Embedded-SQL-Schnittstellen* verbirgt, möchten wir Ihnen jetzt kurz erläutern.

11.6.1 Die ODBC-Schnittstelle

Die ODBC-Schnittstelle (Open Database Connectivity) war die erste genormte *Nicht-Embedded-SQL-Schnittstelle*. Diese Schnittstelle stellt die Funktionalität des SQL-Standards über C-Funktionen zur Verfügung. Da in der Programmiersprache C die Flexibilität einen hohen Stellenwert besitzt und eine Verknüpfung von Daten und Funktionen nicht einfach zu realisieren ist, wurde auch diese Schnittstelle an diese Bedürfnisse angepasst. Genau diese Flexibilität (wie z.B. die Verwendung von

Pointern und das dynamische Allokieren von Speicher für jede benötigte Variable) macht die Nutzung dieser Schnittstelle nicht gerade einfach. Somit wundert es nicht, dass viele Nicht-C-Programmierer die ODBC-Schnittstelle als unnötig kompliziert bezeichnen.

11.6.2 Die CLI-Schnittstelle

Die ODBC-Schnittstelle ist quasi eine Untermenge der Schnittstelle CLI (Call Level Interface). Im Gegensatz zu Microsoft, das eine direkte ODBC-Schnittstelle in seinen DBMS-Produkten implementiert hat, unterstützten IBM sowie andere Datenbankhersteller die ODBC-Schnittstelle nur über so genannte ODBC-Treiber. Diese Treiber sind notwendig, um die entsprechenden genormten ODBC-Aufrufe in die DBMS-spezifischen API-Schnittstellen umzusetzen. Diese Umsetzung bringt natürlich Performance-Einbußen mit sich. Seit der Version DB2 UDB 2 unterstützt IBM die direkte CLI-Schnittstelle. Dies ist eine im Vergleich zur ODBC-Schnittstelle um zusätzliche Funktionalitäten erweiterte Standardschnittstelle. Da diese Schnittstelle direkt vom DBMS der DB2 UDB unterstützt wird, benötigen Sie bei der Nutzung dieser Schnittstelle keine zusätzlichen Treiber.

Tipp:

Da die ODBC-Schnittstelle eine Untermenge der CLI-Schnittstelle ist, können Sie auch ODBC-Programme direkt auf der CLI-Schnittstelle aufsetzen. Sie müssen nur dafür sorgen, dass der entsprechende ODBC-Treiber nicht benutzt wird.

11.6.3 Die OLE DB-Schnittstelle

Durch den Einsatz objektorientierter Programmiersprachen wurde eine standardisierte Schnittstelle zum Austausch von Daten zwischen verschiedenen Programmen wie z.B. Textverarbeitungssystem, Spreadsheets oder E-Mail-Systemen möglich. Die OLE DB-Schnittstelle setzt im Gegensatz zur ODBC- und CLI-Schnittstelle auf der Objekt-Ebene auf und ist somit auch einfacher zu bedienen und nicht so kompliziert. Die OLE DB-Schnittstelle ist eine C++-Programmierschnittstelle. Seit der Version DB2 UDB 7.1 unterstützt die DB2 UDB eine native (direkte) OLE DB-Schnittstelle. Das bedeutet, die DB2 UDB kann sowohl als OLE DB-Client wie auch als Server fungieren. Bis zu der Version DB2 UDB 7.1 wurde die Unterstützung der OLE DB nur über die ODBC- bzw. CLI-Schnittstelle gewährleistet.

11.6.4 Die ADO-Schnittstelle

Da es sich, wie gerade erwähnt, bei der OLE DB um eine für C++ optimierte Schnittstelle handelt, ist die Nutzung dieser Schnittstelle für andere Programmiersprachen wie z.B. Visual Basic nicht immer ganz einfach. Aus diesem Grund hat Microsoft mit der ADO-Schnittstelle (ActiveX Data Objects) eine Programmierschnittstelle geschaffen, die auf einem höheren Level aufsetzt. ADO ist eine Untermenge der *Component Object Model*-Objekte (COM). Die COM-Objekte werden inzwischen von allen Microsoft-Programmiersprachen unterstützt.

11.6.5 Die JDBC-Schnittstelle

Ebenso wie z.B. die Programmiersprache C besitzt die Programmiersprache Java eine auf die Bedürfnisse dieser Sprache und die funktionalen Anforderungen an diese Sprache zugeschnittene SQL-Schnittstelle: die JDBC-Schnittstelle (Java Database Connectivity). Um die Nachteile, die eine rein dynamische SQL-Programmierung mit sich bringt, zu umgehen, wird seit kurzem die statische SQLJ-Schnittstelle zur Verfügung gestellt, deren Compile-Output auf der JDBC-Schnittstelle aufsetzt. Leider werden bis heute nicht alle Funktionalitäten der Embedded-Programmierschnittstelle unterstützt, die Ihnen z.B. in den Programmiersprachen C und Cobol zur Verfügung stehen.

11.6.6 Zusammenfassung

Da wir uns in diesem Buch nicht ausschließlich mit der Datenbankprogrammierung, sondern mit der Administration der Datenbank beschäftigen, war es unsere Absicht, Ihnen in diesem Kapitel einen Überblick über die Programmiermöglichkeiten und verschiedenen Methoden zu geben, die Ihnen zur Verfügung stehen.

An dieser Stelle möchten wir einen wichtigen Hinweis geben. Es handelt sich – bis auf die SQLJ-Schnittstelle – bei allen im Abschnitt 1.6 aufgeführten SQL-Schnittstellen um ausschließlich dynamische SQL-Schnittstellen, und zwar von der Art *mit unbekannter Spaltenliste*. Somit müssen Sie bei der Verwendung einer dieser Schnittstellen mit den entsprechenden Nachteilen rechnen, die diese Programmiermethode mit sich bringt (siehe Abschnitt 11.4.8 „Auswirkungen der Programmierung mit statischem SQL“ und Abschnitt 11.5.6 „Auswirkung der Programmiermethode Embedded-dynamisches SQL“). Denn alle Vorgänge, wie z.B. das Precompilieren und Binden der SQL-Statements sowie der Aufbau des Speicherbereichs für den Datenaustausch der SQL-Ergebnisse, werden bei der Nutzung dieser Schnittstellen zur Laufzeit ausgeführt.

11.7 Sicherheit

Als Abschluss dieses Kapitels fassen wir in diesem Abschnitt noch einmal die speziellen und komplexen Eigenschaften zusammen, die sich in Sachen Sicherheit durch die Komponente des statischen Embedded-SQL ergeben.

Im Anschluss an Abbildung 11.21 werden wir die verschiedenen Szenarien exakt an einigen Beispielen beschreiben. Diese sollen Ihnen die Möglichkeit geben, die Eigenheiten dieser Programmiermethode in puncto Sicherheit besser zu verstehen. In der Praxis müssen wir leider viel zu oft feststellen, dass die Vorteile dieser Eigenschaften deshalb nicht genutzt werden bzw. auch oft zu Fehlern führen, weil sie oft unzureichend bekannt sind.

Voraussetzungen zum Nachvollziehen des Beispiels sind:

- Die DB2 UDB V8 ist korrekt installiert.
- Sie befinden sich bei der Ausführung der Befehle in einem DB2-Befehlsfenster.

- Sie haben die Datenbank BUCHDB mit dem auf der CD befindlichen Batch-File CDATENB.BAT unter dem Benutzer db2admin mit dem Kennwort db2admin in einem DB2-Befehlsfenster erzeugt.
- Auf Ihrem System sind folgende lokalen Benutzer mit lokaler Administrationsberechtigung eingerichtet:
 - db2admin mit dem Passwort db2admin
 - Georg mit einem beliebigen Kennwort
 - Carola mit einem beliebigen Kennwort

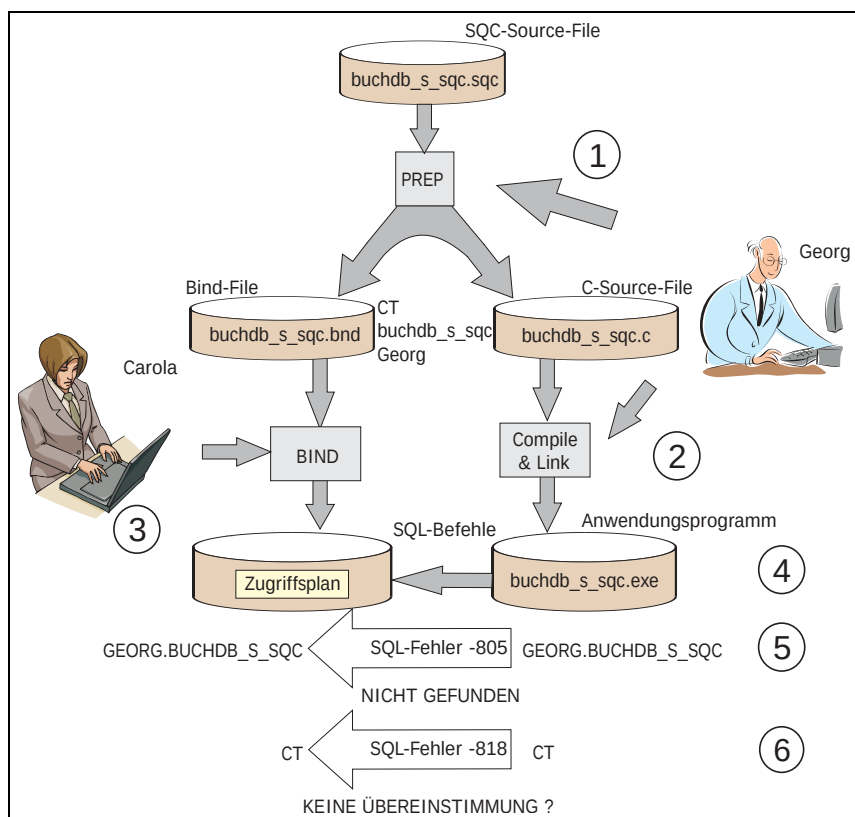


Abbildung 11.21: Sicherheit bei der Verwendung von Embedded-statischem SQL

Schritt 1

Georg programmierte das *Embedded-SQL-Programm* `BuchDB_S_SQC.SQC` und führt in diesem Schritt den Befehl `db2 prep buchdb_s_sqc bindfile` aus. Dazu muss Georg mit seiner Benutzerkennung mit der Datenbank verbunden sein. Da der Benutzer `db2admin` und nicht Georg die Tabellen angelegt hat, besitzen die Tabellenobjekte das Schema `db2admin`. Dies führt dazu, dass der Precompiler die Warnmeldungen in Abbildung 11.22 produziert. Diese können jedoch an dieser Stelle ignoriert wer-

den, da die Tabellenobjekte erst zum Zeitpunkt des Bindens vorhanden sein müssen. Des Weiteren hat der Precompile-Lauf folgende Auswirkungen:

- Das generierte C-Programm erhält die Information, dass das ablauffähige Programm BuchDB_S_SQC.EXE beim Ausführen der jeweiligen SQL-Befehle auf das Zugriffspaket (package) mit dem Namen GEORG.BUCHDB_S zugreifen wird. Der Schema-Name wird also GEORG sein.
- Das Bind-File erhält die Information, dass der Name des Zugriffspakets den Schema-Namen GEORG erhält. Der Name des später durch den Bind generierten Packages wird dann GEORG.BUCHDB_S lauten.

```
ZEILE  NACHRICHTEN FÜR buchdb_s_sqc.sqc
-----
SQL0060W  Der Precompiler "C" wird ausgeführt.
25  SQL0204N  "GEORG.ADRESSE" ist ein nicht definierter Name.
      SQLSTATE= 42704
51  SQL0204N  "GEORG.ADRESSE" ist ein nicht definierter Name.
      SQLSTATE= 42704
84  SQL0204N  "GEORG.ADRESSE" ist ein nicht definierter Name.
      SQLSTATE= 42704
SQL0091W  Vorkompilieren oder Binden mit "0" Fehler(n) und
      "3" Warnung(en) beendet.
```

Abbildung 11.22: Warnhinweise des Precompilers

Schritt 2

Georg kompiliert und linkt das C-Programm BuchDB_S_SQC.C zu dem ablauffähigen Programm BuchDB_S_SQC.EXE.

Schritt 3

Carola bindet das in Schritt 1 generierte Bind-File BuchDB_S_SQC.BND mit dem Befehl db2 bind buchdb_s_sqc.bnd. Carola muss dazu mit ihrer Benutzerkennung zur Datenbank verbunden sein. Falls Carola nicht über ausreichende Berechtigungen zum Binden des Programms verfügt, bekommt sie, je nach nicht vorhandener Berechtigung, eine entsprechende Fehlermeldung (siehe Abbildung 11.23).

```
ZEILE  NACHRICHTEN FÜR buchdb_s_sqc.bnd
-----
SQL0061W  Der Binder wird ausgeführt.
SQL0551N  "CAROLA" verfügt nicht über die Berechtigung, die
      Operation "BIND" für das Objekt "GEORG.BUCHDB_S" auszuführen.
      SQLSTATE= 42501
SQL0082C  Die Verarbeitung wurde aufgrund eines Fehlers
      abgebrochen.
SQL0091N  Bindeprozess mit "2" Fehler(n) und "0" Warnung(en)
      beendet.
```

Abbildung 11.23: Bind-Fehler bei nicht korrekter Berechtigung

Carola benötigt folgende Rechte, um das Bind-File korrekt an die Datenbank zu binden:

- ▶ CONNECT-Berechtigung zur Datenbank BUCHDB
- ▶ Beim erstmaligen Bind-Vorgang die Berechtigung BINDADD und bei jedem folgenden Bind-Prozess die BIND-Berechtigung
- ▶ Alle INSERT-, UPDATE-, DELETE- und SELECT-Berechtigungen auf die entsprechenden Tabellenobjekte, die in den SQL-Befehlen des Programms angesprochen werden.
- ▶ Sollten, wie in unserem Beispiel, die Tabellenobjekte unter dem Benutzer db2admin angelegt worden sein, erhält Carola die Fehlermeldung aus Abbildung 11.24. Wir haben in unserem Beispiel, um dem Benutzer Carola nicht alle Rechte (INSERT-, UPDATE-, DELETE- und SELECT-Berechtigung auf die Tabellenobjekte) einzeln geben zu müssen, den Benutzer mit den lokalen Administrationsrechten ausgestattet.

```
ZEILE  NACHRICHTEN FÜR buchdb_s_sqc.bnd
```

```
-----
SQL0061W  Der Binder wird ausgeführt.
25  SQL0204N  "CAROLA.ADRESSE" ist ein nicht definierter Name.
      SQLSTATE=42704
51  SQL0204N  "CAROLA.ADRESSE" ist ein nicht definierter Name.
      SQLSTATE=42704
84  SQL0204N  "CAROLA.ADRESSE" ist ein nicht definierter Name.
      SQLSTATE=42704
SQL0082C  Die Verarbeitung wurde aufgrund eines Fehlers
          abgebrochen.
SQL0092N  Aufgrund vorheriger Fehler wurde kein Paket
          erstellt.
SQL0091N  Bindeprozess mit "5" Fehler(n) und "0" Warnung(en)
          beendet.
```

Abbildung 11.24: Bind-Fehler bei nicht vorhandenen Tabellenobjekten

Wenn Sie sich mit dem Benutzernamen Carola mit der Datenbank BUCHDB verbinden, können Sie zum Anlegen und Füllen der Tabellenobjekte mit dem Schema Carola folgende Batch-Files benutzen C_TABS und I_TABS.

Hinweis:

Sie finden diese Batch-Files auf der Buch-CD:

- ▶ C_TABS zum Anlegen der Tabellen
- ▶ I_TABS zum Füllen der Tabellen

Danach können Sie den Bind-Befehl `db2 bind buchdb_s_sqc.bnd` erneut und diesmal erfolgreich ausführen.

Schritt 4

In diesem Schritt wird das Programm BuchDB_S_SQC.EXE fehlerfrei ausgeführt.

Schritt 5

Wird das entsprechende Zugriffspaket (package) nicht zu dem Zeitpunkt gefunden, zu dem der SQL-Befehl ausgeführt wird, erhalten Sie die Fehlermeldung aus Abbildung 11.25.

Die Tabelle beinhaltet insgesamt 0 Zeilen -- FehlerCode : -805

Cursor konnte nicht geöffnet werden SQLCODE -805

Msg : GEORG.BUCHDB_S 0X334147704E444C54

SQLRACFI~

Abbildung 11.25: Zugriffsplan nicht gefunden: SQL-Fehler -805

Dieser Fehler tritt z.B. dann auf, wenn das vom Programm aufgerufene Package nicht denselben Schema-Namen wie das Zugriffspaket (package) in der Datenbank besitzt.

Schritt 6

In diesem Schritt entspricht die Aktualität des Zugriffspakets nicht der des Programms. Die *Consistency Token* der beiden Module *ablauffähiges Programm* und *Zugriffspaket* stimmen nicht überein. Die Fehlermeldung, die die DB2 UDB dabei erzeugt, sehen Sie in Abbildung 11.26.

Die Tabelle beinhaltet insgesamt 0 Zeilen -- FehlerCode : -818

Cursor konnte nicht geöffnet werden SQLCODE -818

Msg :

SQLRALDP~

Abbildung 11.26: Consistency-Token-Fehler (Timestamp Error): SQL-Fehler -818

11.7.1 Zugriffspakete der DB2 UDB

Sie können sich alle Zugriffspackages, die in einer Datenbank vorliegen, mit dem Befehl `db2 list packages for all` anzeigen lassen. Wenn Sie die Option `for all` nicht mit angeben, werden nur die Packages für den gerade angemeldeten Benutzer angezeigt. Die Zugriffspakete für das bestimmte Schema `db2admin` können Sie sich mit dem Befehl `db2 list packages for schema db2admin` anzeigen lassen. Abbildung 11.27 zeigt Ihnen, wie ein entsprechender Output des Befehls `list packages...` aussehen kann. Bitte beachten Sie, dass das Package mit der Nummer 1 und das Zugriffspaket mit der Nummer 2 nicht zusammen vorkommen können. Sie besitzen beide denselben Namen und dasselbe Schema. Nur der Benutzer, der den Bind ausgeführt hat, ist ein anderer. Er ist bei Nummer 1 der Benutzer `Carola` und bei Nummer 2 der Benutzer `db2admin`.

db2 list packages

Paket	Schema	Version	Gebunden von	Gesamtanzahl der Bereiche	Gültig	Format	Isolat-stufe	Blockung
BUCHDB_S	GEORG		CAROLA	3	Y	0	CS	U ①
BUCHDB_S	GEORG		DB2ADMIN	3	Y	0	CS	U ②
BUCHDB_D	DB2ADMIN		DB2ADMIN	3	Y	0	CS	U ③
BUCHDB_S	DB2ADMIN		DB2ADMIN	3	Y	0	CS	U ④

Abbildung 11.27: DB2-Packages

Tipp:

Wie Sie in Abbildung 11.27 erkennen können, werden nur acht Stellen für den Namen des Packages verwendet. Somit sind Probleme bei Programmnamen, die sich nicht in den ersten acht Stellen unterscheiden, vorprogrammiert.

12

Konkurrierende Verarbeitung

Eine konkurrierende Verarbeitung entsteht immer dann, wenn wie in Abbildung 12.1 mehrere Benutzer (in unserem Beispiel B1 bis B4) gleichzeitig auf ein und denselben Datenbestand zugreifen. Dabei spielt es keine Rolle, ob diese konkurrierenden Zugriffe über dieselbe oder über verschiedene Anwendungen entstehen. Bei dieser parallelen Verarbeitung kann es somit vorkommen, dass mehrere Benutzer/Anwendungen gleichzeitig auf dieselben Datenbankobjekte zugreifen wollen. Es kann sich dabei um folgende Datenbankobjekte handeln:

- ▶ dieselbe Datenbank
- ▶ denselben Tabellenbereich
- ▶ dieselbe Tabelle
- ▶ denselben Datensatz

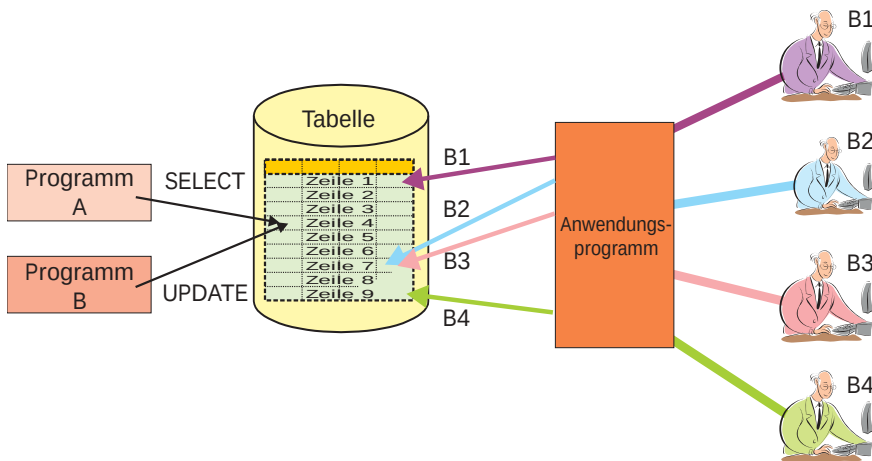


Abbildung 12.1: Parallele Verarbeitung

Je nachdem, in welcher Zugriffsform welches dieser Datenbankobjekte gleichzeitig angefordert wird, können gewisse Konflikte entstehen. Ist z.B. eine Datenbank durch einen exklusiven Zugriff komplett gesperrt worden (exklusiver Connect), wird jeder weitere Connect-Befehl abgewiesen. Ist aber z.B. nur ein Tabellenbereich exklusiv gesperrt, kann durchaus auf andere Tabellenbereiche zugegriffen werden.

Um zu verstehen, wie DB2 mit gleichzeitigen Zugriffen auf ein und dasselbe Datenbankobjekt umgeht und Inkonsistenzen vermeidet, wollen wir uns im nächsten Abschnitt erst einmal damit beschäftigen, welche Anomalien DB2 überhaupt unterscheidet und kennt.

12.1 Welche Problemfälle unterscheidet DB2?

Damit DB2 auf die entsprechenden Anforderungen der parallelen Zugriffe angemessen reagieren kann, muss eine Klassifizierung der entsprechenden Probleme vorgenommen werden.

Die DB2 UDB unterscheidet vier Kategorien von Daten-Anomalien:

- Phantom Reads (PR)
- Nicht wiederholbare Lese-Vorgänge (NWL)
- So genannte Lost Updates (LU)
- Zugriff auf nicht bestätigte (COMMIT) Daten (ZNB)

Damit Sie mit den entsprechenden DB2-Methoden diesen Anomalien gegensteuern können, müssen Sie Ihren jeweiligen Problemfall einer dieser Kategorien zuordnen.

12.1.1 Was sind Phantom Reads (PR)?

Wenn sich die Ergebnismenge einer SELECT-Anweisung innerhalb einer Transaktion so verändert, dass während dieser Transaktion Datensätze zu der Ergebnismenge hinzukommen, spricht man von *Phantom Reads*. Abbildung 12.2 und das folgende, Schritt für Schritt erklärte Beispiel sollen Ihnen helfen, dies besser zu verstehen.

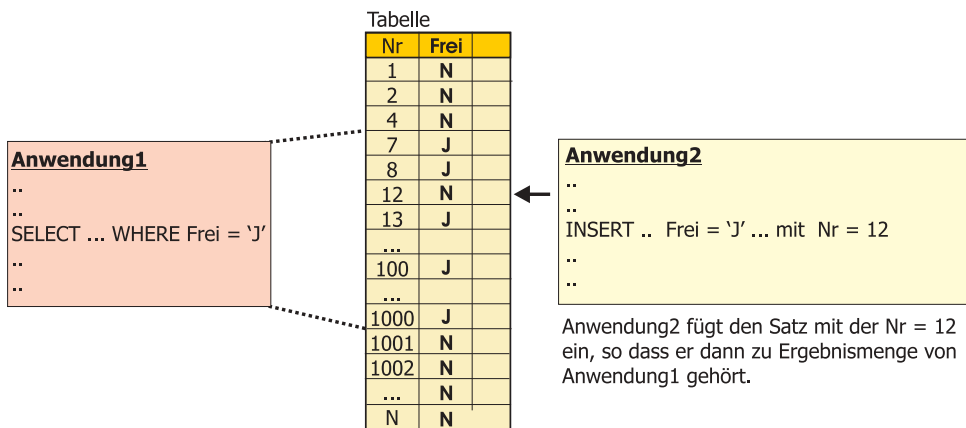


Abbildung 12.2: Phantom Reads (PR)

1. Eine OPEN CURSOR-Anweisung der *Anwendung1* führt den kompletten SELECT-Befehl aus. Somit stellt das RDBMS im Regelfall die komplette Ergebnismenge zur Verfügung.
2. Während der FETCH CURSOR-Anweisungen von *Anwendung1* fügt eine andere Anwendung (*Anwendung2*) Datensätze hinzu, die sich innerhalb des Wertebereichs der SQL-Query befinden, die durch den OPEN CURSOR-Befehl der *Anwendung1* ausgeführt wird. *Anwendung2* bestätigt diese Änderungen mit einem COMMIT. Es entstehen dadurch für die *Anwendung1* so genannte *Phantom-Datensätze*.

- 3. Die Phantom-Datensätze wurden genau an den Stellen in der Ergebnismenge der *Anwendung1* hinzugefügt, die schon von der Anweisung `FETCH CURSOR` der *Anwendung1* gelesen wurden.
- 4. Würde die *Anwendung1* jetzt den Cursor rückwärts positionieren oder die Sätze in derselben Transaktion noch einmal lesen, stünden ihr zusätzliche Datensätze zur Verfügung: *Anwendung1* führt die so genannten „Phantom Reads“ aus.

12.1.2 Was sind nicht wiederholbare Lesevorgänge (NWLV)?

Wenn sich die Ergebnismenge einer `SELECT`-Anweisung innerhalb einer Transaktion so verändert, dass während dieser Transaktion Datensätze von der Ergebnismenge entfernt werden, spricht man von *nicht wiederholbaren Lesevorgängen*. Abbildung 12.3 und das folgende, Schritt für Schritt erklärte Beispiel soll Ihnen helfen, dies besser nachvollziehen zu können:

- 1. Eine `OPEN CURSOR`-Anweisung der *Anwendung1* führt den kompletten `SELECT`-Befehl aus. Somit stellt das RDBMS im Regelfall die komplette Ergebnismenge zur Verfügung.
- 2. Während der `FETCH CURSOR`-Anweisungen von *Anwendung1* ändert eine *Anwendung2* die Datensätze so, dass diese nicht mehr dem Wertebereich der durch den `OPEN CURSOR`-Befehl von *Anwendung1* ausgeführten `SQL`-Query entsprechen. *Anwendung2* bestätigt die Änderungen durch einen `COMMIT`. Es entstehen dadurch für die *Anwendung1* so genannte *nicht wiederholbare Lesevorgänge*.
- 3. Würde die *Anwendung1* jetzt den Cursor rückwärts positionieren oder die Sätze in derselben Transaktion noch einmal lesen, stünden ihr nicht mehr alle zuvor gelesenen Datensätze zur Verfügung: *Anwendung1* führt die so genannten `NWL` aus.

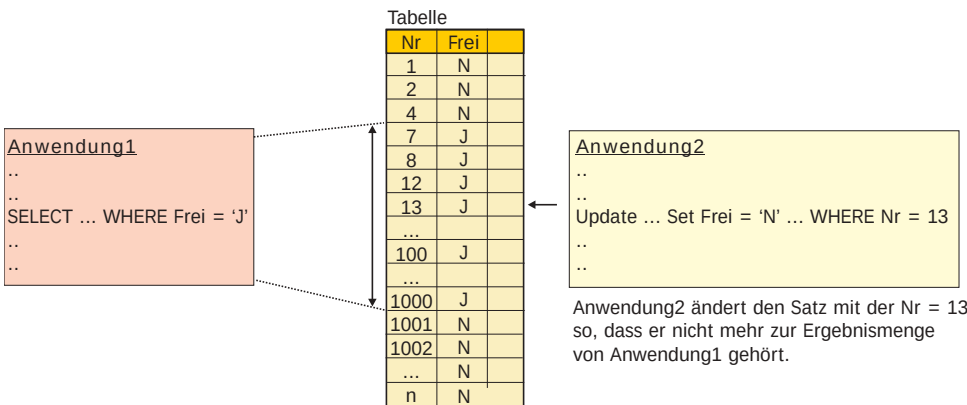


Abbildung 12.3: Nicht wiederholbare Lesevorgänge (NWL)

12.1.3 Was ist ein Lost Update (LU)?

Abbildung 12.4 zeigt Ihnen zwei Anwendungen, die gleichzeitig den gleichen Datensatz, z.B. für einen Auskunftsbildschirm, lesen (z.B. „Ist der Fensterplatz mit der Nummer 13 noch frei?“). Beide Anwendungen ändern den Datensatz mit der Nummer 13. *Anwendung2* ändert den Datensatz als Erste und vergibt somit den Platz an *Anwendung2*. Einen kurzen Augenblick später hat *Anwendung1* denselben Datensatz und vergibt ihn somit an *Anwendung1*. Wer hat eine solche Situation nicht schon einmal erlebt? Da sitzt doch tatsächlich jemand auf Ihrem reservierten Platz. Das Problem ist nur, dass beide eine Reservierungsbestätigung haben.

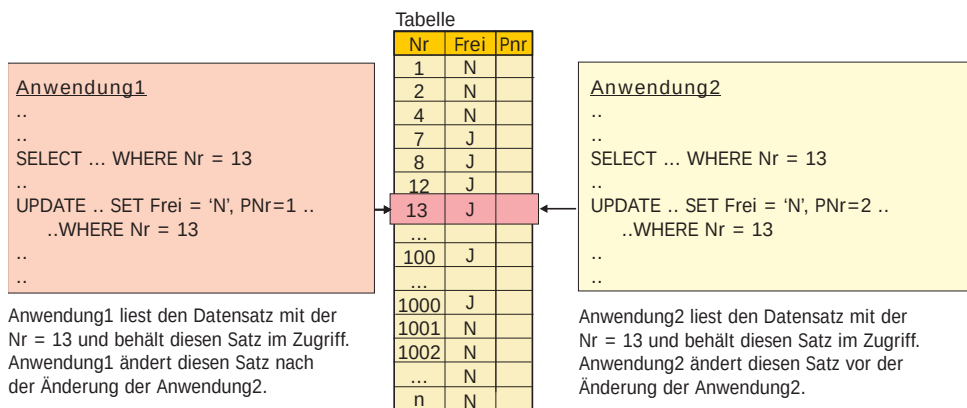


Abbildung 12.4: Lost Update (LU)

12.1.4 Was ist ein Zugriff auf nicht bestätigte Daten (ZNBD)?

Wie Abbildung 12.5 zeigt, liegt ein Zugriff auf nicht bestätigte Daten immer dann vor, wenn ein Datensatz gelesen wird, der von einer anderen Anwendung zwar geändert, aber noch nicht mit einem COMMIT bestätigt worden ist.

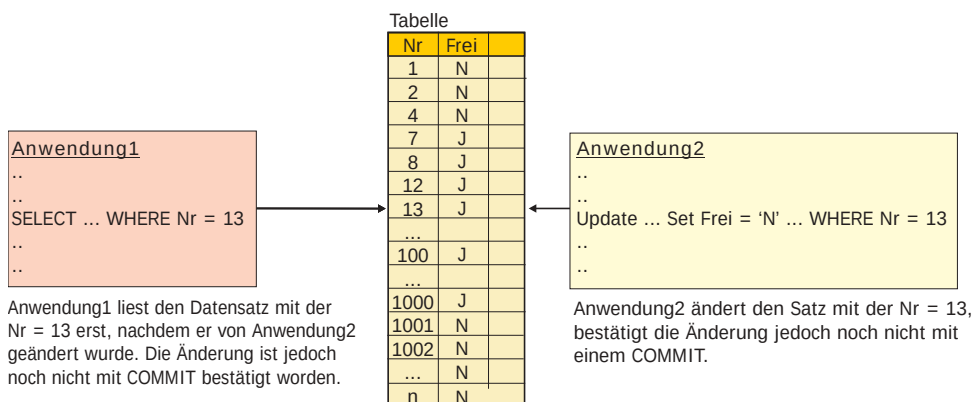


Abbildung 12.5: Zugriff auf nicht bestätigte Daten (ZNBD)

Der Vorgang aus Abbildung 12.5 kann zur Folge haben, dass *Anwendung1* falsche Informationen gelesen hat, wenn *Anwendung2* die Änderung durch einen ROLL-BACK rückgängig macht. Beim Lesevorgang von *Anwendung1* wurde der von *Anwendung2* geänderte Wert ‚N‘ für *Frei* gelesen. Somit könnte der Trugschluss entstehen, dass der Platz auf der Nummer 13 nicht frei sei. Durch die Rückname der Änderung von *Anwendung2* ist der Platz mit der Nummer 13 jedoch nie tatsächlich vergeben worden.

12.2 Welche Lösungen werden angeboten?

Nachdem uns die vier verschiedenen Problem- bzw. Anomalie-Klassen (PR, NWLV, LU und ZNBD) bekannt sind, wollen wir uns den vier Lösungsmöglichkeiten zuwenden, die uns die DB2 UDB zur Verfügung stellt. Die DB2 UDB bietet zur Lösung dieser Problemfälle die so genannten *Isolation-Levels* an. Insgesamt gibt es folgende vier:

- Repeatable Read (RR)
- Read Stability (RS)
- Cursor Stability (CS)
- Uncommitted Read (UR)

Einen für Ihren Problemfall geeigneten Isolation-Level können Sie z.B. beim Binden Ihres Anwendungsprogramms mit angeben.

Abbildung 12.6 zeigt Ihnen, welcher Isolation-Level welche Problemklasse behebt.

So behebt der Isolation-Level RR die Problemklasse PR und alle darunter liegenden wie NWLV, LU und ZNBD.

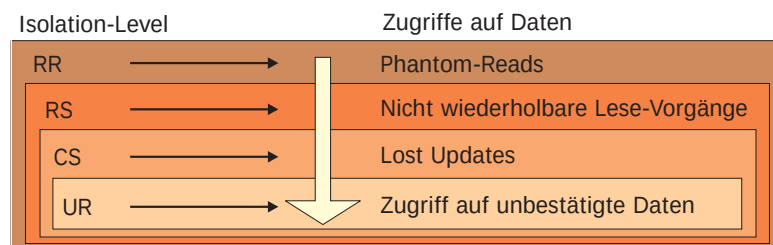


Abbildung 12.6: Welcher Isolation-Level behebt welches Problem?

12.2.1 Repeatable Read (RR)

Im Falle des Isolation-Levels RR sperrt die DB2 UDB alle Datensätze, die zur Bestimmung der Ergebnismenge vom DBMS herangezogen werden.

12.2.2 Read Stability (RS)

Bei der Verwendung des Isolation-Levels RS sperrt die DB2 UDB alle Datensätze der Ergebnismenge.

12.2.3 Cursor Stability (CS)

Die Verwendung des Isolation-Levels CS führt dazu, dass genau der Satz gesperrt ist, auf dem sich der Cursor (zum Thema Cursor siehe Kapitel 11) gerade befindet. Verlässt der Cursor diesen Datensatz, wird das von dem DBMS gesetzte Sperrflag zurückgenommen. Voraussetzung dafür ist jedoch, dass der Datensatz nicht verändert (UPDATE) oder gelöscht (DELETE) wurde. In diesem Fall bleibt der Satz mit einem entsprechenden Sperrflag bis zur nächsten Bestätigung (COMMIT) markiert und somit gesperrt.

12.2.4 Uncommitted Read (UR)

Der Isolation-Level UR, der auch als *Dirty Read* bezeichnet wird, führt dazu, dass die Anwendung keinerlei Sperrflags setzt und auch keine Sperrflags beachtet, die von anderen Anwendungen gesetzt wurden.

12.3 Detailbeschreibung

Abbildung 12.7 zeigt Ihnen, welche Anomalien mit welchem Isolation-Level behoben werden können. Wie das genau funktioniert, wollen wir Ihnen anhand des Beispiels in Abbildung 12.8 verdeutlichen.

		Anomalie		
		Zugriff auf nicht bestätigte (COMMIT) Daten	nicht wiederholbares Lesen	Phantom-Reihen
Isolation-Level	Repeatable Read (RR)			
	Read Stability (RS)			
	Cursor Stability (CS)			
	Uncommitted Read (UR)			

Abbildung 12.7: Mögliche Anomalien

In unserem Beispiel aus Abbildung 12.8 gehen wir davon aus, dass die SELECT-Anweisung `SELECT ... WHERE C1 > 5 and C1 < 1001` zur Bestimmung der entsprechenden Ergebnismenge herangezogen wird.

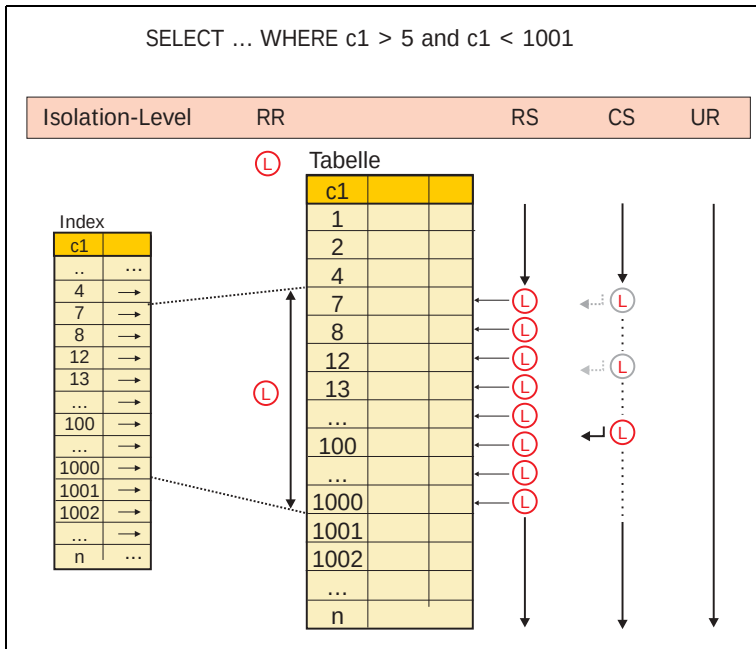


Abbildung 12.8: Auswirkung des Isolation-Levels auf den Lesevorgang

Was jetzt wie und wann bei der Verwendung der entsprechenden Isolation-Level gesperrt wird, wollen wir Ihnen in den nächsten Abschnitten für jeden einzelnen Isolation-Level erklären. Welches Sperrflag die DB2 UDB verwendet, wollen wir an dieser Stelle jedoch noch nicht betrachten.

12.3.1 Was wird bei der Verwendung von RR gesperrt?

Bei der Verwendung des Isolation-Levels RR sperrt das RDBMS:

- ▶ die ganze Tabelle, wenn das RDBMS zur Bestimmung der Ergebnismenge einen Table-Scan herangezogen hat
- ▶ den Bereich, der zur Bestimmung der Ergebnismenge benötigt wird. Dies ist dann der Fall, wenn das RDBMS zur Bestimmung der Ergebnismenge einen Index verwenden kann. Durch den Index-Scan ist DB2 in der Lage, ein so genanntes *Next Key Locking* auszuführen.

In unserem Beispiel bedeutet das, wenn

- ▶ eine zweite *Anwendung2* versucht, einen Datensatz z.B. mit dem Wert 6 für C1 einzufügen,
- ▶ solange *Anwendung1* die Transaktion des SELECT-Befehls nicht mit einem `CLOSE CURSOR` oder eventuell einem `COMMIT` abgeschlossen hat,
- ▶ der Data Manager der DB2 UDB zwar den neuen Datensatz mit dem Wert 6 sperren könnte,

- der Index Manager jedoch versucht, den nächsten Schlüsseleintrag (Indexeintrag) zu dem Wert 6 – in unserem Fall wäre das der Wert 7 im Indexbaum – zu sperren, und dies dann nicht ausführen kann, da dieser schon gesperrt ist.

12.3.2 Was wird bei der Verwendung von RS gesperrt?

In diesem Fall sperrt das RDBMS der DB2 UDB alle Datensätze der Ergebnismenge.

12.3.3 Was wird bei der Verwendung von CS gesperrt?

Durch den Isolation-Level CS wird genau der Datensatz gesperrt, der sich gerade über die FETCH-Anweisung in Bearbeitung befindet. Wenn der Datensatz nicht verändert wurde, wird das von DB2 UDB verwendete Sperrflag zurückgenommen. Bei einer Änderung wird das Sperrflag auf *X-Lock* gesetzt.

12.3.4 Was wird bei der Verwendung von UR gesperrt?

Bei der Verwendung des Isolation-Levels UR werden keinerlei Sperrflags verwendet.

12.4 Welche Datenbankobjekte werden gesperrt?

Um das Verständnis des Sperrverhaltens der DB2 UDB weiter zu vertiefen, wollen wir uns anschauen, welche Datenbankobjekte gesperrt werden können.

Abbildung 12.9 zeigt, auf welchen Ebenen DB2 Sperrungen vornehmen kann:

- Tabellenbereichsebene
- Tabellen-Ebene
- Datensatz-Ebene
- Index-Ebene

Sollte ein kompletter Tabellenbereich gesperrt sein, sind natürlich die darunter liegenden Objekte wie Tabellen und Indizes ebenfalls indirekt als gesperrt markiert.

Die DB2 UDB merkt sich jedes gesperrte Objekt einzeln in der so genannten *Sperrliste* (Lock-List). Dieser Vorgang benötigt sowohl Speicher- als auch Prozessor-Ressourcen. Damit nicht ein unnötiger Verbrauch dieser Ressourcen entsteht, versucht das RDBMS so wenige Objekte wie möglich zu sperren. Das bedeutet: Wenn die DB2 UDB es z.B. durch die Sperrung einer ganzen Tabelle vermeiden kann, dass jeder zu lesende Datensatz einzeln gesperrt werden muss, wird sie dies tun.

12.5 Wie werden die Datenbankobjekte gesperrt?

Um die Datenbankobjekte entsprechend sperren zu können, benutzt die DB2 verschiedene Sperrflags (bzw. Lock-Modes), die sich je nach verwendetem Objekt und Grund der Sperrung unterscheiden.

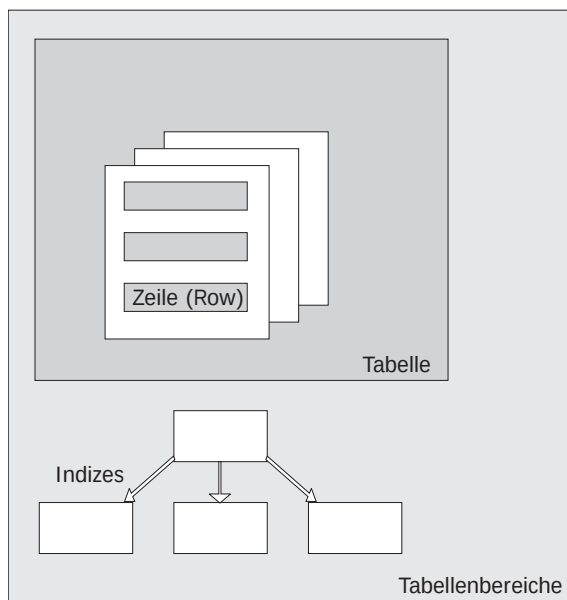


Abbildung 12.9: Welche Datenbankobjekte sperrt DB2?

12.5.1 Sperrflags für die Datenbankobjekte Tabelle und Tabellenbereiche

Je nachdem, welche Umstände dazu führen, dass eine Tabelle bzw. ein Tabellenbereich als gesperrt gekennzeichnet werden muss, markiert die DB2 UDB das Objekt mit folgenden Sperrkennzeichen:

► *Intent None (IN)*

Dieses Sperrflag (*IN-Lock*) wird sowohl für Tabellen als auch für Tabellenbereiche benutzt. Es sagt aus, dass der Sperreigner (Anwendung, die das Sperrflag verursacht) alle, selbst die nicht bestätigten Daten einer Tabelle, lesen, aber nicht ändern kann. Andere Anwendungen können die Daten dieser Tabelle sowohl lesen als auch ändern. Eine Sperrung des Lock-Eigners auf Zeilenebene wird nicht ausgeführt.

► *Intent Share (IS)*

Dieses Sperrflag (*IS-Lock*) wird sowohl für Tabellen als auch für Tabellenbereiche benutzt. Es sagt aus, dass der Lock-Eigner in Verbindung mit einem *S-Lock* auf Datensatzebene die Daten dieser Tabelle zwar lesen, aber nicht ändern kann. Andere Anwendungen können die mit diesem Lock-Mode markierte Tabelle sowohl lesen als auch ändern. Das Ändern ist jedoch nur möglich, wenn der entsprechende Datensatz nicht mit einem *S-Lock* markiert ist.

► *Intent Exclusive (IX)*

Dieses Sperrflag (*IX-Lock*) wird sowohl für Tabellen als auch für Tabellenbereiche benutzt. Der Eigner des Locks ist durch diese Kennzeichnung in der Lage, die Datensätze der entsprechenden Tabelle sowohl zu lesen als auch zu schreiben. Voraussetzung für das Lesen und Schreiben ist, dass für das Lesen auf den entsprechenden Datensätzen ein *S-* bzw. *U-Lock* und beim Schreiben ein *X-Lock* gesetzt werden kann.

► *Share with Intent Exclusive (SIX)*

Dieses Sperrflag (*SIX-Lock*) wird für Tabellen benutzt. Es versetzt den Eigner der Sperre in die Lage, alle Datensätze sowohl zu lesen als auch zu ändern. Voraussetzung für das Ändern des jeweiligen Datensatzes ist jedoch, dass dieser mit einem *X-Lock* gesperrt werden konnte.

► *Share (S)*

Dieses Sperrflag (*S-Lock*) wird für Tabellen benutzt. Es erlaubt dem Eigner der Sperre und allen anderen Anwendungen, die ebenfalls auf die Tabelle zugreifen möchten, Datensätze zu lesen, aber nicht zu ändern. Eine Kennzeichnung der Datensätze durch ein entsprechendes Sperrkennzeichen wird nicht ausgeführt. Die Anweisung `SELECT ... FOR UPDATE` löst diesen Lock-Modus aus.

► *Update (U)*

Dieses Sperrflag (*U-Lock*) wird für Tabellen benutzt. Der Eigner ist durch dieses Sperrflag in der Lage, Datensätze der entsprechenden Tabelle sowohl zu ändern als auch zu lesen. Eine Änderung der Datensätze ist jedoch nur dann möglich, wenn die Tabelle zuvor mit einem *X-Lock* gesperrt werden kann. Andere Applikationen können Datensätze dieser Tabelle lesen, nicht aber ändern. Ein Sperren auf Datensatz-Ebene wird bei der Verwendung dieses Lock-Modus nicht ausgeführt.

► *Exclusive (X)*

Dieses Sperrflag (*X-Lock*) wird für Tabellen benutzt. Durch die Verwendung des *X-Locks* kann der Eigner der Sperre Datensätze der jeweiligen Tabelle sowohl lesen als auch schreiben. Bei der Verwendung dieses Sperrflags werden keine Sperrungen auf Datensatz-Ebene ausgeführt. Andere Anwendungen können Datensätze dieser Tabelle nur lesen, wenn diese den Isolation-Level UR benutzen.

► *Super Exclusive (Z)*

Dieses Sperrflag wird sowohl für Tabellen als auch für Tabellenbereiche benutzt. Dieser Lock-Modus wird z.B. für folgende Operationen genutzt:

- um eine Tabelle zu ändern (`ALTER TABLE`)
- um eine Tabelle zu löschen (`DROP TABLE`)
- für einige Fälle der Tabellenreorganisation

Ist eine Tabelle mit diesem Sperrkennzeichen markiert, können andere Anwendungen Daten dieser Tabelle weder lesen noch schreiben. In Verbindung mit diesem Sperrkennzeichen wird keine Sperrung auf Datensatz-Ebene vorgenommen.

12.5.2 Sperrflags für Datenbankobjekte und Tabellenzeilen

Ebenso wie Tabellen und Tabellenbereiche werden Datensätze mit entsprechenden Sperrkennzeichen, den so genannten *Row Lock Modes*, versehen. Eine Sperrung auf Zeilenebene kommt nur in Zusammenhang mit den *INTENT*-Sperrflags (*IS*, *IX* und *SIX*) auf Tabellenebene vor. Welche Sperrkennzeichen es auf Datensatz-Ebene gibt, wollen wir im Folgenden erläutern:

► *Share (S)*

Der *S-Lock* auf Datensatz-Ebene zeigt an, dass der Eigner der Sperre den markierten Datensatz gerade zum Lesen in der Bearbeitung hat. Andere Anwendungen können lesend auf mit *S-Lock* markierte Datensätze zugreifen.

► *Update (U)*

Das *U-Lock* kennzeichnet, dass der jeweilige Datensatz vom Eigner der Sperre für eine Leseoperation mit eventuellen Änderungsabsichten markiert wurde. Andere Anwendungen können so gesperrte Datensätze für reine Lesezugriffe nutzen. Der Unterschied zum reinen *S-Lock* ist, dass immer nur eine Anwendung einen *U-Lock* setzen kann. Bei einer *CURSOR FOR UPDATE*-Anweisung werden die Datensätze beim Lesen mit einem *U-Lock* markiert.

► *Exclusive (X)*

Bis auf Anwendungen mit dem Isolation-Level *UR* sind die mit einem *X-Lock* markierten Datensätze für jeden weiteren Zugriff gesperrt. Dieses Sperrflag wird bei der Änderung des Datensatzes gesetzt.

► *Weak Exclusive (W)*

Die mit einem *W-Lock* markierte Zeile kann vom Eigner der Sperre gelesen und geändert werden. Dieser Sperrmodus wird nur bei Tabellen verwendet, für die kein Index des Typs 2 definiert wurde. Für das Einfügen in einen eindeutigen Index wird dieses Sperrflag ebenfalls verwendet.

► *Next Key Share (NS)*

Die mit *NS-Lock* gesperrten Datensätze können sowohl vom Lockeigner als auch von anderen gleichzeitig ausgeführten Anwendungen gelesen, aber nicht verändert werden. Dieses Sperrflag wird bei Anwendungen mit dem Isolation-Level *RS* oder *CS* anstelle des *S-Locks* verwendet.

► *Next Key Weak Exclusive (NW)*

Wird in einen Index eine Zeile eingefügt, wird dieses Sperrflag dazu genutzt, die nächste Zeile als gesperrt zu markieren. Die mit *NW-Lock* gesperrte Zeile kann vom Eigner der Sperre zwar gelesen, aber nicht verändert werden. Für Indizes des Typs 2 findet dieses Sperrkennzeichen nur dann Verwendung, wenn die nächste Zeile aufgrund einer Query, die den Isolation-Level *RR* verwendet, gesperrt ist.

Abbildung 12.10 zeigt Ihnen, in welcher Art und Weise diese auf Zeilenebene von DB2 UDB genutzten Sperrkennzeichen zueinander kompatibel sind.

12.6 Lock-Mode-Kompatibilität

		Lock-Mode 1					
Lock		S	U	X	W	NS	NW
Lock-Mode 2	S	✓	✓	✗	✗	✓	✗
	U	✓	✗	✗	✗	✓	✗
	X	✗	✗	✗	✗	✗	✗
	W	✗	✗	✗	✗	✗	✓
	NS	✓	✓	✗	✗	✓	✓
	NW	✗	✗	✗	✓	✓	✗

Abbildung 12.10: Lock-Mode-Kompatibilitäten auf Zeilen-Ebene

Diese Abbildung 12.10 zeigt Ihnen zum Beispiel:

- ▶ Ein Datensatz kann von mehreren Applikationen gleichzeitig durch ein S-Lock gekennzeichnet werden.
- ▶ Ein Datensatz, der durch eine Anwendung mit dem Sperrkennzeichen X-Lock markiert wurde, kann von einer weiteren Anwendung mit keinem weiteren Sperrflag markiert werden.

12.7 Verschiedene Sperrsituationen

In den Beispielen Abschnitt 12.7.1 „Deadlock“ und Abschnitt 12.7.2 „Lock-Timeout“ werden wir Ihnen die zwei verschiedenen Sperrkonflikte *Deadlock* und *Lock-Timeout* näher erklären. In Abschnitt 12.7.3 „Verhinderung von Schreibkonflikten“ zeigen wir ein Beispiel zur Vermeidung von Sperrkonflikten.

12.7.1 Deadlock

Im Anschluss an Abbildung 12.11 werden wir Ihnen Schritt für Schritt erläutern, wie ein *Deadlock* zustande kommen kann.

Voraussetzung

In unserem Beispiel gehen wir davon aus, dass:

- ▶ beiden Applikationen der Isolation-Level CS zugewiesen wurde.
- ▶ zwei verschiedene Anwendungen (*Anwendung1* und *Anwendung2*) auf eine gemeinsame Schnittmenge von Datensätzen (die in der Matrix dargestellt ist) zugreifen möchten. Diese Datensätze sind in Abbildung 12.11 durch die größeren Quadrate gekennzeichnet. Die Schnittmenge enthält genau vier Datensätze.

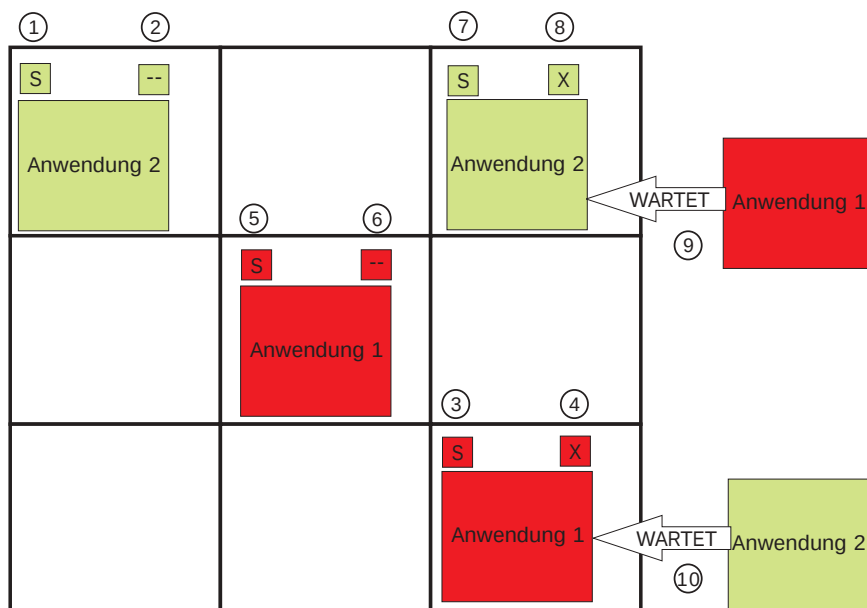


Abbildung 12.11: Der Deadlock

Schritt 1

Anwendung2 greift über einen FETCH lesend auf den Datensatz zu. Das RDBMS kennzeichnet den Satz dadurch mit einem *S-Lock*.

Schritt 2

Anwendung2 ändert den Datensatz nicht. Der nächste FETCH der *Anwendung2* bewegt den CURSOR auf den Datensatz von Schritt 7 und 8. Da der Datensatz von Schritt 1 nicht geändert worden ist, nimmt das RDBMS das Sperrkennzeichen *S-Lock* wieder zurück.

Schritt 3

Anwendung1 greift über einen FETCH lesend auf den Datensatz zu. Das RDBMS kennzeichnet den Satz dadurch mit einem *S-Lock*.

Schritt 4

Anwendung1 ändert den Datensatz. Der nächste FETCH der *Anwendung1* bewegt den CURSOR auf den Datensatz von Schritt 5 und 6. Da der Datensatz in Schritt 3 geändert wurde, markiert das RDBMS den Datensatz jetzt mit einem *X-Lock*.

Schritt 5

Anwendung1 greift über einen FETCH lesend auf den nächsten Datensatz zu. Das RDBMS kennzeichnet den Satz dadurch mit einem *S-Lock*.

Schritt 6

Anwendung1 ändert den Datensatz nicht. Der nächste FETCH der *Anwendung1* bewegt den CURSOR in Richtung des Datensatzes von Schritt 7 und 8. Da der Datensatz von Schritt 5 nicht geändert wurde, nimmt das RDBMS das Sperrkennzeichen *S-Lock* wieder zurück.

Schritt 7

Anwendung2 greift über einen FETCH lesend auf den nächsten Datensatz zu. Das RDBMS kennzeichnet den Satz dadurch mit einem *S-Lock*.

Schritt 8

Anwendung2 ändert den Datensatz. Der nächste FETCH der *Anwendung2* bewegt den CURSOR in Richtung des Datensatzes von Schritt 3 und 4. Da der Datensatz in Schritt 7 geändert wurde, markiert das RDBMS den Datensatz jetzt mit einem *X-Lock*.

Schritt 9

Das RDBMS kann den Datensatz, auf den *Anwendung1* zugreifen will, nicht mit einem *S-Lock* markieren, da dieser in Schritt 8 vom RDBMS durch die Änderung der *Anwendung2* mit einem *X-Lock* markiert wurde. Wie die Kompatibilitätsliste in Abbildung 12.10 zeigt, sind diese beiden Sperrmodi nicht kompatibel. *Anwendung1* wartet jetzt so lange, bis das durch *Anwendung2* gesetzte Sperrkennzeichen entfernt wird. Das Sperrkennzeichen wird durch eine Bestätigung (COMMIT) von *Anwendung2* zurückgesetzt.

Schritt 10

Das RDBMS kann den Datensatz, auf den *Anwendung2* zugreifen will, nicht mit einem *S-Lock* markieren, da dieser in Schritt 4 vom RDBMS durch die Änderung der *Anwendung1* mit einem *X-Lock* markiert wurde. Wie die Kompatibilitätsliste in Abbildung 12.10 zeigt, sind diese beiden Sperrmodi nicht kompatibel. *Anwendung2* wartet jetzt so lange, bis das durch *Anwendung1* gesetzte Sperrkennzeichen entfernt wird. Das Sperrkennzeichen wird durch eine Bestätigung (COMMIT) von *Anwendung1* zurückgesetzt.

Ergebnis

Da jetzt beide Anwendungen auf eine Bestätigung der jeweils anderen Applikation warten, ist der *Deadlock* perfekt.

Reaktion der DB2 UDB bei einem Deadlock

Das RDBMS erkennt diesen *Deadlock* spätestens nach der über den Datenbankparameter DLCHTIME gesetzten Zeit. Zur Aufhebung des Konflikts löst das RDBMS einen ROLLBACK für eine der beiden Anwendungen aus. Der für die Anwendung implizit ausgeführte ROLLBACK führt dazu, dass alle Sperrkennzeichen dieser Anwendung zurückgesetzt werden. Wir nehmen an, dass in unserem Fall die *Anwendung2* den

impliziten ROLLBACK erhalten hat. *Anwendung1* kann dann, falls keine weiteren Konflikte mit anderen Anwendungen auftreten, ihre Transaktion weiterführen.

Hinweis:

Der Standardwert des Parameters DLCHTIME beträgt 10.000 ms.

12.7.2 Lock-Timeout

Nachdem uns jetzt die Bedeutung eines *Deadlocks* und dessen Entstehung bekannt ist, werden wir uns im Anschluss an Abbildung 12.12 dem Lock-Timeout zuwenden.

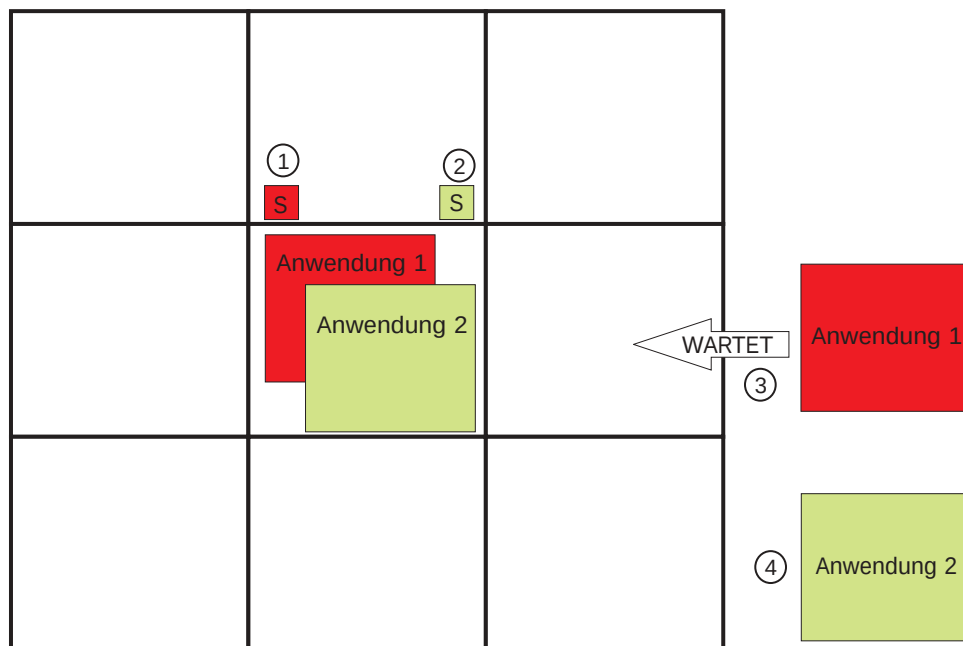


Abbildung 12.12: Der Lock-Timeout

Voraussetzung

In unserem Beispiel gehen wir davon aus, dass:

- ▶ beiden Applikationen der Isolation-Level CS zugewiesen wurde.
- ▶ zwei verschiedene Anwendungen (*Anwendung1* und *Anwendung2*) über CURSOR auf die Ergebnismenge zugreifen, die von einer SELECT-Anweisung im RDBMS zur Verfügung gestellt wurde. In Abbildung 12.12 stellen die beiden größeren Quadrate in der Matrix den gleichzeitigen Zugriff auf ein und denselben Datensatz dar.

Schritt 1

Anwendung1 greift über einen FETCH lesend auf den Datensatz zu. Das RDBMS kennzeichnet den Satz dadurch mit einem *S-Lock*.

Schritt 2

Anwendung2 greift über einen FETCH lesend auf den Datensatz zu. Das RDBMS kennzeichnet den Satz dadurch mit einem *S-Lock*.

Schritt 3

Anwendung1 will den Datensatz mit dem Befehl UPDATE ändern. Das hat zur Folge, dass das RDBMS diesen Datensatz mit einem *X-Lock* markiert. Wie die Kompatibilitätsliste in Abbildung 12.10 zeigt, sind die beiden Lock-Modi S und X nicht kompatibel. Das RDBMS kann die Sperrung des Datensatzes mit einem *X-Lock* zur Markierung der Änderung nicht ausführen, da dieser Satz noch von der *Anwendung2* mit einem *S-Lock* markiert ist.

Schritt 4

Anwendung2 behält den Datensatz weiter in Bearbeitung.

Ergebnis

Anwendung1 wartet so lange, bis die *Anwendung2* den Datensatz mit dem nächsten FETCH-Befehl verlässt.

Sollte jetzt der Benutzer der *Anwendung2* z.B. den Auskunftsbildschirm, der dafür sorgt, dass die Anwendung auf dem Datensatz steht, stehen lassen und in die Pause oder gar in den Urlaub gehen, wartet die *Anwendung1* so lange, bis:

- ▶ der Administrator die *Anwendung2* über den FORCE APPLICATION...-Befehl rauschmeißt.
- ▶ der Benutzer zurückkommt und dafür sorgt, dass die *Applikation2* den Datensatz freigibt.
- ▶ DB2 diese Situation erkennt und der *Applikation1* den SQL-Fehler Lock-Time für den Befehl UPDATE zurückliefert.

Tipp:

Standardmäßig ist der Parameter LOCKTIMEOUT nicht gesetzt. Dieser Parameter sorgt dafür, dass das RDBMS nach der über diesen Parameter eingestellten Zeit den Wartevorgang der entsprechenden Anwendung mit dem zugehörigen SQL-Fehler abbricht.

12.7.3 Verhinderung von Schreibkonflikten

Welche Möglichkeiten Ihnen die DB2 UDB bietet, um einen Lock-Timeout zu vermeiden, erklären wir im Anschluss an Abbildung 12.13.

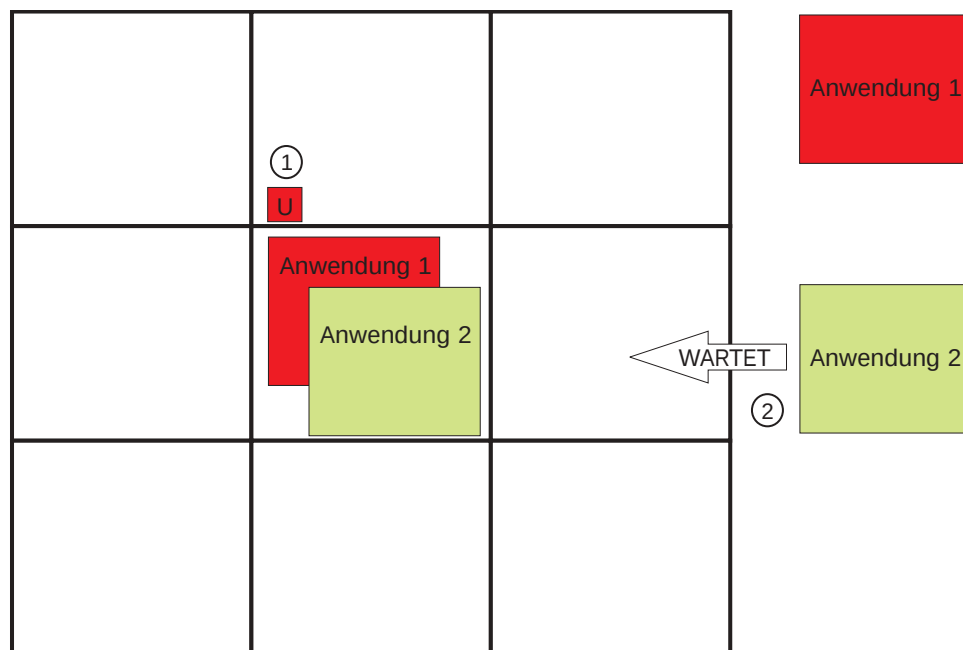


Abbildung 12.13: Sperren mit U-Lock

Voraussetzung

In unserem Beispiel gehen wir davon aus, dass:

- ▶ beiden Applikationen der Isolation-Level CS zugewiesen wurde.
- ▶ zwei verschiedene Anwendungen (*Anwendung1* und *Anwendung2*) über CURSOR auf die Ergebnismenge zugreifen, die von einer SELECT-Anweisung im RDBMS zur Verfügung gestellt wurde. In Abbildung 12.13 stellen die beiden größeren Quadrate innerhalb der Matrix den gleichzeitigen Zugriff auf ein und denselben Datensatz dar.

Die CURSOR beider Anwendungen wurden bei ihrer Deklaration mit der Option ...FOR UPDATE versehen.

Schritt 1

Anwendung1 greift über einen FETCH lesend auf den Datensatz zu. Das RDBMS kennzeichnet den Satz dadurch mit einem U-Lock.

Schritt 2

Anwendung2 greift über einen `FETCH` lesend auf denselben Datensatz zu. Das RDBMS möchte den Datensatz mit einem *U-Lock* markieren. Wie Sie in der Kompatibilitätsliste der Abbildung 12.10 erkennen, ist der Lock-Modus `U` zu sich selbst nicht kompatibel. Dies führt dazu, dass *Anwendung2* so lange wartet, bis:

- ▶ der Administrator die *Anwendung1* über den Befehl `FORCE APPLICATION...` rauswirft
- ▶ die *Anwendung1*:
 - entweder den Datensatz unverändert verlässt und ihn somit freigibt
 - oder die *Anwendung1* den Datensatz ändert und die Transaktion mit einer Bestätigung (`COMMIT`) oder einer Rücknahme (`ROLLBACK`) beendet.
- ▶ DB2 diese Situation erkennt und der *Anwendung2* den SQL-Fehler `lock-time` für den `FETCH`-Befehl zurückliefert.

Ergebnis

Da die *Anwendung1* nicht blockiert ist, wird im Regelfall der Fall 2 eintreffen.

12.8 Konfiguration der Lock-Parameter

Im Anschluss sind die wichtigsten Datenbankparameter aufgeführt, mit denen Sie das Sperrverhalten der DB2 UDB beeinflussen können:

- ▶ `LOCK_LIST`
- ▶ `MAXLOCKS`
- ▶ `DLCHKTIME`
- ▶ `LOCKTIMEOUT`

Nachfolgend werden wir auf die beiden bisher noch nicht erklärten Parameter `MAXLOCKS` und `LOCK_LIST` näher eingehen.

12.8.1 LOCK_LIST

Um sich die ganzen mit Sperrkennzeichen markierten Objekte zu merken, benötigt DB2 einen Speicherbereich, die so genannte *Lockliste*. Über den Parameter `LOCK_LIST` können Sie die Größe dieser Liste bestimmen.

Hinweis:

Sollte jemals der Fall eintreten, dass die Lockliste voll ist, muss jede Anwendung, die ein Datenbankobjekt sperren möchte, warten, bis wieder ein Platz in dieser Liste frei ist.

12.8.2 MAXLOCKS

Eine volle Lockliste wirkt sich negativ auf die Performance aus. Um zu verhindern, dass die Lockliste mit unnötigen Sperrflags belegt wird, versucht die DB2 UDB, so genannte *Lock-Eskalationen* durchzuführen. Das bedeutet, DB2 versucht, verschiedene einzelne Sperrflags zu einem Sperrflag zusammenzufassen.

Ein Beispiel dafür wäre, wenn:

- ▶ eine Anwendung sehr viele Datensatzsperrungen auf einer Tabelle hält.
- ▶ keine andere Anwendung eine Lock-Eskalation durch andere Sperrflags verhindert.

In diesem Fall könnte die DB2 UDB die ganzen Datensatzsperrungen zu einem Table-Lock zusammenfassen bzw. eskalieren.

Der Parameter MAXLOCKS gibt an, wie viel Prozent der Sperrliste eine Anwendung belegen muss, damit der Datenbankmanager eine Eskalation ausführt.

DB2 führt in folgenden Fällen eine Lock-Eskalation durch:

- ▶ Die Anzahl der Sperren, die von einer Anwendung aktiviert wurden, erreicht den Wert von MAXLOCKS.
- ▶ In der Sperrliste steht kein weiterer Platz zur Verfügung.

Hinweis:

Auch die Ausführung einer Lock-Eskalation benötigt Zeit und kann zum Zeitpunkt der Ausführung Einfluss auf die Performance nehmen.

13

Pflege und Wartung von Datenbanken

Auch wenn in einem DB2- bzw. Datenbanksystem alles funktioniert, sich niemand beschwert und ein fundiertes Sicherungskonzept besteht, so müssen trotzdem einige Aufgaben regelmäßig durchgeführt werden. Diese „Pflege“ der Datenbank bzw. des Datenbanksystems ist notwendig – ähnlich wie bei Autos –, um dauerhaft eine hohe Leistung zu bekommen.

Die „Wartung“ umfasst Eingriffe, um Fehler zu analysieren und zu beheben. Dazu gibt es verschiedene Fehlerprotokolle, die von DB2 geführt werden, und eine Reihe von Tools, die Sie als Administrator bei der Wartung unterstützen können.

Natürlich sind auch die Sicherung und die Überwachung des Datenbanksystems Aufgaben, die unter Pflege und Wartung einzuordnen sind. Diese werden aber jeweils in einem separaten Kapitel behandelt und sind deshalb hier nicht noch einmal aufgeführt.

13.1 Pflege von Datenbanken

Die regelmäßige Durchführung von Pflegemaßnahmen ist für eine optimale Leistung unerlässlich. Oft werden in der Praxis diese – eigentlich recht einfach durchzuführenden Schritte – vernachlässigt, was dann häufig zu starken Performance-Einbußen führt. Dabei können einige dieser Schritte sogar automatisiert werden, um zu verhindern, dass man sie vergisst. Scripts und die Scheduling-Funktionalität (zeitliche Steuerung) durch die Taskzentrale sind hier geeignete Hilfsmittel.

13.1.1 Runstats

Bereits beim Anlegen der Datenbank wurden in dem Systemkatalog-Tabellenbereich nicht nur die Tabellen zur Strukturbeschreibung erstellt, sondern auch die so genannten Statistik-Tabellen bzw. -Sichten. Diese Statistikinformationen wurden in den Systemkatalog-Sichten dem Schema SYSSTAT zugeordnet und enthalten viele wichtige Daten über die Anzahl der Zeilen, Verteilungen (clustering) und über die Ausprägungen der Inhalte verschiedener Spalten. Diese Statistikinformationen sind ein elementarer Ausgangspunkt für die Zugriffsoptimierungen des DB2-Optimizers und so sicherlich eine wichtige leistungsrelevante Komponente.

Die Statistiken müssen vom Administrator gepflegt werden, d.h. er muss dafür sorgen, dass der Optimizer aktuelle Statistiken vorfindet, da diese nicht automatisch auf dem aktuellen Stand gehalten werden. Das erfordert nicht, dass Sie sich Gedanken über die Aktualisierung (update) der entsprechenden Spalten und Informationen der Statistiken machen müssen. Ein Tool – runstats (run statistics) – ist für das Sammeln dieser Statistiken zuständig. Sie müssen nur die Ausführung dieses Tools sicherstellen.

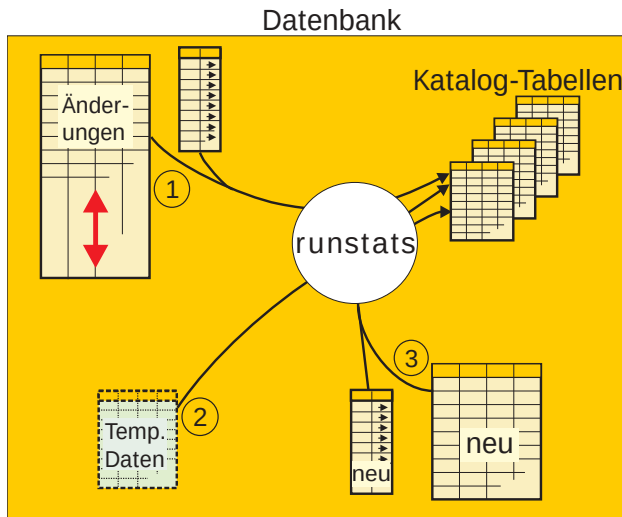


Abbildung 13.1: Runstats

Abbildung 13.1 zeigt, dass das runstats-Tool eingesetzt werden sollte, wenn Änderungen stattfinden. Diese Änderungen können – wie in Punkt ① angedeutet – durch große Lade- oder Löschoperationen bedingt sein. Auch Zugriffe auf temporäre Tabellen (siehe auch „global temporary tables“ in Abschnitt 7.2.2) – Punkt ② – können von Statistikinformationen profitieren, wenn der Optimizer dadurch die Datenmenge und andere Werte für die Zugriffswegberechnung nutzen kann. Werden neue Objekte erstellt (und gefüllt), müssen auch dafür Statistiken gesammelt werden (Punkt ③).

Generell gilt also, dass nach jedem *größeren* Eingriff, wie z.B. Lade-Operationen, großen Imports, Reorganisationen oder nach der Erstellung neuer Tabellen und Indizes, die Statistiken auf den neuesten Stand gebracht werden sollten. Sind 10 bis 20% der Daten einer Tabelle von Änderungen betroffen, kann schon von *größeren* Änderungen gesprochen werden.

Die verfügbaren Statistik-Sichten einer Datenbank sind in Abbildung 13.2 gezeigt. Dabei sind die Statistiken (Spalteninhalte) von neuen Objekten bzw. allen Objekten, für die noch nie Statistikinformationen gesammelt wurden, mit „-1“ (nicht verfügbar) belegt.

Beschreibungen der Inhalte bzw. der Spalten finden Sie in [SQLREF] (siehe Literaturverzeichnis).

Das Runstats-Tool

Das runstats-Tool bezieht sich grundsätzlich auf die Tabellen-Ebene. Die Statistiken können also für eine einzelne Tabelle gesammelt werden und – optional – für die zur Tabelle gehörenden Indizes. Dabei lassen sich mit diesem Tool unter DB2 UDB V8 zahlreiche weitere Detaillierungsgrade einstellen, um festzulegen, mit welchem Aufwand und in welchem Umfang Statistiken bereitgestellt werden sollen.

D:\SQLLIB\BIN>db2 list tables for schema SYSSTAT

Tabelle/Sicht	Schema	Typ	Erstellungszeit
COLDIST	SYSSTAT	V	2003-08-08-13.00.10.731001
COLGROUPDIST	SYSSTAT	V	2003-08-08-13.00.10.791003
COLGROUPDISTCOUNTS	SYSSTAT	V	2003-08-08-13.00.10.791005
COLGROUPS	SYSSTAT	V	2003-08-08-13.00.10.791001
COLUMNS	SYSSTAT	V	2003-08-08-13.00.10.741001
FUNCTIONS	SYSSTAT	V	2003-08-08-13.00.10.791007
INDEXES	SYSSTAT	V	2003-08-08-13.00.10.891001
ROUTINES	SYSSTAT	V	2003-08-08-13.00.11.021001
TABLES	SYSSTAT	V	2003-08-08-13.00.11.101001
XMLSTATS	SYSSTAT	V	2003-08-08-13.00.11.272001

10 Satz/Sätze ausgewählt.

Abbildung 13.2: Statistik-Sichten einer Datenbank

Wir werden in diesem Buch nur eine Auswahl der wichtigsten Optionen und Detaillierungsgrade vorstellen, was für die allermeisten Situationen völlig ausreichend ist. Alle weitergehenden Optionen erfordern eine sehr gute Kenntnis der Daten und der Zugriffe auf diese Daten und werden nur bei sehr wenigen, sehr speziellen Anfragen Vorteile bringen. Wichtiger ist hierbei, dass die Statistiken (regelmäßig) gepflegt werden!

Bevor wir uns die Befehlssyntax anschauen, gehen wir den „grafischen Weg“ über die DB2-Steuerzentrale. Der Aufruf des runstats-Tools erfolgt, wie erwähnt, auf Tabellen-Ebene und somit aus dem Kontextmenü einer Tabelle. Der Eintrag STATISTIK AUSFÜHREN... führt Sie zum STATISTIK AUSFÜHREN-Fenster der Tabelle (siehe Abbildung 13.3).

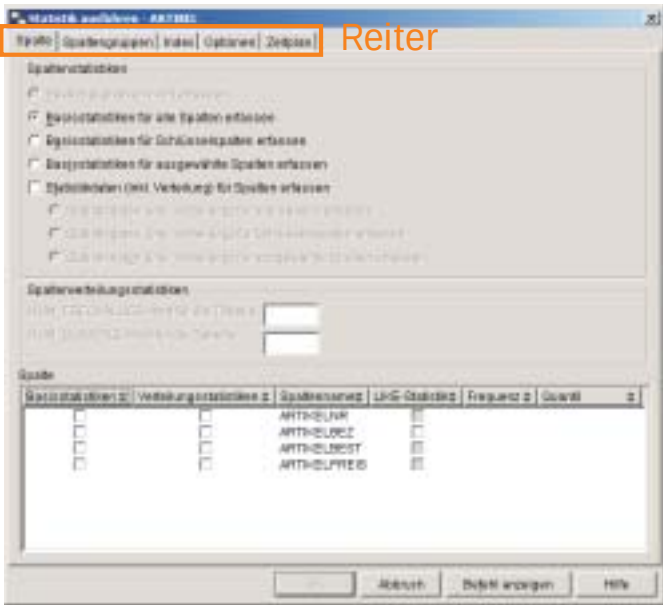


Abbildung 13.3: Statistik ausführen – Schritt 1 – Spalte

Die verschiedenen Reiter zeigen bereits die Vielfalt der Einstellmöglichkeiten. Die Registerkarte SPALTENGRUPPEN betrachten wir hierbei nicht weiter, da dies in den meisten Fällen unnötig ist und die Sache nur verkompliziert.

Die erste Registerkarte (SPALTE) erlaubt es, die Spalten auszuwählen, die für die Statistikinformationen gesammelt werden sollen. Die Standardeinstellung BASISSTATISTIKEN FÜR ALLE SPALTEN ERFASSEN ist dabei sicherlich vernünftig. Um die Statistiken nur für ausgewählte Spalten, die im Spalten-Teilfenster markiert werden können, zu sammeln, ist schon eine sehr gute Kenntnis der Zugriffe und deren interner Abarbeitung erforderlich.

Einige Begriffe, die in diesem Zusammenhang immer wieder auftauchen, sollen zunächst geklärt werden:

► Verteilungsstatistiken

Die Verteilungsstatistiken beziehen sich auf die Verteilung der Inhalte (Spaltenwerte) von Spalten. Dies ist besonders für Spalten interessant, die in WHERE- und GROUP BY-Klauseln von SQL angegeben werden. Werden diese Statistiken *nicht* gesammelt, kennt der Optimizer z.B. nur die Anzahl der Zeilen und den jeweils größten und kleinsten Wert einer Spalte. Des Weiteren geht er dann aber von einer Gleichverteilung der Daten zwischen diesem Maximal- und Minimalwert aus. Sind die Daten tatsächlich aber ungleichmäßig verteilt, kann es zu suboptimalen Zugriffsweg-Bestimmungen durch den Optimizer kommen.

Beispiel: Wenn Sie das Telefonbuch zur Hand nehmen, sehen Sie über die Markierung der einzelnen Buchstaben im Register, dass es mehr Einträge für Leute gibt, deren Nachnamen mit „M“ oder „S“ beginnen als mit „X“ oder „Z“. Die Daten sind also ungleich verteilt. Die Annahme, dass eine Abfrage aller mit „S“ beginnenden Namen genauso viele „Treffer“ liefern würde wie die Abfrage der Namen, die mit „Z“ beginnen, ist somit falsch und kann zu falschen Zugriffsweg-Bestimmungen führen.

Man benötigt Verteilungsstatistiken, um diese Ungleichverteilung erfassen zu können, die für die so genannte Selektivität (Ergebnismengen-Einschränkung) einer Abfrage und so für die Zugriffsbestimmung einer Abfrage entscheidend sein kann.

Zwei Werte unterstützen diese Verteilungsstatistiken im Besonderen:

- Frequenz bzw. Häufigkeit

Wie oft kommt ein bestimmter Wert vor?

- Quantil

Diesen Wert können Sie sich wie die Beschriftung einer Reihe von Leitz-Ordern vorstellen, in die Sie Kundenakten eingeordnet haben. Auf jedem Ordner wird der größte Wert (Kundennummer oder Name – je nach Sortierung) stehen, dessen Akte sich noch im Ordner befindet. Größere Werte sind dann im nächsten Ordner abgeheftet usw. DB2 legt dabei zusätzlich fest, dass in jedem „Ordner“ die gleiche Anzahl von Akten abgelegt ist. So kann bei einem Zugriff, der alle Kunden mit einer Kundennummer betrifft, die kleiner als „4711“ sein soll, schnell abgeschätzt werden, wie viele Ordner benötigt werden. Entsprechend kennt man dann die Anzahl der „Treffer“.

► LIKE-Statistiken

Diese Option ermöglicht es, für SQL-Abfragen mit LIKE-Prädikat spezielle Statistiken zu sammeln.

Statistikinformationen können – wie Sie gesehen haben – sehr detailliert sein und sind die Basisdaten jeder Optimierung. Dabei können sich diese Informationen nicht nur auf die Spalten beziehen, sondern auch auf die zugehörigen Indizes, die ja hoffentlich für den Zugriff verwendet werden. Ob Indizes verwendet werden können, und wie gut (effizient) der Zugriff über die Indizes sein wird, schätzt der Optimizer ebenfalls durch Statistikdaten ab.

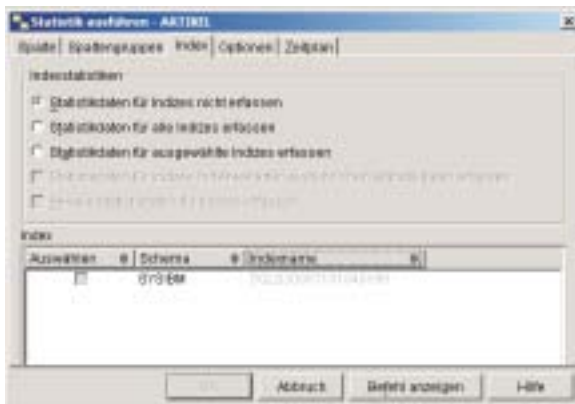


Abbildung 13.4: Statistik ausführen – Schritt 3 – Index

Auf der dritten Registerkarte – INDEX – im STATISTIK AUSFÜHREN-Fenster kann bestimmt werden, ob diese Indexstatistiken mit erfasst werden sollen. Während der Standard dies nicht vorsieht, können Sie entweder alle oder bestimmte Indizes auswählen, für die Statistikwerte gesammelt werden. Sollen Indexstatistiken erfasst werden, können Sie noch den Detaillierungsgrad erhöhen, indem Sie ...MIT AUSFÜHRICHEN STATISTIKEN ERFASSEN selektieren. BEISPIELSTATISTIKDATEN FÜR INDIZES ERFASSEN kann nur dann ausgewählt werden, wenn Sie die detaillierten Indexstatistiken gewählt haben. Dabei wird dann mit Stichproben gearbeitet und eine Art Hochrechnung erstellt, was schnell geht und gute Werte liefert, die aber nicht hundertprozentig korrekt sein werden. Diese Möglichkeit dient besonders zur Leistungssteigerung des RUNSTATS-Befehls und kann bedenkenlos verwendet werden.

Hinweis:

Je detaillierter Sie Statistiken erfassen und je mehr Optionen Sie angeben, desto mehr Speicher wird einerseits für die eigentliche Erstellung der Statistiken und andererseits für die Speicherung dieser Statistikwerte im Systemkatalog benötigt. Für die Erstellung der Statistikwerte mit der Option BEISPIELSTATISTIKEN sind beispielsweise 2 Mbyte zusätzlicher Speicher notwendig, die im Datenbankparameter `stat_heap_sz` bereitgestellt werden müssen.

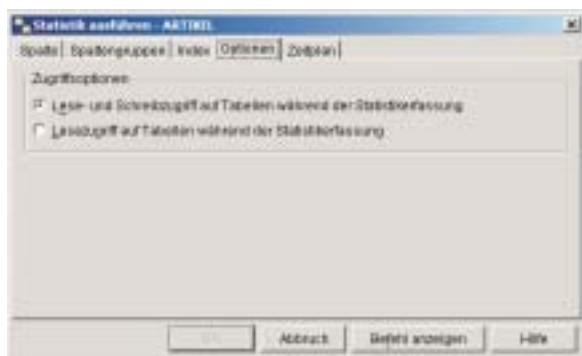


Abbildung 13.5: Statistik ausführen – Schritt 4 – Optionen

Im vierten Schritt (siehe Abbildung 13.5) kann festgelegt werden, welche Zugriffsmöglichkeiten während des Statistiklaufs vorhanden sein sollen. Lese- und Schreibzugriff sind dabei die Standardvorgabe.

Auf der letzten Registerkarte können Sie festlegen, ob das Tool entweder zeitgesteuert oder sofort aufgeführt wird. Über den Button BEFEHL ANZEIGEN können Sie sich den Befehl vorher anschauen.

Der Befehl lässt sich einfach in ein Script verpacken. Dabei sieht die Syntax wie in Abbildung 13.6 aus.

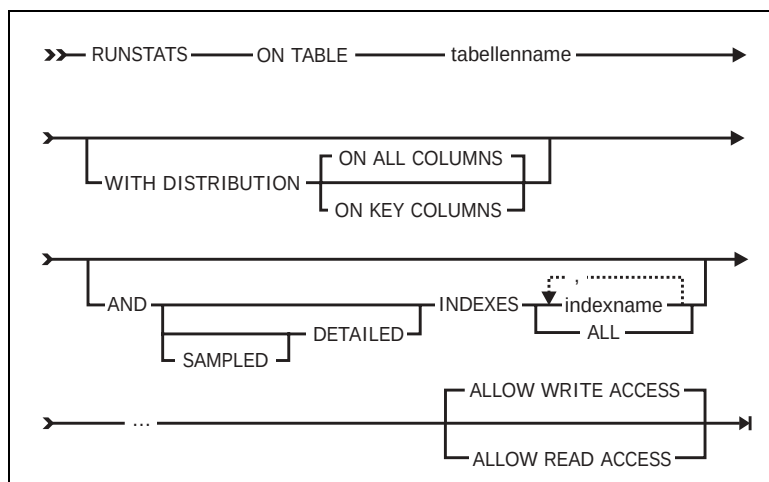


Abbildung 13.6: runstats-Syntaxdiagramm

Die Optionen aus Abbildung 13.6 haben folgende Bedeutungen:

- **tabellenname** – Hier muss der vollqualifizierte Tabellename, also in der Form „Schema-Name.Tabellenname“ angegeben werden.

- ▶ **WITH DISTRIBUTION** – ist die Option, um die Werteverteilung zu berücksichtigen. Dabei werden standardmäßig alle Spalten mit einbezogen. Wahlweise können Sie die Spalten aussuchen (in Abbildung 13.6 nicht explizit gezeigt) oder nur die Schlüsselspalten (**ON KEY COLUMNS**) berücksichtigen.
- ▶ **AND ... INDEXES** – ermöglicht es, Statistikinformationen für einen, mehrere oder alle Indizes sammeln zu lassen. Dabei erhöht die Option **DETAILED** den Detaillierungsgrad, und **SAMPLED** ermöglicht die stichprobenartige Auswertung und Hochrechnung für eine schnellere Durchführung des Statistiklaufs.
- ▶ **ALLOW WRITE ACCES** – erlaubt vollen Zugriff während der Statistiksammlung, während bei **ALLOW READ ACCESS** nur der lesende Zugriff möglich ist.

So sieht der **RUNSTATS**-Befehl mit dem höchsten Detaillierungsgrad auf die Bestellungstabelle aus:

```
runstats on table produktion.bestellung  
with distribution  
and detailed indexes all
```

Tipp:

Ob die Statistiken für eine Tabelle mit **RUNSTATS** gesammelt wurden, sehen Sie daran, dass die **SYSSTAT.TABLES**-Sicht kaum noch „1“-Werte für die jeweilige Tabelle besitzt. Über die Spalte **STATS_TIME** der **SYSCAT.TABLES**-Sicht können Sie ablesen, wann das letzte Mal ein **RUNSTATS** auf diese Tabelle durchgeführt wurde.

Die Statistiken sind nur so gut wie die Daten. Da das Pflegen der Statistiken in der Praxis häufig vernachlässigt wird, müsste es eigentlich heißen: Die Statistiken sind *maximal* so gut wie die Daten.

Was man tun kann, um die Daten zu verbessern, sehen Sie in den nächsten Abschnitten.

13.1.2 Reorgchk

Die Reorganisation – also die Neu-Anordnung – der Daten(-reihenfolge) ist eine Möglichkeit, einen effizienteren Zugriff zu ermöglichen. Ob eine Reorganisation für bestimmte Tabellen oder Indizes notwendig ist, können Sie mit dem **REORGCHK**-Tool herausfinden.

Dieses Tool arbeitet auch mit Statistikinformationen, um festzustellen, ob die Reorganisation einer Tabelle (oder eines Index) notwendig ist.

Dabei spielt die Clustering-Eigenschaft, die wir schon im Zusammenhang mit dem „Clustered Index“ (siehe Kapitel 7.5.2) besprochen haben, eine große Rolle. Kurz gesagt ging es dabei darum, die Daten physikalisch in der Reihenfolge eines ausgewählten Index anzuordnen, um einen effizienten E/A-Zugriff gewährleisten zu können. Sind die Daten in der benötigten Reihenfolge, werden zusammengehörige Daten schnell über wenige E/A-Zugriffe gelesen. Diese Clustering-Eigenschaft der

Daten zu einem Index wird in verschiedenen Statistikwerten (z.B. CLUSTERFACTOR oder CLUSTERRATIO) in den Systemkatalog-Tabellen gespeichert und geht auch in die Zugriffsberechnungen des Optimizers mit ein.

Ein weiterer Punkt ist die Freigabe von Speicherplatz. Werden in einer Tabelle Zeilen gelöscht, werden diese aus Performance-Gründen nicht sofort entfernt, sondern nur als gelöscht markiert. Um diesen eigentlich frei gewordenen Platz tatsächlich nutzen zu können, muss erst eine Reorganisation durchgeführt werden. Auch dazu findet das REORGCHK-Tool entsprechende Hinweise in den Systemtabellen.

Diese Prüfung, ob eine Reorganisation notwendig ist, führt das REORGCHK-Tool durch. Allerdings gibt es für dieses Tool keine grafische Schnittstelle, und so müssen wir mit der Befehlsvariante vorlieb nehmen, die allerdings nur wenige Parameter hat und sehr einfach zu handhaben ist.

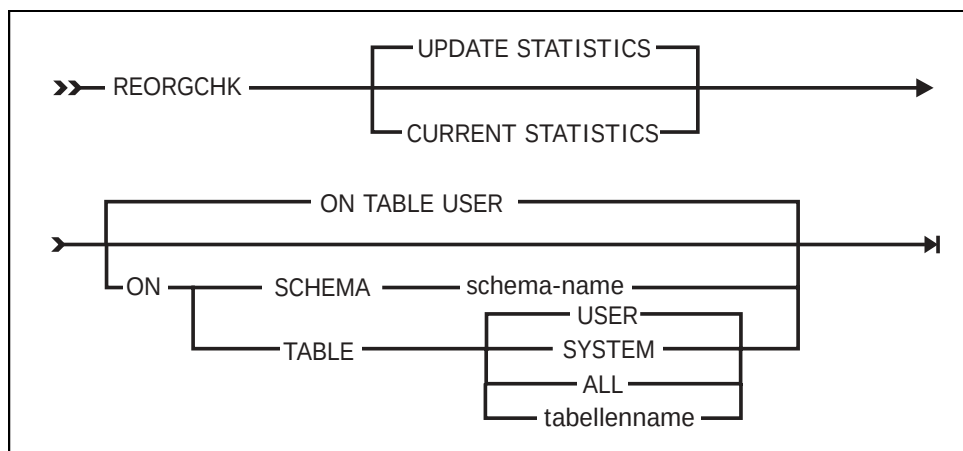


Abbildung 13.7: reorgchk-Syntaxdiagramm

Das Syntaxdiagramm (siehe Abbildung 13.7) zeigt, dass das REORGCHK-Tool auch gleich Statistikinformationen sammeln kann. Weiterhin kann es auf die Tabellen des aktuellen Benutzers (ON TABLE USER), auf ein bestimmtes Schema (SCHEMA) oder sogar auf eine bestimmte einzelne Tabelle eingeschränkt werden. TABLE SYSTEM bedeutet, dass nur Reorganisationsinformationen über die Systemkatalog-Tabellen ausgegeben werden. Mit der Option ALL werden alle Tabellen und Indizes in der Datenbank überprüft.

Tipp:

Da mit REORGCHK auch gleich die entsprechenden Statistiken für die gewählten Tabellen gepflegt werden können, ist dies eine Möglichkeit, mit einem einzelnen Befehl die Statistiken für mehrere Tabellen zu sammeln, während das Tool RUNSTATS zwar verschiedene Detaillierungsgrade kennt, aber für jede Tabelle extra aufgerufen werden muss.

Wie Sie dem Syntaxdiagramm aus Abbildung 13.7 entnehmen können, ist keine Option erforderlich, und im einfachsten Fall reicht ein einfaches REORGCHK aus. Die Ausgabe ist dann etwas umfangreicher und durchaus detailliert, wie Abbildung 13.8 zeigt.

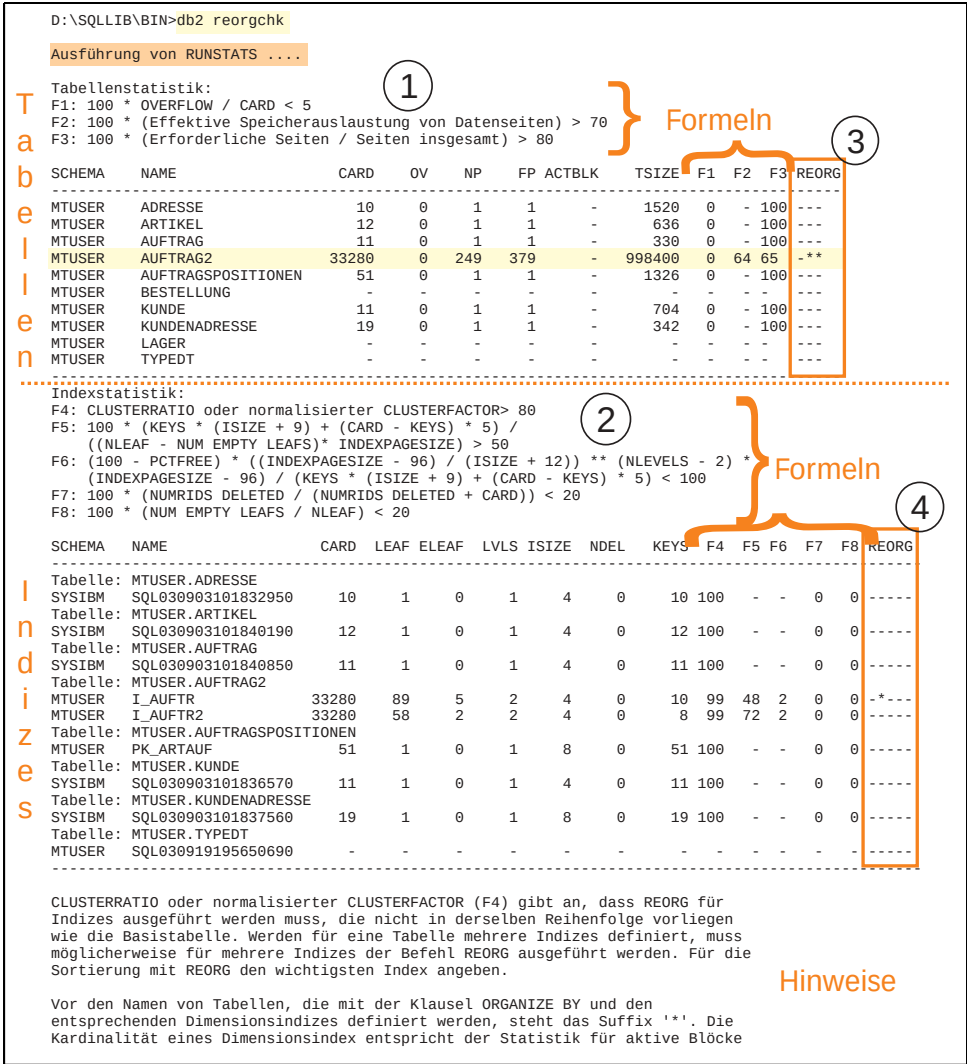


Abbildung 13.8: Beispiel für eine Ausgabe von REORGCHK

Standardmäßig wird zuerst ein RUNSTATS ausgeführt. Im ersten Teil der Ausgabe (bei Punkt ①) werden zuerst verschiedene Kennzahlen (F1 bis F3) durch spezielle Formeln berechnet und in der Tabelle ausgegeben. Diese haben alle mit der Speicherseitenbelegung zu tun und können ein Hinweis dafür sein, ob die Daten nach einer Reorganisation weniger Platz auf der Platte einnehmen werden. Zusätzlich finden Sie noch andere Statistikwerte in den Ergebnistabellen aufgelistet.

Folgende Inhalte sind dabei besonders interessant:

- ▶ CARD zeigt die Anzahl der Datensätze in der Tabelle an.
- ▶ NP (NPAGES) zeigt die Anzahl der Seiten an, die Daten beinhalten. Sie können sich das vielleicht besser merken, wenn Sie sich das N von NPAGES als Abkürzung für „Netto“ merken.
- ▶ FP (FPAGES) zeigt die Anzahl der Seiten an, die der Tabelle zugeordnet sind.
- ▶ TSIZE ist die Tabellengröße in Bytes.

Im Punkt ② werden Formeln für die Indexstatistiken und -Auswertungen gezeigt, die ebenfalls dann in der Ausgabetabelle aufgeführt sind.

Der letzte Abschnitt besteht dann noch aus verschiedenen Hinweisen.

Wirklich wichtig sind die Ergebnisse unter Punkt ③ und Punkt ④ der Abbildung 13.8. Dort zeigen „*“ an, wie wichtig und dringend eine Reorganisation ist. Ist der Eintrag ein Strich „–“, ist alles in Ordnung. Je mehr „*“ auftauchen, desto notwendiger ist die Reorganisation dieses Objekts. Bei zwei oder drei Sternchen sollte die Reorganisation dringend durchgeführt werden.

13.1.3 Reorganisation von Tabellendaten

Nachdem Sie mit REORGCHK herausgefunden haben, welche Tabellen oder Indizes reorganisiert werden sollten, schauen wir uns diesen Schritt etwas genauer an.

Mit dem REORG-Tool wird der nicht mehr benötigte Speicherplatz wieder freigegeben, und die Daten werden nach einem Index angeordnet, falls einer angegeben bzw. vorhanden ist.

Besonders interessant ist die in DB2 UDB V8 hinzugekommene Möglichkeit, die Reorganisation „im laufenden Betrieb“ – also „online“ – durchzuführen. Diese Technik bietet gleich noch einen weiteren Vorteil, da damit die Reorganisation „am Platz“, d.h. im Tabellenbereich der Tabelle erfolgen kann. Während für die „Offline“-Reorganisation die Tabelle kopiert wird und viel zusätzlicher Platz (etwa das Zwei- bis Dreifache der Tabelle) notwendig ist, werden bei der „Online“-Variante nur wenige zusätzliche Seiten benötigt. Besonders in Data Warehouse-Umgebungen scheiterte die Reorganisation von großen Tabellen oft am mangelnden Platz für die temporären Zwischendaten.

Sie können sich diese Online-Funktionalität, die durch die Option INPLACE aktiviert wird, folgendermaßen vorstellen (siehe Abbildung 13.9):

- ▶ Im ersten Schritt wird Platz geschaffen, indem die als gelöscht markierten Datensätze entfernt werden und gültige Datensätze temporär auf andere Seiten ausgelagert werden.
- ▶ Auf dem frei gewordenen Speicherplatz werden im Schritt ② die Datensätze geordnet – d.h. in der Sortierung des angegebenen Index zurückgeschrieben.
- ▶ Der letzte Schritt (Punkt ③) zeigt die reorganisierte Struktur. Dabei wurde der FREESPACE-Parameter berücksichtigt, der auf jeder Seite eine bestimmte definierbare Menge an freiem Platz lässt, um neue Daten dann an der „richtigen“ Stelle

hinzufügen zu können (siehe auch Abschnitt 7.1.5, der dieses Prinzip bei Indexseiten zeigt). Außerdem werden insgesamt weniger Speicherseiten benötigt, da der Platz der gelöschten Datensätze nun freigegeben worden ist.

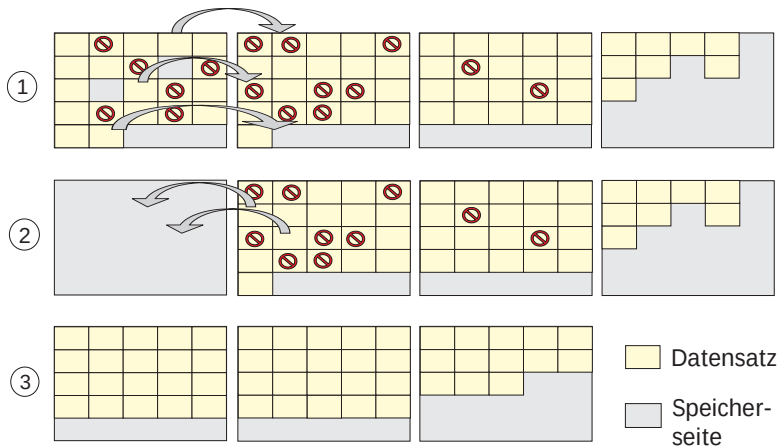


Abbildung 13.9: Reorganisation mit der INPLACE-Option

Der Aufruf dieses Tools erfolgt über die Steuerzentrale oder alternativ als Befehl von der Befehlszeile.

In der Steuerzentrale finden Sie die REORGANISIEREN...-Auswahl im Kontextmenü der jeweiligen Tabelle. Im TABELLE REORGANISIEREN-Fenster, das nach der Auswahl erscheint (siehe Abbildung 13.10), finden Sie zwei Registerkarten. Auf der ersten können Sie Einstellungen für die Reorganisation vornehmen, und auf der zweiten finden Sie die gewohnten Möglichkeiten der Ausführungssteuerung.

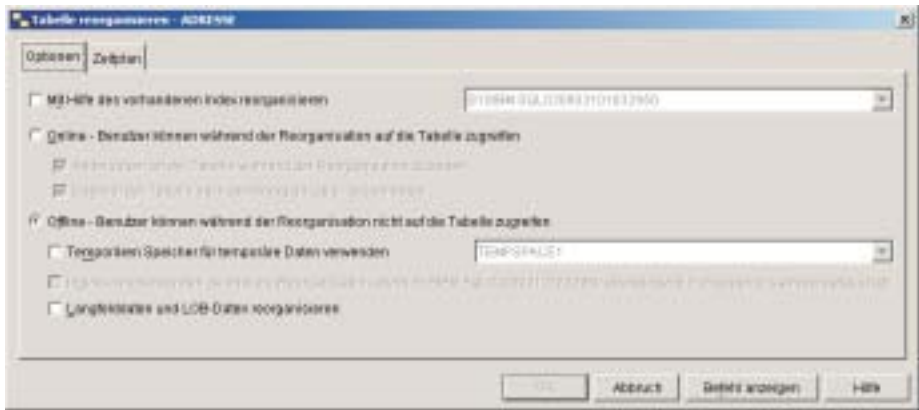


Abbildung 13.10: Fenster Tabelle reorganisieren

Die Option MIT HILFE DES VORHANDENEN INDEX REORGANISIEREN bezieht sich auf die Reihenfolge, in die die Daten gebracht werden sollen. Dieser Punkt steht nur zur Auswahl, wenn ein Index vorhanden ist. Wird diese Option *nicht* ausgewählt, wird

als Sortiervorgabe der Clustered Index auf diese Tabelle verwendet, falls ein solcher vorhanden ist. Anderenfalls wird keine Sortierreihenfolge vorgegeben, und es wird nur der nicht mehr benötigte Platz freigegeben.

Online bedeutet, dass Benutzer während der Reorganisation auf die Daten zugreifen und diese (optional) sogar verändern können. Diese zusätzlichen Änderungen resultieren dann aber in einer längeren Laufzeit der Reorganisation. Die Option DATEN IN DER TABELLE NACH DER REORGANISATION ABSCHNEIDEN ist sehr schlecht übersetzt und im Handbuch kaum beschrieben. Sie bedeutet letztendlich, dass die durch die Reorganisation frei gewordenen Seiten tatsächlich freigegeben werden und nicht mehr der Tabelle zugeordnet bleiben. Während dieses Vorgangs wird die Tabelle mit einem so genannten „Share“-Lock gesperrt, das kurzfristig keine Änderungen durch andere Anwendungen erlaubt. Wird diese Option *nicht* gewählt, kann nach der Reorganisation der eigentlich frei gewordene Platz im Tabellenbereich nicht für andere Tabellen verwendet werden.

Offline stellt die „klassische“ Reorganisationsmethode dar, die der Reorganisation vor DB2 UDB V8 entspricht. Dabei sollte immer die Option TEMPORÄREN SPEICHER FÜR TEMPORÄRE DATEN VERWENDEN genutzt werden, bei der der temporäre Tabellenbereich angegeben wird, in dem dann die Reorganisation stattfindet. Wird diese Option nicht verwendet, findet die Reorganisation in dem Tabellenbereich statt, in dem die Tabelle liegt. Dies kann schnell zu einem Überlaufen des Tabellenbereichs führen, wenn dort die temporären Daten, die ja das Zwei- bis Dreifache der Tabellendaten ausmachen können, keinen Platz finden. Dies sollte unbedingt vermieden werden, und für diesen Zweck wurde ja ein temporärer Tabellenbereich bereits bei der Datenbankerstellung angelegt.

Wenn Sie auf der zweiten Registerkarte JETZT AUSFÜHREN... WÄHLEN, können Sie sich auch den REORG-Befehl über den Button BEFEHL ANZEIGEN anschauen. Abbildung 13.11 zeigt die zugehörige Syntax des REORG-Befehls, wobei hier nur Optionen für die Reorganisation von Tabellen gezeigt werden. Die Reorganisation der Indizes besprechen wir im nächsten Abschnitt.

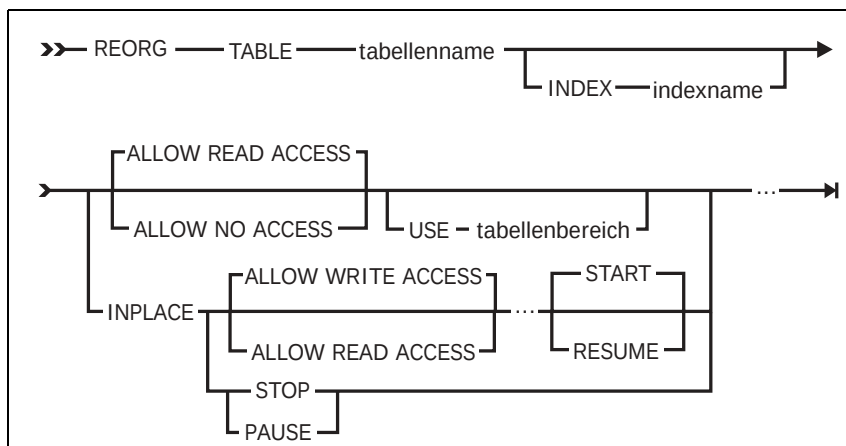


Abbildung 13.11: reorg table-Syntaxdiagramm

Nach dem Tabellennamen sollte der Indexname angegeben werden, nach dem die Daten sortiert – also „geclustert“ – werden sollen. Wird keiner angegeben, wird ein Clustered Index verwendet, falls einer definiert ist.

Ohne die INPLACE-Option, das bedeutet: im „Offline“-Fall, werden die Daten im Tabellenbereich der Tabelle umsortiert. Wurde über die USE-Option ein temporärer Tabellenbereich angegeben, wird der zusätzliche Platz dort belegt.

Der „Ast“ mit der INPLACE-Option stellt die Online-Variante dar, die ja auch „am Platz“ stattfindet. Die Reorganisation lässt sich so starten und stoppen sowie eine Zeit lang unterbrechen (PAUSE) und wieder fortsetzen (RESUME). Dabei wird sozusagen gleich von Beginn der Reorganisation an die Cluster-Eigenschaft Schritt für Schritt verbessert. Da aber diese Reorganisation eine zusätzliche Systemlast darstellt, die Sie vielleicht nicht zusätzlich während der Hauptlast auf der Datenbank haben wollen, könnten Sie diese Reorganisationstätigkeit pausieren lassen und zu einem späteren Zeitpunkt, z.B. nachts, per RESUME weiterlaufen lassen.

Hinweis:

Während bei der Offline-Variante die Reorganisation im Wiederherstellungsfall neu aufgesetzt werden muss, da alle temporären Zwischendaten verloren gehen, ist die Online-Variante durch die INPLACE-Option voll wiederherstellungsfähig. Das heißt für Sie einerseits, dass Ihnen auch bei einem Absturz keine Daten verloren gehen, die gerade umsortiert wurden. Andererseits entsteht durch diese Online-Reorganisation ein erhöhter Protokollierungsaufwand. Sie sollten also unbedingt einen ausreichenden Protokollplatz (Anzahl und Größe der Protokolldateien) zur Verfügung stellen, bevor Sie diese Online-Reorganisation durchführen.

Bitte beachten Sie, dass dadurch auch die „Restart“-Zeiten länger werden können, also die Zeit, die Sie nach einer Fehlersituation benötigen, bis die Datenbank wieder voll hergestellt ist.

Neben den Daten können mit DB2 UDB V8 auch Indizes reorganisiert werden.

13.1.4 Reorganisation von Indizes

Diese Funktionalität ist in DB2 UDB V8 neu, da mit dieser Version auch ein neuer Indextyp eingeführt wurde: Typ-2-Indizes. Diese Indizes haben einige Vorteile. Unter anderem können Sie mit ihnen gelöschte Inhalte (wie bei Tabellendaten) als gelöscht markieren, ohne diese gleich physikalisch entfernen zu müssen. Man nennt dies auch Pseudo-Löschung. Dies hat den Vorteil, dass die teilweise aufwändigen Umstrukturierungen des Indexbaums zu einem späteren Zeitpunkt – bei der Reorganisation des Index – erfolgen und nicht während der Löschung selbst. Zusätzlich sind dadurch auch weniger Sperrungen der Indexstruktur und so letztendlich der Datenzugriffe nötig. Bis auf einen minimal größeren Speicherbedarf gibt es keine Nachteile. Einige der „Online“-Tools, wie z.B. die Online-Reorganisation oder das Online-Load, setzen diese neue Indexstruktur zwingend voraus.

Die Index-Reorganisation dient somit:

- ▶ der Freigabe von Speicherplatz, also dem physikalischen Löschen der als gelöscht markierten Inhalte.
- ▶ der Neuordnung der Indexstruktur – die logische Reihenfolge eines Index ist immer gegeben, allerdings können sich durch viele Veränderungen auch Fragmentierungen der Indexseiten ergeben.
- ▶ der Wiederherstellung des mit PCTFREE konfigurierten freien Platzes auf jeder Indexseite, um neu hinzugefügte Daten am „richtigen“ Ort abspeichern zu können – also auf der Index-Datenseite, die der Sortierung entspricht.

Eine zusätzliche Eigenschaft der Indexreorganisation ist die Möglichkeit, einen Index vom Typ 1 auf den neuen Typ 2 umzustellen. Dieser Schritt wird typischerweise bei der Migration auf DB2 UDB V8 durchgeführt. Wurde die Datenbank mit DB2 UDB V8 neu erstellt, sind alle Indizes automatisch in der neuen Typ-2-Struktur.

Schauen wir uns zuerst den Aufruf über die Steuerzentrale an, bevor wir zum REORG-Befehl mit den Optionen für die Index-Reorganisation kommen.

Wie die Reorganisation der Tabellen erfolgt der Aufruf in der Steuerzentrale auf Tabellen-Ebene durch den Eintrag INDIZES REORGANISIEREN... im Kontextmenü der Tabelle.

Abbildung 13.12 zeigt das Fenster INDIZES REORGANISIEREN mit seinen Einstellmöglichkeiten.

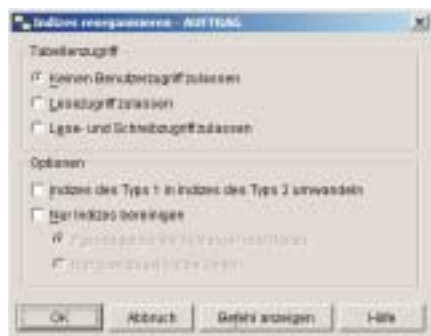


Abbildung 13.12: Fenster Indizes reorganisieren

Im TABELLENZUGRIFF-Teil des Fensters können Sie festlegen, welche Aktionen während der Reorganisation erlaubt sein sollen.

Folgende Optionen stehen Ihnen darüber hinaus zur Verfügung:

- ▶ INDIZES DES TYP 1 IN INDIZES DES TYP 2 UMWANDELN
Diese Umwandlung ist nur *einmal* bei der Migration von DB2 auf DB2 UDB V8 notwendig – danach ist dieser Punkt ohne Bedeutung.

► NUR INDIZES BEREINIGEN – Dabei gibt es zwei Alternativen:

– PSEUDOGELÖSCHTE SCHLÜSSEL UND SEITEN

Mit dieser Option werden alle Daten, die als „gelöscht markiert“ sind, entfernt, und Seiten, die dadurch komplett leer werden, werden freigegeben. Können Indexseiten durch diesen neu frei gewordenen Platz zusammengelegt werden, wird auch dies durchgeführt.

– NUR PSEUDOGELÖSCHTE SEITEN

Diese Ausprägung gibt nur Seiten frei, die nur mit „gelöscht markierten“ Inhalten belegt sind.

Die zugehörige Befehlssyntax ist in Abbildung 13.13 gezeigt. Dabei wurde der REORG-Befehl nur auf die Belange der Index-Reorganisation gekürzt.

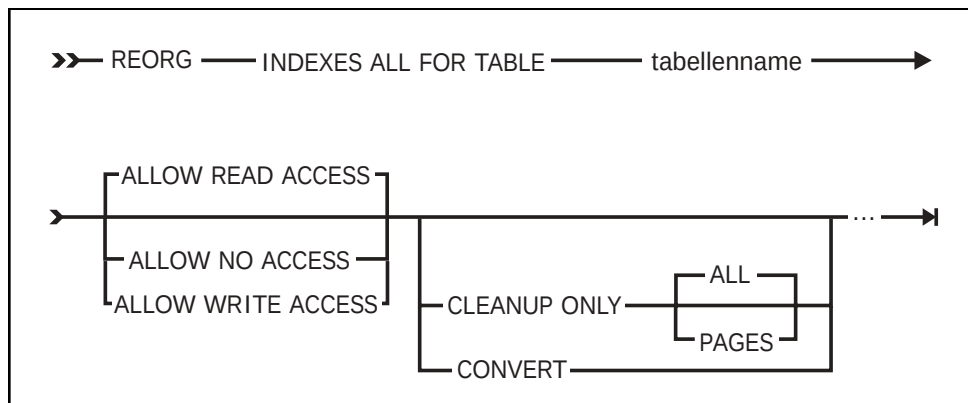


Abbildung 13.13: reorg indexes-Syntaxdiagramm

Die Optionen für die Index-Reorganisation beziehen sich immer auf *alle* Indizes einer Tabelle. Neben der Festlegung der Zugriffsmöglichkeiten kann hier auch angegeben werden, ob „nur“ als „gelöscht markierte“ Einträge bereinigt werden sollen (CLEANUP ONLY) und in welchem Umfang dies geschehen soll (ALL oder nur PAGES, also auf Seitenbasis) oder ob zusätzlich die physikalische Indexreihenfolge wiederhergestellt werden soll. Dieser letzte Fall ist der Standard und hat daher keine extra Option.

CONVERT ist – wie oben erwähnt – nur die einmalige Konvertierung der Indexstruktur in das Typ-2-Format (siehe auch den Abschnitt „Migration“ im Anhang).

13.1.5 Zusammenspiel und Abhängigkeiten

Nachdem wir diese wichtigen Pflege-Tools vorgestellt haben, stellt sich noch die Frage, wann welches Tool auszuführen ist und vor allem in welcher Reihenfolge. Denn die Reihenfolge stellt tatsächlich den entscheidenden Faktor dar, um den maximalen Nutzen aus diesen Maßnahmen zu ziehen.

Handlungsbedarf besteht, wenn sich Änderungen ergeben, die für die Bestimmung des Zugriffswegs durch den Optimizer relevant sind. Dies sind:

- ▶ Änderungen an den Objekten bzw. am Datenmodell der Datenbank
Neue Tabellen oder Indizes können alternative Zugriffswege ermöglichen und so eventuell eine Leistungssteigerung durch einen effizienteren Zugriff bewirken.
- ▶ Änderungen der Datenmengen
Je mehr Daten in einer Tabelle sind, desto wichtiger und günstiger wird ein Indexzugriff im Vergleich zu einem Tabellen-Scan – also im Vergleich zum sequenziellen Lesen der Tabelle. Bei sehr geringen Datenmengen ist allerdings dieses sequenzielle Lesen schneller als der Zugriff über einen oder sogar mehrere Indizes.
- ▶ Änderungen der Konfiguration
Durch die dynamische Änderung der Konfiguration können sich sogar im laufenden Betrieb wesentliche Datenbankeinstellungen ändern. Mehr Speicherplatz für das Caching der Daten kann speicherintensivere Zugriffe verbessern und so eventuell den eher rechenzeitintensiven, also CPU-lastigen Zugriffen überlegen sein.

Werden also größere Änderungen durchgeführt – d.h. sind über 10% der Daten betroffen oder wurde ein Parameter verändert –, besteht Handlungsbedarf, um eine optimale Performance erzielen zu können.

Anhand von Abbildung 13.14 wollen wir uns die Reihenfolge der Tool-Ausführung erschließen.

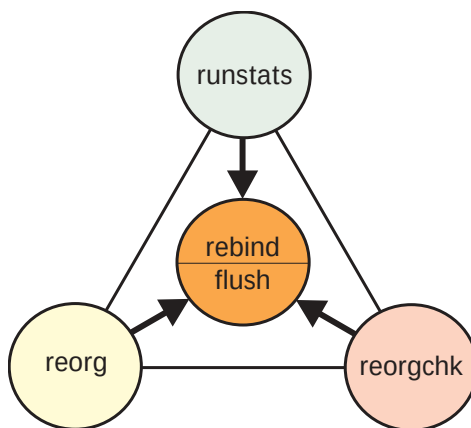


Abbildung 13.14: Zusammenspiel und Reihenfolge der Pflege-Tools

Die Statistiken sind der Ausgangspunkt. Diese werden vom REORGCHK-Tool genutzt, um festzustellen, welche Tabellen reorganisiert werden sollten. Man beginnt also mit einem RUNSTATS, um aktuelle Statistiken zu bekommen. Alternativ können Sie auch die Statistiken gleich durch das REORGCHK-Tools sammeln lassen, was aber keine Einstellung der Granularität bzgl. der zu sammelnden Statistiken zulässt. Durch das Ergebnis des REORGCHK wissen Sie dann, welche Tabellen bzw. Indizes zu reorganisieren sind.

So weit, so gut, aber wie Abbildung 13.14 zeigt, sind Sie damit noch nicht fertig. Die Befehle `REBIND` und `FLUSH PACKAGE CACHE` sind noch zentrale Punkte, um diesen „Pflege-Durchgang“ abzuschließen.

Dabei dient ein so genanntes `BIND` oder `REBIND` der erneuten Festlegung der Zugriffswege (durch den Optimizer) für statische SQL-Programme. Die Details dazu finden Sie in Kapitel 11.

Für dynamisches SQL bewirkt ein `FLUSH PACKAGE CACHE DYNAMIC`, dass die zwischengespeicherten Zugriffswege bei der nächsten Ausführung neu berechnet und festgelegt werden, um von der neuen Konfiguration und den neuen Statistikinformationen profitieren zu können.

Dieser erneute `BIND`-Vorgang bzw. ein `REBIND` sollte alle diese Pflege-Vorgänge abschließen, damit von diesem neuen und aktuellen Zustand profitiert werden kann.

13.2 Wartung und Problemanalyse

Dieser Abschnitt soll Ihnen zeigen, welche Möglichkeiten Ihnen als Datenbankadministrator zur Verfügung stehen, um in Fehlersituationen eine eigene Diagnose stellen zu können. Während das Monitoring später beschrieben wird, konzentrieren wir uns hier auf Fehlerprotokolle und andere Hilfsmittel. Als weiteren Punkt zeigen wir Ihnen, wie Sie vorgehen sollten, um bei Fehlern Hilfe von IBM zu bekommen.

13.2.1 Benachrichtigungsprotokoll für die Systemverwaltung (notification log)

Standardmäßig werden in einem Exemplar Systemverwaltungsprotokolle geführt, die Fehlermeldungen in dem Exemplar aufzeichnen.

Das „notification log“ oder „Benachrichtigungsprotokoll für die Systemverwaltung“ ist dabei für Sie als Administrator gedacht. Mit DB2 UDB V8 wurde dieses Protokoll neu eingeführt, um Ihnen eine besser dokumentierte und lesbarere Struktur zu bieten. Auch die Sprachumgebungen werden unterstützt, das heißt, die Fehlermeldungen werden Ihnen auf Deutsch angezeigt.

In diesem Protokoll sind durchaus Fehlercodes enthalten, die einer weiteren Untersuchung lohnen.

Über zwei Parameter der Exemplar-Konfiguration haben Sie Einfluss auf dieses Benachrichtigungsprotokoll:

► NOTIFYLEVEL

Dieser Wert gibt den Detaillierungsgrad an, in dem die Diagnosedaten von DB2 gesammelt werden. Sie haben die Auswahl zwischen fünf Stufen (0-4), wobei bei `NOTIFYLEVEL=0` keine Informationen geschrieben werden und `NOTIFYLEVEL=4` den höchsten Detaillierungsgrad darstellt, bei dem auch informative Meldungen geschrieben werden. Der Standardwert ist 3. Alle wichtigen Informationen, Warnungen und Fehler werden hierbei protokolliert.

▶ DIAGPATH

Über den DIAGPATH können Sie für Unix-Systeme spezifizieren, in welches Verzeichnis diese Diagnosehinweise gespeichert werden sollen. Standardmäßig wird die Datei unter Unix in /sqllib/db2dump/ mit der Namenskonvention <exemplareigner>.nfy gespeichert.

In Windows-Systemen existiert diese Datei *nicht*. Stattdessen wird das Ereignisprotokoll des Windows-Betriebssystems genutzt. Sie erreichen es über START/EINSTELLUNGEN/SYSTEMSTEUERUNG/VERWALTUNG/EREIGNISANZEIGE.

In Abbildung 13.19 sehen Sie ein Beispiel für das Benachrichtigungsprotokoll einer Windows-Umgebung.

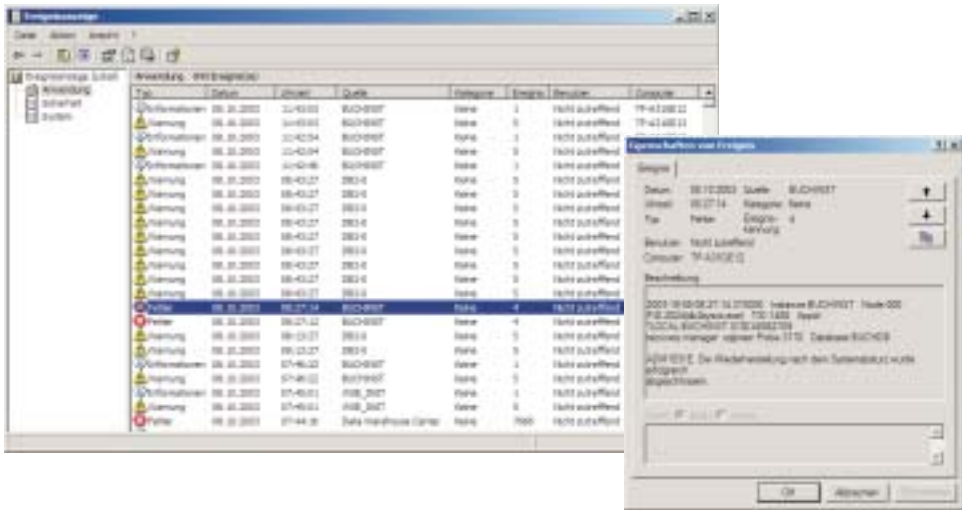


Abbildung 13.15: Ereignisanzeige unter Windows

Während Sie im Überblicksfenster die Art der Meldung (Information, Warnung und Alarm) anhand der Icons sehen können, bekommen Sie auf Doppelklick die Details zu dieser Meldung, mit Datum, Uhrzeit, Quelle etc. angezeigt.

Hinweis:

Die Fehlercodes können Sie in den DB2-Handbüchern nachschlagen.

In Abbildung 13.16 sehen Sie den Inhalt des Benachrichtigungsprotokolls einer englisch installierten Linux-Maschine. Dieser Datei-Auszug besteht aus fünf Einträgen, die durch die gestrichelte Linie verdeutlicht wurden. Neue Einträge werden immer hinten an die Datei angefügt. Das heißt für Sie, dass Sie den neuesten Eintrag am Ende der Datei finden. Von dort blättern Sie etwas zurück und lesen sich dann die Meldungen von diesem Punkt bis zum Ende durch.

... Zeitstempel		auslösendes Exemplar	
1	2003-08-24-14.09.35.107815	Instance:db2inst1	Node:000
	PID:927(db2star2)	TID:1024	Appid:none
	base sys utilities	startdbm	Probe:911
	ADM7513W	Database manager has started.	Meldung
2	2003-08-24-14.19.35.672457	Instance:db2inst1	Node:000
	PID:1839(db2agent (SAMPLE))	TID:1024	Appid:*LOCAL.db2inst1.00BF14121933
	base sys utilities	sqlDatabaseQuiesce	Probe:1
	ADM7506W	Database quiesce has been requested.	betroffene Datenbank
3	2003-08-24-14.19.35.988697	Instance:db2inst1	Node:000
	PID:1839(db2agent (SAMPLE))	TID:1024	Appid:*LOCAL.db2inst1.00BF14121933
	base sys utilities	sqlDatabaseQuiesce	Probe:2
	ADM7507W	Database quiesce request has completed successfully.	Anwendung
4	2003-08-24-14.19.36.031905	Instance:db2inst1	Node:000
	PID:1839(db2agent (SAMPLE))	TID:1024	Appid:*LOCAL.db2inst1.00BF14121933
	base sys utilities	sqlDatabaseUnquiesce	Probe:1
	ADM7510W	Database unquiesce has been requested.	
5	2003-08-24-14.19.36.155557	Instance:db2inst1	Node:000
	PID:1839(db2agent (SAMPLE))	TID:1024	Appid:*LOCAL.db2inst1.00BF14121933
	base sys utilities	sqlDatabaseUnquiesce	Probe:2
	ADM7509W	Database unquiesce request has completed successfully.	

Abbildung 13.16: Beispiel-Inhalt eines Linux-Benachrichtigungsprotokolls

Jeder Eintrag beginnt mit einem Zeitstempel. Die Einträge können unterschiedlich lang sein und enthalten meist den Exemplarnamen, den Datenbanknamen und den Anwendungsnamen. Die Meldungen mit den zugehörigen Fehlercodes können Sie auch in der DB2-Dokumentation nachlesen.

Zusätzlich sind noch die Prozesse und Thread-IDs angegeben, die die jeweilige Meldung ausgelöst haben, und häufig noch die betroffene Komponente im Datenbanksystem.

Sie brauchen vielleicht etwas Übung, um diese Strukturen zu lesen und sich in den Meldungen zurechtzufinden, aber nach einer Weile sind diese Meldungen sehr hilfreich und können oft den entscheidenden Hinweis auf den Fehler geben.

13.2.2 Diagnoseprotokoll db2diag.log

Dieses Diagnoseprotokoll war bis DB2 UDB V8 die erste Anlaufstelle für den Administrator. Durch das zusätzliche Benachrichtigungsprotokoll (notification log) hat diese Datei aber etwas an Nutzen verloren. Die Einträge sind nicht so „sprechend“, und die enthaltenen Informationen sind für die IBM-Benutzerunterstützung gedacht. Trotzdem kann ein Blick in diese Datei nichts schaden, wenn Sie an anderer Stelle nicht weiterkommen.

Auch hier können Sie die Protokollierung über zwei Parameter in der Exemplar-Konfiguration steuern:

► DIAGLEVEL

Dieser Parameter legt – genau wie NOTIFYLEVEL – den Detaillierungsgrad der zu protokollierenden Informationen fest. Auch hier ist der Standardwert 3, und der Wertebereich geht ebenfalls von 0 bis 4.

► DIAGPATH

Diesen Parameter haben Sie schon beim Benachrichtigungsprotokoll kennen gelernt. Mit ihm können Sie das Zielverzeichnis für die Datei db2diag.log bestimmen. Die Datei existiert in Windows-Umgebungen standardmäßig unter ...\\sqlllib\\BUCHINST\\db2diag.log, wobei BUCHINST der Name des Exemplars ist. In Unix-Umgebungen entspricht der Pfad dem des Benachrichtigungsprotokolls, also .../sqlllib/db2dump/db2diag.log.

Der Inhalt ist sehr ähnlich aufgebaut wie bei dem Benachrichtigungsprotokoll im vorhergehenden Abschnitt, was ja auch kein Wunder ist, da das Benachrichtigungsprotokoll aus dem db2diag.log sozusagen „herausgelöst“ wurde. Abbildung 13.17 zeigt einen Auszug aus einer db2diag.log-Datei. Die sechs Einträge sind wieder jeweils mit einem Zeitstempel gekennzeichnet und wurden von uns durch gepunktete Linien getrennt, um sie übersichtlicher zu gestalten.

	Zeitstempel	verursachendes Exemplar
1	2003-10-08-07.46.21.347000	Instance: BUCHINST Node: 000 PID: 2024(db2syscs.exe) TID: 3296 Appid: none base sys utilities startdbm Probe: 911 ADM7513W Database manager has started.
2	2003-10-08-08.27.10.097000	Instance: BUCHINST Node: 000 PID: 2024(db2syscs.exe) TID: 1488 Appid: *LOCAL.BUCHINST.010C48062709 base sys utilities sqledint Probe: 30 Crash Recovery is needed.
3	2003-10-08-08.27.12.530000	Instance: BUCHINST Node: 000 PID: 2024(db2syscs.exe) TID: 1488 Appid: *LOCAL.BUCHINST.010C48062709 recovery manager sqlpresr Probe: 410 Database: BUCHDB betroffene Datenbank
4	2003-10-08-08.27.12.540001	Instance: BUCHINST Node: 000 PID: 2024(db2syscs.exe) TID: 1488 Appid: *LOCAL.BUCHINST.010C48062709 recovery manager sqlprecm Probe: 125 Database: BUCHDB Using parallel recovery with 3 agents 10 QSets 70 queues and 2 chunks
5	2003-10-08-08.27.13.963000	Instance: BUCHINST Node: 000 PID: 2024(db2syscs.exe) TID: 1488 Appid: *LOCAL.BUCHINST.010C48062709 recovery manager sqlprecm Probe: 400 Database: BUCHDB DIA2051W Forward phase of crash recovery has completed. Next LSN is "00000000035D3DC3".
6	2003-10-08-08.27.14.523000	Instance: BUCHINST Node: 000 PID: 2024(db2syscs.exe) TID: 1488 Appid: *LOCAL.BUCHINST.010C48062709 recovery manager sqlpresr Probe: 3170 Database: BUCHDB Crash recovery completed. Next LSN is 00000000035D3DC3

Abbildung 13.17: Beispiel-Inhalt der Diagnosedatei db2diag.log

Diese Datei sollte in Fehlersituationen unbedingt an den IBM-Support geschickt werden, doch dazu später noch mehr.

13.2.3 Integritätsprüfung mit dem INSPECT-Befehl

Mit INSPECT ist in DB2 UDB V8 eine neue Möglichkeit hinzugekommen, die Integrität der Datenbankarchitektur zu überprüfen. Im Gegensatz zu anderen Tools ist dieser Check „online“ möglich, also während Benutzer und Anwendungen in der Datenbank aktiv sind.

Dieses Tool ermöglicht es beispielsweise zu sehen, welcher Indextyp für eine Tabelle vorliegt, also ob noch der alte Typ-1-Index verwendet wird oder ob bei der Migration auf die neue Typ-2-Indexart umgestellt wurde (per REORG INDEX CONVERT-Befehl).

Wie das Syntaxdiagramm in Abbildung 13.18 zeigt, stehen folgende Überprüfungsebenen zur Auswahl:

- Datenbank-Ebene – Alternative ①
Das kann die komplette Datenbank sein oder mehrere Tabellenbereiche.
- Tabellenbereichsebene – Alternative ②
beinhaltet die Überprüfung eines einzelnen Tabellenbereichs oder mehrerer Tabellen in diesem
- Tabellen-Ebene – Alternative ③
Eine einzelne Tabelle wird überprüft.

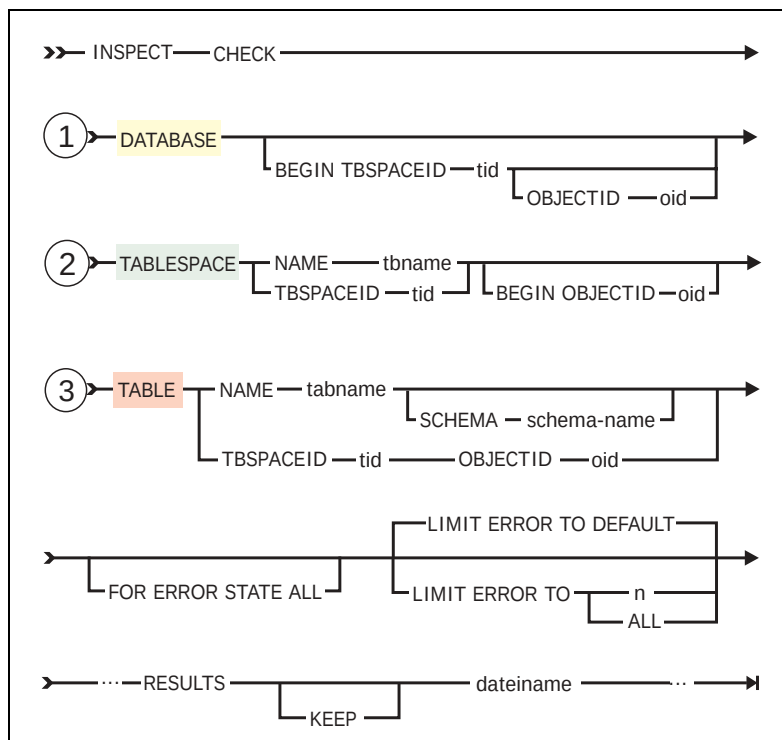


Abbildung 13.18: inspect-Syntaxdiagramm

Wichtig ist dabei, dass Sie durch diese Ebenen nur bestimmen, für welche bzw. für wie viele Objekte die Integritätsprüfungen vorgenommen werden. Es werden immer die gleichen Informationen gesammelt, entweder für alle Objekte in der Datenbank oder eben für nur eine bestimmte Tabelle.

Bevor wir zu Beispielen kommen, schauen wir uns die Optionen im Einzelnen an:

► **BEGIN TBSPACEID und TBSPACEID**

Hier wird die Tabellenbereichs-ID angegeben für die bzw. ab der die Daten gesammelt werden sollen. Diese Tabellenbereichs-ID bekommen Sie über die Systemkatalog-Sicht SYSCAT.TABLESPACES oder über den Befehl LIST TABLESPACES.

► **BEGIN OBJECTID und OBJECTID**

Die OBJECTID bezieht sich auf die Tabellen-ID (TABLEID-Spalte im Systemkatalog) für die bzw. ab der die Integritätsinformationen überprüft werden sollen. Diese Tabellen-ID bekommen Sie durch folgende SQL-Abfrage des Systemkatalogs:

```
select tabschema, tabname, tableid, tbpaceid
from syscat.tables
```

► **NAME**

Nach diesem Schlüsselwort kann der Objektname angegeben werden, also der Name des Tabellenbereichs oder der Tabelle.

► **FOR ERROR STATE ALL**

Diese Ausprägung bedeutet, dass die Integrität von Tabellen, die intern bereits einen Fehlerstatus haben, trotzdem überprüft wird. Ohne diese Option wird nur der Fehlerstatus zurückgegeben ohne Integritätsprüfung des Objekts.

► **LIMIT ERROR TO**

Mit diesem Parameter können Sie angeben, nach wie vielen fehlerhaften Speicherseiten die Überprüfung eines Objekts abgebrochen werden soll. Sie können hier entweder eine Anzahl der Seiten angeben oder per ALL spezifizieren, dass eine Überprüfung komplett stattfinden soll. Standardmäßig ist dies durch LIMIT ERROR TO DEFAULT auf die Extent-Größe beschränkt.

► **KEEP**

Diese Option ist sehr wichtig, da die Ergebnisdatei verworfen wird, falls kein Fehler gefunden und diese Option nicht angegeben wurde. Wollen Sie beispielsweise die Indexart, wie oben erwähnt, angezeigt bekommen, ist diese Option ein Muss, da weder Typ-1-Indizes noch Indizes vom Typ-2 einen Fehler darstellen und die Ergebnisse sonst gelöscht würden.

► **dateiname**

Zum Dateinamen sollte es eigentlich nichts zu sagen geben, es steht nur nirgends, dass hier *kein* Pfad angegeben werden darf und die Datei *immer* im Exemplarverzeichnis unterhalb des SQLLIB-Verzeichnisses gespeichert wird, also z.B. in ...\\sqllib\\BUCHINST\\. Die erzeugte Datei ist im binären Format und muss also noch separat aufbereitet werden; dazu gleich mehr.

Weitere Optionen für diesen Befehl bezüglich des Detaillierungsgrads und der partitionierten Umgebung wurden in Abbildung 13.18 weggelassen, da wir einerseits in diesem Buch die partitionierte Umgebung nicht näher betrachten und andererseits standardmäßig der höchste Detaillierungsgrad eingestellt ist, was grundsätzlich sinnvoll ist. Man muss es ja nicht unbedingt komplizierter machen als nötig!

Beispiele für die verschiedenen INSPECT-Ebenen wären:

```
inspect check database results keep inspect_db_buchdb.out
inspect check tablespace defaulttb results keep inspect_tb_defaulttb.out
inspect check table name auftrag results keep inspect_tab_auftrag.out
```

Die Ausgabedatei muss, bevor man sie analysieren kann, mit dem Befehl `db2inspf` aufbereitet werden. Auch dieser Befehl hat Optionen, nur eine Teilmenge der Ergebnisdatei zu formatieren, aber das können Sie schließlich auch im Editor, nachdem Sie sich alles angeschaut haben. Somit genügt folgender Befehl zur Ergebnisaufbereitung:

```
db2inspf <dateiname> <ausgabedatei>
```

Konkret könnte das so aussehen:

```
db2inspf d:\sqllib\buchinst\inspect_tb_defaulttb.out inspect_tb_defaulttb.txt
```

Wie wir beschrieben haben, sind die gesammelten Datendetails gleich, und nur der Umfang, für welche Objekte die Prüfung stattfindet, ist je nach Ebene unterschiedlich. Schauen wir uns in Abbildung 13.19 also nur den Ausschnitt so eines INSPECT-Ergebnisses an.

Nach einigen Informationen über die Datei selbst sehen Sie in Abbildung 13.19 die Tabellenbereichsinformationen mit den Behälterinformationen. Anschließend werden in den Analysedaten alle Objekte in diesem Tabellenbereich aufgeführt. Hier finden Sie auch die Informationen zum Indextyp, die Summe der der Tabelle zugeordneten Speicherseiten und die verwendeten Seiten. Auch die Informationen zu dem bzw. den zugehörigen Indizes werden an dieser Stelle aufgeführt. Anschließend kommt das nächste Objekt usw..

Hinweis:

Alternativ steht das Tool `db2dart` in DB2 zur Verfügung. Dieses kann aber nur auf inaktive Datenbanken angewendet werden. Dafür geht diese Integritätsprüfung tiefer, und es können auch Aktionen angestoßen werden, während der INSPECT-Befehl rein informativ ist.

```

DATABASE: BUCHDB
VERSION : SQL08013
2003-10-08-08.27.54.295000

Aktion: CHECK DATABASE
Ergebnisdateiname: inspect_db_buchdb.out

...

Beginn der Tabellenbereichsphase. Tabellenbereichs-ID: 3
  Tabellenbereichsname: DEFAULTTB
  Tabellenbereichstyp: SMS - System Managed Space; Speicherbereichsgröße: 16; Seitengröße: 4096; Behälteranzahl: 1
  Behältername: c:\behälter\buchdb\defaultTB\b1
  ...
  Beginn der Datenbankphase. Objekt: 2 Tabellenbereich: 3
  Der Indextyp für diese Tabelle ist 2.
  DAT Objektzusammenfassung: Summe der Seiten 1 - Verwendete Seiten 1 - Freier Speicherbereich 32 %
  Ende der Datenbankphase.
  Beginn der Indexphase. Objekt: 2 Tabellenbereich: 3
  INX Objektzusammenfassung: Summe der Seiten 3 - Verwendete Seiten 3
  Ende der Indexphase.
  Ende der Tabellenphase.
  ...
  Beginn der Tabellenbereichsphase. Tabellenbereichs-ID: 3
  ...
  Beginn der Datenbankphase. Objekt: 3 Tabellenbereich: 3
  Der Indextyp für diese Tabelle ist 2.
  DAT Objektzusammenfassung: Summe der Seiten 1 - Verwendete Seiten 1 - Freier Speicherbereich 52 %
  Ende der Datenbankphase.
  Beginn der Indexphase. Objekt: 3 Tabellenbereich: 3
  INX Objektzusammenfassung: Summe der Seiten 3 - Verwendete Seiten 3
  Ende der Indexphase.
  Ende der Tabellenphase.
  ...

```

Allgemeine
Datei-Informationen

Tabellenbereichsinformation

Tabellen-Information

Index-Information

Abbildung 13.19: Beispiel eines INSPECT-Ergebnisses

Achtung:

Mit db2dart können Sie bei falscher Anwendung Schaden in der Datenbank anrichten! Es ist nur für erfahrene Administratoren zu empfehlen. Es ist teilweise per Kennwort geschützt und kann nur gemeinsam mit dem DB2-Support verwendet werden.

Durch dieses Tool können Sie sich beispielsweise die so genannte „high water mark“, also den höchsten Füllstand eines Tabellenbereichs, ausgeben lassen.

13.2.4 Problemmeldungen und Hilfestellungen

Mit dem Kauf der DB2-Software haben Sie in den allermeisten Fällen auch eine so genannte *maintenance* mit gekauft. Das bedeutet, Sie bekommen innerhalb der Laufzeit dieser *maintenance*, die jährlich verlängert werden kann, alle neuen DB2-Versionen ohne zusätzliche Kosten. Zusätzlich haben Sie die Möglichkeit, bei Fehlern die Hilfe des DB2-Supports zu nutzen. Dieser Support – also diese Hilfestellung – bezieht sich auf Problembehebungen und nicht auf Fragestellungen wie:

- Wie kann ich dies und jenes besser machen?
- Wie sollte ein Script aufgebaut sein?

Diese Fragen kann Ihnen am besten Ihr IBM Business-Partner beantworten, der oft auch gute Fachkenntnisse über die Produkte besitzt.

Den IBM-Support erreichen Sie am besten über die DB2-Supportseite im Internet:

www.ibm.com/software/data/db2/udb/support.html

Dort können Sie z.B. unter SUBMIT & TRACK PROBLEMS Ihre Probleme eingeben. Sie brauchen dazu Ihre Kundennummer, die Sie beim Kauf der Software erhalten haben.

Bevor Sie dort Angaben machen oder die Support-Hotline nutzen, sollten Sie sich die folgenden Punkte anschauen und berücksichtigen, um kostbare Zeit zu sparen. Es ist wichtig, dass Sie genaue Angaben zu Ihrem System, Ihrer Installation und Ihrem Problem machen können. So stellen Sie sicher, dass nicht durch Nachfragen und Nachforderungen von Detailinformationen unnötig viel Zeit bis zur Problemlösung vergeht.

DB2LEVEL

Der Befehl `db2level` zeigt schnell und einfach die installierte DB2-Version an. Sie sehen zudem das aktuelle Exemplar und das Installationsverzeichnis. Musste man in DB2 UDB V7 noch den Fixpack-Stand über die Support-Webseite nachschlagen, weil nur der Release-Code (hier: WR21324) angezeigt wurde, so ist die Ausgabe mit DB2 UDB V8 vollständig.

```
D:\>db2level
DB21085I  Exemplar "BUCHINST" verwendet "32" Bit und DB2-Codefreigabe
"SQL08013" mit Aktualitäts-ID "02040106".
Informationstokens: "DB2 v8.1.3.132", "s030728", "WR21324" und FixPak "3".
Produkt ist in "D:\SQLLIB" installiert.
```

Abbildung 13.20: Beispiel für eine Ausgabe von `db2level`

Diese Informationen brauchen Sie bereits, um einen Fehler oder ein Problem zu melden.

DB2SUPPORT

Der Name sagt es schon, dieses Tool ist sehr hilfreich, wenn Sie mit dem DB2-Support Kontakt aufnehmen. Dieses Tool stellt selbstständig alle wichtigen Diagnose- und Konfigurationsinformationen für den Support in einer ZIP-Datei zusammen. Es gibt zahlreiche Optionen, um den Inhalt, der von Grund auf schon sehr gut und umfangreich ist, noch zu erweitern.

Beim Syntaxdiagramm in Abbildung 13.21 haben wir uns auf die wichtigsten Optionen beschränkt – diese sollten absolut ausreichend sein.

Die Optionen im Einzelnen:

► `ausgabepfad`

Pfad, in dem die Datei `db2support.zip` erstellt werden soll.

► `-f` oder `-flow`

Es werden Pausen und Eingabeaufforderungen unterdrückt, um dieses Tool auch aus Scripten aufrufen zu können.

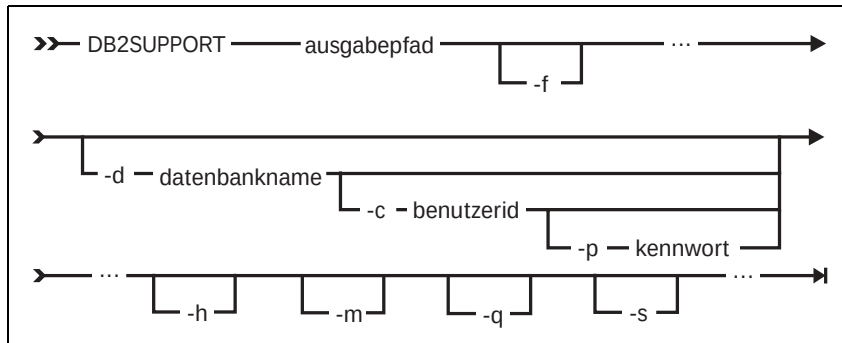


Abbildung 13.21: db2support-Syntaxdiagramm

► **-d datenbankname**

Gibt an, für welche Datenbank die Informationen gesammelt werden sollen.

► **-c benutzerid und -p kennwort**

Durch diese Option wird zusätzlich versucht, mit der angegebenen Benutzer-ID und dem Kennwort eine Verbindung zur Datenbank herzustellen. So können mit dieser Option Daten über Pufferpools, Tabellenbereiche, Tabellen, Pakete und Anwendungen etc. gesammelt werden.

► **-h**

Hilfe-Funktion, Sie bekommen die wichtigsten Optionen in der Befehlszeile angezeigt.

► **-m oder -html**

Mit dieser Option haben Sie die Möglichkeit, die Ergebnisse im HTML-Format statt als Textdateien erstellen zu lassen. Diese Option bietet somit auch eine gute Möglichkeit der Systemdokumentation für die eigenen Zwecke!

► **-q oder -question_response**

Diese Option versucht in einem interaktiven Modus Ihre Fehlersituation abzufragen. Dabei können Sie immer wieder eine Auswahl aus Menüpunkten treffen oder Ihre Fehlermeldung textuell beschreiben.

► **-s oder -system_detail**

Zusätzliche Systemdetails mit Informationen über die Hardware und das Betriebssystem werden gesammelt.

Um die Informationen zu sammeln, ist es am besten, dieses Tool von der Befehlszeile des Servers zu starten, um nicht in einem Client-Server-Umfeld versehentlich die Client-Informationen zu sammeln.

Dieses Tool bietet Ihnen eine schöne und einfache Methode, sich alle Informationen automatisch zusammenstellen zu lassen, und ist sogar durch die **-m**-Option dazu geeignet, die eigene Dokumentation des Datenbanksystems zu erweitern und sauber aufzubereiten.

TRACE

Das Trace-Tool (db2trc) kommt bei schweren Fehlersituationen zum Einsatz und dokumentiert alle Schritte auf unterster Prozess-Ebene. Die wichtigste Voraussetzung für dieses Tool ist, dass Sie den Fehler nachvollziehen – also reproduzieren – können. Ist das der Fall, haben Sie bereits gute Karten und können dieses Tool einsetzen, um auch diese Informationen – zusätzlich – zur Ausgabe des db2support-Befehls an den Support zu schicken. Unserer Erfahrung nach wird diese Information fast immer (automatisch) verlangt, wenn ein Problem an das DB2-Labor weitergereicht wird. Warum also warten, bis die Nachfrage kommt?

Für die Ausführung dieses Tools müssen Sie sich nicht viel merken, sondern folgen einfach dieser Anleitung:

1. Schalten Sie den Trace durch folgenden Befehl ein:

```
db2trc on
```

2. Versuchen Sie, den Fehler nachzuvollziehen.

3. Lassen Sie die Trace-Informationen in eine so genannte Dump-Datei (z.B. db2trc.dmp) schreiben:

```
db2trc dmp db2trc.dmp
```

4. Beenden Sie den Trace:

```
db2trc stop
```

5. Über folgenden Befehl können Sie die binäre Dump-Datei in eine Textdatei umwandeln:

```
db2trc fmt db2trc.dmp db2trc.txt
```

Das Ergebnis ist extrem umfangreich, und Sie sollten die Fehlersituation vorher so weit wie möglich einkreisen, um nur den fehlerhaften Teil in dem Trace schreiben zu lassen. Machen Sie nicht den Fehler, das Trace-Ergebnis auszudrucken – das verschwendet nur Unmengen Papier.

Der IBM-Support hat Hilfsmittel, um diese Ergebnisse auszuwerten und so hoffentlich die Ursache des Problems zu ermitteln.

Im Support-Fall kann es leicht sein, dass man zu wenige Informationen mitschickt, zu viele waren dagegen noch nie ein Problem – abgesehen von irgendwelchen Mail-Limitierungen.

14 Überwachung von DB2

14.1 Warum ist Überwachung (Monitoring) notwendig?

Denken Sie an all die Einstellungen, über die Sie bisher gelesen haben. Sind Sie sich sicher, dass diese alle „optimal“ in Ihrem Datenbanksystem eingestellt sind?

- ▶ Kennen Sie alle SQL-Abfragen, die Sie bzw. die Anwendungen gegenüber der Datenbank abgesetzt haben?
- ▶ Wie hat sich das Datenvolumen innerhalb der letzten Wochen und Monate innerhalb Ihrer Datenbank geändert?
- ▶ Welchen Einfluss hat das Datenvolumen auf die Abfragezeiten Ihrer Anwendungen bzw. der zugehörigen SQL-Abfragen?
- ▶ Hatten Sie Performance-Probleme beim Einfügen von Datensätzen?
- ▶ Gibt es Probleme, die nur zu bestimmten Zeitpunkten bzw. nur an bestimmten Tagen auftreten? Welche Anwendungen verursachen Engpässe im Datenbanksystem?
- ▶ Wie können Sie Probleme schon frühzeitig erkennen, bevor diese Ihr Datenbanksystem und somit auch die Anwendungen behindern?

Dies sind nur einige der typischen Fragen, mit denen Sie sich auseinander setzen müssen, wenn Sie eine Datenbank dauerhaft betreiben und diese von mehreren Anwendern und Anwendungen genutzt wird.

14.2 Überwachungs-Tools in DB2

In DB2 gibt es unterschiedliche Überwachungs-Tools – *Monitore*, die Sie bei der Beantwortung der oben aufgeführten Fragen unterstützen.

- ▶ Monitor des Datenbanksystems (Systemmonitor)
Dieser Monitor ist die „Mutter aller Monitore“ in DB2. Dieser DB2-Systemmonitor wird von fast allen Monitoren verwendet, mit Ausnahme des Diagnosemonitors. Die Überwachungsinformationen werden aus den Daten, die dieser Monitor sammelt, weiterverarbeitet und für den jeweiligen Zweck aufbereitet und dargestellt. Welche Daten für welche Anwendung hier gesammelt werden, kann von den einzelnen Monitoren ganz gezielt bestimmt werden.
- ▶ Monitor zur Diagnose (Diagnosezentrale bzw. health center)
Die Aufgaben des Diagnosemonitors und somit der Diagnosezentrale sind nicht nur, eine vorbeugende Überwachung Ihrer Datenbank bzw. Ihres Datenbanksystems durchzuführen, sondern auch Vorschläge zu geben, um eine „Erkrankung“

zu verhindern. Die Diagnosezentrale zeigt Ihnen, wenn ein bestimmter Schwellenwert (z.B. der Füllgrad eines DMS-Tabellenbereichs) überschritten wurde, mit welchen Maßnahmen Sie die Größe dieses Tabellenbereichs erhöhen können.

► **Monitor zum Visualisieren von genutzten Speicherbereichen (memory visualizer)**

Dieser Monitor zeigt Ihnen in grafischer Form die Auslastung bestimmter Speicherbereiche an, die Sie in der DBM- bzw. DB-Konfiguration gesetzt haben. Es wird hier aufgezeigt, welcher Wert konfiguriert wurde und wie die tatsächliche aktuelle Nutzung dieser Speicherbereiche ist. Sie bekommen einen schnellen Überblick über die aktuelle Speicherauslastung und können schnell sehen, ob es bald notwendig wird, die Konfigurationen anzupassen.

► **Monitor für Momentaufnahmen (Snapshot-Monitor)**

Um ganz gezielt Monitorinformationen auszuwerten, die vom DB2-Systemmonitor gesammelt wurden, kann der Monitor für die Momentaufnahmen verwendet werden. Dieser holt die zu diesem *Zeitpunkt* vorhandenen Daten aus dem Monitorpuffer und zeigt diese Informationen auf dem Bildschirm an oder schreibt diese in eine Datei bzw. eine Tabelle. Bei dieser Abfrage des Monitorpuffers können viele Einschränkungen angegeben werden, um die Ausgabedaten zu begrenzen.

► **Monitor für die Ereignisverfolgung (Event-Monitor)**

Wollen Sie eine Anwendung oder Aktivitäten eines bestimmten Benutzers verfolgen, so ist dieser Monitor hierfür geeignet, da er eine *Zeitraum*-Überwachung durchführt. Der Ereignismonitor (Event-Monitor) erfasst die DB2-Systemmonitordaten *nach dem Eintreten* bestimmter Ereignisse. Zum Beispiel können Sie das Ende einer Transaktion, eines SQL-Befehls oder auch das Ende einer gegenseitigen Sperre hier nachvollziehen.

► **Fehler-Monitor (nur in Unix-Systemen vorhanden)**

Mit dem Fehlermonitor kann im laufenden Betrieb überprüft werden, ob ein Exemplar ordnungsgemäß beendet wurde oder nicht. Wird durch den Fehlermonitor festgestellt, dass das zu überwachende Exemplar abgebrochen bzw. unsachgemäß beendet wurde, so startet der Fehlermonitor das Exemplar wieder neu.

► **SQL-Monitor**

Als SQL-Monitor werden auch die EXPLAIN-Tools bezeichnet, mit denen es möglich ist, sich den Zugriffsweg anzeigen zu lassen, den der DB2-Optimizer für einen SQL-Befehl auswählt. Mit diesen EXPLAIN-Tools sehen Sie, wie DB2 Ihre Daten zusammenführt (JOIN), sortiert, welche Zwischentabellen angelegt werden und welche Indizes von DB2 ausgewählt wurden, um Ihre Daten kostenoptimal zu lesen und zu verarbeiten.

14.3 Der DB2-Systemmonitor

Der DB2-Systemmonitor sammelt Daten über das aktuelle Exemplar (instance), die dazugehörigen Datenbanken und die Anwendungen, die mit diesen Datenbanken verbunden sind.

Der Systemmonitor sammelt immer bestimmte Basisdaten schon ab dem Start des Exemplars (instance), also ab einem db2start. Diese Informationen sind z.B. die Anzahl der lokalen und fernen Verbindungen zur Datenbank oder die Anzahl genutzter Agenten.

Die meisten Monitor-Detaildaten werden nicht gesammelt, um das System nicht unnötig zu belasten. Sie können aber den Detaillierungsgrad des Monitors über so genannte Monitorschalter (switches) bestimmen.

Starten Sie z.B. den Snapshot-Monitor, so erhält diese Monitorschnittstelle ihre angeforderten Informationen vom Systemmonitor (siehe Abbildung 14.1). Diese „Informationssicht“ ist absolut unabhängig von anderen Monitoren. Das heißt, die Spezifikation des Detaillierungsgrads oder das Zurücksetzen von Monitorwerten geschieht auf Ebene des jeweiligen Monitorkaufrufs.

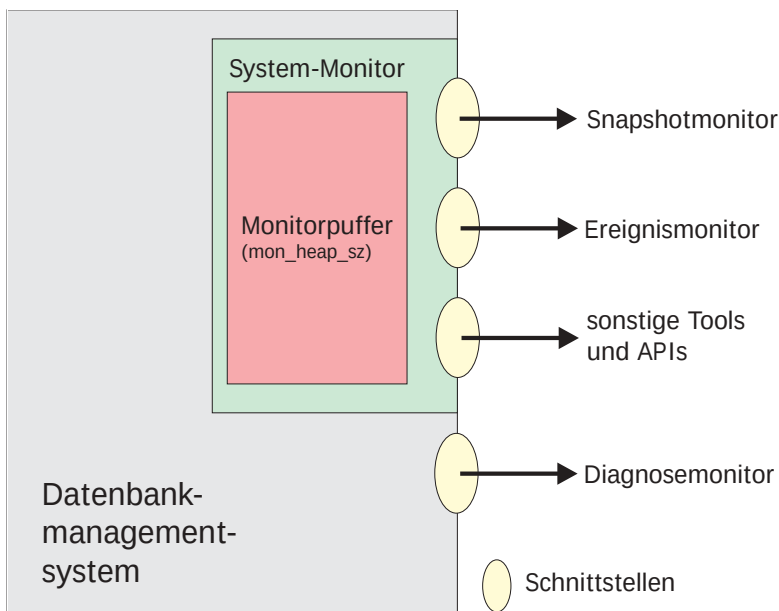


Abbildung 14.1: Systemmonitor

Der Systemmonitor nutzt den Monitorpuffer, um die Informationen zwischenspeichern. Der Snapshot-Monitor, der Ereignismonitor und andere Tools nutzen die Daten des Systemmonitors und stellen, genau betrachtet, eigentlich nur Benutzerschnittstellen dar.

14.4 Der Snapshot-Monitor

Der Snapshot-Monitor wird verwendet, um Augenblicksbetrachtungen vom Exemplar, der Datenbank und deren Anwendungen durchzuführen. In diesen *Momentaufnahmen* wird der Inhalt aus dem Monitorpuffer (mon_heap_sz) ausgelesen und in die dafür vorgesehene Ausgabe gebracht. Diese Ausgabe kann der Bildschirm, eine Datei oder auch eine Tabelle sein. Die Daten, die im Systemmonitor gesammelt wurden, hängen von den so genannten Monitorschaltern ab.

Damit der Systemmonitor überhaupt mit dem Datensammeln beginnt, muss eine Verbindung zum Exemplar, z.B. über ein ATTACH oder zur Datenbank über ein CONNECT, bestehen.

Der Systemmonitor löscht alle gesammelten Daten über eine Datenbank, wenn diese inaktiv wird, es also keine bestehende Verbindung mehr gibt. Dies kann verhindert werden, wenn Sie ihre Datenbank über ein ACTIVATE DATABASE gestartet haben und die Datenbank so im aktiven Modus bleibt, auch wenn es keine bestehende Verbindung zur Datenbank mehr gibt.

14.4.1 Welche Daten sollen gesammelt werden?

Die Informationen, die der Snapshot-Monitor anzeigen soll, müssen vorher über den Systemmonitor gesammelt worden sein. Einige von den Monitordaten werden immer gesammelt; andere Monitordaten werden erst gesammelt, wenn der zugehörige Schalter (switch) aktiviert wurde.

Welcher Schalter aktiv ist, sehen Sie am schnellsten über den Befehl:

```
get monitor switches
```

```
C:\BuchDB\db2batch>db2 get monitor switches
```

Schalter für Monitoraufzeichnungen

Schalterliste für Datenbankpartitionsnummer 0

Inform. zu Pufferpoolaktivitäten	(BUFFERPOOL) = ON	30/09/2003 09.59.50.813318
Informationen zu Sperren	(LOCK) = ON	30/09/2003 10.00.09.731425
Informationen zur Sortierung	(SORT) = ON	30/09/2003 10.00.48.205476
Informationen zu SQL-Anweisungen	(STATEMENT) = ON	30/09/2003 10.00.48.205476
Informationen zu Tabellenaktivitäten	(TABLE) = ON	30/09/2003 10.00.48.205476
Informationen zu Zeitmarke abrufen	(TIMESTAMP) = ON	29/09/2003 17.32.12.000035
Informationen zu Arbeitseinheiten	(UOW) = ON	30/09/2003 10.00.18.835562

Abbildung 14.2: get monitor switches-Syntax mit Ausgabe

Wurden die Schalter schon zu einem früheren Zeitpunkt eingeschaltet oder über den Datenbankmanager als Standard gesetzt, so sehen Sie bei der Abfrage über den Befehl get monitor switches hinter dem ON einen Zeitstempel (siehe Abbildung 14.2). Dieser Zeitstempel zeigt Ihnen, seit wann für diese Monitorgruppe Daten gesammelt werden.

Hinweis:

Jede Anwendung, also jede Verbindung, die zu einer Datenbank besteht, hat eine eigene Sicht auf die Monitor Daten und kann Monitorinformationen individuell zusammenstellen.

Um zu bestimmen, welche Daten gesammelt werden sollen, müssen Sie zuerst die „Schalter für Monitoraufzeichnungen“ (switches) einschalten, damit alle angeforderten Daten auch in den Monitorpuffer (mon_heap_sz) geschrieben werden. Die Monitor Daten sind in funktionale Gruppen unterteilt und können immer für eine oder mehrere dieser Gruppen eingeschaltet werden.

Der Befehl, um Schalter für einen einzelnen Monitorvorgang zu setzen, lautet:

```
update monitor switches using <schalter-name> on
```

Die zugehörigen „Schalter für Monitoraufzeichnungen“ (<schalter-name>) sind: BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, TIMESTAMP und UOW.

Alternativ können Sie Schalter über die Exemplarkonfiguration vorgeben, um diese als Standard zu verwenden. Der zugehörige Befehl lautet:

```
update dbm cfg using <standard-schalter-name> on
```

Die zugehörigen Schalter für die Standardvorgaben (<standard-schalter-name>) sind: DFT_MON_BUFPOOL, DFT_MON_LOCK, DFT_MON_SORT, DFT_MON_STMT, DFT_MON_TABLE, DFT_MON_TIMESTAMP und DFT_MON_UOW.

Die genaue Bedeutung wird im Folgenden beschrieben:

► BUFFERPOOL oder DFT_MON_BUFPOOL

Mit diesem Monitorschalter werden u.a. alle logischen und physikalischen Lese- und Schreibvorgänge erfasst, die im Pufferpool durchgeführt werden.

► LOCK oder DFT_MON_LOCK

Wird dieser Schalter aktiviert, so werden u.a. die Anzahl und die Dauer von Datenbanksperren sowie der aufgetretenen Prozessverklebungen (deadlocks) gesammelt.

► SORT oder DFT_MON_SORT

Es werden u.a. die Anzahl, die Art der Sortierung sowie der benötigte Speicherbedarf für alle angefallenen Sortieroperationen gesammelt.

► STATEMENT oder DFT_MON_STMT

Hier werden u.a. die Laufzeiten gemessen sowie die Anzahl der Reihen registriert, die verarbeitet wurden.

► TABLE oder DFT_MON_TABLE

Mit diesem Monitorschalter werden u.a. alle logischen und physikalischen Lese- und Schreibvorgänge erfasst, die auf eine Tabelle erfolgen.

- ▶ UOW oder DFT_MON_UOW
Die Start- und Stopp-Zeiten sowie der Rückgabewerte einer Transaktion werden hiermit festgehalten.
- ▶ TIMESTAMP oder DFT_MON_TIMESTAMP
Zeitmarken werden nicht mehr geschrieben, wenn dieser Wert auf OFF gesetzt wird. ON ist die Standardeinstellung.

Die Monitorschalter können in jeder Anwendung gezielt gesetzt werden, also auch aus dem DB2-Befehlsfenster heraus. Die Monitorschalter in der Exemplarkonfiguration fangen dann alle mit DFT_MON_ (siehe die Liste unmittelbar über diesem Text) an. Diese Veränderung kann auch über das GUI durchgeführt werden, wie in Abbildung 14.3 gezeigt wird.

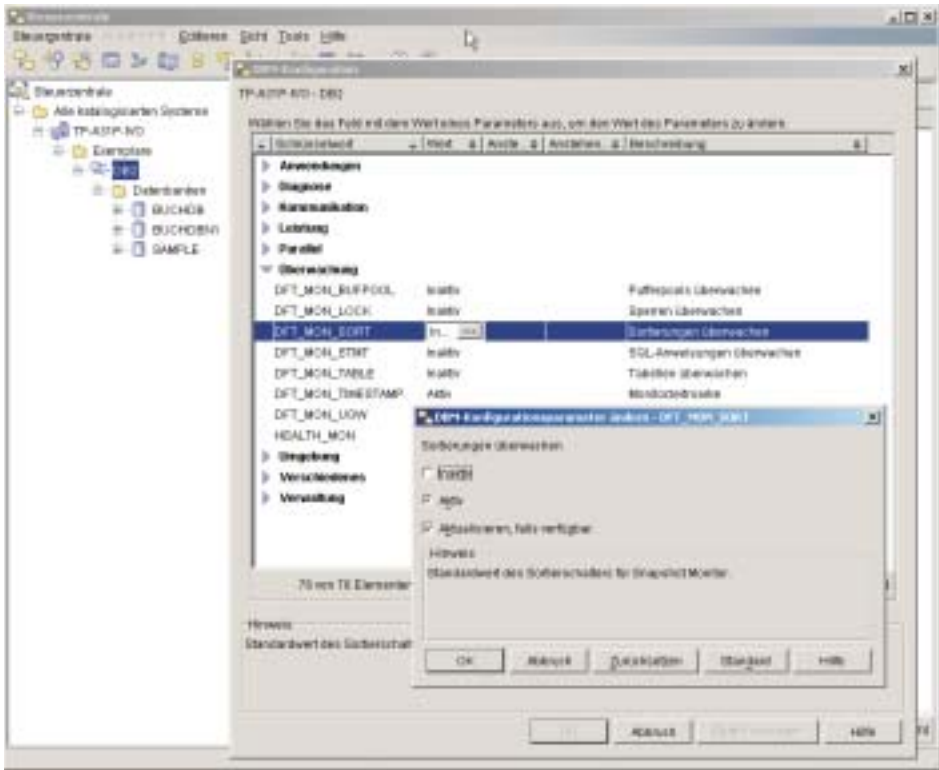


Abbildung 14.3: Monitorschalter in der Exemplarkonfiguration

In der Exemplarkonfiguration, die Sie über den MENÜPUNKT/BUTTON ÜBERWACHUNG AUFRUFEN, erhalten Sie die Monitorschalter zum Verändern der Standardeinstellung und zum Setzen der neuen Werte für jede künftige Verbindung zu diesem Exemplar bzw. zu den dazugehörigen Datenbanken. Diese Veränderungen können auch über die Befehlszeile herbeigeführt werden. Zum Beispiel sieht die Einstellung, dass die Sortierung nun immer bei jedem Monitor automatisch mit dabei ist, so aus:

```
update dbm cfg using DFT_MON_SORT ON
```

14.4.2 Wohin sollen die Monitordaten geschrieben werden?

Ihnen stehen für die Ausgabe der Snapshot-Informationen drei Möglichkeiten zur Verfügung:

- ▶ Ausgabe auf den Bildschirm (Standardfall)
- ▶ Ausgabe in eine Datei
- ▶ Ausgabe in eine Tabelle

Die Auswertung über eine SQL-Abfrage ist sehr sinnvoll, vor allem, wenn man Vergleiche und ausführliche Analysen durchführen möchte. Dazu können die Monitorinformationen in Tabellen gespeichert werden (siehe Abschnitt 14.4.4 „Wie führen Sie den Snapshot-Monitor über Tabellen-Funktionen aus?“).

14.4.3 Wie führe ich den Snapshot-Monitor per Befehl aus?

Um detaillierte Informationen aus dem System monitor zu erhalten, gibt es den Befehl:
get snapshot for ...

Mit diesem Befehl bzw. mit den zusätzlichen Optionen können Sie die Ausgabe auf wesentliche Informationen einschränken, um nicht mit einer Flut an Informationen überhäuft zu werden. Abbildung 14.4 zeigt eine kurze Übersicht über die wichtigsten Optionen.

get snapshot for	database manager	
	all	databases
		applications
		bufferpools
	all	on <db-name>
	database	
	bufferpools	
	applications	
	tables	
	tablespaces	
	locks	
	dynamic sql	

Abbildung 14.4: get snapshot for-Befehlssyntax

Der Grundbefehl ist immer GET SNAPSHOT FOR, danach kommen die Einschränkungen. Sie können alle Monitorinformationen zum Exemplar (DBM) erhalten bzw. sie dann weiter für *alle* Datenbanken, Anwendungen und Puffer-Pools einschränken. Bei diesen Anzeigen werden hauptsächlich die aufsummierten Informationen angezeigt. Wollen Sie mehr Details zu einer Datenbank haben, so ist der folgende Befehl einzugeben:

get snapshot for all on <datenbank-name>

Diese Informationen beinhalten alle Monitorinformationen, falls diese gesammelt wurden, also die zugehörigen Monitorschalter eingeschaltet waren. Sind Ihnen diese Informationen zu viel, so können Sie mit den einzelnen Optionen, die oben in Abbildung 14.4 aufgeführt sind (z.B. database, applications oder lock), die Ausgabe auf die jeweilige Monitorgruppe einschränken. Ein praktisches Beispiel sehen Sie in Abbildung 14.5.

```
C:\BuchDB\db2batch>db2 get snapshot for bufferpools on buchdb
```

Momentaufnahme eines Pufferpools

Pufferpoolname	= IBMDEFAULTBP
Datenbankname	= BUCHDB
Datenbankpfad	= C:\DB2\NODE00
00\SQL00005\	
Aliasname der Eingabedatenbank	= BUCHDB
Logische Lesevorgänge im Pufferpool	= 13229
Physische Lesevorgänge im Pufferpool	= 5
Schreibvorgänge im Pufferpool	= 0
Logische Lesevorgänge im Pufferpoolindex	= 1487
Physische Lesevorgänge im Pufferpoolindex	= 15
Gesamtzeit der Lesevorgänge im Pufferpool (ms)	= 0
Gesamtzeit der Schreibvorgänge im Pufferpool (ms)	= 0
Asynchrone Lesevorgänge in Poolseiten	= 0
Asynchrone Schreibvorgänge in Poolseiten	= 0
Schreibvorgänge im Pufferpoolindex	= 0
Asynchrones Lesen der Poolindexseiten	= 0
Asynchrone Schreibvorgänge in Poolindexseiten	= 0
Gesamtzeit für asynchrone Lesevorgänge	= 0
Gesamtzeit für asynchrone Schreibvorgänge	= 0
Asynchrone Leseanforderungen	= 0
Direkte Lesevorgänge	= 34
Direkte Schreibvorgänge	= 0
Direkte Leseanforderungen	= 12
Direkte Schreibanforderungen	= 0
Abgelaufene Zeit für direktes Lesen (ms)	= 325
Abgelaufene Zeit für direktes Schreiben (ms)	= 0
Geschlossene Datenbankdateien	= 0
In erweiterten Speicher kopierte Datenseiten	= 0
In erweitertem Speicher kopierte Indexseiten	= 0
Aus erweitertem Speicher kopierte Datenseiten	= 0
Aus erweitertem Speicher kopierte Indexseiten	= 0
Ungelesene Seiten für Vorableszugriff	= 0
Über einen Vektor definierte E/A-Operationen	= 0
Seiten aus über einen Vektor definierten E/A-Operationen	= 0
Block-E/A-Operationen	= 0
Seiten aus Block-E/A-Operationen	= 0
Zuordnungen physischer Seiten	= 0
Node number	= 0
Tabellenbereiche, die Pufferpool verwenden	= 3
Pufferpoolinformationen ändern:	
Noch zu entfernende Seiten	= 0
Aktuelle Größe	= 250
Größe nach Änderung	= 250

Abbildung 14.5: get snapshot for bufferpools-Syntax mit Ausgabe

In der Abbildung 14.5 wurde eine Momentaufnahme für die Pufferpools durchgeführt. Aus dem hier sehr sprechenden Text ist z.B. zu erkennen, dass die Anzahl der Logischen Lesevorgänge im Pufferpoolindex hier 1487 beträgt, und dass dafür 15 physikalische Lesevorgänge nötig sind, um die Indexseiten in den Pufferpool einzulesen. Im weiteren Text (letzte Zeile) ist die aktuelle Größe des Pufferpools zu sehen.

Bei den Informationen, die mit dem GET SNAPSHOT-Befehl angezeigt werden, ist noch zu berücksichtigen, dass Daten von Anwendungen, die vor dem Aktivieren eines Monitorschalters aktiv waren, in der aktuellen Auflistung nicht berücksichtigt werden. Abbildung 14.6 verdeutlicht dies noch einmal.

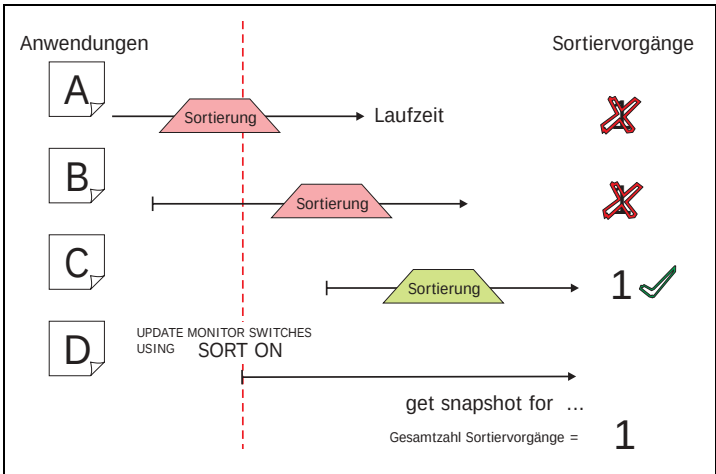


Abbildung 14.6: Start der Informationssammlung beim Snapshot

In Abbildung 14.6 schaltet die Anwendung D den Monitorschalter SORT ein, um alle aktiven Sortierungen mitzubekommen, die in der Datenbank ablaufen. Doch die Informationen, die die Anwendung D bekommt, stammen nur von dem Sortiervorgang der Anwendung C, da diese Anwendung nach dem Aktivieren des Sort-Schalters (update monitor switches using sort on) gestartet wurde und somit vom Monitor der Anwendung D erfasst werden konnte.

Alle Monitorelemente (zurzeit ca. 350) zu besprechen würde hier viel zu weit führen. Um sämtliche Monitorelemente kennen zu lernen, empfehlen wir das DB2-Monitor-Handbuch [MONGUIDE].

Oft wird die Frage gestellt, ob es nicht eine Übersicht gibt, mit der man in der Lage ist, relativ schnell zu sehen, welche Informationen mit welchem Befehl abgefragt werden können und welche Monitorschalter dazu benötigt werden. Abbildung 14.7 zeigt eine solche kompakte Darstellung.

Monitorinformationen (Informationsgruppen)			Speicher-Pools	Tabellenbereiche	Agenten	Zeilen	lesen/schreiben	Pkg./Sect./Cat. Speicher	Anwendungs- status	SQL-Aktivitäten SEL/INS/UPD/DEL	Protokollplatz	Nutzung/Verbrauch DB-Sperrungen	Gesamtzahl	Sortiervorgänge	dynamisches SQL	Tabellen	Pufferpool lesen/schreiben	Sperrungen(Details)	Verbrauch von Prozessorzeit
Befehle																			
get snapshot for	database manager													S					
	all	databases											S	S			S		S
		applications											S	S			S		
		bufferpools															S		
	all	on <db-name>										S	S	S			S		S
	database										S	S				S			
	bufferpools															S		S	
	applications											S	S						
	tables														S	S			
	tablespaces																		
locks										S							S		
dynamic sql													S						
														S					

Monitordaten werden immer gesammelt

S

 Teile der Daten werden immer gesammelt, alle Daten erst, wenn Switch ON ist

S

 Switch muss eingeschaltet sein

Abbildung 14.7: Informationsgruppen, bezogen auf den get snapshot-Befehl

Abbildung 14.7 soll Ihnen helfen, sich einen Überblick über die Monitorgruppen und die Informationen, die in diesen gesammelt werden, zu verschaffen. Sie sehen, mit welcher GET SNAPSHOT-Option Sie welche Monitorgruppen abfragen können, wenn die zugehörigen Monitorschalter eingeschaltet bzw. ausgeschaltet sind.

Wollen Sie diese Monitorinformationen über einen längeren Zeitraum sammeln, so sollten Sie noch einiges dazu bedenken:

Bestimmte Monitorinformationen sind Zähler und werden ständig weitergeführt. Diese Zähler werden erst auf „0“ gesetzt, wenn die zugehörigen Schalter auf OFF gesetzt werden bzw. die Anwendung oder die Datenbank sich „abmeldet“. Wollen Sie jedoch Monitorwerte gezielt zurücksetzen, so gibt es dazu die Befehle

```
reset monitor for database <db-name>
```

und

```
reset monitor all
```

Bei der Einschränkung ...for database <db-name> werden nur die Informationen für diese Datenbank aus der Sicht der aufrufenden Anwendung gelöscht. Die anderen Anwendungen sind davon nicht betroffen. Bei der Option reset monitor all werden alle Werte aus den Datenbanken und auch einige aus dem Exemplar gelöscht bzw. auf 0 zurückgesetzt.

Die Ausgabe, die der Snapshot-Monitor erzeugt, kann nur über eine Umleitung in eine Datei geschrieben werden.

```
get snapshot for all on buchdb > snapshot.out
```

Diese Ausgabedatei (snapshot.out) kann über einen Editor oder ein Programm weiterverarbeitet werden.

Hinweis:

In Windows-Umgebungen sollten Sie den Wordpad-Editor verwenden, da dieser im Gegensatz zum Notepad eine korrekte Formatierung der Ausgabedatei erzeugt.

14.4.4 Wie führen Sie den Snapshot-Monitor über Tabellen-Funktionen aus?

Als Alternative zum GET SNAPSHOT-Befehl kann der Snapshot-Monitor auch über SQL ausgeführt werden. Bei dieser Variante werden Tabellen-Funktionen (siehe Abschnitt 7.10) eingesetzt, die in DB2 UDB V8 standardmäßig mitgeliefert werden. Diese Technik ist die Voraussetzung dafür, dass Sie die Snapshot-Ergebnisse nicht nur am Bildschirm oder in Dateien ausgeben, sondern auch in Tabellen speichern können.

Snapshot-Befehle	SQL-Tabellen-Funktionen
get snapshot for dbm	SNAPSHOT_DBM
	SNAPSHOT_FCM
	SNAPSHOT_FCMNODE
get dbm monitor switches	SNAPSHOT_SWITCHES
get snapshot for database on <dbname> get snapshot for all database	SNAPSHOT_DATABASESE
get snapshot for applications on <dbname> get snapshot for all applications	SNAPSHOT_APPL
	SNAPSHOT_APPL_INFO
	SNAPSHOT_LOCKWAIT
	SNAPSHOT_STATEMENT
	SNAPSHOT_AGENT
	SNAPSHOT_SUBSECT
get snapshot for tables on <dbname>	SNAPSHOT_TABLE
	SNAPSHOT_TABREORG
get snapshot for locks on <dbname>	SNAPSHOT_LOCK
get snapshot for tablespaces <dbname>	SNAPSHOT_TBS
	SNAPSHOT_TBS_CFG
	SNAPSHOT QUIESCERS
	SNAPSHOT_CONTAINERS
	SNAPSHOT_RANGES
get snapshot for all bufferpools get snapshot for bufferpools on <dbname>	SNAPSHOT_BP
get snapshot for dynamic sql on <dbname>	SNAPSHOT_DYN_SQL

Tabelle 14.1: Relation einer SQL-Funktion zum get snapshot-Befehl

Die Ergebnisdaten einer Tabellen-Funktion sind dabei meist nur ein Teilauszug der Ergebnisse des zugehörigen GET SNAPSHOT-Befehls. Um die gleichen Informationen, die Sie mit dem GET SNAPSHOT-Befehl erzeugen können, auch über eine SQL-Abfrage zu erhalten, sollten Sie wissen, mit welchen Tabellen-Funktionen Sie welche Snapshot-Daten erhalten. Tabelle 14.1 zeigt diese Zusammenhänge.

Mit den in Tabelle 14.1 gezeigten Tabellen-Funktionen können Sie nun die Snapshot-Daten mit einem SQL-Befehl abfragen. Wollen Sie z.B. Informationen über die Aktivitäten im Pufferpool haben, so setzen Sie folgenden SQL-Befehl ab:

```
select * from table(snapshot_bp('<db-name>','-1') as temp_tab
```

Diese Abfrage ist eine SQL-Tabellen-Funktion auf eine temporäre Datei, d.h. bei jeder neuen Abfrage wird ein neuer Snapshot angelegt. Die „-1“ bedeutet, dass diese Abfrage auf der aktuellen Datenbankpartition durchgeführt wird. (In einer partitionierten Umgebung könnten Sie anstatt der „-1“ auch die Partitionsnummer angeben oder „-2“ für alle Partitionen; dies würde der GLOBAL-Option beim GET SNAPSHOT entsprechen.)

Wollen Sie nun die Spaltennamen herausbekommen, um eine gezielte Abfrage starten zu können, ist folgender Befehl hilfreich:

```
describe select * from table(snapshot_bp('buchdn','-1')) as temp_tab
```

Sie erhalten nun eine Auflistung aller Spalten und der zugehörigen Definitionen.

Über diese Tabellenbeschreibung ist es Ihnen möglich, ganz gezielt Abfragen auf einzelne Spalten durchzuführen. Ein Beispiel hierfür:

```
select pool_data_L_reads,pool_data_p_reads,bp_name,db_name from
table(snapshot_bp('buchdb','-1')) as temp_tab
```

POOL_DATA_L_READS	POOL_DATA_P_READS	BP_NAME	DB_NAME
33492	2241	IBMDEFAULTBP	BUCHDB

In der SQL-Abfrage wurden die Spalten POOL_DATA_L_READS für die logischen Leseanforderungen, und die Spalte POOL_DATA_P_READS für die physikalische Leseanforderungen von Datenseiten in den Pufferpool, sowie der Pufferpool-Name (BP_NAME) und der Datenbankname (DB_NAME) ausgewählt.

Vielleicht fragen Sie sich, wozu Sie die Snapshot-Daten in einer Tabelle benötigen. Doch stellen Sie sich einmal vor, Sie führen eine Analyse in einem Umfeld mit sehr vielen Anwendungen und somit auch vielen Benutzern einer Datenbank durch. In der Summe können dabei Tausende von unterschiedlichen SQL-Abfragen durchgeführt worden sein. Die Analyse solcher Snapshots ist sehr umständlich und nur zu bewältigen, wenn sie mit Hilfe von Scripts oder Programmen durchgeführt wird. Nun hat aber nicht jeder Administrator die Zeit, solche Analyse-Scripts oder -programme selbst zu schreiben. Viel einfacher ist es, diese Datenflut in Tabellen zu laden und mit einfachen SQL-Befehlen zu sortieren und zu analysieren. Wie Sie dies am einfachsten durchführen, wollen wir Ihnen in diesem Abschnitt aufzeigen.

C:\>db2 "describe output select * from table(snapshot_bp('buchdb',-1)) as temp_tab"

SQLDA-Informationen

sqldaid : SQLDA sqldabc: 1424 sqln: 32 sqld: 32

Spalteninformationen

	sqltype	sqllen	sqlname.data	sqlname.length
393	TIMESTAMP	26	SNAPSHOT_TIMESTAMP	18
493	BIGINT	8	POOL_DATA_L_READS	17
493	BIGINT	8	POOL_DATA_P_READS	17
493	BIGINT	8	POOL_DATA_WRITES	16
493	BIGINT	8	POOL_INDEX_L_READS	18
493	BIGINT	8	POOL_INDEX_P_READS	18
493	BIGINT	8	POOL_INDEX_WRITES	17
493	BIGINT	8	POOL_READ_TIME	14
493	BIGINT	8	POOL_WRITE_TIME	15
493	BIGINT	8	POOL_ASYNC_DATA_READS	21
493	BIGINT	8	POOL_ASYNC_DATA_WRITES	22
493	BIGINT	8	POOL_ASYNC_INDEX_WRITES	23
493	BIGINT	8	POOL_ASYNC_READ_TIME	20
493	BIGINT	8	POOL_ASYNC_WRITE_TIME	21
493	BIGINT	8	POOL_ASYNC_DATA_READ_REQS	25
493	BIGINT	8	DIRECT_READS	12
493	BIGINT	8	DIRECT_WRITES	13
493	BIGINT	8	DIRECT_READ_REQS	16
493	BIGINT	8	DIRECT_WRITE_REQS	17
493	BIGINT	8	DIRECT_READ_TIME	16
493	BIGINT	8	DIRECT_WRITE_TIME	17
493	BIGINT	8	POOL_ASYNC_INDEX_READS	22
493	BIGINT	8	POOL_DATA_TO_ESTORE	19
493	BIGINT	8	POOL_INDEX_TO_ESTORE	20
493	BIGINT	8	POOL_INDEX_FROM_ESTORE	22
493	BIGINT	8	POOL_DATA_FROM_ESTORE	21
493	BIGINT	8	UNREAD_PREFETCH_PAGES	21
493	BIGINT	8	FILES_CLOSED	12
449	VARCHAR	20	BP_NAME	7
449	VARCHAR	8	DB_NAME	7
449	VARCHAR	256	DB_PATH	7
449	VARCHAR	8	INPUT_DB_ALIAS	14

Abbildung 14.8: describe snapshot_bp-BildschirmAusgabe

Wie bekommen Sie die Monitordaten in eine Tabelle?

Hier eine kurze Vorgehensbeschreibung:

1. Bauen Sie Tabellen für die Aufnahme der Snapshot-Informationen auf.

Für jede Tabellen-Funktion kann eine Tabelle angelegt werden. Dies geschieht am einfachsten über den folgenden Befehl, der ein Beispiel für die SNAPSHOT_DYN_SQL-Funktion ist:

```
create table snapshot_dyn_sql as (select * from table snapshot_dyn_sql
('buchdb',-1)) as temp) definition only;
```

Entsprechend können Sie Tabellen für die anderen Tabellen-Funktionen anlegen.

Hierbei gibt es noch Folgendes zu beachten:

Sie sollten sich darüber im Klaren sein, welcher Tabellenbereich Ihr Standard-Tabellenbereich ist, also wo diese neu angelegte Tabelle abgelegt wird, wenn der

Tabellenbereich nicht mit angegeben worden ist. Hier sei noch einmal auf den IBMDEFAULTGROUP-Tabellenbereich hingewiesen, den wir schon in *Kapitel 5.9.2* angesprochen haben.

2. Erzeugen Sie Snapshot-Daten, und legen Sie sie in der Tabelle ab.
3. Die Snapshot-Daten müssen nun erzeugt und in die dafür vorgesehene Tabelle eingefügt werden. Dies geschieht am einfachsten mit dem folgenden Befehl:

```
insert into snapshot_dyn_sql select * from table (snapshot_dyn_sql
('buchdb',-1)) as temp
```

Dieser Befehl kann nun in bestimmten Zeitabständen wiederholt werden, um einen Trend bzw. Zustände zu diversen Zeitpunkten festzuhalten. Dafür kann z.B. die DB2-Taskzentrale eingesetzt werden.

Hinweis:

Bedenken Sie, dass die Monitorwerte kumulierte Werte enthalten. Wenn Sie die Differenz zwischen zwei Messungen haben wollen, so müssen Sie die Monitorzähler auf „0“ zurücksetzen. Dies geschieht am besten mit dem Befehl:

```
RESET MONITOR FOR DATABASE <dbname>
```

4. Fragen Sie gezielt die Daten aus der Snapshot-Tabelle ab.

Die Daten, die nun in der Snapshot-Tabelle (hier: `SNAPSHOT_DYN_SQL`) eingefügt wurden, erhalten alle den gleichen Zeitstempel. Somit können Sie, wenn der Befehl aus Punkt 2 öfter ausgeführt wurde, immer verfolgen, von welchem Zeitpunkt Ihre Daten stammen. Dies kann dann bei weiteren SQL-Abfragen berücksichtigt werden. Zum Beispiel:

```
Select
snapshot_timestamp,
num_executions,
total_exec_time,
stmt_sorts,
substr(stmt_text,1,20) as "SQL-Text"
from snapshot_dyn_sql
order by num_executions desc
fetch first 20 rows only
```

14.5 Der Ereignismonitor

Der Ereignismonitor (Event-Monitor) arbeitet im Gegensatz zum Monitor für Momentaufnahmen (Snapshot-Monitor) *erst*, wenn ein vordefiniertes Ereignis (event type) stattgefunden hat. Dieses Ereignis könnte z.B. sein:

- das Erkennen einer gegenseitigen Sperre (deadlock)
- das Ende eines SQL-Befehls

- ▶ das Ende eine Transaktion
- ▶ das Ende einer Verbindung zur Datenbank
- ▶ das Stoppen/Inaktivieren einer Datenbank

Tritt ein bestimmtes Ereignis (event type) auf, werden Informationen aus dem Monitorpuffer ausgeschrieben und in eine Datei, Pipe oder in Tabellen geschrieben.

Dies kann mit der FLUSH-Option des Ereignismonitors umgangen werden. Mit dieser Option werden die Monitorpuffer *sofort*, also nicht erst *nach* dem Eintreten eines Ereignisses, mit den bis zu diesem Zeitpunkt gesammelten Daten ausgeschrieben. Tritt das vorgesehene Ereignis, z.B. das Beenden einer Transaktion (UOW), ein, so werden die zugehörigen Daten nochmals – nur aktueller – ausgeschrieben.

Doch bevor der Ereignismonitor Daten sammeln kann, muss er eingeschaltet werden.

Bevor dies passiert, muss er angelegt und definiert werden. Dazu müssen Sie Folgendes klären:

- ▶ Welche Daten – event types – werden benötigt?
- ▶ Wohin sollen diese Daten geschrieben werden?

14.5.1 Welche Daten sollen gesammelt werden?

Der Ereignismonitor sammelt die Daten, die der Systemmonitor liefert. Es ist nicht notwendig, so wie beim Snapshot-Monitor, Schalter (monitor switches) zu aktivieren, um die benötigten Daten zu sammeln.

Sie bestimmen über die Ereignistypen, bei welchen Ereignissen Daten gesammelt werden sollen. Folgende Ereignistypen (event types) sind möglich:

▶ DATABASE

Beim Stoppen der Datenbank werden alle zugehörigen Zähler ausgegeben.

▶ TABLES

Für alle Tabellen innerhalb der Datenbank wird die Anzahl der gelesenen und geschriebenen Sätze geschrieben.

▶ DEADLOCKS

Eine erkannte gegenseitige Sperrung wird genauestens protokolliert, d.h. mit den einzelnen Sperrungen und den verursachenden SQL-Befehlen.

▶ TABLESPACES

Beim Beenden der Datenbank werden u.a. die zugehörigen Zähler für die Pufferpool-Zugriffe, die Vorabesezugriffe (prefetchers), die Seitenlöschfunktion (page cleaners) und die direkten Zugriffe (direct I/O) für jeden Tabellenbereich geschrieben.

► BUFFERPOOLS

Beim Beenden der Datenbank werden u.a. die zugehörigen Zähler für die Pufferpool-Zugriffe, die Vorabesezugriffe (prefetchers), die Seitenlöschfunktion (page cleaners) und die direkten Zugriffe (direct I/O) für jeden Bufferpool ausgegeben.

► CONNECTIONS

Beim Beenden der Verbindung werden die zugehörigen Zähler über diese Anwendung ausgegeben.

► STATEMENTS

Wird ein SQL-Befehl beendet, so werden u.a. die zugehörigen Start- und Stoppzeiten, CPU-Zeiten, der SQL-Text von dynamischen SQL-Abfragen und die SQLCA (Rückgabewerte von SQL-Abfragen) geschrieben.

► TRANSAKTIONS

Wird eine Transaktion beendet, so werden u.a. die Start- und Stopp-Zeit, die vorausgegangene Transaktion, der Verbrauch an CPU-Zeit sowie Sperr- und Protokollinformationen herausgeschrieben.

Der Teilbefehl, um einen Ereignismonitor anzulegen, würde folgendermaßen aussehen:

```
CREATE EVENT MONITOR <evm-name> FOR <event-type> ...
```

14.5.2 Wohin sollen die Monitordaten geschrieben werden?

Da das Datenaufkommen beim Event-Monitor um einiges höher ist als beim Snapshot-Monitor, werden die Monitordaten nach jedem Ereignis aus dem Monitorpuffer geschrieben. Mit der Option ...WRITE TO... bestimmen Sie das Ziel der Ausgabe. Es gibt die Möglichkeit, die Daten in Dateien (FILE), in Pipes (PIPE) oder in Tabellen (TABLE) auszugeben.

Die zugehörige Befehlsoption lautet:

```
... WRITE TO TABLE | PIPE | FILE ...
```

Wir schauen uns hier Optionen wie WRITE TO FILE und WRITE TO TABLE etwas näher an, da diese am häufigsten eingesetzt werden.

► WRITE TO FILE

Bei der WRITE TO FILE-Option werden die Daten in eine bzw. mehrere Dateien (z.B. 00000000.evt, 00000001.evt) eingefügt. Weitere Optionen verhindern, dass das zugehörige Dateisystem vollläuft und somit das System bzw. die Datenbank zum Absturz bringt.

```
... WRITE TO FILE 'C:\buchdb\evm\statements' MAXFILES 10 MAXFILESIZE 1024 BLOCKED
APPEND
```

– MAXFILES

Mit dieser Option schränken Sie die Anzahl der Dateien ein, die vom Ereignismonitor angelegt werden dürfen.

- MAXFILESIZE

Definiert die Maximalgröße einer Event-Monitordatei. Der Standard für Windows ist 200 x 4 K-Seiten und 1.000 x 4 K-Seiten für Unix.

- BLOCKED/NONBLOCKED

BLOCKED ist die Standardeinstellung und stellt sicher, dass keine Daten verloren gehen, wenn Monitorpuffer überlaufen und geleert werden müssen. In diesem Fall wartet der Monitor mit seiner Schreibaktion, bis die zugehörigen Monitorpuffer geleert wurden. Diese Vorgehensweise kann das Datenbanksystem in seinem Durchsatz behindern.

Wird die NONBLOCKED-Option eingesetzt, so warten die Monitorprozesse nicht bis die Puffer geleert wurden, sondern vergessen die Daten, die zu diesem Zeitpunkt eventuell in den Monitorpuffer geschrieben werden sollten. Falls ein solcher Datenverlust auftritt, wird dies in einer Protokolldatei festgehalten.

► WRITE TO TABLE

Das Schreiben in eine Tabelle hat den großen Vorteil, dass die Daten anschließend flexibel mit Hilfe von SQL analysiert werden können. Wie muss eine solche Tabelle aussehen bzw. wie legt man diese Tabellen am einfachsten an?

Die benötigte Tabelle wird von DB2 *automatisch bei der Erstellung* des Ereignismonitors angelegt.

Wollen Sie die Tabelle(n) gezielt in einem bestimmten Tabellenbereich platzieren, werden Sie von dem Tool db2evtbl unterstützt. Dieses Tool generiert Ihnen die notwendigen DDL-Befehle (CREATE TABLE-Befehle) für die benötigten Tabellen der zugehörigen Ereignismonitore. Ein Beispiel:

```
db2evtbl -schema <schema-name> -evm <evm-name> database, tables, tablespaces,
bufferpools > evm_alle.ddl
```

Nun müssen Sie nur noch die evm_alle.ddl-Datei ausführen, z.B. durch:

```
db2 -tf evm_alle.ddl
```

Abbildung 14.9 enthält eine Übersicht über die Tabellen, die angelegt werden bzw. angelegt sein müssen, wenn Sie Monitordaten für diese Ereignistypen in Tabellen schreiben wollen.

Die Namenskonvention, die beim Erstellen dieser Event-Monitor-Tabellen verwendet wird, sieht wie folgt aus:

```
substr(evmGroup||"_"||<ereignis-monitor-name>,1,128)
```

Wird ein Ereignismonitor evm_tab angelegt, so werden die zugehörigen Tabellen folgenden Namen haben: TABLE_EVM_TAB und CONTROL_EVM_TAB.

Ereignismonitor-Typen	Tabellen (evmGroup)
Database/Datenbank	DB
	CONTROL
Tables/Tabellen	TABLE
	CONTROL
Deadlocks/ gegenseitige Sperren	CONNHEADER
	DEADLOCK
	DLCONN
	CONTROL
Deadlocks with Details/ gegenseitige Sperren mit Details	CONNHEADER
	DEADLOCK
	DLCONN
	DLLOCK
Tablesaces/ Tabellenbereiche	CONTROL
	TABLESPACES
Bufferpools/Pufferpools	BUFFERPOOL
	CONTROL
Connections/ Verbindungen	CONNHEADER
	CONN
	CONTROL
Statements/ Anweisungen	CONNHEADER
	STMT
	SUBSECTION
	CONTROL
Transactions/ Transaktionen	CONNHEADER
	XACT
	CONTROL

Abbildung 14.9: Beziehungen: Ereignistypen zu Ereignistabellen

Nachdem Sie nun wissen, welche Daten (event types) Sie benötigen und wohin die gesammelten Daten geschrieben werden sollen, könnte ein Befehl zum Anlegen eines Ereignismonitors folgendermaßen aussehen:

```
CREATE EVENT MONITOR evm_stmt FOR statements
WHERE APPL_NAME LIKE 'sqltp1%' and
AUTHID ='IGUSER'
WRITE TO FILE 'C:\dbbuch\evm\statement'
```

In dem oberen Befehl haben wir noch eine Einschränkung auf den Anwendungs-namen (APPL_NAME) und die Benutzer-ID (AUTH_ID) eingebaut, um die Monitorausgabe einzuschränken. Somit erhalten wir nur Daten, wenn der Benutzer IGUSER die Anwendung sqltp1 startet.

Hinweis:

Die Einschränkungen für die Ausgabe gelten zurzeit nur, wenn Sie den Ereignis-monitor über einen DB2-Befehl anlegen. Führen Sie die gleiche Aktion über das GUI durch, so wird Ihnen diese Möglichkeit zur Einschränkung nicht geboten. Im GUI ist diese Einschränkung erst möglich, wenn die Daten komplett erstellt wor-den sind. Dann kann über die Spaltenauswahl diese Einschränkung erfolgen.

Wollen Sie den Ereignismonitor aus der Steuerzentrale aufrufen bzw. über die Steuer-zentrale anlegen, so werden Sie die Fenster aus Abbildung 14.10 sehen.

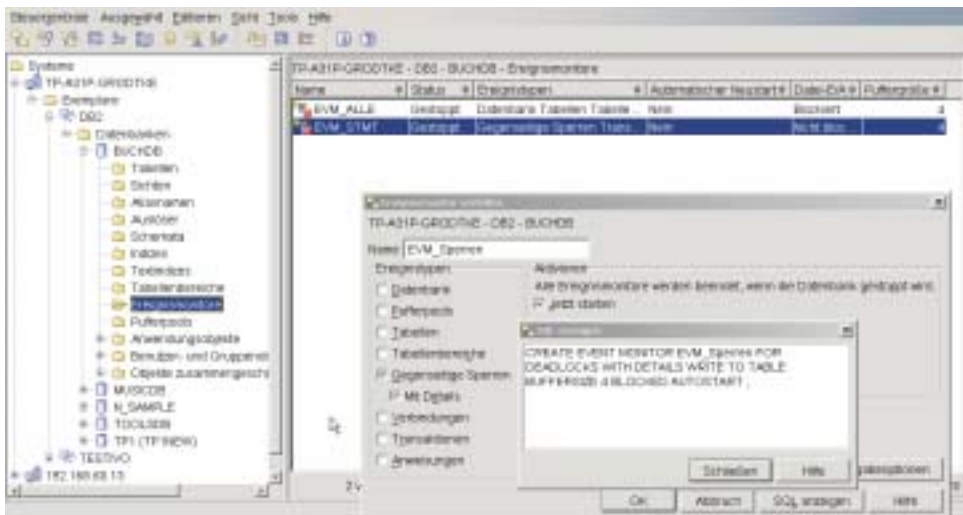


Abbildung 14.10: Ereignismonitor über die Steuerzentrale erstellen

Jetzt ist der Ereignismonitor angelegt, aber nicht gestartet.

14.5.3 Wie aktiviere ich einen Ereignismonitor?

Der Ereignismonitor kann über die DB2-Kommandozeile gestartet werden, wenn Sie mit der zugehörigen Datenbank verbunden sind:

```
SET EVENT MONITOR <evm-name> STATE 1
```

Aus der Steuerzentrale gehen Sie im Objektbaum auf EREIGNISMONITORE, klicken den gewünschten Ereignismonitor an und wählen EREIGNISÜBERWACHUNG STARTEN aus.

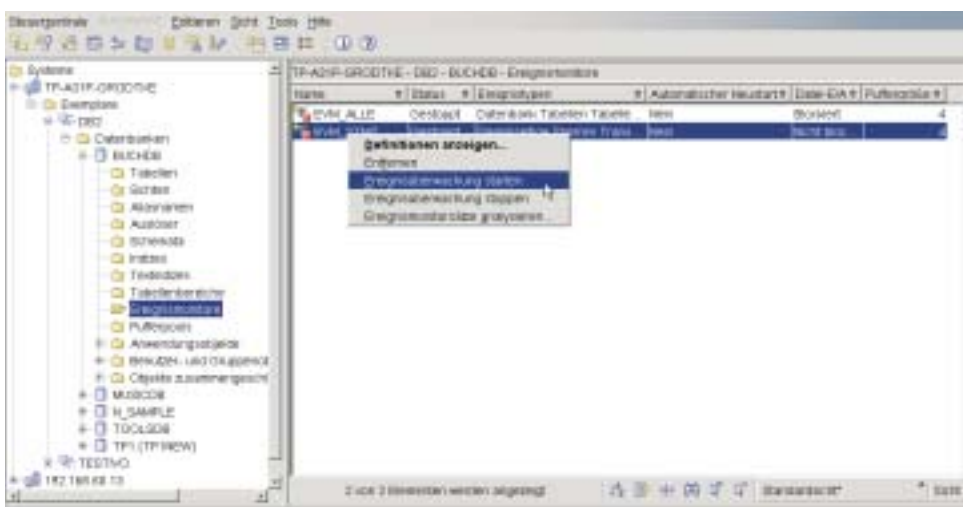


Abbildung 14.11: Ereignismonitor starten

14.5.4 Wie stoppe ich einen Ereignismonitor?

Hier gibt es wiederum mehrere Möglichkeiten. Sie können den Ereignismonitor „von Hand“, also mit einem Befehl über die DB2-Kommandozeile, oder über das GUI der Steuerzentrale stoppen – also fast so, wie Sie diesen Monitor auch gestartet haben:

```
SET EVENT MONITOR <evm-name> STATE 0
```

STATE=0 steht für das Inaktivieren dieses Monitors. Oder Sie gehen über die Steuerzentrale (siehe Abbildung 14.11) und wählen in diesem Fenster die Auswahl EREIGNISÜBERWACHUNG STOPPEN, um den ausgewählten Monitor zu beenden.

Nun kann es aber vorkommen, dass Sie sehr viele Monitordaten sammeln und Ihre Dateisysteme bzw. auch die Tabellenbereiche überlaufen würden, wenn hier nicht ein Mechanismus implementiert wäre, der dies verhindert.

Dies sind einmal die Optionen MAXFILESIZE und MAXFILES, die beim Schreiben in eine Datei (WRITE TO FILE) verhindern, dass der Ereignismonitor zu viele Daten schreibt. Wenn diese Parameter ihre Werte erreicht haben, wird der Ereignismonitor automatisch von DB2 gestoppt. Haben Sie z.B. MAXFILESIZE auf 1024 und MAXFILES auf 10 gesetzt, können die Monitordaten maximal zehn Dateien mit einer Größe von 4 Mbyte belegen; danach wird der Monitor gestoppt.

Werden die Monitordaten in Tabellen ausgegeben, was der Standard ist, wenn Sie über die Steuerzentrale die Ereignismonitore anlegen, so gibt es hier einen Parameter, der ein Überlaufen der Tabellenbereiche verhindern soll. Der Parameter heißt PCTDEACTIVATE und ist für DMS-Tabellenbereiche gedacht. Hier kann ein Prozentsatz angegeben werden, bei dem der Ereignismonitor das Schreiben von Daten stoppen muss. So bedeutet PCTDEACTIVATE=80, dass die Überwachung (monitoring) stoppt, wenn der DMS-Tabellenbereich, in dem die Ereignismonitor-Tabellen stehen, zu 80% gefüllt ist. Falls Sie einen SMS-Tabellenbereich nutzen, hat dieser Parameter keine Gültigkeit.

14.5.5 Wie analysieren Sie die angefallenen Daten?

Um die Ereignismonitor-Daten zu analysieren, gibt es drei Wege:

- ▶ eigene SQL-Abfragen auf die Tabellen des Ereignismonitors
- ▶ das grafische Event-Analyzer-Tool DB2EVA. Dieses kann über die Steuerzentrale und auch direkt über die Kommandozeile aufgerufen werden. Die Einschränkung ist, dass dieses Analyse-Tool nur die Ereignismonitor-Daten aus den Tabellen auswerten kann und somit nicht einsetzbar ist, wenn Sie Ihre Monitordaten in Dateien (FILES) oder Pipes (PIPE) geschrieben haben.
- ▶ Haben Sie die Monitordaten jedoch in Dateien geschrieben, so kann nur das textorientierte Tool DB2EVMON zum Einsatz kommen, um Sie bei der Analyse zu unterstützen.

Abfragen direkt auf die Tabellen sind sehr sinnvoll, da Sie die benötigten Informationen ganz gezielt mit einer SQL-Abfrage analysieren können. Um z.B. herauszubekommen, welche SQL-Abfragen einen Überlauf der Sortierspeicher (SORTHEAP) verursachen, könnte die Abfrage aus Abbildung 14.12 hilfreich sein.

C:\>db2 "select sort_overflows,substr(cast(stmt_text as varchar(2048)),1,256) as stmt from stmt_evm_alle where sort_overflows > 0"

SORT_OVERFLOWS

STMT

1 SELECT TabSchema, TabName, Type, TBspace, INDEX_TBSPACE, LONG_TBSPACE, DataCapture, Definer, Remarks FROM SYSCAT.Tables WHERE Type in ('T','S','U','H') ORDER BY TabName FOR FETCH ONLY WITH UR

Abbildung 14.12: Abfrage auf sort_overflows in den Monitortabellen

Diese Information kann natürlich auch über die anderen Tools gefunden werden, doch dies ist mit mehr Zeitaufwand verbunden.

Um über das GUI an die gleichen Informationen wie bei der SQL-Abfrage in Abbildung 14.12 zu kommen, müssen Sie die Monitordaten, die vom Ereignismonitor gesammelt wurden, analysieren (siehe Abbildung 14.13).

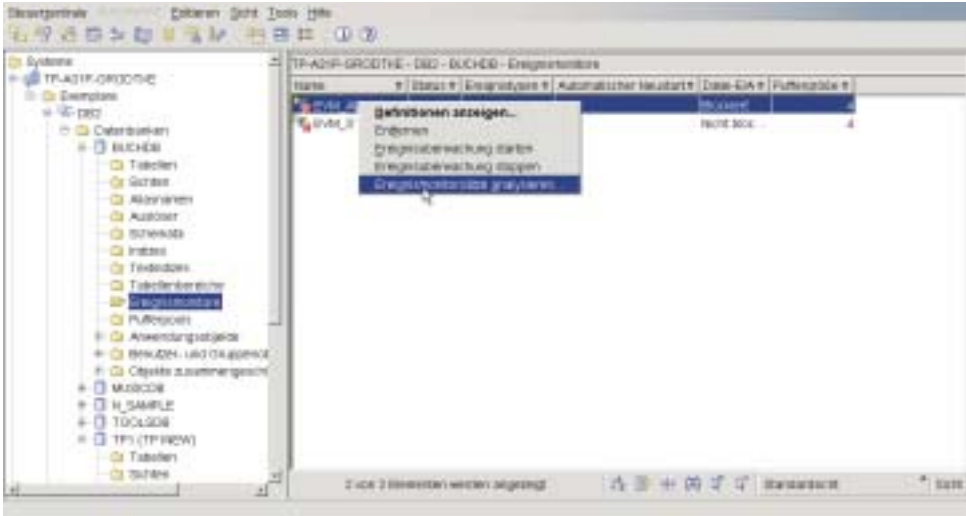


Abbildung 14.13: Ereignismonitor: Ereignisse analysieren

Nach der Auswahl des Ereignismonitors (hier: EVM_ALLE) und des dazugehörigen Menüeintrags EREIGNISMONITORSÄTZE ANALYSIEREN... erhalten Sie alle Zeiträume aufgelistet (siehe Abbildung 14.14), in denen dieser Ereignismonitor Daten gesammelt hat.

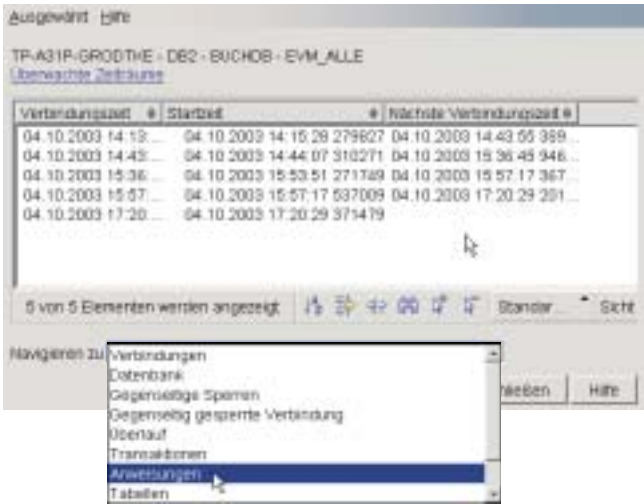


Abbildung 14.14: Ereignismonitor: Monitor-Zeiträume

Im unteren Fensterausschnitt (siehe Abbildung 14.14) selektieren Sie die ANWEISUNGEN, um alle SQL-Befehle, die in diesem Ereignismonitor enthalten sind, anzuzeigen.

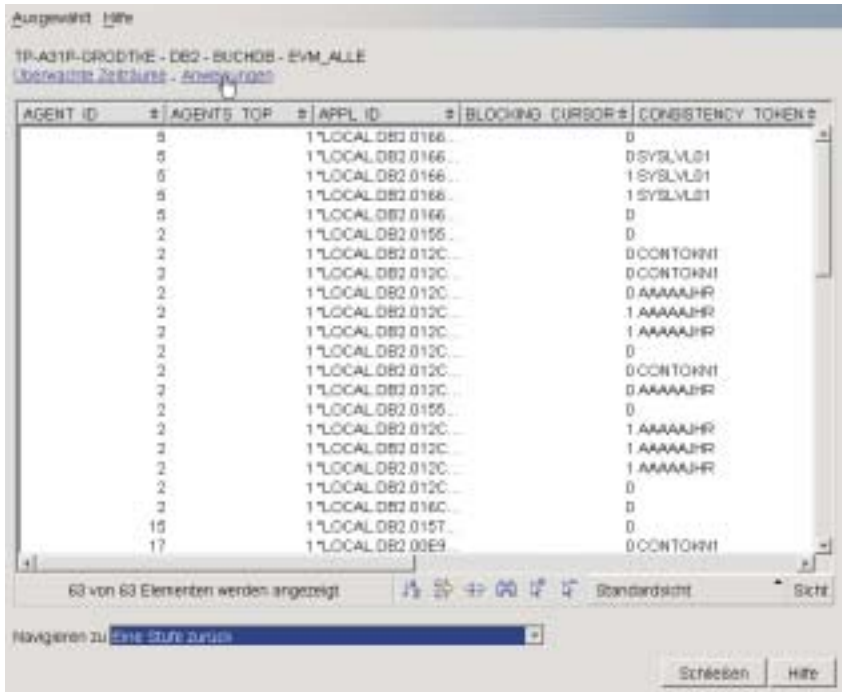


Abbildung 14.15: Ereignismonitor: Auflistung aller Anweisungen

Wenn Sie nun alle ANWEISUNGEN erhalten, wie in Abbildung 14.15 gezeigt, fehlen hier eine Menge an Informationen bzw. Spalten. In unserem Fall wollen wir etwas über die übergelaufenen Sortierungen wissen, doch diese werden im normalen Fenster nicht angezeigt. Diese Zusatzspalten müssen erst über das Icon SPALTEN ANPASSEN (siehe den Kreis in Abbildung 14.16) für das Hinzufügen von Spalten ausgewählt werden und nach ANGEZEIGTE SPALTEN verschoben werden. Hierbei sollte auch gleich die Position der neuen Spalte nach oben geschoben werden, damit diese im Fenster sofort sichtbar wird. Dieser Ablauf ist in Abbildung 14.16 mit Pfeilen angedeutet.

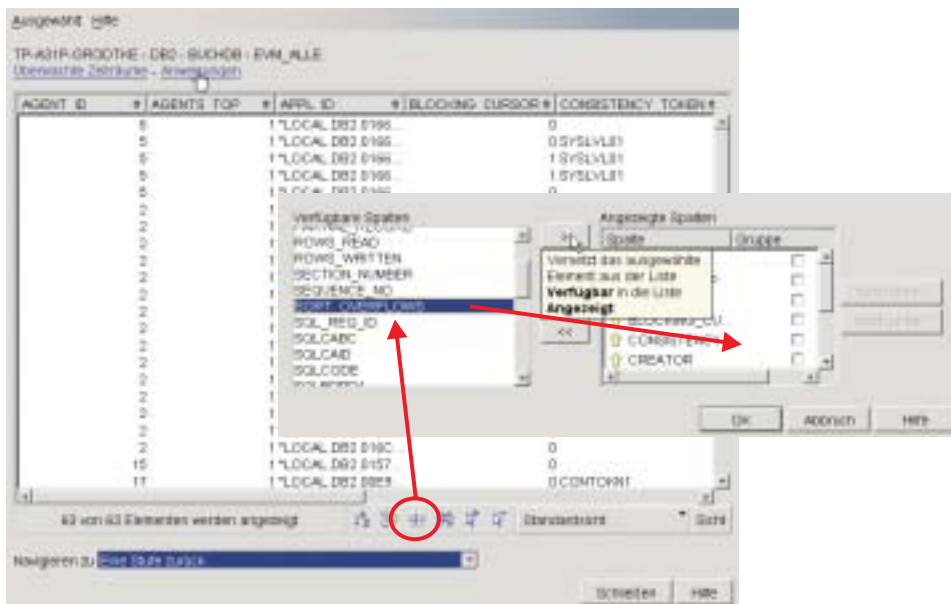


Abbildung 14.16: Ereignismonitor: Spalten hinzufügen

Ist nun die Spalte SORT_OVERFLOW im Fenster sichtbar (siehe auch Abbildung 14.17), so sollte noch ein Filter gesetzt werden, um auch nur die Befehle zu sehen, bei denen ein Sortierüberlauf stattgefunden hat. Über die Auswahl der Filterfunktion erhalten Sie ein neues Fenster mit allen Spalten, die für diese Auswahl zur Verfügung steht. Dort können Sie Bedingungen eingeben, z.B. bei der SORT_OVERFLOW-Spalte den Wert auf „ungleich“ „0“ setzen, wie in Abbildung 14.17 gezeigt.

So erhalten Sie nur noch die Sätze, die diese Bedingung erfüllen. In unserem Beispiel ist es noch ein Satz. Über die Option DATENELEMENTE sehen Sie im Überblick alle Monitorwerte in einem Fenster (siehe Abbildung 14.18), das Sie auf die passende Größe anpassen können.

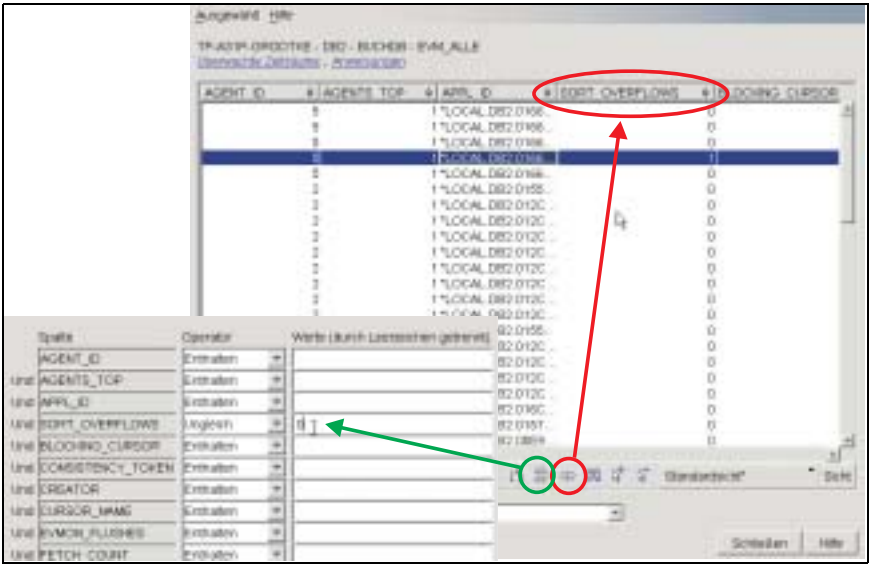


Abbildung 14.17: Ereignismonitor: Filter setzen

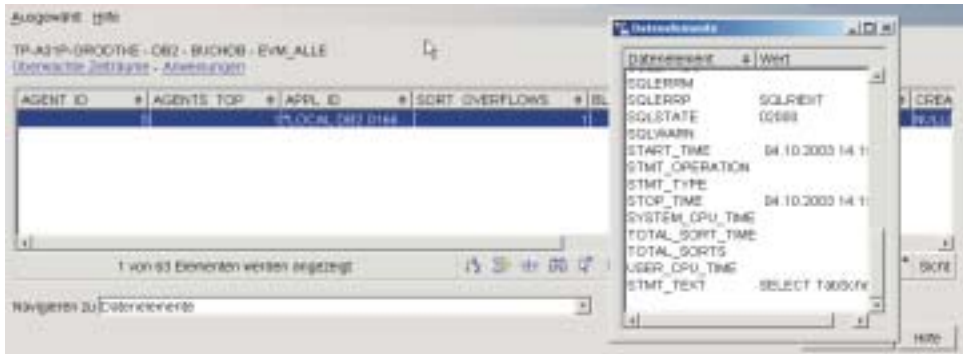


Abbildung 14.18: Ereignismonitor: Datenelemente

Das Tool DB2EVMON

Haben Sie sich dafür entschieden, die Ereignismonitor-Daten in eine Datei (FILE) oder eine Pipe (PIPE) statt in Tabellen zu schreiben, können Sie das z.B. mit folgender Option tun:

```
CREATE EVENT MONITOR <evm-name> FOR <evm-type>  
WRITE TO FILE 'C:\buchdb\evm'
```

Hinweis:

Der Pfad, der nach dem WRITE TO angegeben wird (hier: C:\buchdb\evm), muss vorhanden sein. Dies wird aber erst beim Aktivieren des Ereignismonitors überprüft!

Die Monitordaten werden von DB2 im binären Format abgelegt. Mit Hilfe des Tools db2evmon müssen Sie diese Daten erst umformatieren, um sie mit einem Texteditor anzeigen und analysieren zu können.

Der Aufruf kann wie folgt aussehen:

```
db2evmon -db <dbname> -evm <evm-name>
```

oder

```
db2evmon -path <evm-path>
```

Die Standardausgabe ist der Bildschirm. Es gibt keine weiteren Optionen dieses Tools, um die Ausgabe umzuleiten. Sie sollten aber unbedingt diese Ausgabe mit Betriebssystemmitteln in eine Textdatei umleiten, um darin besser blättern zu können.

Die Ausgabe des Ereignismonitors ist in drei Abschnitte unterteilt:

► Prolog

Der Prolog beinhaltet den Header mit Informationen über den Monitorstart, das Exemplar und die Datenbank (siehe Abbildung 14.19).

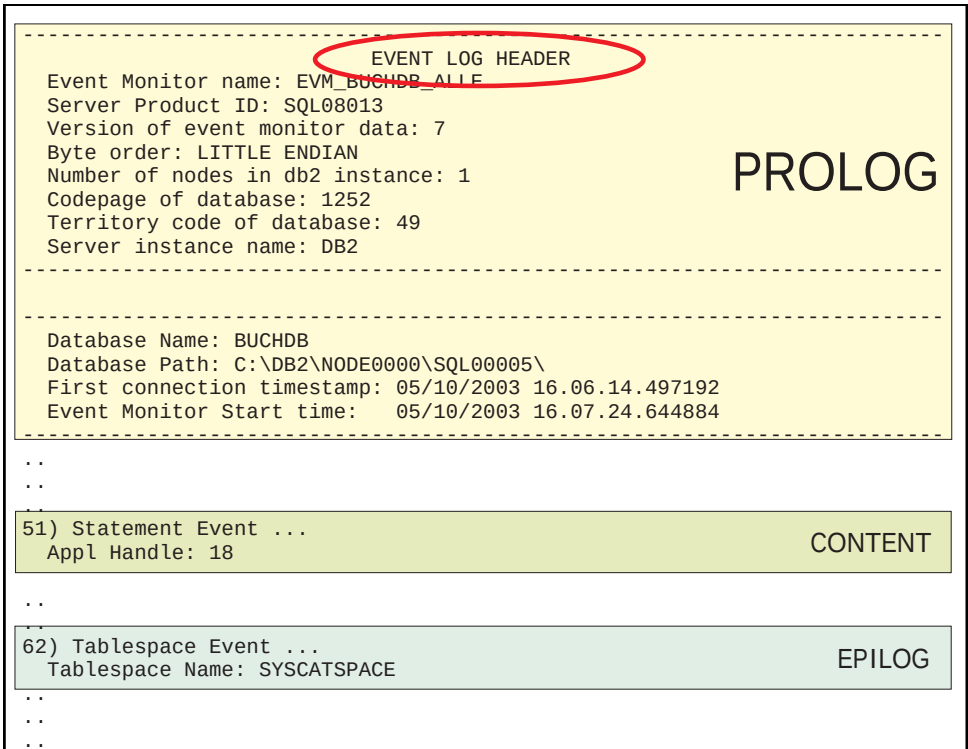


Abbildung 14.19: Prolog des Ereignismonitors

► Content

In diesem Teil der Ausgabe sehen Sie die einzelnen Ereignistypen (STATEMENT, DEAD-LOCKS, CONNECTION, TRANSACTION), die Sie beim Anlegen des Ereignismonitors eingeschaltet haben, falls dazu Informationen vorhanden sind. Abbildung 14.20 wurde auf nur eine Position gekürzt, um den wesentlichen Unterschied darzustellen.

EVENT LOG HEADER	
..	
51) Statement Event ...	CONTENT
Appl Handle: 18	
Appl Id: *LOCAL.DB2.005B85140615	
Appl Seq number: 0014	
Record is the result of a flush: FALSE	

Type : Dynamic	
Operation: Describe	
Section : 201	
Creator : NULLID	
Package : SQLC2E03	
Consistency Token : AAAAJHR	
Package Version ID :	
Cursor : SQLCUR201	
Cursor was blocking: TRUE	
Text : select status,tablename from syscat.tables order by 1	

Start Time: 05/10/2003 16.20.11.280346	
Stop Time: 05/10/2003 16.20.11.283159	
Exec Time: 0.002813 seconds	
Number of Agents created: 1	
User CPU: 0.000000 seconds	
System CPU: 0.000000 seconds	
Fetch Count: 263	
Sorts: 0	
Total sort time: 0	
Sort overflows: 0	
Rows read: 263	
Rows written: 0	
Internal rows deleted: 0	
Internal rows updated: 0	
Internal rows inserted: 0	
Bufferpool data logical reads: 263	
Bufferpool data physical reads: 0	
Bufferpool temporary data logical reads: 0	
Bufferpool temporary data physical reads: 0	
Bufferpool index logical reads: 1	
Bufferpool index physical reads: 0	
Bufferpool temporary index logical reads: 0	
Bufferpool temporary index physical reads: 0	
SQLCA:	
sqlcode: 0	
sqlstate: 00000	
..	
62) Tablespace Event ...	EPILOG
Tablespace Name: SYSCATSPACE	

Abbildung 14.20: Content des Ereignismonitors

► Epilog

Im Epilog-Teil (siehe Abbildung 14.21) werden die Monitorelemente angezeigt, die nach einem Beenden bzw. Deaktivieren der Datenbank geschrieben werden. Diese Monitorelemente beziehen sich dann auch auf die Ereignistypen wie BUFFERPOOL, TABLES, TABLESPACES.

```

-----
EVENT LOG HEADER
-----
51) Statement Event ...
62) Tablespace Event ...
   Tablespace Name: SYSCATSPACE
   Record is the result of a flush: FALSE

   Buffer Pool Statistics:
   Buffer pool logical data page reads: 589
   Buffer pool physical data page reads: 31
   Buffer pool data page writes: 3
   Buffer pool logical index page reads: 28
   Buffer pool physical index page reads: 12
   Buffer pool index page writes: 2
   Buffer pool read time: 327
   Buffer pool write time: 33
   Files closed: 0
   Buffer pool asynch data page reads: 6
   Buffer pool asynch data page read reqs: 1
   Buffer pool asynch data page writes: 2
   Buffer pool asynch index page reads: 0
   Buffer pool asynch index page read reqs: 0
   Buffer pool asynch index page writes: 2
   Buffer pool asynch read time: 0
   Buffer pool asynch write time: 32
   Unread prefetch pages: 0

   Direct I/O Statistics:
   Sectors read directly: 56
   Sectors written directly: 0
   Direct read requests: 8
   Direct write requests: 0
   Direct read time: 444
   Direct write time: 0
   Tablespace event timestamp: 05/10/2003 16.20.21.938510
   ..
   ..
   ..
EPILOG

```

Abbildung 14.21: Epilog des Ereignismonitors

14.6 Diagnosemonitor und Diagnosezentrale (Health Center)

Der Diagnosemonitor ist ein Tool, das Ihnen hilft, sich anbahnende Fehlersituationen zu erkennen, bevor sich diese zu einem kritischen Problem entwickeln. Der Diagnosemonitor bietet dazu vorgefertigte Schwellenwert-Einstellungen, die Sie jederzeit an Ihre Belange anpassen können.

Die Informationen, die der Diagnosemonitor erstellt, können über grafische Tools wie die Diagnosezentrale und die webbasierende Diagnosezentrale abgefragt werden. Die gleichen Informationen erhalten Sie auch über die DB2-Befehlszeile oder die Programmierschnittstelle (API).

Der Diagnosemonitor ist ähnlich wie der Systemmonitor in Funktionsgruppen aufgeteilt und enthält in jeder dieser Funktionsgruppen weitere Attribute, die gesetzt werden können, um eine feinere Spezifizierung für die Diagnose zu erhalten. Es gibt folgende Funktionsgruppen:

- ▶ Datenbankmanagementsystem (Exemplar)
- ▶ Betriebszustand
- ▶ Speicher für das Monitoring
- ▶ Speicher für Sortierungen
- ▶ Datenbanken
- ▶ Betriebszustand
- ▶ Gemeinsamer Zugriff auf Anwendungen
- ▶ Paket- und Katalog-Cache sowie gemeinsame Arbeitsbereiche
- ▶ Datenbankprotokollierung
- ▶ Sortierung
- ▶ Speicher
- ▶ Tabellenbereichsspeicher

Der Diagnosemonitor bietet einmal die Möglichkeit, bei Überschreitung bestimmter Schwellenwerte Aktionen auszulösen sowie ein zusätzliches Programm laufen zu lassen bzw. einen Verantwortlichen zu informieren. Andererseits haben Sie auch die Möglichkeit eines so genannten „Ausnahmemanagements“ bei Überschreitung von Schwellenwerten. Dieses Ausnahmemanagement zeigt Ihnen in einfacher Weise, welche Schritte Sie unternehmen sollten, um ein weiteres Über- oder Unterschreiten der gesetzten Schwellenwerte zu verhindern.

Die Diagnosezentrale ist die grafische Darstellung der gesetzten Schwellenwerte im Diagnosemonitor und dient zur besseren und einfacheren Konfiguration sowie zum Einleiten von Aktionen, um eventuell aufkommende Probleme zu vermeiden.

Um mit dem Diagnosemonitor zu arbeiten, müssen Sie diesen erst aktivieren. Dies geht über den DB2-Befehl

```
update dbm cfg using HEALTH_MON ON
```

oder über die Steuerzentrale. In dieser gibt es zwei Möglichkeiten des Aufrufs. Bei der ersten Möglichkeit gehen Sie über das TOOLS-Menü und dann in die Auswahl DIAGNOSEZENTRALE. Die zweite Möglichkeit besteht nur, wenn der Diagnosemonitor entsprechend konfiguriert und schon aktiv ist. In der rechten unteren Ecke der grafischen Oberfläche, z.B. der Steuerzentrale, wird dann eine Fehlermeldung angezeigt, z.B. „8 Alerts der Diagnosezentrale – Anklicken“ (siehe Abbildung 14.22). Wenn Sie nun auf den Button mit dem „!“ klicken, kommen Sie direkt in die Diagnosezentrale.

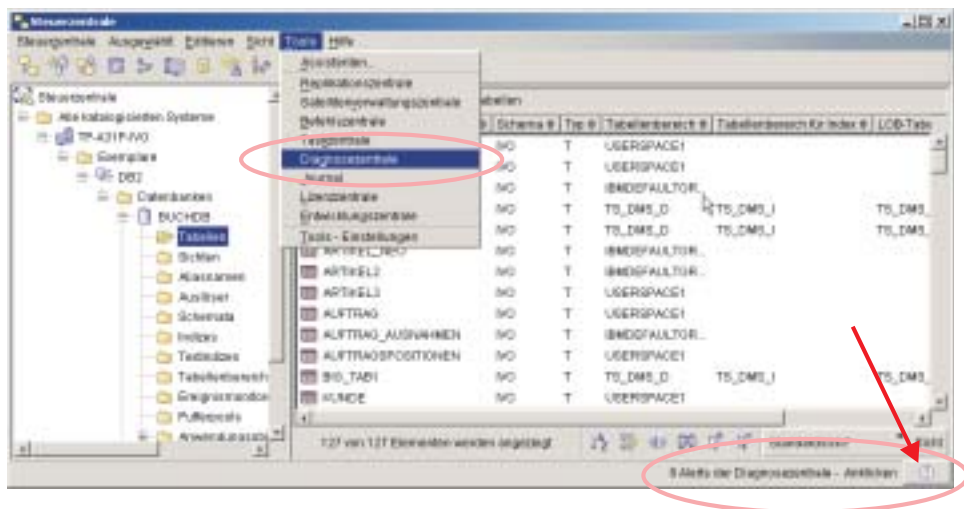


Abbildung 14.22: Aufrufmöglichkeiten der Diagnosezentrale

In der Diagnosezentrale angelangt, ist die Anzeige sehr unterschiedlich. Wurde der Diagnosemonitor schon früher eingeschaltet, so sehen Sie hier schon möglicherweise einige Alarme (Alerts; siehe Abbildung 14.23). Steigen Sie ganz neu ein, so erhalten Sie einen fast leeren Bildschirm angezeigt und müssen den Diagnosemonitor erst starten und konfigurieren (siehe Abbildung 14.25).

Die Diagnosezentrale (siehe Abbildung 14.23) zeigt Ihnen eine Übersicht über alle aufgetretenen Alarme (Alarm oder Alert – ein Signal, das einen Status eines Objekts darstellt), also alle über- bzw. unterschrittenen Schwellenwerte, die vom DB2 schon standardmäßig gesetzt sind. Um diese Standardeinstellungen zu ändern, sehen wir uns Abbildung 14.25 an.

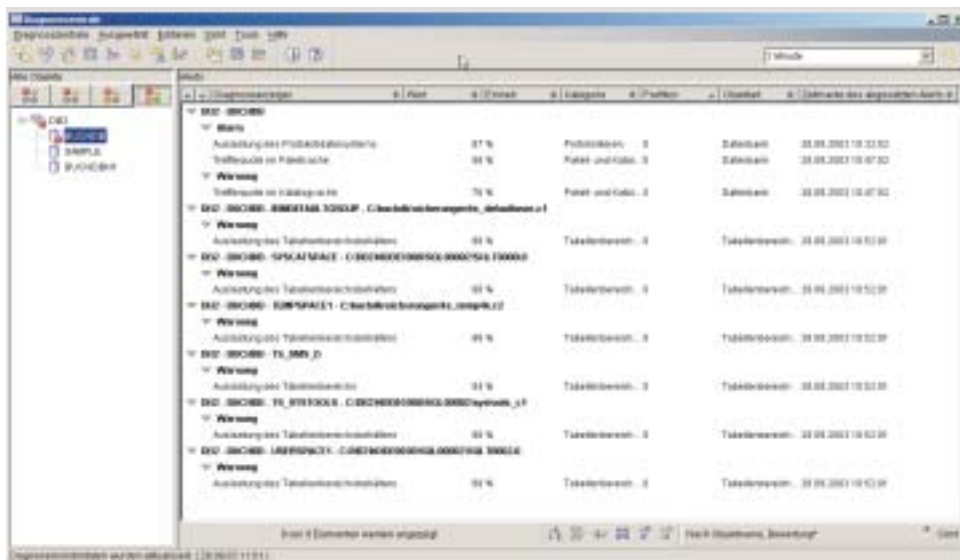


Abbildung 14.23: Diagnosezentrale mit Übersicht über Alarmer

Hinweis:

Um Informationen über die anstehenden Alarmer (alerts) aus der Diagnosezentrale zu erhalten, sollten Sie in den TOOLS-Einstellungen den Reiter BEACONNACHRICHT DER DIAGNOSEZENTRALE auswählen, um dort mindestens BENACHRICHTIGUNG ÜBER STATUSZEILE auszuwählen. Die andere Option (BENACHRICHTIGUNG ÜBER DIALOGNACHRICHT) sollte mit Bedacht ausgewählt werden, da diese Benachrichtigungsdialoge auf dem Bildschirm stehen bleiben und so die Weiterarbeit behindern.

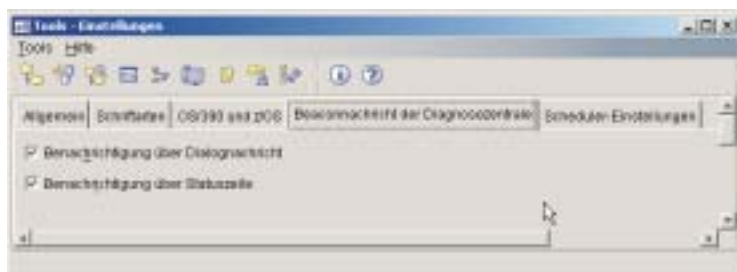


Abbildung 14.24: Beaconnachrichten der Diagnosezentrale

Wollen Sie nun den Diagnosemonitor über die Steuerzentrale aktivieren, so reicht es, wenn Sie auf den Namen des Exemplars klicken und dann aus dem Kontextmenü DIAGNOSEMONITOR STARTEN auswählen. Der Diagnosemonitor wird sofort gestartet. Sie erhalten ein Farbsymbol über den Status des Exemplars und die

darunter liegenden Datenbanken, das Ihnen signalisiert, ob eine Warnung oder ein Alarm ansteht. Bis die Alarm-Anzeige komplett ist, kann es einige Minuten dauern, haben Sie also hier etwas Geduld. Eventuell haben Sie in Ihrem Exemplar bzw. Ihrer Datenbank auch keine Schwellenwerte, die überschritten worden sind.

Zur Auswahl (siehe Abbildung 14.25) der globalen Standardwerte auf Exemplar-Ebene wählen Sie GLOBALE STANDARDWERTE FÜR DIAGNOSEANZEIGER aus. Zur Auswahl der Einstellungen auf Datenbank-Ebene müssen Sie den Eintrag EINSTELLUNGEN FÜR DIAGNOSEANZEIGER VON DATENBANKOBJEKTEN selektieren.

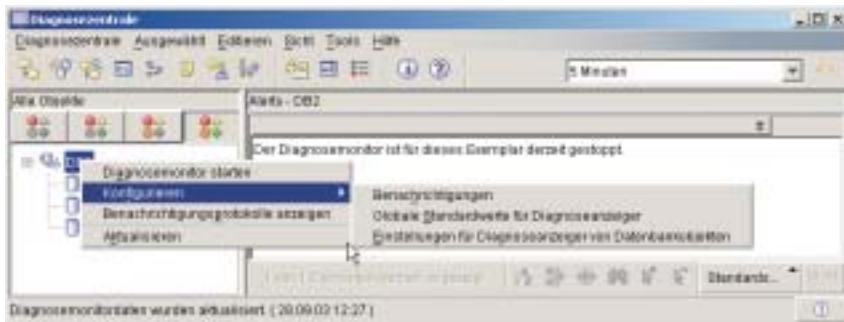


Abbildung 14.25: Diagnosemonitor starten und konfigurieren

Werden die Datenbankobjekte zur Konfiguration ausgewählt, so erhalten Sie das Konfigurationsmenü für alle möglichen Diagnosezeiger, die fast alle mit Schwellenwerten belegt sind und hier auch verändert werden können. Diese Auswahl zeigt keine Informationen über Tabellenbereiche an. Doch in der Praxis ist oft die Frage nach der Auslastung des Tabellenbereichs sehr wichtig, vor allem für DMS-Tabellenbereiche.

Um auf die Ebene der Tabellenbereiche zu kommen, müssen Sie bei der Objekt-Auswahl den rechten „...“-Button (siehe Abbildung 14.26) anklicken, um in die Auswahl der Datenbankobjekte zu kommen. Von hier aus können Sie nun die Tabellenbereiche einzeln auswählen – ja sogar der einzelne Behälter kann ausgewählt werden – und dort die jeweiligen Schwellenwerte neu vergeben. Der Standard für den Tabellenbereich steht auf folgenden Werten: WARNUNG bei 80%-Füllstand und ALARM bei 90%-Füllstand.

Beim Erreichen dieser Schwellenwerte können auch Aktionen eingeleitet werden. Diese Aktionen können vorbereitete Prozeduren (z.B. Anruf durch ein Telefonwählprogramm) oder eine Task aus der eigenen Taskzentrale (z.B. ein REORG) sein.

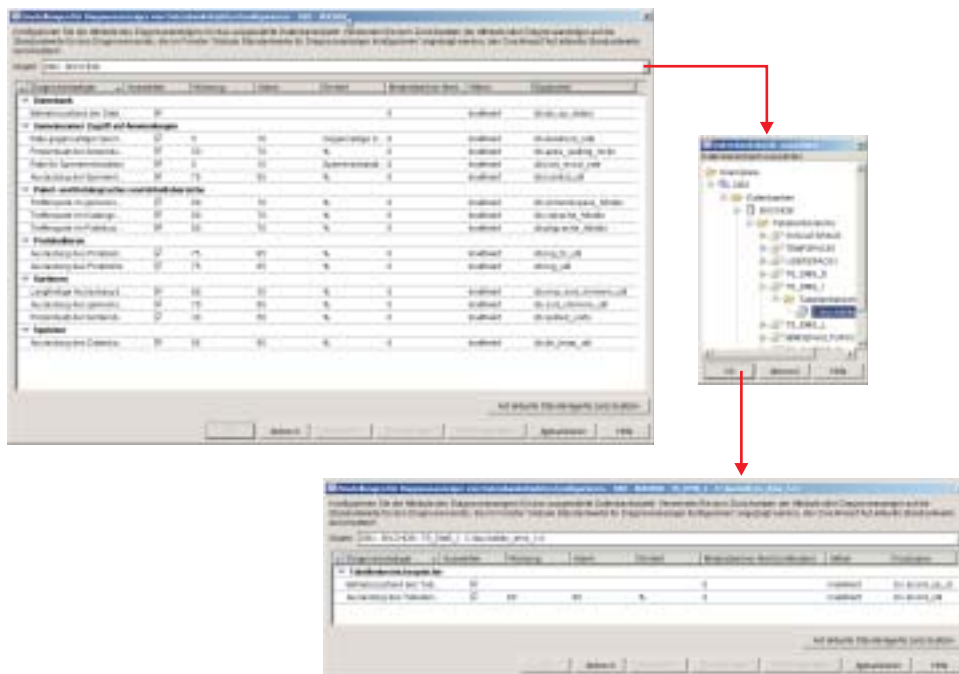


Abbildung 14.26: Tabellenbereiche im Diagnosemonitor

Diese Konfiguration, die Sie über die Diagnosezentrale durchgeführt haben, ist auch über die Befehlszeile möglich. Um die Konfiguration des Diagnosemonitors auf Exemplar-Ebene zu erhalten, geben Sie folgenden Befehl ein (siehe Abbildung 14.27):



Abbildung 14.27: Bildschirmausgabe von db2 get alert cfg for dbm

Wenn Sie wissen wollen, wie die Schwellenwerte für einen bestimmten Tabellenbereich (hier: TS_DMS_D) gesetzt sind, so müssen Sie folgenden Befehl absetzen – siehe Abbildung 14.28:

```
C:\buchdb>db2 get alert cfg for tablespace ts_dms_d on buchdb
```

Alertkonfiguration	
Anzeigername	= ts.ts_op_status
Typ	= Statusbasiert
Sicherheitsstufe	= 0
Formel	= ts.ts_status;
Aktionen	= Disabled
Schwellenwert- oder Statusüberprüfung	= Enabled
Anzeigername	= ts.ts_util
Typ	= Schwellenwertbasiert
Warnung	= 80
Alarm	= 90
Sicherheitsstufe	= 0
Formel	= ((ts.ts_used_pages/ts.ts_usable_pages)*100);
Aktionen	= Disabled
Schwellenwert- oder Statusüberprüfung	= Enabled

Abbildung 14.28: Bildschirmausgabe von db2 get alert cfg for tablespace ...

Um nun einen Schwellenwert anzupassen, müssen Sie den Anzeigernamen (KURZNAME im GUI) kennen. Diesen erhalten Sie am einfachsten aus der Übersicht in der Diagnosezentrale, unter der Spalte Kurzname (siehe Abbildung 14.26). Die Beschreibung dieses Kurznamens können Sie sich über den Befehl aus Abbildung 14.29 anzeigen lassen.

```
C:\buchdb>db2 get description for health indicator ts.ts_util
```

DESCRIPTION FOR ts.ts_util
Dieser Diagnoseanzeiger zeigt die Belegung des Speichers für jeden DMS-Tabellenbereich an. Der DMS-Tabellenbereich wird als voll betrachtet, wenn alle Behälter voll sind. Der Bezugswert wird anhand der Formel $(ts.used / ts.useable) * 100$ berechnet, wobei ts.used und ts.useable den Systemmonitordatenelementen tablespace_used_pages bzw. tablespace_usable_pages entsprechen. Die Auslastung des Tabellenbereichs wird als Prozentsatz des verwendeten Speichers gemessen, wobei ein hoher Prozentsatz eine nicht optimale Funktion für diesen Bezugswert anzeigt. Mit den kurzfristigen und langfristigen Wachstumsraten, die in den zusätzlichen Informationen für diesen Bezugswert enthalten sind, kann ermittelt werden, ob die derzeitige Wachstumsrate eine kurzfristige Abweichung darstellt oder ob sie längerfristig Wachstum entspricht. Die Berechnung der verbleibenden Zeit bis zur vollen Belegung in den zusätzlichen Informationen ist eine Prognose, wie viel Zeit verbleibt, bis der gesamte freie Speicher belegt ist.

Abbildung 14.29: Bildschirmausgabe von db2 get description for health indicator....

Die Schwellenwerte für einen Diagnosezeiger können mit dem Befehl UPDATE ALERT CFG FOR... angepasst werden. Die Optionen sind sehr vielfältig, da auch Aktionen über die Befehle definiert werden können, wie auch schon im GUI kurz vorgestellt wurde (siehe Abbildung 14.26). Ein kompletter Befehl könnte folgendermaßen lauten:

```
update alert cfg for tablespace ts_dms_d on buchdb using ts.ts_util set alarm 80, warning 60
```

In diesem Befehl wird für den Tabellenbereich TS_DMS_D der Schwellenwert für die Warnungen auf 60% gesetzt. Das heißt, bei einem Füllgrad von 60% erhalten Sie eine Nachricht, falls Sie die Beaconnachrichten, wie in Abbildung 14.24 gezeigt, aktiviert haben. Diese Information wird dann auch in der Diagnosezentrale sichtbar.

Wollen Sie nun wissen, welcher Diagnosezeiger welchen Status hat – bezogen auf eine Datenbank –, so können Sie dies über folgenden Befehl herausbekommen:

```
get health snapshot for all on buchdb
```

Die Ausgabe ist die gleiche, jedoch nicht ganz so übersichtlich wie die Anzeige in der Diagnosezentrale. In der Diagnosezentrale haben Sie die Möglichkeit, die einzelnen Diagnosezeiger „einzuklappen“, so dass Sie nicht alle untergeordneten Optionen sehen, was die Übersichtlichkeit verbessert (siehe Abbildung 14.30).

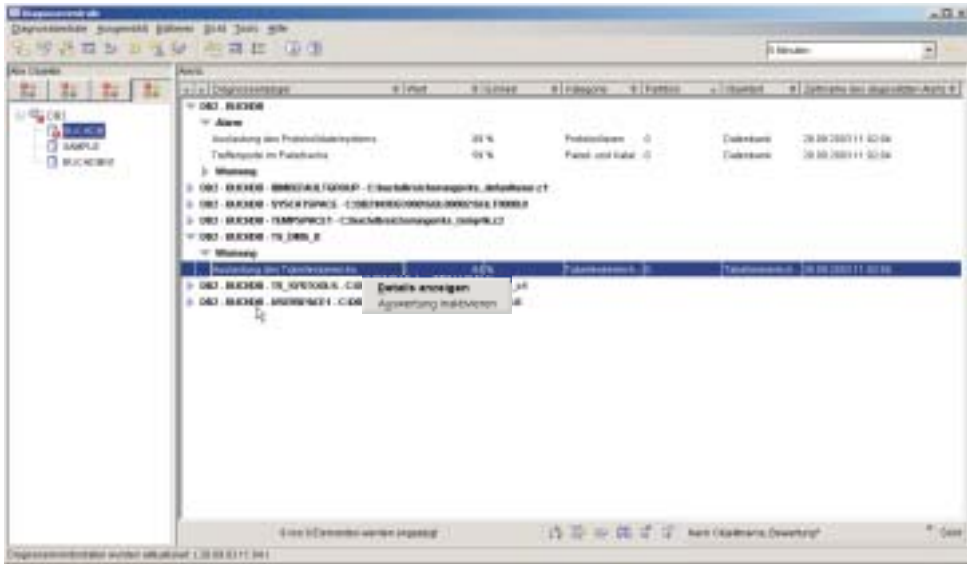


Abbildung 14.30: Diagnosezentrale mit Warnungen zum Tabellenbereich

Die Vorteile, die die Diagnosezentrale nun voll ausspielen kann, bestehen in der Möglichkeit, direkt aus dem Menü die Details aufzurufen (siehe Abbildung 14.30) und durch weiteres Navigieren auf den Button PROTOKOLL und den Reiter EMPFEHLUNGEN zu gelangen (siehe Abbildung 14.31).

Wenn Sie auf den PROTOKOLL-Button klicken, werden Ihnen in einem Fenster die Zeiten angezeigt, an denen die Schwellenwerte erreicht wurden, und die tatsächlichen Werte, die zu diesem Zeitpunkt im System gemessen wurden. Im Beschreibungsfeld sehen Sie die genaue Bedeutung des Diagnosezeigers mit der zugehörigen Formel, die verwendet wird, um den Wert des Diagnosezeigers zu berechnen.

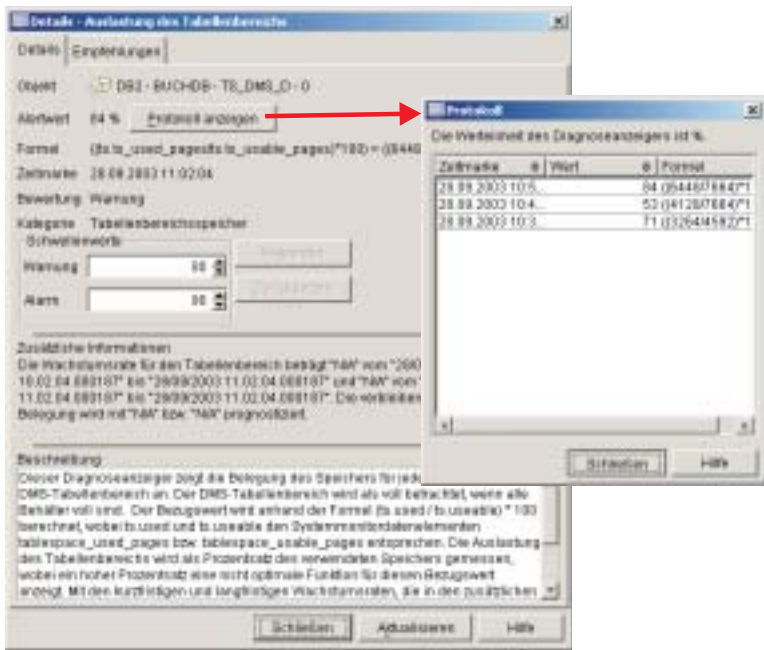


Abbildung 14.31: Detailbeschreibung der Ausnahme mit Protokoll

Gehen Sie nun zur Registerkarte EMPFEHLUNGEN, so erhalten Sie eine Auflistung von Vorschlägen, hier AKTION (siehe Abbildung 14.32) genannt, die Sie dabei unterstützen, die überschrittenen Schwellenwerte wieder zu unterschreiten. Sie erhalten Vorschläge, wie Sie am besten diese Warnung bzw. den Alarm beseitigen. Diese Ratschläge münden nicht immer in einer Änderung von Konfigurationsparametern oder in einer Vergrößerung eines Tabellenbereichs, sondern sind oft auch nur Hinweise zur weiteren Analyse der Fehlersituation. Es steht Ihnen danach offen, diese Vorschläge umzusetzen.

Wenn Sie eine der Empfehlungen auswählen, erhalten Sie im folgenden Text eine genaue Anweisung bzw. Beschreibung der empfohlenen Vorgehensweise (siehe Abbildung 14.33).



Abbildung 14.32: Empfehlungen zur Ausnahmebehandlung

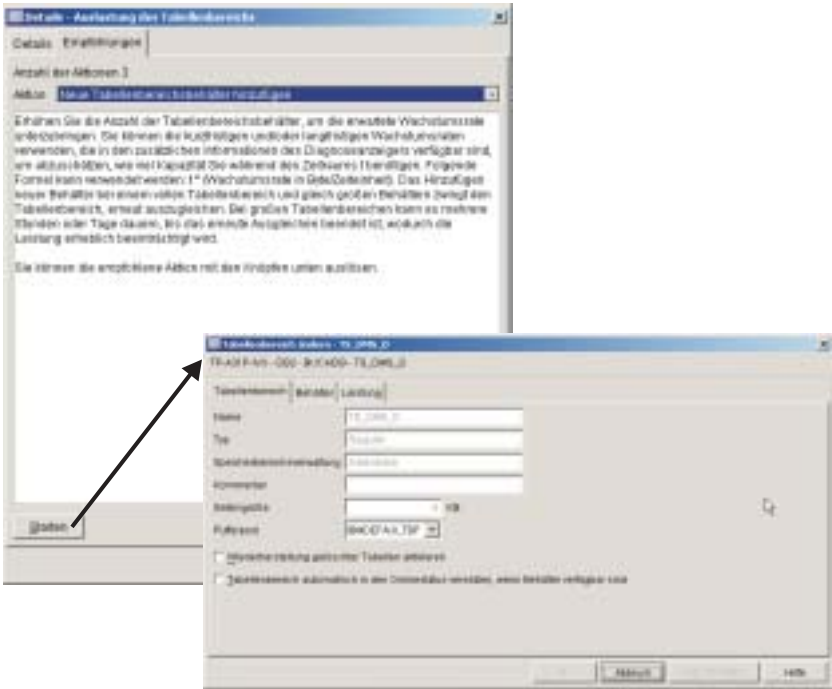


Abbildung 14.33: Ausführung der Empfehlung

Sind Sie mit dieser Vorgehensweise einverstanden und wollen Sie diese Aktion auch gleich in die Tat umsetzen, so klicken Sie auf den START-Button. Daraufhin wird das zugehörige Menü oder Fenster aus der Steuerzentrale aufgerufen, um die vorgeschlagene Aktivität sofort in eine richtige Aktion in DB2 umzuleiten bzw. durchzuführen.

Diese Möglichkeiten bietet die Befehlszeile nicht. Hier können Sie sich zwar die Empfehlungen anzeigen lassen, aber die Möglichkeit, diese Empfehlungen gleich in die Tat umzusetzen, ist nicht gegeben.

Ein Beispiel von der Befehlszeile könnte so wie in Abbildung 14.34 aussehen (die Ausgabe ist aus Platzgründen stark gekürzt).

```
C:\buchdb>db2 get recommendations for health indicator ts.ts_util

RECOMMENDATIONS FOR ts.ts_util

Hauptspeicherauslastung untersuchen
Um eine umfassende Sicht der Auslastung des Tabellenbereichsspeichers zu analysieren,
starten Sie das Speicherverwaltungstool. Dieses Tool kann so konfiguriert
werden, dass es Momentaufnahmen vom Speicher als Basis für eine Trendanalyse nimmt.

..

Neue Tabellenbereichsbehälter hinzufügen
Erhöhen Sie die Anzahl der Tabellenbereichsbehälter, um die erwartete Wachstumsrate
unterzubringen. Sie können die kurzfristigen und/oder langfristigen Wachstum
sraten verwenden, die in den zusätzlichen Informationen des Diagnoseanzeigers verfügbar sind,
um abzuschätzen, wie viel Kapazität Sie während des Zeitraums t be
nötigen. Folgende Formel kann verwendet werden: t * (Wachstumsrate in Byte/Zeiteinheit).
Das Hinzufügen neuer Behälter bei einem vollen Tabellenbereich und glei
ch großen Behältern zwingt den Tabellenbereich, erneut auszugleichen. Bei großen
Tabellenbereichen kann es mehrere Stunden oder Tage dauern, bis das erneute Aus
gleichen beendet ist, wodurch die Leistung erheblich beeinträchtigt wird.

Über die Steuerzentrale:
1. Erweitern Sie die Objektbaumstruktur unter der Elterndatenbank für den
gewünschten Tabellenbereich, und wählen Sie den Ordner aus.
2. Klicken Sie mit Maustaste 2 zuerst den gewünschten Tabellenbereich
in der Detailsicht an, und klicken Sie danach im Popup-Menü 'Ändern' an.
Das Notizbuch 'Tabellenbereich ändern' wird geöffnet.
3. Klicken Sie die Seite 'Behälter' an. Wählen Sie 'Hinzufügen' aus, um
den Dialog 'Behälter definieren' zu öffnen.
4. Füllen Sie die erforderlichen Felder im Dialog 'Behälter definieren' aus,
und klicken Sie 'OK' an, um den neuen Behälter hinzuzufügen. Der Dialog
'Behälter definieren' wird geschlossen, und die Seite 'Behälter' des
Notizbuchs 'Tabellenbereich ändern' wird wieder in den Vordergrund gestellt.

Im Befehlszeilenprozessor geben Sie wie im folgenden Beispiel dargestellt ein:
ALTER TABLESPACE TS1 (FILE 'FILE3' 100, DEVICE '/dev/rdisk3' 200)

Dabei gilt Folgendes:

TS1
Der Name des Tabellenbereichs

(FILE 'FILE3' 100, DEVICE '/dev/rdisk3' 200)
Der Name und der Pfad des Behälters
```

Abbildung 14.34: get recommendation for health indicators-Syntax mit Ausgabe

Bei der Überschreitung der Schwellenwerte von genutzten Datenbankspeichern, wie z.B. die Trefferquote im Katalogcache, Trefferquote im Packagecache oder Auslastung des DatenbankzwischenSpeichers, lautet eine der möglichen Empfehlungen, den Speicher über einen längeren Zeitraum zu verfolgen. Dazu wird dann der Memory Visualizer aufgerufen.

14.7 Memory Visualizer

Der Memory Visualizer hilft Ihnen, einen Überblick in grafischer Form über die Speichernutzung in *einem* DB2-Exemplar zu erhalten. Es können hier zusätzlich Schwellenwerte für die einzelnen Speicherbereiche gesetzt werden. Beim Erreichen dieser Schwellenwerte wird bei den jeweiligen Objekten ein Farbcode (roter Kreis = Alarm oder gelbes Dreieck = Warnung) angezeigt, um eine schnelle Übersicht zu erhalten.

Sie können den Memory Visualizer aus einer Empfehlung der Diagnosezentrale aufrufen oder ihn eigenständig als Überwachungstool mit dem Befehl `db2memvis` starten.

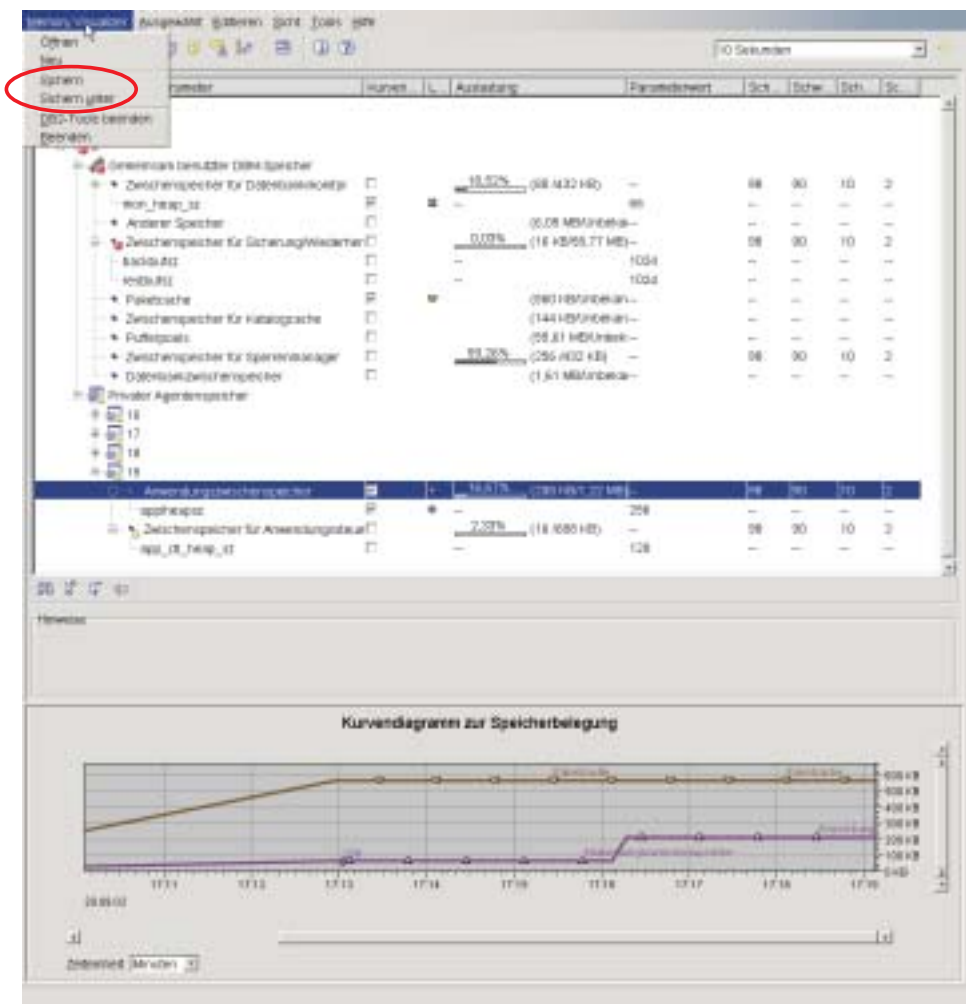


Abbildung 14.35: Memory Visualizer

Längere Zeiträume können protokolliert und gesichert werden, indem Sie die Daten unter einem eigenen Namen sichern (siehe Abbildung 14.35). Diese Sicherung kann jederzeit wieder angesehen werden, und durch das Verschieben des unteren Scroll-Balkens unter dem Kurvendiagramm ist eine Ansicht über die Zeit möglich, die hier gespeichert wurde. Mit dem rechten Scroll-Balken kann die Skala mit den Speicherangaben angepasst werden, um so alle Speicherwerte, große und kleine, eindeutig zu erkennen.

In der oberen rechten Ecke ist die Eingabe der Zeitintervalle möglich. Diese können Sie in Stufen von fünf Sekunden bis zu vier Stunden angeben. Diese Intervalle sollten Sie mit Bedacht wählen, da zu diesem Zeitpunkt ein Snapshot durchgeführt wird und die erhaltenen Daten im Memory Visualizer grafisch angezeigt werden (siehe Abbildung 14.35). Hier sollte kein zu kleiner Wert gewählt werden, um nicht unnötig das Betriebssystem zu belasten.

Da die grafische Darstellung oft auch zu einem Performance-Problem führt, wird vor allem beim Monitoring auf grafische Darstellungen verzichtet bzw. werden die Tools zur Darstellung separat auf einem eigenen System betrieben, das getrennt von den Datenbank- bzw. Anwendungsservern läuft.

Der Memory Visualizer kann auch von der Befehlszeile aus aufgerufen werden und hat auch noch eine nützliche Zusatzanzeige über die Summe des Speicherverbrauchs.

Über die Befehlszeile rufen Sie den Memory Visualizer wie folgt auf: db2mtrk

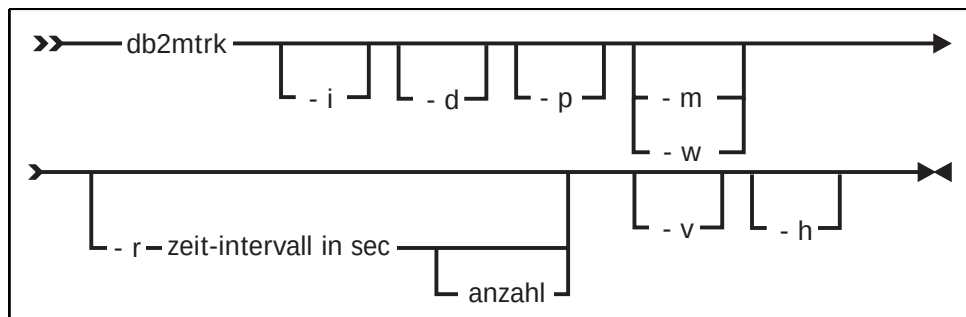


Abbildung 14.36: db2mtrk-Syntaxdiagramm

Die wichtigsten Optionen beim db2mtrk-Befehl sind:

- ▶ -i
Speicher auf Exemplarebene. Unter Windows werden auch alle Datenbankspeicher mit aufgeführt.
- ▶ -d (nur für Unix-Systeme)
Speicher auf Datenbank-Ebene. Jede Datenbank wird einzeln aufgelistet.
- ▶ -p
Speicher auf Anwendungsebene. Jeder Agent, der mit einer Datenbank verbunden ist, wird einzeln mit seinem Speicherverbrauch aufgelistet.

► -m

Maximalbereiche werden hier aufgezeigt, d.h. welche Speichergröße für den einzelnen Bereich genutzt werden könnte.

► -w

Hier wird der Maximalwert, der bisher benötigt wurde, angezeigt.

► -r zeit-intervall in sekunden

Für Wiederholungen des Befehls kann ein Zeitintervall in Sekunden angegeben werden.

► anzahl

Die Anzahl der Wiederholungen wird hier eingeschränkt.

► -v

Die Ausgabe ist „sprechender“, da die Abkürzungen in Langform ausgeschrieben werden. So wird statt

utilh	pckcacheh	catcacheh	bph	bph	bph
16,0K	160,0K	80,0K	1,1M	640,0K	384,0K

der gleiche Inhalt in ausführlicher Form dargestellt:

Backup/Restore/Util Heap hat die Größe 16384 Byte
 Package Cache hat die Größe 163840 Byte
 Catalog Cache Heap hat die Größe 81920 Byte
 Buffer Pool Heap hat die Größe 1179648 Byte
 Buffer Pool Heap hat die Größe 393216 Byte

► -h

Hier wird die Syntax erklärt.

Bei den folgenden Beispielen sollten Sie beachten, dass mit der Option „-v“ nicht nur ein ausführlicher Text für die Speicherobjekte angezeigt wird, sondern auch immer eine „Gesamt“-Summe der Speicherbelegung zum Ende jeder Teilauflistung gezeigt wird.

Im ersten Beispiel wurde der Speicherverbrauch eines Exemplars abgefragt, in dem noch keine einzige Datenbank gestartet war bzw. in dem es noch keine aktive Verbindung zu einer der Datenbanken des Exemplars gab.

```
C:\BuchDB>db2mtrk -i -w -v
```

```
Speicher wird überwacht am 2003/09/29 um 17:42:53.
```

```
Speicher für Exemplar
```

```
Database Monitor Heap hat eine Grenze von 16384 Byte
```

```
Other Memory hat eine Grenze von 6029312 Byte
```

```
Gesamt: 6045696 Byte
```

Das zweite Beispiel zeigt den Speicherverbrauch eines Exemplars mit fünf Datenbanken und den dazugehörigen Verbindungen.

```
C:\BuchDB>db2mtrk -i -p -w -v
```

```
Speicher wird überwacht am 2003/09/29 um 17:47:16.
```

```
Speicher für Exemplar
```

```
Backup/Restore/Util Heap hat eine Grenze von 16384 Byte
Package Cache hat eine Grenze von 573440 Byte
Catalog Cache Heap hat eine Grenze von 147456 Byte
Buffer Pool Heap hat eine Grenze von 56803328 Byte
Buffer Pool Heap hat eine Grenze von 655360 Byte
Buffer Pool Heap hat eine Grenze von 393216 Byte
Buffer Pool Heap hat eine Grenze von 262144 Byte
Buffer Pool Heap hat eine Grenze von 196608 Byte
Lock Manager Heap hat eine Grenze von 262144 Byte
Database Heap hat eine Grenze von 1687552 Byte
Database Monitor Heap hat eine Grenze von 16384 Byte
Other Memory hat eine Grenze von 6225920 Byte
Gesamt: 67239936 Byte
```

```
Speicher für Agent 4280
```

```
Application Heap hat eine Grenze von 212992 Byte
Application Control Heap hat eine Grenze von 16384 Byte
Gesamt: 229376 Byte
```

```
Speicher für Agent 3684
```

```
Application Heap hat eine Grenze von 147456 Byte
Application Control Heap hat eine Grenze von 16384 Byte
Gesamt: 163840 Byte
```

Diese Informationen sind ein Auszug und eine Zusammenfassung von DB2-System-Monitorinformationen, die auch über die Snapshot-Funktion abgefragt werden können.

14.8 Fehlermonitor (nur in Unix-Umgebungen)

Auf Unix-Systemen kann die Verfügbarkeit eines DB2-Exemplars (instance) erhöht werden, wenn der Fehlermonitor (db2fmd) eingeschaltet wird.

Nach einer DB2-Installation bzw. nach dem darauf folgenden „boot“ startet automatisch der Fehlermonitor-Koordinator (Prozess: db2fmc). Dieser Fehlermonitor-Koordinator überwacht die oder den Fehlermonitor (Prozess: db2fmd), und ein Fehlermonitor überwacht jeweils ein Exemplar bzw. auch den DAS-Prozess.

Wird nun ein Fehlermonitor (db2fmd) unnormal beendet, so wird er vom Fehlermonitor-Koordinator (db2fmc) wieder automatisch gestartet. Wird ein Exemplar (db2sysc) nicht korrekt beendet, so wird das DB2-Exemplar vom Fehlermonitor (db2fmd) wiederum automatisch gestartet.

Nach welchen Regeln der Fehlermonitor arbeitet, ist in der Fehlermonitor-Konfigurationsdatei im \$INSTHOME/sql11b/-Verzeichnis des jeweiligen Exemplars verzeichnet. Der Dateiname ist wie folgt aufgebaut: fm.<maschinen-name>.reg. Diese Datei kann mit Hilfe des db2fm-Befehls geändert werden. Folgende Einträge stehen in dieser Datei:

```
FM_ON = no
FM_ACTIVE = yes
START_TIMEOUT = 600
STOP_TIMEOUT = 600
STATUS_TIMEOUT = 20
STATUS_INTERVAL = 20
RESTART_RETRIES = 3
ACTION_RETRIES = 3
NOTIFY_ADDRESS = <exemplar-name>@<maschinen-name>
```

Die Optionen haben folgende Bedeutungen:

- ▶ **FM_ON**
Sie gibt an, ob der Fehlermonitor gestartet werden soll oder nicht. Der Standardwert ist NO. Wird der Wert für einen bereits aktivierten Fehlermonitor auf NO gesetzt, so wird dieser gestoppt.
- ▶ **FM_ACTIVE**
Der Fehlermonitor führt nur dann Aktionen aus, wenn die Parameter FM_ON und FM_ACTIVE auf YES gesetzt sind. Der Standardwert ist YES.
- ▶ **START_TIMEOUT**
Gibt die Zeitspanne an, innerhalb derer der Fehlermonitor den Service starten muss, den er überwacht. Der Standardwert ist 600 Sekunden.
- ▶ **STOP_TIMEOUT**
Gibt die Zeitspanne an, innerhalb derer der Fehlermonitor den Service beenden muss, den er überwacht. Der Standardwert ist 600 Sekunden.
- ▶ **STATUS_TIMEOUT**
Gibt die Zeitspanne an, innerhalb derer der Fehlermonitor den Status des Service abrufen muss, den er überwacht. Der Standardwert ist 20 Sekunden.
- ▶ **STATUS_INTERVAL**
Gibt die Mindestzeit des Messintervalls des überwachten Service an. Der Standardwert ist 20 Sekunden.
- ▶ **RESTART_RETRIES**
Gibt an, wie oft der Fehlermonitor nach einem fehlgeschlagenen Aufruf versuchen soll, den Status des überwachten Service erneut abzurufen. Wird der hier angegebene Wert erreicht, versucht der Fehlermonitor, den Service erneut zu starten.
Der Standardwert ist 3.

► ACTION_RETRIES

Gibt an, wie oft der Fehlermonitor versuchen soll, den überwachten Service wieder in den Online-Status zu versetzen. Der Standardwert ist 3.

► NOTIFY_ADDRESS

Gibt die E-Mail-Adresse an, an die der Fehlermonitor eine Hinweismnachricht senden soll.

Um Veränderungen in dieser Datei durchzuführen, gibt es den db2fm-Befehl. Mit Hilfe dieses Befehls können Sie alle Optionen ändern sowie den Status des Monitors abfragen.

Um den Status des Exemplars abzufragen, setzen Sie folgenden Befehl ab:

```
db2fm -i <exemplar-name> -s
```

Um alle möglichen Optionen zu erhalten, geben Sie folgenden Befehl ein:

```
db2fm
```

Wollen Sie die Überwachung durch den Fehlermonitor aktivieren, so setzen Sie folgenden Befehl ab:

```
db2fm -i <exemplar-name> -f yes
```

Wenn Sie den Wert für START_TIMEOUT auf 20 Sekunden setzen wollen und den Wert für STOP_TIMEOUT auf 40 Sekunden, setzen Sie folgenden Befehl ab:

```
db2fm -i db2inst1 -T 20/40
```

14.9 Tipps aus der Praxis

14.9.1 db2batch

Das db2batch-Tool ermöglicht es Ihnen, die Leistungsdaten eines oder mehrerer SQL-Befehle zu analysieren und zu vergleichen. Die Ausgabe des db2batch-Tools sind Snapshot-Monitorinformationen, die für jede SQL-Abfrage, die im db2batch-Tool gesetzt ist, festgehalten werden. Die Vergleiche, die vom Tool direkt aufgezeigt werden, beziehen sich auf den CPU-Verbrauch und die Laufzeit.

Folgende Einschränkungen sind zu berücksichtigen:

- Alle SQL-Befehle werden mit dem Isolation Level RR ausgeführt.
- Die maximale Größe eines SQL-Befehls beträgt 65.535 Zeichen.
- Die maximale Länge einer Eingabe-Zeile beträgt 3.898 Zeichen.
- LOB-Spalten dürfen in der Ausgabe nicht vorkommen.
- Alle SQL-Befehle müssen mit einem Terminierungszeichen beendet werden.
Der Standard ist ein Semikolon (;) bzw. kann mit der --#SET DELIMITER-Option neu gesetzt werden.

Der db2batch-Befehl hat folgenden Aufbau:

```
db2batch -d <dbname> -f <eingabe-datei> -r <ausgabe-datei>
```

Bestimmte Optionen, wie z.B. die Anzahl der Ausgabesätze oder die Optimierungsparameter, können in der Eingabedatei oder auch direkt mit dem Befehl angegeben werden. Wir zeigen Ihnen hier als Erstes den typischen Aufbau der Eingabedatei.

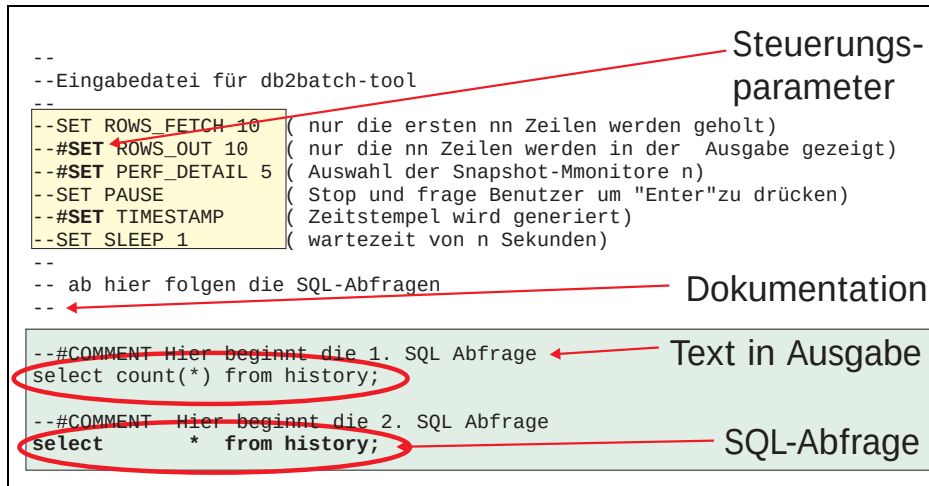


Abbildung 14.37: db2batch-Eingabedatei

In der Eingabedatei gelten Zeilen, die nur zwei Bindestriche „--“ enthalten, als Dokumentationszeilen. Diese Zeilen werden nicht in der Ausgabedatei angezeigt. Im Gegensatz dazu ist die „--#COMMENT“-Option für die Dokumentation in der Ausgabedatei gedacht.

Optionen, die mit „--#SET“ beginnen, sind zur Steuerung gedacht. In unserem Beispiel sehen Sie:

- --#SET ROWS_OUT – Hiermit bestimmen Sie die Anzahl der SQL-Ergebniszeilen, die in der Ausgabe angezeigt werden. Dieser Wert sollte sehr gering sein, da Sie normalerweise nicht an der Ausgabe, sondern an den Monitordaten interessiert sind.
- --#SET PERF_DETAIL – Hiermit bestimmen Sie den Detaillierungsgrad der Monitorinformationen, die in der Ausgabe angezeigt werden sollen. Abbildung 14.38 zeigt, welcher Detaillierungsgrad (1 bis 5) welche Informationen liefert.

Wie aus Abbildung 14.38 zu entnehmen ist, erhalten Sie mit PERF_DETAIL=5 alle verfügbaren Monitorinformationen.

Nun fehlt noch die SQL-Abfrage. In unserem Beispiel (siehe Abbildung 14.37) sind es zwei SQL-Abfragen, die hier miteinander verglichen werden sollen. Die SQL-Abfragen müssen immer mit einem „;“ beendet werden. Falls Sie hier ein anderes Zeichen wünschen, so ist dies mit der Option --#SET DELIMITER zu ändern.

db2batch	#SET PERF_DETAIL n oder -o p n				
Inhalt der Ausgabedatei	1	2	3	4	5
ROWS_OUT					
SUMMARY					
Database Snapshot - Kurzform					
Database Snapshot					
Application Snapshot					
Statement Details					
Database Manager Snapshot					
Bufferpool Snapshot					
Tablespace Snapshot					
Table Snapshot					

Abbildung 14.38: db2batch – PERF_DETAIL-Tabelle

```
--#SET ROWS_OUT 10 ( NUR DIE NN ZEILEN WERDEN IN DER AUSGABE GEZEIGT)
--#SET PERF_DETAIL 5 ( OPTIMIERER LEVEL N)
--#SET TIMESTAMP ( ZEITSTEMPEL WIRD GENERIERT)
Current Timestamp: Sun Oct 05 21:46:53 2003
Hier beginnt die 1. SQL Abfrage

Statement number: 1

select count(*) from history

1
-----
105836

Number of rows retrieved is: 1
Number of rows sent to output is: 1

Prepare Time is: 0,000 seconds
Execute Time is: 0,030 seconds
Fetch Time is: 0,000 seconds
Elapsed Time is: 0,030 seconds

*** Database Snapshot ***

Input database alias = TP1
Database Name = TP1NEW
Database Path = C:\DB2\NODE0000\SQL00002\
Database status = Active
Location of DB to snapshot application = Local
Server Platform = NT

Summary of Results

Statement # Elapsed Time (s) Agent CPU Time (s) Rows Fetched Rows Printed
1 0,030 0,020 1 1
2 0,611 0,180 105836 10

Arith. mean 0,321 0,1
Geom. mean 0,135 0,0601
```

Steuerungsparameter

Text in Ausgabe

SQL-Abfrage

ROWS_OUT

PERF_DETAIL

Abbildung 14.39: db2batch-Ausgabedatei

Die Ausgabe des db2batch-Tools erfolgt am Bildschirm, falls Sie sie nicht mit der Option „-r“ in eine Datei umleiten. Die Ausgabedatei hat – abhängig von der PERF_DETAIL-Option – die Merkmale, die in Abbildung 14.39 dargestellt sind.

In Abbildung 14.39 sehen Sie nochmals Teile der Eingabedatei. Die Optionen, die Sie gesetzt hatten, werden hier angezeigt. Felder, die Sie in der Eingabedatei mit --#COMMENT gekennzeichnet haben, werden nun ohne das Optionsfeld angezeigt.

Mit dem ROWS_OUT-Parameter können Sie die Anzahl der Ausgabezeilen einschränken. Die Snapshot-Informationen sind so ausführlich, wie Sie es mit der Option PERF_DETAIL eingestellt haben.

Am Ende des Reports sehen Sie die Zusammenfassung, bezogen auf die genutzte CPU-Zeit, sowie die tatsächliche Laufzeit (Elapsed Time) der SQL-Befehle.

15 Performance

Leistungssteigerung ist immer ein Thema – natürlich auch für Datenbanksysteme. Ein performantes System kann Kosten für neue, schnellere Systeme einsparen und gleichzeitig die Zufriedenheit der Anwender durch kürzere Antwortzeiten verbessern.

Dieses Kapitel behandelt grundlegende Konzepte der Performance und soll nur ein erster Einstieg in dieses vielfältige und sehr umfassende Themengebiet sein.

15.1 Der DB2-Optimizer

In Gesprächen zwischen Datenbankadministratoren werden Sie oft die Worte hören: „Der Optimizer ...“ oder „...der Optimizer hat umgeschrieben ...“. Was ist „der Optimizer“? Der *Optimizer* ist ein Teil des SQL-Compilers, der wiederum eine feste Komponente der DB2-Datenbank ist und für die *korrekte* und *optimale* Durchführung aller SQL-Befehle zuständig ist. Der SQL-Compiler hat mehrere Phasen, die jeder SQL-Befehl durchlaufen muss. Diese zeigen wir Ihnen in Abbildung 15.1.

15.1.1 Der SQL-Compiler

Eine SQL-Abfrage muss die Schritte 1 bis 7 durchlaufen, wenn der Zugriff auf eine zusammengeschlossene Datenbank (federated database) erfolgt. Wird *nicht* eine zusammengeschlossene Datenbank abgefragt, so werden die Schritte 4 und 6 nicht durchgeführt.

Die Schritte 1 bis 7 sind:

1. Abfrage analysieren

Die SQL-Anweisung wird im ersten Schritt analysiert, um die Syntaxfehler herauszubekommen. Für die Prüfung wird der SQL92-Standard mit Erweiterungen der SQL3-Spezifikation verwendet. Wird ein Fehler festgestellt, so wird der zugehörige Fehlercode an die Anwendung zurückgegeben. Wird kein Fehler festgestellt, so wird eine interne Darstellung der Anweisung im Abfragediagrammmodell (query graph model) abgelegt. Dieses Modell ist die Ausgangsbasis für die Bearbeitung durch die weiteren Schritte.

2. Semantik prüfen

In diesem Schritt prüft der Compiler, ob die Datentypen auch zu den Funktionen passen, z.B. ob beim Einsatz der Skalarfunktion YEAR auch der Datentyp DATE verwendet wird. Es wird geprüft, ob Sichten (Views) aufgelöst werden können oder ob Zwischentabellen angelegt werden müssen. Referenzielle Abhängigkeiten sowie Eindeutigkeitsregeln und Auslöser, falls vorhanden, werden in das Abfragediagrammmodell übernommen.

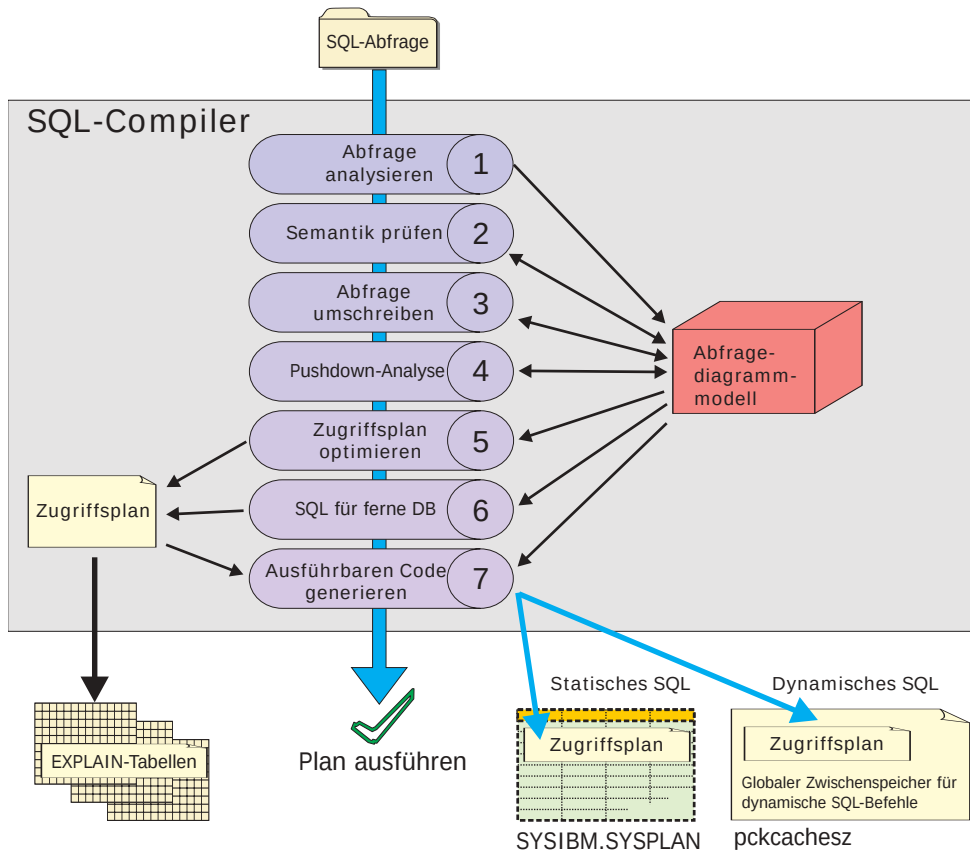


Abbildung 15.1: SQL-Compiler

3. Abfrage umschreiben

Aufgrund des Abfragediagrammmodells ist es dem DB2-Compiler oder auch dem DB2-Optimizer möglich, die SQL-Abfragen umzusetzen und dadurch zu optimieren. Hierbei wird garantiert, dass die umformulierte Abfrage die gleiche Ergebnismenge liefert wie die original formulierte Abfrage.

Bei der Umformulierung werden grundsätzlich drei Ziele verfolgt:

- Zusammenfügen von Operationen
 - Zusammenfügen von Sichten
 - Unterabfragen in Verknüpfungen umwandeln
 - Eliminierung von überflüssigen Verknüpfungen
 - Anzahl erforderlicher Berechnungen reduzieren

- Verschieben von Operationen
 - Eliminierung von DISTINCT
 - Allgemeine Verschiebung von Vergleichselementen, um die Vergleichselemente, die die Auswahl in größerem Maße einschränken, zum frühestmöglichen Zeitpunkt anzuwenden
 - Zwischenergebnisse auf anderen Datenbankpartitionen erstellen und damit die Anzahl der Verschiebevorgänge (Broadcast-Vorgänge) reduzieren
- Übersetzen von Vergleichselementen
 - Hinzufügen implizierter Vergleichselemente, um weitere Verknüpfungen zu ermöglichen
 - Transformationen von OR zu IN oder umgekehrt

4. Pushdown-Analyse

Dieser Punkt wird nur bei zusammengeschlossenen Datenbanken durchgeführt. Diese Funktion liefert eine Empfehlung, ob eine Operation an eine Datenquelle verschoben und dort auch ausgewertet werden kann.

5. Zugriffsplan optimieren

Mit dem aktuellen Inhalt des Abfragediagrammmodells wird nun der optimale Zugriffsplan ermittelt. Hierfür werden die Statistiken in den Systemkatalog-Tabellen über die abhängigen Tabellen, Indizes, Spalten und Funktionen verwendet, um den Zugriffsplan mit dem geringsten Ausführungsaufwand zu generieren. Bei dieser Generierung werden auch Konfigurationsparameter berücksichtigt, z.B. die Speicher für Sortierungen, die Größe des Pufferpools, die Anzahl der Prozessoren.

Dieser Plan wird in den Explain-Tabellen abgelegt, falls dies mit den zugehörigen Optionen gewünscht wurde.

6. Generierung von SQL für ferne Datenbanken

Der endgültige Plan kann aus mehreren Schritten bestehen, die zum Teil auch auf anderen Datenbanken ausgeführt werden. In diesem Schritt wird der SQL-Befehl, der auf einer fernen Datenbank ablaufen soll, für diese optimiert.

7. Ausführbaren Code generieren

Im letzten Schritt wird der Zugriffsplan in einen ausführbaren Zugriffsplan umgewandelt. Dabei wird die Umsetzung von Codepages und Hostvariablen berücksichtigt.

Der Zugriffsplan für ein statisches SQL wird in den Systemkatalog-Tabellen abgelegt (SYSIBM.SYSPLAN). Für dynamisches SQL wird dieser Zugriffsplan im Paket-Cache (pckcachesz) der Datenbank zwischengespeichert.

Sie fragen sich nun sicherlich, an welchem Punkt Sie den Optimizer beeinflussen können.

Sie können nicht die Regeln ändern bzw. nicht die internen Vorgehensweisen des Optimizers direkt beeinflussen. Doch Sie haben die Möglichkeit, über einige Konfigurationsparameter das Verhalten indirekt zu verändern. Beispielsweise sagen Sie dem DB2-Optimizer mit den Optimierungsklassen, welche internen Mechanismen

er einsetzen soll, ob bestimmte Systemstatistiken mit berücksichtigt werden sollen, welche Join-Konzepte Vorrang haben und vieles mehr.

15.1.2 Optimierungsklassen (QUERYOPT)

Mit den Optimierungsklassen bestimmen Sie die Mechanismen, nach denen der DB2-Optimizer eine SQL-Anweisung betrachtet, umschreibt und optimiert. Die Optimierungsklassen unterscheiden sich wie folgt:

- ▶ Optimierungsklasse 0
steht für einen minimalen Optimierungsaufwand und für die Bevorzugung von Index-Scans.
- ▶ Optimierungsklasse 1
steht auch für minimalen Aufwand, doch hier werden schon die Kosten gegenübergestellt, um festzustellen, ob ein Index-Scan oder ein Tabellen-Scan günstiger ist.
- ▶ Optimierungsklasse 2
Dies ist die ideale Einstellung für eine OLAP-(OnLine Analytical Processing) oder eine DSS-(Decision Support-)Umgebung, da hier der geringste Zeitaufwand für sehr komplexe SQL-Abfragen entsteht.
- ▶ Optimierungsklasse 3
Dieser Level entspricht dem Level des DB2 für z/OS-Optimizers.
- ▶ Optimierungsklasse 5
Dies ist die Standardeinstellung nach der Installation. Es ist die einzige Optimierungsklasse, die die Fähigkeit besitzt, sich nach „unten“ zu verändern, um nicht unnötige Zeit für die Optimierung zu verschwenden. Somit ist diese Klasse optimal für alle Arten von SQL-Abfragen.
- ▶ Optimierungsklasse 7
Sie ist der Optimierungsklasse 5 gleichzusetzen, aber ohne die Fähigkeit, die Optimierungsklasse herabzusetzen.
- ▶ Optimierungsklasse 9
Bei diesem Level gibt es keine Einschränkungen mehr. Unsere Empfehlung ist, diese Klasse nur bei sehr großen SQL-Abfragen bzw. bei Abfragen mit sehr großen Datenmengen einzusetzen. In der Regel ist die Optimierungsklasse 5 immer ausreichend.

15.1.3 Einstellen der Optimierungsklassen

Für statisches und dynamisches SQL werden die Optimierungsklassen unterschiedlich gesetzt.

Statisches SQL

Für eine statische SQL-Abfrage kann die Optimierungsklasse beim PreCompile (PREP) und beim Binden (BIND) des zugehörigen Programms als Option QUERYOPT mitgegeben werden. Hier ein Beispiel:

```
PREP testpgm.sqc QUERYOPT 3
```

Beim Binden der BND-Datei sähe es so aus:

```
BIND testpgm.bnd QUERYOPT 3
```

Wird die Option QUERYOPT nicht beim Precompile oder beim Binden der Anwendung angegeben, so gilt die Standardeinstellung. Diese ist in der Datenbankkonfiguration mit dem Parameter DFT_QUERYOPT festgelegt und hat nach der Installation den Wert 5.

Dynamisches SQL

Für dynamisches SQL kann das Register CURRENT QUERY OPTIMIZATION mit dem SQL-Befehl SET CURRENT QUERY OPTIMIZATION gesetzt werden.

Werden die SQL-Abfragen über eine CLI-Schnittstelle abgesetzt, so werden die Optionen für CLI-Programme in einer separaten Datei, der db2cli.ini, auf dem zugehörigen Client konfiguriert. In dieser Datei sollte dann der Parameter DB2OPTIMIZATION mit dem gewünschten Wert eingefügt werden.

Tipp:

Die db2cli.ini-Datei kann mit folgenden Befehlen ausgelesen

```
GET CLI CFG
```

und verändert werden:

```
UPDATE CLI CFG FOR SECTION section-name  
USING <schlüsselwort> <wert>
```

Die db2cli.ini-Datei steht im ../sqllib/-Verzeichnis und kann nur mit einer SYS-ADM-Berechtigung verändert werden.

Auch hier gilt wie bei dem statischen SQL-Beispiel: Wenn die Optimierungsklasse nicht explizit gesetzt wird, so gilt der Wert, der in der Standardeinstellung im Datenbankkonfigurationsparameter DFT_QUERYOPT steht.

15.2 Visual Explain

Wie schon in Abschnitt Abbildung 15.1 erläutert wurde, untersucht der SQL-Compiler Ihre SQL-Abfragen und ermittelt die beste Zugriffsmöglichkeit auf Ihre Daten. Dieser Pfad zu den Daten wird *Zugriffsplan* oder *Zugriffsweg* genannt. Sie können sich den Zugriffsplan mit dem DB2-Tool Visual Explain als Diagramm oder über das Tool db2explain in Text-Form anzeigen lassen. Das Diagramm ist eine visuelle Dar-

stellung der Datenbankobjekte, die an einer Abfrage beteiligt sind (z.B. Tabellen und Indizes). Darüber hinaus enthält das Diagramm die Operationen, die an diesen Objekten ausgeführt wurden (z.B. Verknüpfungsmethoden und Sortierungen) und zeigt den Datenfluss zwischen den Operationen und den Objekten an.

Sie können aufgrund der Analyse des Zugriffspfads mit den Explain-Tools die Performance des SQL-Befehls verbessern, indem Sie eine oder alle der folgenden Optimierungsaktivitäten ausführen:

- ▶ Optimieren Sie Ihren Tabellenentwurf, und reorganisieren Sie die Tabellendaten (siehe dazu auch Kapitel 13).
- ▶ Erstellen Sie die entsprechenden Indizes. Nutzen Sie dazu das db2advis-Tool, mit dem Sie fehlende Indizes finden können.
- ▶ Verwenden Sie den Befehl RUNSTATS, um dem Optimierungsprogramm die aktuellen Statistikdaten zur Verfügung zu stellen (siehe auch Kapitel 13).
- ▶ Anpassung der entsprechenden Konfigurationsparameter, z.B.:
 - bufferpool
 - avg_appls
 - sortheap
 - stmheap
 - maxlocks
- ▶ Wählen Sie die entsprechend Bindeoption aus (siehe auch Kapitel 11).
 - BLOCKING ALL
- ▶ Entwerfen Sie Abfragen, die nur die wirklich erforderlichen Spalten und Daten abrufen.
- ▶ Erstellen Sie EXPLAIN-Momentaufnahmen:
 - mit der Option FOR SNAPSHOT beim PREP- oder BIND-Befehl
Es wird nur ein grafischer Zugriffsplan erstellt.
 - mit der Option WITH SNAPSHOT beim PREP- oder BIND-Befehl
Es wird ein SNAPSHOT (grafischer Zugriffsplan) erstellt, und weitere Details werden in den EXPLAIN-Tabellen abgelegt.
 - mit der Option EXPLAIN SNAPSHOT EXPLAIN für dynamisches SQL
Es wird nur ein SNAPSHOT (grafischer Zugriffsplan) erstellt. Das SQL wird dabei nicht ausgeführt.
 - mit der Option EXPLAIN MODE YES für dynamisches SQL
In den EXPLAIN-Tabellen werden ein SNAPSHOT (grafischer Zugriffsplan) und weitere Details abgelegt.
 - mit dem Vergleichstool db2batch mit der Option -e 1|2|3

- ▶ Verwenden Sie ein Zugriffsplan-Diagramm, um einen Zugriffsplan zu verbessern:
 - Visual-Explain
 - SQL-Explain (db2expln oder dynexpln)

15.2.1 Wie kommen Sie an die Informationen des Tools Visual Explain?

Hier gibt es unterschiedliche Möglichkeiten:

- ▶ über einen Aufruf in der Steuerzentrale
- ▶ über einen Aufruf als interaktiven Befehl oder als eine Prozedur in der Befehlszentrale oder
- ▶ über eine direkte SQL-Abfrage auf die Explain-Tabellen. Bei diesen Abfragen ist es aber nicht möglich, die grafische Darstellung des Ablaufplans zu sehen.

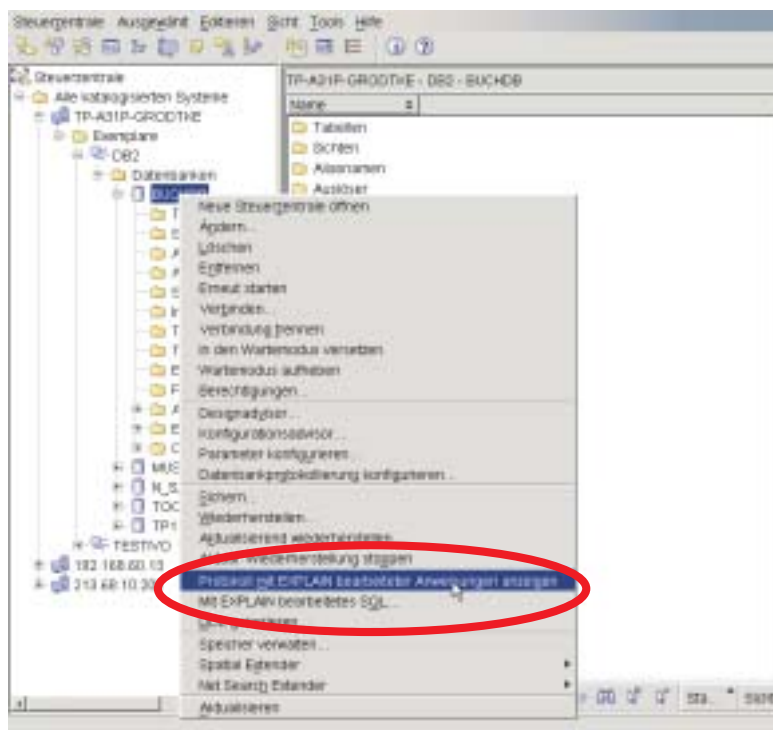


Abbildung 15.2: Aufruf des Explain-Tools über die Steuerzentrale

Beim Aufruf aus der Steuerzentrale haben Sie die Möglichkeit, wie in Abbildung 15.2 gezeigt, bestehende Informationen aus den Explain-Tabellen abzufragen. Entweder erfolgt dies durch der Auswahl PROTOKOLL MIT EXPLAIN BEARBEITETER ANWEISUNGEN ANZEIGEN, oder Sie analysieren einen neuen SQL-Befehl mit dem Explain-Tool über die Auswahl MIT EXPLAIN BEARBEITETETS SQL....

Hinweis:

Wenn Sie das Visual Explain-Tool zum ersten Mal nutzen, werden die notwendigen Explain-Tabellen automatisch vom DB2 in Ihrem Standard-Tabellenbereich angelegt. Dies kann der USERSPACE1, der IBMDEFAULTGROUP-Tabellenbereich oder der erste von Ihnen angelegte Tabellenbereich sein. Da in den Explain-Tabellen die Zugriffspläne als Bilder gespeichert werden (als LOBs) und somit sehr viel Platz benötigen, sollten Sie es vermeiden, dass diese Daten in einen Tabellenbereich gelegt werden, der eventuell für die Produktionsdaten vorgesehen war. Deswegen hier die Empfehlung: Legen Sie die Tabellen für das Explain-Tool bewusst selbst an, und fügen Sie die Tabellen in einen von Ihnen dafür vorgesehenen Tabellenbereich ein.

Die Tabellendefinition, also die DDL für diese EXPLAIN-Tabellen, steht im Verzeichnis ./sqlib/misc/ und lautet EXPLAIN.DDL. Kopieren Sie diese Datei in ein eigenes Arbeitsverzeichnis, und ergänzen Sie in der EXPLAIN.DDL-Datei die CREATE TABLE-Befehle um den Zusatz der eigenen Tabellenbereiche; also fügen Sie hinter jede Tabellendefinition noch ein IN <tabellebereichsname> an.

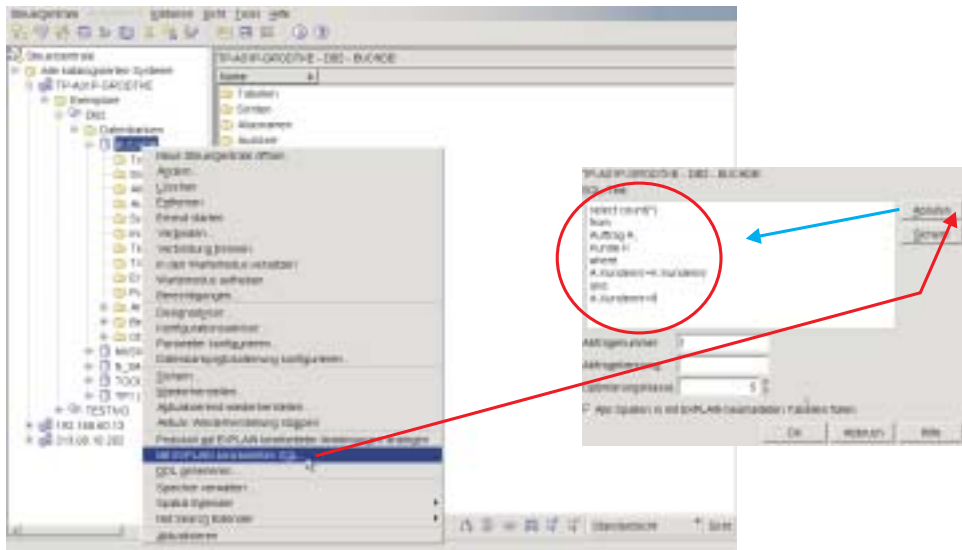


Abbildung 15.4: SQL-Befehlseingabe für das Visual Explain-Tool

Haben Sie den gewünschten SQL-Befehl geschrieben bzw. mit dem Button ABRUFEN in das Auswahlfenster geholt, so stehen Ihnen die Felder ABFRAGENUMMER und ABFRAGEKENNUNG zu Dokumentationszwecken zur Verfügung. Nutzen Sie diese Möglichkeit!

Im Auswahlfeld OPTIMIERUNGSKLASSE können Sie den Optimierungsgrad, den der DB2-Optimizer einsetzt, bestimmen, wie auch schon im Abschnitt 15.1.2 „Optimierungsklassen (QUERYOPT)“ beschrieben wurde.

Tipp:

Werden Tests mit der Optimierungsklasse durchgeführt, wird also der Wert im Feld OPTIMIERUNGSKLASSE verändert, um zu sehen, wie der DB2-Optimizer eventuell seinen Zugriffsplan ändert, so sollte der Wert auch in dem Feld ABFRAGEKENNUNG eingetragen werden, da bei der Übersicht die OPTIMIERUNGSKLASSE nicht angezeigt wird.

Haben Sie alle Einstellungen vorgenommen, können Sie nun den SQL-Befehl abschicken, um ihn analysieren zu lassen (einfach auf OK klicken).

Nach einer kurzen Zeitspanne, die von der Komplexität des SQL-Befehls sowie der ausgewählten Optimierungsklasse abhängt, wird Ihnen der Zugriffsplan angezeigt (siehe Abbildung 15.5).

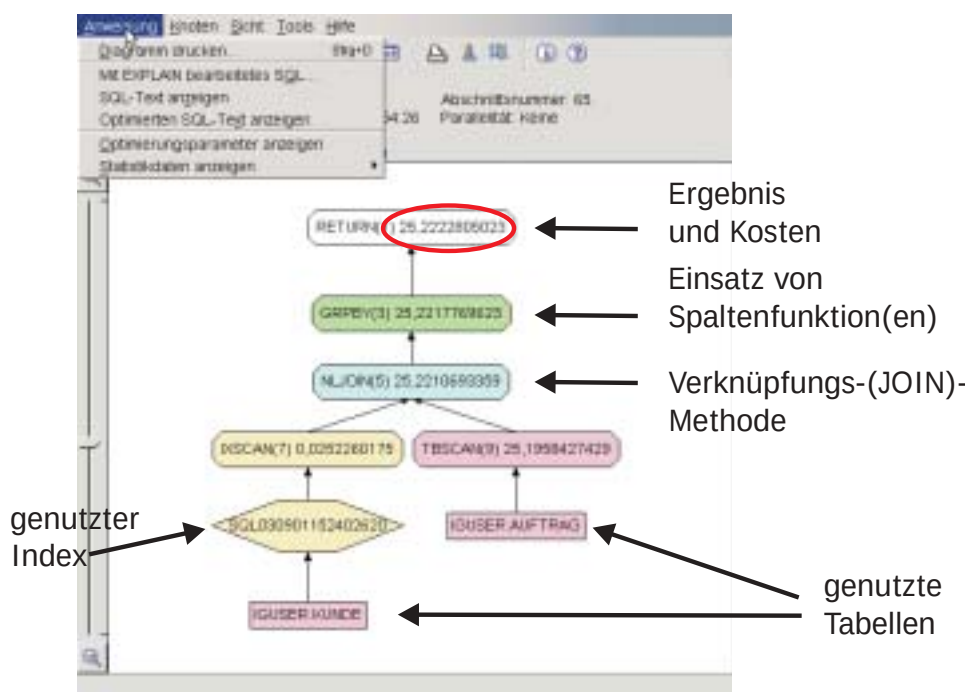


Abbildung 15.5: Visual Explain-Zugriffsplan mit Kurzbeschreibung

Die Abbildung 15.5 zeigt Ihnen den zugehörigen Zugriffsplan zu dem SQL-Befehl, der in Abbildung 15.4 abgesetzt wurde. Sie sehen, dass für den Zugriff auf die Tabelle KUNDE der Index SQL030901152402620 verwendet wird. Der Zugriff auf die zweite Tabelle, hier AUFTRAG, erfolgt über einen TBSCAN. Das steht für eine Tabellensuche oder auch Relationssuche; dabei wird die ganze Tabelle AUFTRAG sequenziell durchsucht. Die Daten der beiden Tabellen werden über ein NLJOIN (Nested Loop Join), also eine verschachtelte Schleifenverknüpfung, miteinander verbunden. Das Zwischenergebnis wird von der Spaltenfunktion, hier ein COUNT(*),

bearbeitet, und das Endergebnis daraus wird an den Abfragenden (Anwendung oder Benutzer) zurückgegeben.

Wollen Sie nun wissen, welcher SQL-Befehl hier verwendet wurde? Nein, werden Sie sagen, Ich habe doch das SQL geschrieben, wozu also die Nachfrage? Nun, Sie werden staunen, was DB2 aus einer Abfrage macht, um diese zu *optimieren*. Sie können über das Auswahlmenü dieser Anzeige OPTIMIERTEN SQL-TEXT ANZEIGEN auswählen und erhalten den SQL-Befehl, den DB2 generiert hat (siehe Abbildung 15.6) und der in Abbildung 15.5 dargestellt wird.

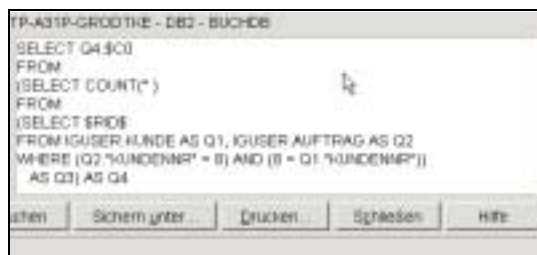


Abbildung 15.6: Visual Explain – optimierter SQL-Befehl

Wenn Sie diesen SQL-Befehl genauer anschauen, wird Ihnen auffallen, dass er nicht mehr dem SQL92-Standard entspricht und somit nicht kopiert und nicht wiederverwendet werden kann und darf. Diese Darstellung ist ein DB2-internes Format, das dazu dient, dem DB2-Optimizer ein ideales SQL zu liefern, um einen optimalen Zugriffsweg ermitteln zu können.

Die Entscheidungen für einen optimalen Zugriffsweg beruhen auf so genannten *Kosten*, auch *Timerons* genannt.

15.2.2 Was ist ein Timeron?

Timeron ist die Einheit für den „geschätzter Aufwand“. Ein Timeron entspricht nicht direkt der tatsächlich abgelaufenen Zeit, gibt jedoch eine grobe relative Schätzung der Ressourcen (Aufwand) an, die der Datenbankmanager benötigt, um einen Zugriffsplan auszuführen.

Der Aufwand, der in jedem Operator eines Zugriffsplan-Diagramms angezeigt wird, ist der kumulative Aufwand, vom Beginn der Ausführung des Zugriffsplans bis einschließlich zur Ausführung dieses bestimmten Operators.

Es werden keine Faktoren wie die Auslastung des Systems oder der Aufwand für das Zurückgeben von Datenzeilen an den Benutzer berücksichtigt.

Welche Parameter für die Optimierung des SQL-Befehls verwendet wurden, kann auch aus den Explain-Tabellen bzw. aus Visual Explain ausgelesen werden. Mit der Auswahl OPTIMIERUNGSPARAMETER ANZEIGEN erhalten Sie diese aufgelistet (siehe auch Abbildung 15.7).

Der Vorteil dieser Anzeige ist, dass Sie mit einem Blick sehen, welche Parameter zum Zeitpunkt des Explains gültig waren und wie diese aktuell belegt sind. Wei-

chen die Werte der Spalte AKTUELL stark von den Werten der Spalte MIT EXPLAIN BEARBEITET ab, kann sich bei der wiederholten Zugriffsweg-Bestimmung ein anderer Zugriffsweg und somit Zugriffsgraph ergeben.

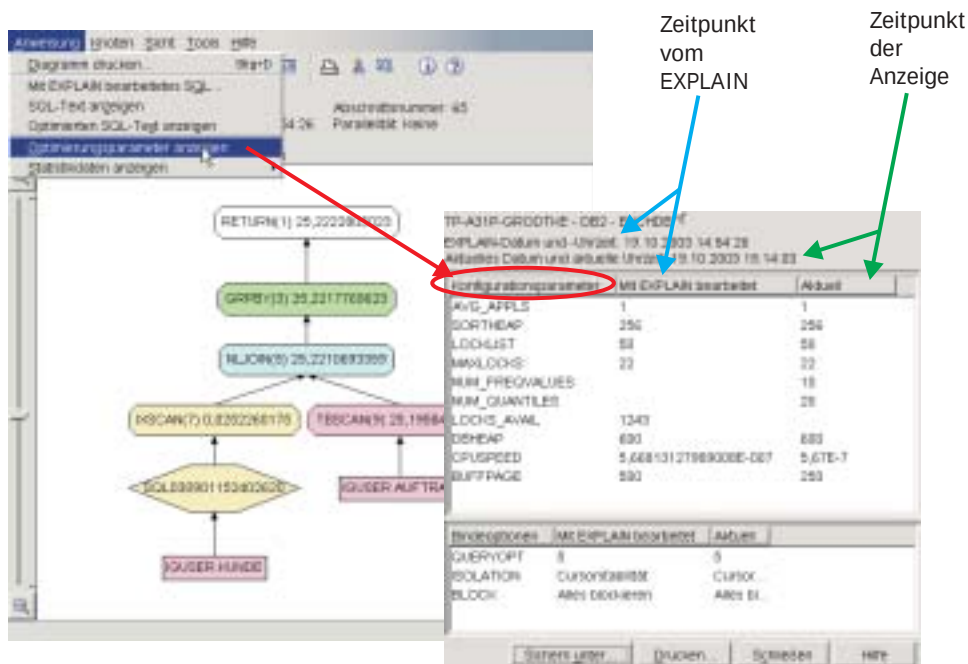


Abbildung 15.7: Visual Explain – Optimierungsparameter anzeigen

Doch nicht nur die Konfigurationswerte sind ausschlaggebend für die Berechnung des Zugriffswegs, sondern auch die Informationen über die Objekte. Beispielsweise üben die Größe der Tabellen und die Qualität des Indizes einen großen Einfluss auf die Berechnung des Zugriffswegs aus. Daher ist es ebenso wichtig, sich die Statistiken der Objekte anzusehen, die vom DB2-Optimizer verwendet wurden.

Die Statistiken erhalten Sie jeweils, wenn Sie auf das Objekt (Index oder Tabelle) gehen und die rechte Maustaste drücken. In den Statistiken der Tabelle AUFTRAG (siehe Abbildung 15.8) sehen Sie, dass im Feld STATS_TIME die Information „Statistiken nicht aktualisiert“ steht. Das heißt, dass für diese Tabelle noch nie das RUNSTATS-Tool ausgeführt wurde. Somit hat DB2 keine genauen Informationen über den Füllstand der Tabelle. Auch weitere Felder sind mit -1 belegt, was wiederum bedeutet, dass hier keine Informationen vorliegen. Schauen Sie jedoch in der Spalte MIT EXPLAIN BEARBEITET nach, so stellen Sie fest, dass DB2 hier Werte über die Größe der Datenmenge (CARD) sowie die Anzahl von genutzten Datenseiten (NPAGES) angenommen hat. In der Klammer wird dies zwar „Standardwert“ genannt, es ist aber nur ein grob geschätzter Wert anhand der belegten Datenseiten.

Wurden die Statistiken erstellt, wurde also das RUNSTATS-Tool durchgeführt, so sehen Sie, wie in den Statistiken der Tabelle KUNDE gezeigt, das aktuelle Datum und die zugehörigen Werte in den entsprechenden Zeilen.

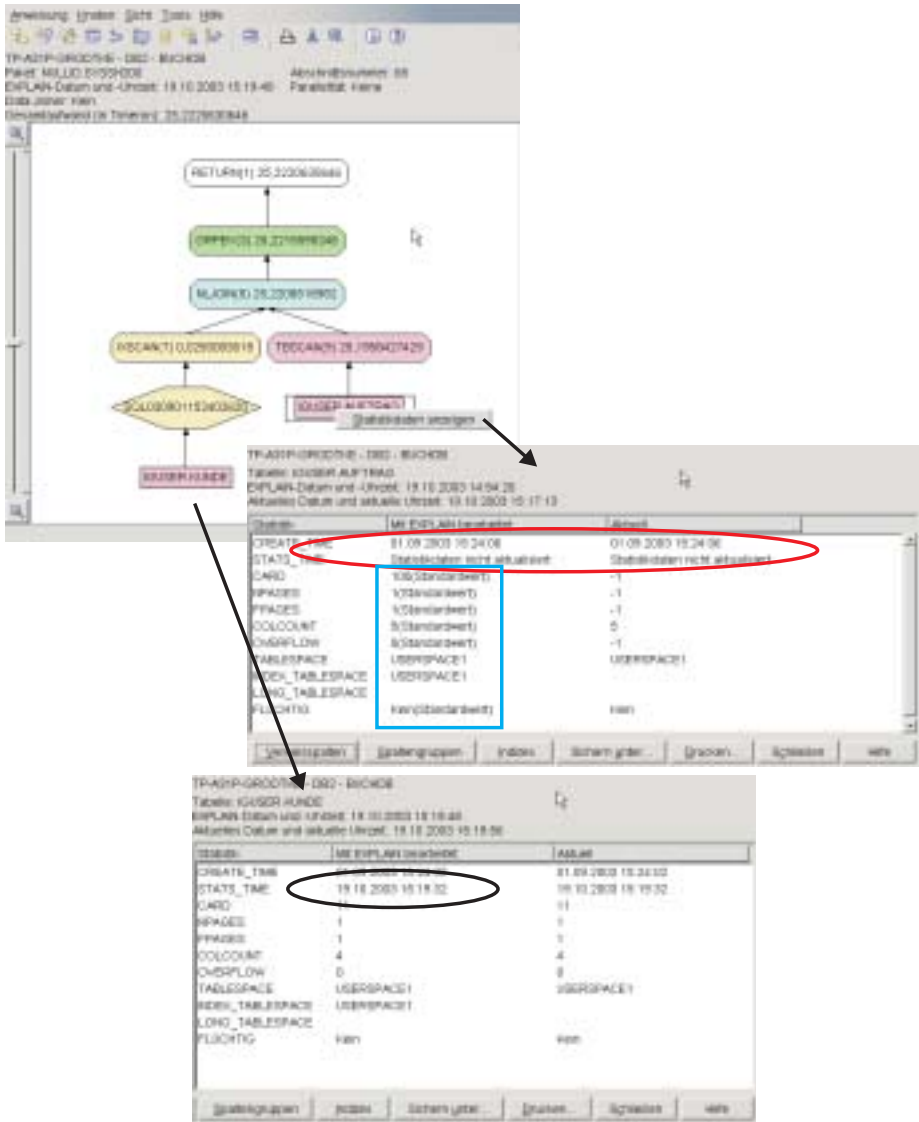


Abbildung 15.8: Visual Explain – Statistiken anzeigen

Hinweis:

Wenn Sie sich genauer mit dem Thema Optimierung bzw. mit der Analyse über das Visual Explain-Tool befassen wollen, empfehlen wir Ihnen das „Lernprogramm für Visual Explain“, das mit dem DB2 ausgeliefert wird und unter folgendem Namen und Pfad zu finden ist:

C:\Programme\IBM\SQLLIB\doc\htmlcd\de_DE\tutr\db2tv\db2tv.htm

15.3 Prozessmodell

Wozu brauchen Sie ein *Prozessmodell*? Nun, ein Prozessmodell soll Ihnen helfen, sich in der Funktionsweise und mit dem Zusammenspiel einiger DB2-Komponenten besser zurechtzufinden. Es soll Ihnen einen besseren Überblick über interne Abläufe im Bereich der Ressourcen und des Speicherverbrauchs innerhalb Ihres Betriebssystems geben. Abbildung 15.9 zeigt eine Prozesshierarchie, die aufgebaut wird, wenn Folgendes in Ihrem System abläuft:

- ▶ Das Exemplar wird mit dem Befehl `db2start` gestartet.
- ▶ Die Datenbank wird aktiviert. Dies kann mit dem Befehl `activate database <dbname>` bzw. mit dem ersten `connect to <dbname>` geschehen.
- ▶ Der erste `connect to <dbname>` erfolgt zur Datenbank, dieser CONNECT kann „lokal“, also direkt am Server stattfinden oder über eine entfernte Verbindung von einem Client aus geschehen.

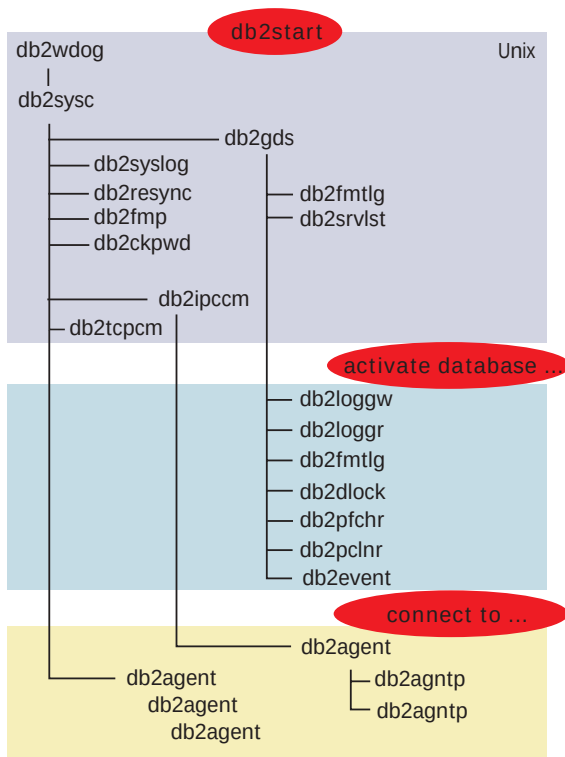


Abbildung 15.9: Prozessmodell – Unix

Die Unix-Umgebung arbeitet mit einer Architektur, die auf Systemprozessen basiert. Die Windows-Umgebung dagegen arbeitet mit einer Architektur, die zur Leistungsmaximierung Threads verwendet (Threads sind quasi parallel ablaufende Steuerflüsse innerhalb eines Programms). In der DB2-Literatur werden Sie auch oft den Begriff EDU (Engine Despatching Unit, zuteilbare Einheiten der Steuerkomponente) finden. Dieser Begriff wird oft verwendet, um die Bezeichnung „Prozesse“ im Unix-Umfeld und „Threads“ unter Windows zu umgehen bzw. allgemein gültig zu halten.

Folgende Prozesse sind in Abbildung 15.9 dargestellt:

► db2wdog

ist der Wachprozess (watchdog), der auf Unix-Systemen die abnormalen Beendigungen behandelt. Dies ist notwendig, da unter Unix ein Prozess nur seinen Vorläuferprozess prüfen kann. Jedes Mal, wenn DB2 einen neuen Prozess startet, informiert der db2gds den db2wdog über diesen neuen Zustand. Wird nun ein abhängiger Prozess mit `[Strg]+[C]` oder einem anderen abnormalen Signal beendet, so wird dieses Signal zu db2wdog gesendet, und dieser schickt dann das Signal weiter an alle von diesem Prozess abhängigen Prozesse, um diese auch zu terminieren.

► db2sysc

der DB2-Systemkontrollprozess, ohne den gar nichts geht. Alle anderen Prozesse sind von diesem Kontrollprozess des Exemplars abhängig.

► db2gds

Das ist der globale „Daemon“-Prozess, der alle weiteren Datenbankprozesse startet.

► db2syslog

Der „System Logger“-Prozess fügt Informationen über aufgetretene Fehler in die Fehlerprotokolldatei des Betriebssystems ein. Unter Unix muss dies separat über die `syslog.conf` konfiguriert werden. Unter Windows schreibt DB2 automatisch in die Event Log-Datei von Windows.

► db2resync

Dies ist der Prozess für die Resynchronisation von Anwendungen, die eine zweistufige Festschreibung (two-phase commit) verwendet.

► db2fmp

Der db2fmp-Prozess ist immer ein separater Prozess, der für die Ausführung von abgeschirmten gespeicherten Prozeduren und benutzerdefinierten Funktionen zuständig ist.

► db2ckpwd

wird zur Authentifizierungsüberprüfung genutzt, also zur Überprüfung der Benutzer-ID und des Kennworts.

► db2ipccm

ist ein Horchprozess (listener) für die interne Kommunikation, wenn Sie sich also auf dem gleichen Rechner zu einer „lokalen“ Datenbank verbinden.

- ▶ `db2tcpcm` (nur wenn `DB2COMM=TCPIP` und `SVCENAME=portnummer/name`)

ist ein Horchprozess für das TCP/IP-Netzwerkprotokoll. Der Prozess horcht auf dem Port, der in der DBM-CFG unter `SVCENAME=<portnummer>` eingegeben wurde. Der Prozess wird aber nur gestartet, wenn auch die Registry-Variable `DB2COMM=TCPIP` gesetzt wurde.

Folgende Prozesse werden beim activate database gestartet:

- ▶ `db2loggw`

Der „database log writer“, d.h. der DB2-Protokoll-Schreibprozess, schreibt die Protokollseiten aus dem Protokoll-Puffer (`logbufsz`) auf die Protokolldateien aus.

- ▶ `db2loggr`

Der „database log reader“, d.h. der DB2-Protokoll-Leseprozess, liest die Protokollsätze aus den Protokolldateien aus, wenn eine Wiederherstellung eines konsistenten Zustands (rollback oder crash recovery) oder eine aktualisierende Wiederherstellung (rollforward) notwendig ist.

- ▶ `db2fmtlg`

Protokolldateien werden im Protokollpfad im Voraus formatiert, wenn `LOGRETAIN=ON` und `USEREXT=OFF` gesetzt sind.

- ▶ `db2dlock`

erkennt gegenseitige Sperren, indem dieser Prozess die Lock-Listen (`locklist`) durchsucht und bei Erkennung einer gegenseitigen Sperre einen der verursachenden Prozesse/Anwendungen beendet. In einer partitionierten Datenbank gibt es noch einen weiteren Prozess, den `db2glock`, der über die Partitionen feststellt, ob eine gegenseitige Sperrung vorhanden ist. Dieser Prozess läuft aber nur auf der Partition, auf der der DB2-Katalog liegt.

- ▶ `db2pfchr`

die Vorablesefunktion für Pufferpools. Dieser Prozess liest Daten- und Index-Seiten im Voraus, also asynchron, von Platte und kopiert diese in den zugehörigen Pufferpool. Die Anzahl der verfügbaren Prozesse wird mit dem DB-Parameter `NUM_IOSERVERS` konfiguriert. Der Standard ist bei 3; somit werden auch immer drei `db2pfchr`-Prozesse (pro Datenbank) im System gestartet.

- ▶ `db2pclnr`

die Seitenlöschfunktion für Pufferpools. Dieser Prozess schreibt veränderte Daten und Indexseiten (dirty pages) aus dem Pufferpool in die Tabellenbereiche zurück, und zwar asynchron. Die Anzahl der Prozesse, die gestartet werden, hängt vom DB-Konfigurationsparameter `NUM_IOCLEANERS` ab.

- ▶ `db2event`

Dieser Prozess ist für jeden Ereignismonitor (Event-Monitor), der gestartet wird, zu sehen. Er ist für das Sammeln der Ereignismonitor-Daten und das Heraus-schreiben der Daten in die Ausgabedatei, Pipe oder Tabelle zuständig.

Die folgenden Prozesse starten, wenn Sie sich mit der Datenbank über den Befehl CONNECT verbinden. Kommen Sie als „lokaler“ Client, so erfolgt die Kommunikation über einen so genannten *gemeinsamen Speicherbereich* (shared memory segment). Der Prozess db2ipccm (ipccm, Internal Process Communication) ist der „Horch“-Prozess für lokale Verbindungen. Kommen Sie aber als „entfernter“ Client über das Netzwerkprotokoll TCP/IP, so steht hier ein weiterer „Horch“-Prozess (db2tcpcm) in Wartestellung (nur wenn DB2COMM=TCPIP gesetzt wurde). Kommt eine Anforderung zur Verbindung mit einer Datenbank, so wird von dem DB2-System-Controller (db2sysc) ein Prozess speziell für diese Anwendung angelegt (db2agent).

Folgende Prozesse können bei einer Verbindung zur Datenbank noch gesehen werden:

► db2agent

Dieser Prozess ist der Koordinatorprozess. Alle Anforderungen zur Datenbank gehen über diesen Prozess. Wenn Sie mit einer partitionierten Datenbank arbeiten, so wird dieser db2agent-Prozess weitere Sub-Prozesse starten, wie z.B. db2agntp.

► db2agnsc

Diesen Prozess können Sie sehen, wenn ein Rollforward oder eine Crash Recovery im parallelen Modus durchgeführt wird.

► db2agnta

Das ist ein „idle subagent“, ein „nicht zugeordneter Unterprozess“. Wenn Subprozesse nicht mehr vom Koordinierungsagenten benötigt werden, so gehen diese in einen Warte-Pool, um bei Bedarf wiederverwendet zu werden. Diese sind dann zu sehen, wenn der DBM-Konfigurationsparameter INTRA_PARALLEL=YES ist.

15.3.1 Windows-Prozesse

Im Windows-System haben wir keine Prozesse, sondern Threads, daher gibt es auch kein Prozessmodell wie bei Unix-Systemen.

► db2dasrrm.exe

DB2-Administrationsserver – unterstützt entfernte und lokale Anforderungen von der DB2-Steuerzentrale.

► dasstmt.exe

DB2-Admin-Server-Tools – Informationen für und aus der Tools-Datenbank werden mit diesem Prozess verwaltet.

► db2fmp.exe

Der db2fmp-Prozess ist für die Ausführung von abgeschirmten gespeicherten Prozeduren und benutzerdefinierten Funktionen zuständig.

► db2jds.exe

JDBC-Anwendungsanforderungen werden hierüber ausgeführt.

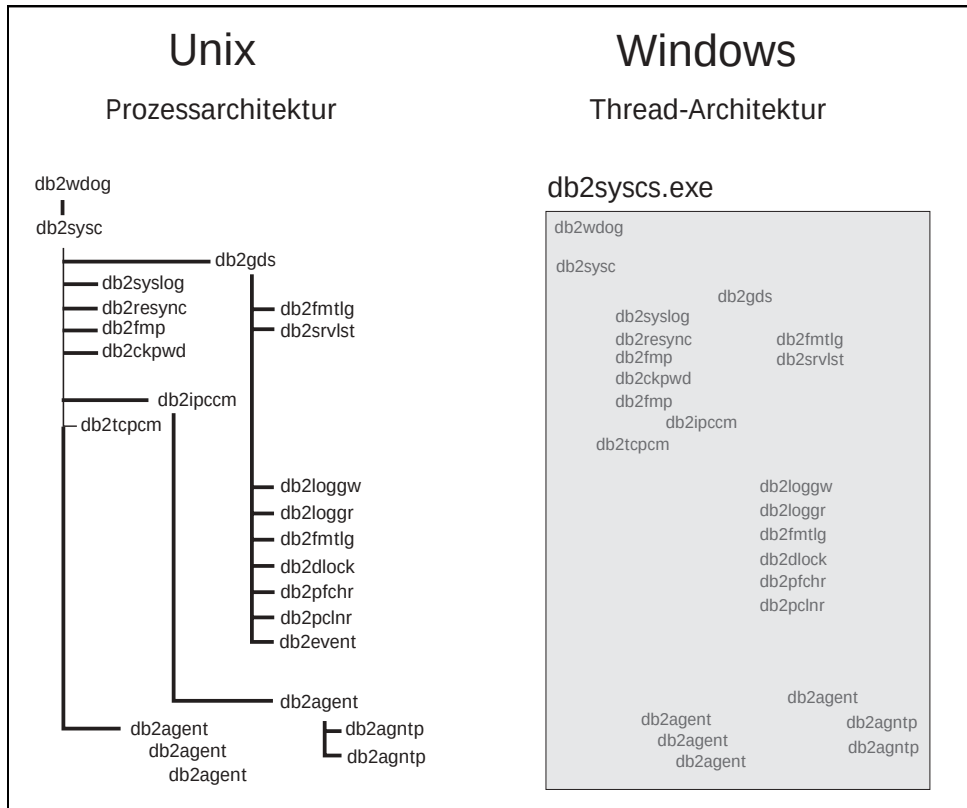


Abbildung 15.10: Windows-Prozessmodell

► `db2licd.exe`

Die Lizenzen werden beim Start des Exemplars überprüft und bei Ungereimtheiten in die `db2diag.log`-Datei geschrieben.

► `db2sec.exe`

wird zur Authentifizierungsüberprüfung genutzt, also zur Überprüfung der Benutzer-ID und des Kennworts.

► `db2syscs.exe` (siehe Abbildung 15.10)

Dies ist der Hauptprozess und das eigentliche DBMS. Alle weiteren Prozesse werden als Threads innerhalb dieses Hauptprozesses ausgeführt.

15.3.2 Prozesse beim Sichern einer Datenbank

Beim Sichern einer Datenbank sind andere Prozesse zu sehen als z.B. beim Laden von Daten.

Um den Ablauf einer Sicherung auf Prozess-Ebene zu verstehen, haben wir hier ein Beispiel aufgebaut.

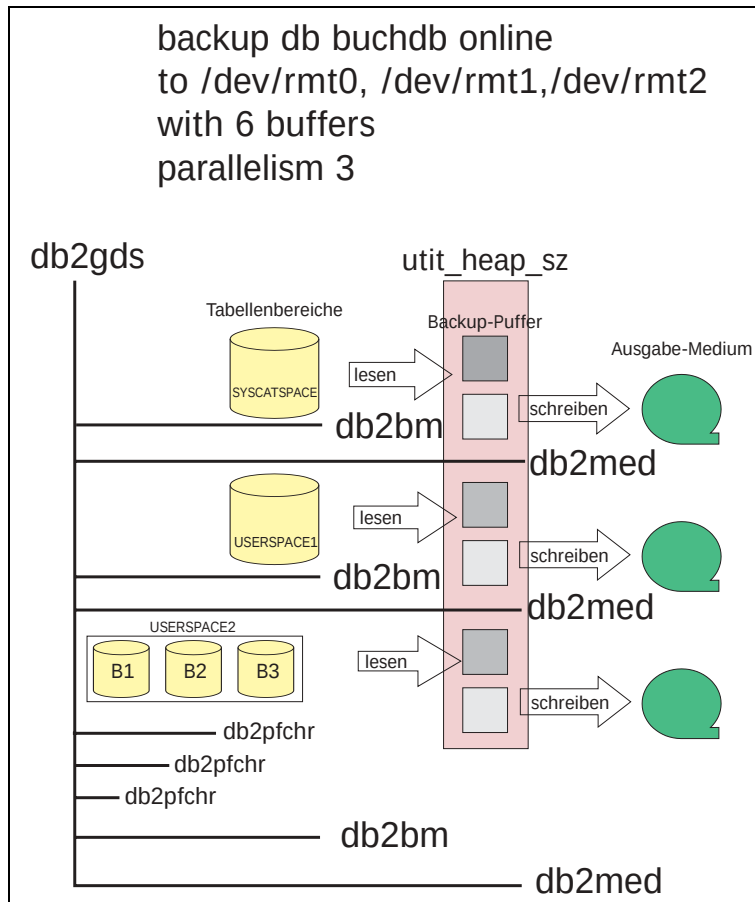


Abbildung 15.11: Prozesse beim Backup auf drei Ausgabe-Einheiten

Wird ein Backup mit den folgenden Optionen durchgeführt

to /dev/rmt0, /dev/rmt1,/dev/rmt2

so wird die Ausgabe der Sicherung auf drei Ausgabe-Einheiten verteilt. Der Grund dafür kann einmal die Performance oder auch mangelnder Speicherplatz auf dem Ausgabemedium sein.

Mit der Anzahl der Puffer, hier 6,

with 6 buffers

sagen Sie dem DB2, dass Sie sechs Backup-Puffer haben möchten. Die Größe des Sicherungspuffers ist standardmäßig 1.024 x 4 4-Kbyte-Seiten. Damit haben Sie je Tabellenbereich (wir haben hier drei Tabellenbereiche) zwei Backup-Puffer – einen zum Schreiben und einen zum Auslesen, intern wird immer umgeschaltet. Auch die Anzahl der Schreibprozesse ist von der Anzahl der Ausgabe-Einheiten abhängig, da je Ausgabebeeinheit ein DB2 Media-Controller (db2med) gestartet wird.

Die Option

parallelism 3

gibt an, wie viele „Puffer-Manipulatoren“ – db2bm – gestartet werden sollen, um Daten gleichzeitig aus den Tabellenbereichen zu lesen.

Hier müssen Sie unbedingt darauf achten, dass Sie, wenn Sie einen Tabellenbereich über mehrere Behälter (Container) verteilt haben, auch die Anzahl der „Vorauslesefunktion für Pufferpool“ (Prefetcher) ausreichend groß gesetzt haben (NUM_IOSERVERS), um auch bei dieser Sicherung die Vorteile der Vorauslesefunktion zu nutzen.

15.4 Speichermodell

Die Übersicht in Abbildung 15.12 zeigt vereinfacht die Speichergrundstruktur von DB2.

Diese ist aufgeteilt in:

- ▶ den *globalen und gemeinsam benutzten Speicher des Datenbankmanagers* (DBM shared memory), der reserviert wird, wenn das Exemplar, also das DBMS, mit dem Befehl db2start aktiviert wird. Der maximale Speicherverbrauch kann über die DBM-Konfigurationsvariable INSTANCE_MEMORY limitiert werden. Der Standardwert steht auf AUTOMATIC, so dass DB2 selbst die Größe des Speichers bestimmen kann. Dieser Wert ist auch die empfohlene Einstellung und sollte nicht verändert werden.
- ▶ den *globalen Datenbankspeicher* (DB global memory), der für jede Datenbank angelegt wird, wenn diese aktiviert wird, also mit dem ersten CONNECT oder über den Befehl ACTIVATE DATABASE. Auch hier kann der Speicherverbrauch limitiert werden. Der Datenbankkonfigurationsparameter DATABASE_MEMORY steht auf dem Standardwert AUTOMATIC. Dieser Wert sollte auch so bleiben, damit die Datenbank die benötigten Speicher, wie z.B. für den Pufferpool oder den Katalog-Zwischenspeicher, dynamisch reservieren und aufbauen kann.
- ▶ den *globalen Anwendungsspeicher* (application global memory). Dieser „globale“ Anwendungsspeicher wird nur benötigt, wenn Sie:
 - mit einer partitionierten Datenbank arbeiten,
 - mit dem Verbindungskonzentrator (connection concentrator) arbeiten und
 - den Konfigurationsparameter für Prozessorparallelität intra_parallel=YES und CURRENT DEGREE größer als 1 gesetzt haben.

Dann werden hier prozessübergreifende Informationen festgehalten, wie z.B. über „angelegte temporäre Tabellen“. In einer partitionierten Umgebung werden hier die SQL-Befehle auch für die Subprozesse gehalten.

Sie haben die Möglichkeit, mit dem Parameter appgroup_mem_sz den maximalen Speicher für alle Anwendungen zu limitieren, und mit dem app_ctl_heap_sz-Parameter sagen Sie, wie viel Speicher jede Anwendung sich aus diesem Speicher-Pool nehmen darf.

- Der *private Anwendungsspeicher* (agent private memory) wird jeder Anwendung zugeordnet, wenn diese sich zur Datenbank verbindet. In diesem Speicherbereich sind *nur* Informationen für diese eine Anwendung vorhanden, z.B. für die privaten Sortierbereiche sowie für die Anwendungszwischenspeicher, die wiederum Informationen, z.B. über die SQLCA, SQLDA oder den Cursor-Status, beinhalten.

Der benötigte Speicher, den alle Agenten gemeinsam nutzen, kann nicht direkt eingeschränkt werden. Dies geht nur über den Umweg der Prozesslimitierung. Dies können Sie erreichen, wenn Sie die DBM-Parameter MAXAGENTS oder MAXC-AGENTS bei der Parallelverarbeitung einschränken.

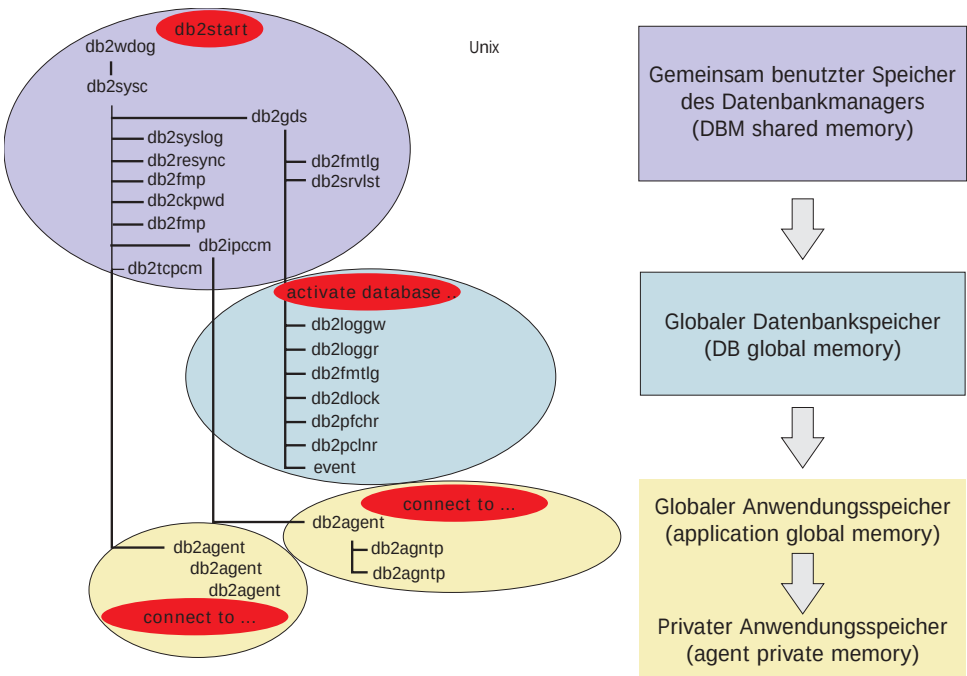


Abbildung 15.12: Speichermodell – Überblick

15.5 E/A-Performance und Pufferpool

Die Größe und die Nutzung des Pufferpools sind die wichtigsten Faktoren für die Datenbank-Performance. Die Funktionsweise des Pufferpools haben wir schon in den vorangegangenen Kapiteln ausführlich erklärt. Hier zeigen wir Ihnen nur noch die weiteren Optimierungsmöglichkeiten auf.

Beim Lesen von großen Datenmengen ist es wichtig, dass diese schnell in den Pufferpool gelesen werden können, damit die verarbeitende Anwendung die Daten immer im Speicher findet und nie warten muss, bis das Datenbanksystem fehlende Daten vom zugehörigen Tabellenbereich (Platte) holen muss. Damit diese Vorgänge optimal innerhalb der Datenbank ablaufen können, müssen einige Dinge bei der Konfiguration beachtet werden. Eine Zusammenfassung zeigen wir in Abbildung 15.13.

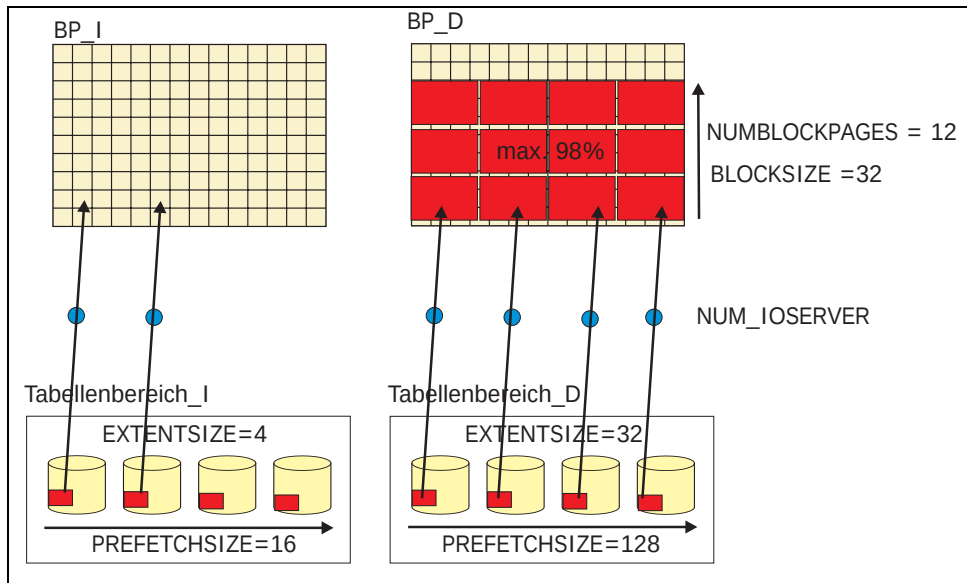


Abbildung 15.13: Parallel-E/A und Blocked Pufferpool

Erkennt DB2, dass es große Datenblöcke sequenziell lesen kann (sequential detection), dann wird die „Vorablesfunktion“ (prefetching) aktiviert, um die angeforderten Daten schneller und effektiver von den Tabellenbereichen zu lesen. Die angeforderten Daten sind in diesem Fall nicht die Datenseiten, die die Anwendung direkt benötigt, sondern die Datenseiten, die mit größter Wahrscheinlichkeit die nächsten Datenseiten sein werden, die von der Anwendung als Nächstes benötigt werden.

Folgende Optionen sollten für die Optimierung berücksichtigt werden:

- Beim Anlegen der Tabellenbereiche sollte die PREFETCHSIZE nach folgender Formel angelegt werden:

Anzahl der Tabellenbereichsbehälter * EXTENTSIZE = PREFETCHSIZE (maximal 512)

Jetzt ist DB2 in der Lage, die Vorablesfunktion zu beauftragen, Daten *blockweise* (extent-weise) zu holen. Damit dies parallel erfolgen kann, müssen die E/A-Prozesse in ausreichender Anzahl vorhanden sein.

- NUM_IOSERVER ist ein DB-Konfigurationsparameter, mit dem die Anzahl der Lese-Prozesse zur Verfügung gestellt wird. Die Anzahl dieser Prozesse sollte wie folgt gewählt werden:

Anzahl der NUM_IOSERVERS = Anzahl der Tabellenbereichsbehälter (Container) der Tabellenbereiche (tablespaces), die gleichzeitig durchsucht werden.

In unserem Beispiel:

2 Tabellenbereiche * 4 Behälter = 8 NUM_IOSERVERS

- Nun kann der Pufferpool auch noch für diese BLOCK-Daten, die ja nun EXTENT-weise in den Pufferpool gelesen werden, angepasst bzw. optimiert werden.

In früheren DB2-Versionen gab es diese Option nicht. Sie ist auch nur geeignet, um die Vorablesfunktion von DB2 zu optimieren.

Mit der Option NUMBLOCKPAGES und BLOCKSIZE beim CREATE BUFFERPOOL bzw. beim ALTER BUFFERPOOL können Sie Folgendes bewirken (siehe Abbildung 15.13):

NUMBLOCKPAGES gibt die Anzahl der Blöcke an, die im Pufferpool angelegt werden. Die BLOCKSIZE gibt die Anzahl der Datenseiten an, die ein Block enthält. Die Summe der NUMBLOCKPAGES * BLOCKSIZE darf nicht größer sein als 98% des Pufferpools (SIZE #Seiten).

Tipp:

Die BLOCKSIZE sollte immer gleich oder kleiner sein als die EXTENTSIZE der zugehörigen Tabellenbereiche, also *nie* größer!

Vorsicht:

NUMBLOCKPAGES sollte beim Anlegen oder Ändern des Pufferpools nur eingesetzt werden, wenn sehr hohe Vorauslesevorgänge durchgeführt werden. Der blockbasierte Teil des Pufferpools kann *nur* für diese Vorauslese-Funktion verwendet werden und steht somit *nicht* mehr für „normale“ (seitenbasierte) Leseoperationen zur Verfügung!

Die Größe des blockbasierten Anteils des Pufferpools sollte den tatsächlichen Vorausleseanforderungen optimal angepasst werden, um keinen Speicherplatz zu verschenken.

15.6 SQL-Performance

Die Analyse von SQL-Befehlen bzw. die daraus resultierenden Umstellungen und Aktionen können einen enormen Einfluss auf die Abfrage-Performance Ihrer Datenbank haben. Alle diese Möglichkeiten zu besprechen würde ein ganzes Buch füllen; deswegen beschränken wir uns hier nur auf einige allgemeine Tipps, die jedoch bei genauester Beachtung schon die größten Optimierungen bringen. Ob diese Vorschläge auch die gewünschten Effekte erzielen, testen Sie am besten mit dem Tool Visual Explain, db2exfmt oder db2explain.

Folgende Punkte sollten Sie immer beim Schreiben eines SQL-Befehls beachten:

- Wählen Sie nur Spalten aus, die *wirklich* benötigt werden (*kein* select *).
- Verwenden Sie immer eine Einschränkung (WHERE SpaltenName = wert).
- Verwenden Sie immer die niedrigste sinnvolle Isolationsstufe (UR| CS| RS| RR).
- Mit der Option WITH kann dies für jeden SQL-Befehl angegeben werden.

- ▶ Definieren Sie den Cursor immer *eindeutig* (FOR READ ONLY| FOR UPDATE OF).
- ▶ Vermeiden Sie alles, was zur einer Sortieroperation führt!
- ▶ Sortierungen können auf der Festplatte durchgeführt werden, wenn der Sortierbereich (SORTHEAP) nicht groß genug angelegt worden ist.
- ▶ Wenn in der Summe alle Sortierbereiche den Grenzwert des DBM-Konfigurationsparameters SHEAPTHRES überschreiten, kann dies auch dazu führen, dass DB2 auf der Festplatte (Temp-Tabellenbereiche) sortiert.
- ▶ Setzen Sie FETCH FIRST n ROWS ONLY ein.
- ▶ Diese Option sollten Sie sich generell angewöhnen, nicht nur in Programmen, sondern auch von der DB2-Befehlszeile, da Sie mit dieser Option die mögliche Ausgabe der Daten auf ein Minimum einschränken können, z.B.:

```
SELECT  KundenNr, KundenName, KundenRabatt
FROM    Kunde
WHERE   KundenRabatt > 15
FETCH FIRST 20 ROWS ONLY;
```

15.7 Tipps

Um alle Datenbankmanager und Datenbankkonfigurationsparameter optimal zu setzen, gibt es den Konfigurationsadvisor (der schon in Kapitel 5 behandelt wurde). Hier noch ein Hinweis zum Zielspeicher (Standard ist hier immer 80%): Achten Sie darauf, was noch *außer* DB2 auf dem Rechner läuft, den Sie konfigurieren.

Stellen Sie sich noch folgende Fragen:

- ▶ Welcher Speicher steht mir tatsächlich zur Verfügung?
- ▶ Welche Datenbanken laufen zur gleichen Zeit auf diesem Rechner, und wie sind diese konfiguriert (speziell die Pufferpools)?
- ▶ Wie viele Benutzer arbeiten gleichzeitig mit dem System, und welchen Speicherverbrauch haben diese Benutzer?

A Migration auf DB2 V8

Bei der Migration von DB2 sind zusätzlich zu den Installationsschritten noch andere Gesichtspunkte zu beachten, da ja bestehende Daten bzw. Datenbanken übernommen werden sollen.

In DB2 ist es generell möglich, von den letzten zwei Versionen direkt auf die neueste Version – also ohne Zwischenschritt – zu migrieren. Das bedeutet für DB2 UDB V8, dass Sie direkt von folgenden Versionen aus migrieren können:

- ▶ DB2 UDB V7
- ▶ DB2 UDB V6
- ▶ Data Joiner V2.1.1

Genauso können Sie von Client-Installationen dieser Versionen auf einen DB2 V8 Server zugreifen. Ein Client mit der Version 5 bekommt eine Fehlermeldung, dass seine Installationsversion für den Zugriff auf DB2 UDB V8 zu alt ist.

Umgekehrt ist von DB2 UDB V8-Clients aus ein Zugriff auf DB2 UDB V7 möglich – allerdings (in dieser Version) mit starken Einschränkungen (später mehr dazu).

Wir wollen mit Ihnen nun im Folgenden einen Migrationsablauf durchgehen, in dem wir Ihnen die wichtigen Punkte und Überlegungen in einer empfohlenen Abfolge präsentieren.

A.1 Vorbereitung der Migration

Einige Schritte sollten Sie vor der Migration bedenken und durchführen, um dann einen möglichst reibungslosen Ablauf der Migration gewährleisten zu können. Für die Migration müssen Sie unbedingt ein Wartungsfenster reservieren, da Sie über längere Zeit einen exklusiven Zugriff auf die Exemplare benötigen. Wie groß dieses Fenster ist bzw. wie lange die Migration in Anspruch nimmt, hängt maßgeblich von der Anzahl Ihrer Datenbanken und der darin gespeicherten Datenmenge ab. Eine Testmigration auf einer Test-Maschine ist anzuraten, um den ungefähren Aufwand abschätzen zu können.

A.1.1 Kompatibilität zwischen den Versionen

Durch die Umstellung vom privaten Protokoll auf den DRDA-Standard bei der Client/Server-Kommunikation mit Version 8 sind größere Inkompatibilitäten zu der vorhergehenden Version zu erwarten, als Sie bisher vielleicht gewöhnt waren. Davon ist besonders der Zugriff von DB2 V8-Clients auf ältere Server (also z.B. DB2 V7) betroffen. Einige der Einschränkungen sind:

- ▶ Bestimmte Datentypen werden nicht unterstützt (z.B. LOBs).
- ▶ Die Authentifizierungseinstellung `SERVER_ENCRYPT` wird nicht unterstützt.

- ▶ Die Verbindung kann *nur* über TCP/IP hergestellt werden.
- ▶ Ein ATTACH zum DB2 V7-Exemplar ist weder von der Befehlszeile noch implizit über das GUI möglich.
- ▶ Die meisten Tools der grafischen Oberfläche (z.B. Steuerzentrale, Taskzentrale, Journal etc.) funktionieren nicht mit älteren Versionen.
- ▶ Einige grafische Tools funktionieren nur eingeschränkt, z.B. der Konfigurationsassistent oder die Befehlszentrale.
- ▶ Etliche weitere Programme und Funktionen funktionieren nicht.

Diese Liste der Einschränkungen ist schon sehr umfangreich, aber Sie haben die Möglichkeit, diese Einschränkungen zu umgehen, indem Sie zuerst die Server migrieren und dann die Clients.

Wichtig:

Migrieren Sie zuerst Ihre Server (möglichst alle) auf DB2 UDB V8, bevor Sie die Clients migrieren! Dadurch lassen sich die Inkompatibilitäten größtenteils vermeiden.

Kommen wir also zu den vorbereitenden Schritten:

1. Stellen Sie ausreichend Platz zur Verfügung.

Das betrifft einerseits den SYSCATSPACE, in dem neue Systemkatalog-Tabellen hinzukommen und andere Änderungen vorgenommen werden. Andererseits sollten Sie auf dem Server genug temporären Speicher zur Verfügung haben. Das heißt, das /tmp-Dateisystem sollte noch gut Platz haben.

2. Um alle Informationen angezeigt zu bekommen, stellen Sie den Diagnose-Level auf die höchste Stufe – für den Fall der Fälle:

```
update dbm cfg using diaglevel 4 noftiylevel 4
```

3. Erstellen Sie eine *Offline*-Sicherung Ihrer Datenbanken.

Gesicherte Datenbanken älterer DB2-Versionen können in DB2 V8 wiederhergestellt werden, aber eine aktualisierende Wiederherstellung ist *nicht* möglich, da DB2 V8 keine Protokolldateien der V7 verarbeiten kann. Deshalb müssen Sie eine *Offline*-Sicherung anlegen.

4. Überprüfen Sie Ihre Datenbanken daraufhin, ob sie migriert werden können und alle Stati in Ordnung sind.

Hinweis:

Dafür steht Ihnen der Befehl db2ckmig zur Verfügung, den Sie im utilities-Verzeichnis auf der Produkt-CD-ROM der neuen Version – also von DB2 UDB V8 – finden.

```
db2ckmig <datenbankname> -l <migrations-protokolldatei>
```

Dieses Tool sollte unbedingt genutzt werden! Es wird Ihnen anzeigen, ob die Datenbank in einem „migrierbaren“ Zustand ist. Unter anderem werden die Datenbank- und Tabellenbereichsstatus überprüft. Die angegebene Protokoll-datei enthält dann die Information, dass die Migration möglich ist, oder eventuell auch eine Warnung, weil z.B. Sichten auf Systemtabellen von Benutzern definiert wurden, die sich durch die Migration ändern. Bei Fehlerhinweisen sollten Sie diese unbedingt beheben, bevor Sie fortfahren.

Sie sehen: Bereits hier gibt es einige Punkte, die nur im Wartungsfenster durchgeführt werden können.

A.2 Durchführung der Migration

Haben Sie die vorbereitenden Aktionen durchgeführt, liegt die Software bereit und sind Sie mit den entsprechenden Rechten am Server ausgestattet, kann die Migration beginnen.

DB2 UDB V8 hat teilweise geänderte Installationspfade, also geänderte Ziel-Pfade für die Installationsdateien. DB2 UDB V8 wird in folgenden Pfaden installiert:

- ▶ Unix-Systeme: /opt/IBM/db2/V8.1/
- ▶ AIX: /usr/opt/db2_08_01/
- ▶ Windows: \Programme\IBM\sql1ib\

Um den neuen Code zu installieren, sind folgende Schritte abzuarbeiten:

1. Stoppen Sie Ihre DB2-Exemplare und das DAS-Exemplar.
2. Starten Sie die Installation von DB2 (durch db2setup), wie in Kapitel 3 beschrieben.
 - Sie sollten die Toolskatalog-Datenbank automatisch anlegen lassen, wenn das Setup Ihnen die Auswahl gibt, um bestehende Zeitpläne und Scripts übernehmen zu können.
 - Die Benachrichtigungsfunktion mit SMTP-Server und Kontakten können Sie auch zu einem späteren Zeitpunkt anpassen bzw. eingeben.
 - Beachten Sie, dass die Dokumentation in DB2 V8 separat zu installieren ist.

Hat die Installation funktioniert, sind bei Windows-Systemen bereits alle Exemplare auf die neue Version umgestellt worden. Bei Unix-Systemen ist dazu ein manueller Eingriff notwendig, da bei diesen mehrere Versionen parallel installiert sein können und Sie ja auch Exemplare in verschiedenen Versionen parallel ausführen. So können Sie hier entscheiden, welches Exemplar auf die neue Version umgestellt werden soll.

Folgende Schritte sind auf Unix-Systemen für die Exemplar-Migration durchzuführen:

1. Migrieren eines Exemplars durch den Befehl db2imigr

Den Befehl db2imigr finden Sie im instance-Unterverzeichnis des DB2-Installationspfads, also z.B. in /opt/IBM/db2/V8.1/instance/ für Linux-Systeme. Diesen Befehl müssen Sie mit root-Berechtigung ausführen. Der typische Befehl für die Exemplar-Migration sieht so aus:

```
db2imigr -u <abgeschirmte benutzer-id> <exemplarname>
```

2. Migrieren des DAS (DB2 administration servers)

Die Migration des DAS ermöglicht die Übernahme von Scripts und Zeitplänen etc. aus dem DAS-Exemplar der V7. Da in DB2 V8 der DAS ein Prozess ist und kein Exemplar mehr, muss der DB2-Toolskatalog angelegt sein, um diese Migration erfolgreich umsetzen zu können. Falls der Toolskatalog während der Installation nicht automatisch angelegt wurde oder falls dabei ein Fehler aufgetreten ist, können Sie die Installation mit folgendem Befehl nachholen:

```
create tools catalog toolsdb create new database toolsdb
```

Die Migration des DAS erfolgt dann durch folgenden Schritt:

```
dasmigr <alter_DAS_name> <V8_DAS_name>
```

Auch für diesen Schritt müssen Sie mit root-Berechtigung arbeiten.

Damit ist die Migration der Exemplare abgeschlossen. Der nächste Schritt ist nun die Migration der Datenbanken in dem bzw. den migrierten Exemplaren. Dieser Schritt gilt auch wieder für die Windows-Umgebungen. Ist eine Datenbank noch nicht auf den aktuellen Versionsstand gebracht, schlägt jeder Verbindungsversuch fehl. Die Fehlermeldung weist aber darauf hin, dass auf den aktuellen Release-Stand umgestellt werden muss.

Die Datenbank-Migration wird durch folgenden Befehl angestoßen:

```
migrate db <datenbank-aliasname> [user <benutzer-id> using <kennwort>]
```

Dabei werden neue Katalogtabellen und benutzerdefinierte Funktionen angelegt und die Steuerdateien auf die V8-Strukturen umgestellt. Die Datenbankinhalte der Benutzer werden dabei *nicht* angefasst!

Hinweis:

Unter Umständen kann es vorkommen, dass bei dieser Umstellung die Fehlermeldung SQL0101N ausgegeben wird, mit dem Hinweis, dass der SQLSTMT zu komplex ist. Durch eine einfache Erhöhung des Datenbankparameters STMHEAP können Sie dieses Problem lösen.

A.3 Erste Schritte nach der Migration

Ist die Migration der Datenbank abgeschlossen, können Sie wieder eine Verbindung zu ihr aufbauen. Trotzdem sollten Sie in Ihrem Wartungsfenster noch weitere wichtige Schritte durchführen, um einen reibungslosen Ablauf zu gewährleisten.

Folgende Punkte sollten Sie durchführen:

- ▶ *Konvertierung der Indizes* auf die neue Typ-2-Struktur mittels `REORG INDEX ... CONVERT`. Während die Indizes für die Systemtabellen bei der Migration umgestellt wurden, müssen Sie diesen Schritt für Ihre Benutzertabellen explizit durchführen. Um später nicht beide Indextypen in der Datenbank zu haben, empfiehlt sich hier die Umstellung während Ihres Wartungsfensters!
- ▶ *Aktualisieren und Füllen der neuen Statistik-Tabellen* mit dem `RUNSTATS`-Befehl. In diesem Fall wäre sogar der `REORGCHK`-Befehl vorzuziehen, da Sie so alle Tabellen in einem Befehl abhandeln und durch Ihr Wartungsfenster auch die größere Systemlast keine Rolle spielt.
- ▶ *Erneutes Binden aller Pakete* in der Datenbank, um die Zugriffspfade in der neuen DB2 UDB V8-Umgebung vom Optimizer neu bestimmen zu lassen. Sie werden dabei durch das Tool `db2rbind` unterstützt, das mit einem Befehl alle Pakete neu binden lassen kann. Folgende Struktur können Sie beim Aufrufen dieses Tools nutzen:
`db2rbind <datenbankaliasname> /1 <protokolldatei> all [/u <benutzerid> /p <kennwort>]`
- ▶ *Überprüfen Sie die Einstellungen der DB2-Registrierung*, da hier sowohl Variablen weggefallen als auch neue Variablen und neue Standardeinstellungen hinzugekommen sind. Beispiele dafür sind der neue Standardwert für `DB2_HASH_JOIN` (dieser ist in DB2 UDB V8 YES) und die in die Datenbankkonfiguration übernommene Einstellung des entfallenen Parameters `DB2_NEWLOGPATH2`.
- ▶ *Überprüfen der gesetzten Berechtigungen* innerhalb der Datenbank. Neue „Rechte“, wie z.B. `CREATE_EXTERNAL_ROUTINE`, können genutzt werden, da die neuen Standardrechte eventuell noch nicht Ihren Sicherheitsanforderungen entsprechen.
- ▶ Nachdem Sie alle Migrationsschritte ausgeführt und auch alle Einstellungen angepasst haben, sollten Sie *unbedingt* eine (Offline-)Sicherung Ihrer Datenbanken durchführen, um immer auf diesen konsistenten Zeitpunkt zurückgreifen zu können.

Tipp:

Ein weiteres Tool `db2cidmg` kann bei der automatischen Migration entfernter Datenbanken helfen.

Anschließend haben Sie alle Aufgaben abgeschlossen, die innerhalb des Wartungsfensters erledigt werden sollten. Weitere Tests können nun im laufenden (Test-)Betrieb erfolgen.

Sollten Sie zur Grundversion von DB2 UDB V8.1 noch zusätzliche Fixpacks einspielen wollen, beachten Sie bitte die weiteren Punkte des folgenden Abschnitts.

A.4 Migration bzw. Update nach Fixpack-Installation

Es mag vielleicht den einen oder anderen von Ihnen überraschen, aber auch nach der Installation von Fixpacks sind noch Migrations- oder Update-Prozesse durchzuführen. Dies gilt vor allem für die Unix-Welt, aber auch in der Windows-Welt gibt es unter Umständen etwas zu tun.

A.4.1 Spezifische Aufgaben nach der Installation von Fixpacks unter Unix

In Unix-Umgebungen müssen nach der Installation eines Fixpacks zusätzliche Befehle für die Aktualisierung der Umgebung ausgeführt werden. Diese Befehle erfordern eine root-Berechtigung im Betriebssystem und sind im Verzeichnis DB2DIR/instance/ zu finden.

DB2DIR ist dabei das DB2-Installationsverzeichnis und entspricht je nach Betriebssystem folgendem Pfad:

► AIX

/usr/opt/db2_08_01 bzw. /usr/opt/db2_08_FPN bei einer Multi-Fixpack-Installation, n entspricht dabei der Fixpack-Nummer.

► alle anderen Unix-Systeme

/opt/IBM/db2/V8.1 bzw. /opt/IBM/db2/V8.FPN bei einer Multi-Fixpack-Installation, n entspricht dabei der Fixpack-Nummer.

Aktualisierung des DAS (DB2 Administration Server)

Der Befehl dasupdt aktualisiert den DAS auf den neuen Fixpack-Stand. Wichtig ist dabei, dass dieser Befehl bei einer Multi-Fixpack-Installation aus dem DB2DIR/instance/-Unterverzeichnis des neuen (bzw. neuesten) Fixpack-Installationsverzeichnisses ausgeführt wird. DB2DIR ist dabei das Installationsverzeichnis von DB2 UDB.

Der „normale“ Aufruf erfolgt ohne Option. Folgende Optionen stehen für spezielle Situationen zur Verfügung:

► -h – Anzeige der Hilfe

► -d – Debug-Modus

► -D – Versetzen des DAS auf einen niedrigeren Programmstand (level)

Die letzte Option ist für Multi-Fixpack-Installationen gedacht, um einen DAS auf einen früheren Fixpack-Stand „zurückzusetzen“.

Aktualisierung der Exemplare

Der Befehl db2iupdt bezieht sich nun auf das einzelne DB2-Exemplar. Neben den Optionen des dasupdt-Befehls stehen hier noch weitere zur Verfügung, die wir in Form eines Syntaxdiagramms abgebildet haben.

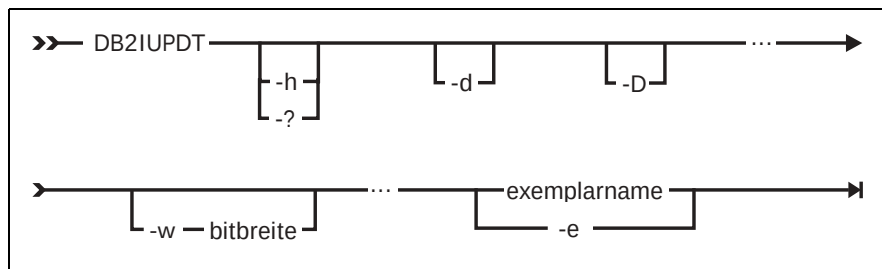


Abbildung A.1: db2iupdt-Syntaxdiagramm

Die Optionen im Einzelnen:

- ▶ -h oder -?
Zeigt die Hilfe zu diesem Befehl an.
- ▶ -d
Debug-Modus aktivieren.
- ▶ -D
Versetzt das Exemplar auf einen niedrigeren Programmstand.
- ▶ -w bitbreite
Bitbreite eines Exemplars ändern. 32 und 64 sind gültige Angaben für die Bitbreite. Diese Option gilt *nicht* für Linux!
- ▶ exemplarname oder -e
Hier geben Sie den Namen des Exemplars an, das aktualisiert werden soll, oder der Befehl bezieht sich durch -e auf alle Exemplare.

Hinweis:

Voraussichtlich wird bei einer Multifixpack-Installation der alternative Fixpack-Stand nicht durch einen DB2-Fixpack aktualisiert werden können. Diese Aktualisierung ist somit nur für die Grundinstallation möglich!

A.4.2 Aufgaben nach der Installation des Fixpacks 2 (DB2 UDB V8.1.2)

Dieser Abschnitt bezieht sich zwar auf alle Plattformen bzw. Betriebssysteme, ist aber erst ab dem Fixpack 2 brauchbar. Durch die Installation des DB2 Fixpack 2 bekommen Sie eine DB2-Version 8.1.2.

Mit dem Befehl db2updv8 werden die Systemkatalog-Tabellen um neue Einträge erweitert. Diese Einträge beziehen sich z.B. auf Routinen für den Diagnosemonitor und die Einbindung heterogener Datenquellen.

Die zugehörige Befehlsstruktur sieht so aus:

```
db2updv8 -d <datenbankaliasname> [-u <benutzerid> -p <kenwort> -h]
```

Dieser Befehl stellt eine Datenbankverbindung her. Dazu benötigt er den entsprechenden Datenbank-Aliasnamen und eventuell eine gültige Kombination aus Benutzer-ID und Kennwort. Wie bei fast allen Befehlen steht die -h-Option zur Verfügung, um einen Hilfetext anzuzeigen.

Um diese Routinen endgültig zu aktivieren, müssen Sie sicherstellen, dass die Datenbank zur Ruhe kommt, dass also alle Verbindungen beendet werden und eine eventuelle Aktivierung (kurzzeitig) aufgehoben wird.

B Beschreibung des CD-Inhalts

Auf der CD zum Buch sind folgende Komponenten enthalten:

- ▶ DB2 Universal Database Express Edition – Trial-Version (DB2_Express-Verzeichnis)
- ▶ DB2-Demonstrationsprogramm (DB2_Demo-Verzeichnis)
- ▶ Beispiel-Datenbank ToyStore (Toystore_Install-Verzeichnis)
- ▶ Beispiel-Datenbank BuchDB und Beispiel-Programme (DB2_Buch-Verzeichnis)
- ▶ Zusatzinformationen (pdf-Verzeichnis)

Die CD startet beim Einlegen in das CD-ROM-Laufwerk automatisch. Sollte dies nicht der Fall sein, können Sie es über den Aufruf `x:\starter.exe` manuell nachholen, wobei `x` der Laufwerksbuchstabe Ihres CD-Laufwerks ist.



Abbildung B.1: Start-Fenster der beigelegten CD

B.1 DB2 UDB Express Edition V8.1 – Trial-Version

Die CD enthält die Windows-Variante der DB2 UDB Express Edition. Der Link **INSTALLATION IBM DB2 UNIVERSAL DATABASE EXPRESS EDITION, VERSION 8.1.** führt Sie direkt zum Willkommen-Fenster der DB2-Installation (siehe Kapitel 3).

Für die Linux-Version der DB2 UDB Express Edition finden Sie einen Link zum kostenlosen Download einer Trial-Version ebenfalls direkt im Start-Fenster. Ebenso gibt es die Möglichkeit, weitere DB2-Dokumentation aus dem Internet herunterzuladen – der entsprechende Verweis befindet sich ebenfalls im Start-Fenster.

Diese Version unterstützt die Betriebssysteme Windows NT, Windows XP Professional Edition, Windows 2000 Professional und den Windows Server 2003. Eine Installation auf der Windows XP Home Edition ist leider nicht möglich!

Wenn Sie die DB2-Installation direkt anstoßen wollen, ohne über die Startanwendung zu gehen, finden Sie die `setup.exe` im Verzeichnis `x:\EB2_Express\`, wobei `x` wieder der Laufwerksbuchstabe Ihres CD-ROM-Laufwerks ist.

B.2 DB2DEMO-Programm

Über den Link **DB2-DEMONSTRATIONSANWENDUNG** können Sie die Installation einer gelungenen DB2-Demo-Umgebung starten, die allerdings nur in Englisch verfügbar ist.

Wollen Sie die Installation des DB2-Demoprogramms per Befehl ausführen, finden Sie die Datei `db2demosetup` im Verzeichnis `x:\DB2_Demo\`, wobei `x` wieder der Laufwerksbuchstabe Ihres CD-ROM-Laufwerks ist.

Voraussetzung für dieses Programm ist die Möglichkeit, auf eine DB2-Datenbank zugreifen zu können. Eine Installation funktioniert auch auf Windows XP Home Edition. Voll einsatzfähig ist dieses Programm nur dann, wenn dort eine Verbindung zu einem entfernten Datenbankserver konfiguriert ist.

Dieses Programm ist hervorragend geeignet, um verschiedene DB2 UDB V8-Produkteigenschaften zu demonstrieren. Das Fenster besteht, wie Abbildung B.2 zeigt, aus vier Teilfenstern:

► Übersichtsfenster

Hier können Sie die verschiedenen Themengebiete auswählen und erhalten dann eine einleitende Präsentation zum Thema und die Möglichkeit, diese V8.1-„Features“ auszuprobieren, indem Sie die Einzelschritte anklicken. Die zugehörige Syntax erscheint dann automatisch im Befehlsfenster.

► Befehlsfenster

Hier sehen Sie die Befehlssyntax für die DB2-Befehle bzw. die SQL-Abfragen.

► Hilfefenster

In diesem Teilfenster wird erläuternder Text zu den von Ihnen durchgeführten Schritten angezeigt.

- Ergebnisfenster
Hier sehen Sie die Ergebnisse der Befehle.

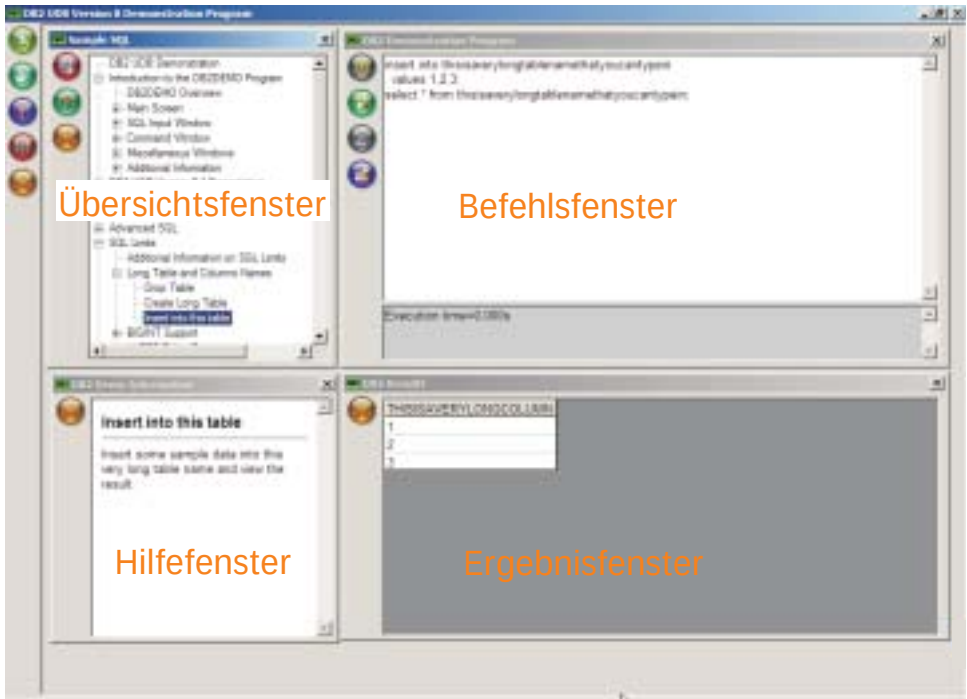


Abbildung B.2: DB2 Demonstrationsprogramm

Neben einer eigenen Einführungssektion finden Sie auch noch weitere Produktpräsentationen über die Themen-Hierarchie des Übersichtsfensters.

B.3 BuchDB-Scripts

Im Verzeichnis DB2_Buch finden Sie in den entsprechenden Unterverzeichnissen die Scripts und Dateien, auf die sich die Beispiele in diesem Buch beziehen:

- BUCHDB-Datenbank
Dieses Verzeichnis enthält die nötigen Scripts, um die BuchDB-Datenbank in Ihrem Exemplar zu erstellen. Rufen Sie das Script C_Datenbank.bat aus einer DB2-Befehlszeile auf, um dadurch die Datenbank BuchDB mit Inhalt zu erstellen.

Hinweis:
Stellen Sie sicher, dass in der Datei C_Datenbank.bat eine gültige Benutzer-ID/Kennwort-Kombination im CONNECT-Befehl angegeben ist. Standardmäßig haben wir das mit db2admin / db2admin vorbelegt.

► DB2EBUCH_Response_File

Dieses DB2-Response-File (für DB2 UDB ESE) ist die Ausgangsdatei, auf die in Kapitel 3 „Installation der DB2“ Bezug genommen wird. Diese Datei kann als Vorlage für Ihre eigenen Anpassungen dienen.

Ein Original-Response-File (db2exp.rsp) für die mitgelieferte DB2 UDB Express Edition finden Sie im Verzeichnis x:\DB2_Express\db2\Windows\samples\.

Eine Anpassung an Ihre Bedürfnisse sollte mit Hilfe der Vorlage aus diesem Verzeichnis ohne Schwierigkeiten machbar sein.

► Links

Eine HTML-Datei mit den Links aus dem Anhang des Buchs ist in diesem Unterverzeichnis enthalten. So können Sie sich nach Aufruf der Datei durch die Internet-Links klicken, ohne diese abtippen zu müssen.

► Datenbankprogramme

Die CD enthält auch verschiedene Beispiel-Datenbankprogramme, die in Kapitel 11 gezeigt werden.

Das Programm BuchDB gibt den Inhalt der Tabelle Adresse aus und liegt in verschiedenen Programmierarten vor. So enthält das Unterverzeichnis C eine Variante mit statischem SQL (BuchDB_S_SQL.SQC) und eine mit dynamischem SQL (BuchDB_D_SQL.SQC). Im Unterverzeichnis Java finden Sie die SQLJ-Variante des Beispiel-Programms (BuchDB_S_SQLJ.SQLJ).

Neben den Datenbankprogrammen finden Sie hilfreiche Scripts zur Vorbereitung der Programme. Fortgeschrittene können diese Batch-Files auch für ihre eigenen Anwendungen nutzen.

► SQL

Scripts mit SQL-Abfragen der BuchDB-Datenbank, die teilweise auch im Buch aufgeführt sind.

B.3.1 Nutzung der Beispiel-Datenbankprogramme

Um die mitgelieferten Datenbankprogramme auszuführen, sollten folgende Voraussetzungen erfüllt sein:

- DB2 UDB V8 muss installiert sein.
- Für die C-Programme benötigen Sie einen Microsoft C/C++-Compiler. Wenn Sie andere Compiler einsetzen, sind weitere Anpassungen nötig, die hier nicht näher erläutert werden.
- Die Datenbank BUCHDB muss existieren.
- Die Standard-Benutzer-ID db2admin mit dem Kennwort db2admin wird vorausgesetzt.
- Das Exemplar ist für die TCP/IP-Kommunikation über den Port 50000 konfiguriert und aktiv.

Sollte Ihre Installation von diesen Voraussetzungen abweichen, sind neben den im Folgenden beschriebenen Schritten weitere durchzuführen, wie verschiedene Anpassungen der Batch- und Programmdateien.

Ausführung der C-Datenbankprogramme (BuchDB_S_SQC und BuchDB_D_SQC)

1. Führen Sie die Datei `bcls.bat` für das statische Programm bzw. `bclld.bat` für das dynamische Programm aus.

`bcls` steht für „bind compile link statisch“ und `bclld` für „bind compile link dynamisch“.

Beide Programme rufen das Batch-File `bclmsemb.bat` mit den jeweiligen Parametern auf. Die Datei `bclmsemb.bat` stellt eine Verbindung zur Datenbank her und bindet, kompiliert und linkt das Programm. Diese Datei stellt die Programm-Vorbereitungsschritte durch den Microsoft C-Compiler sicher und kann auch für Ihre eigenen Programme nützlich sein, da bei der Erstellung extra auf Wiederverwendbarkeit geachtet wurde.

2. Ausführen des Programms durch den Aufruf:

`BuchDB_S_SQC.exe` bzw. `BuchDB_D_SQC.exe`

Ausführung des Java-Datenbankprogramms (BuchDB_S_SQLJ)

1. Passen Sie die Datei `setjava.bat` an, und führen Sie sie aus.

Diese Datei dient zur Anpassung Ihrer `PATH`-Umgebungsvariable, um den Pfad zum Java-Installationsverzeichnis zu setzen. Standardmäßig ist `D:\IBM\SQL-LIB\java\jdk\bin` eingestellt, das Java-Verzeichnis der DB2-Installation. Bitte überprüfen Sie diesem Pfad, und passen Sie ihn gegebenenfalls an.

2. Aufruf von `bcljava.bat`

Die Datei `bcljava.bat` ruft die Datei `bclsqlj.bat` mit dem Parameter *Programmname* (`buchdb_s_sqlj`) auf.

3. Ausführen des Programms durch den Aufruf von:

`java BuchDB_S_SQLJ`

Die weiteren Dateien können Ihnen als nützliche Beispiele dienen und Hilfestellungen für eigene Programme sein. Für die reine Ausführung des Beispiels werden sie jedoch nicht benötigt. Besonders interessant ist die Datei `bclsqlj.bat`, wenn Sie selbst Programme umwandeln wollen. Sie prekompiliert, kompiliert und bindet das SQLJ-Programm.

B.4 ToyStore-Beispiel-Datenbank und Datenbankapplication

Im Unterverzeichnis `x:\Toystore_Install` finden Sie Scripts zum Anlegen einer weiteren Datenbank, der ToyStore-Datenbank und einer Beispiel-Anwendung.

Diese Installation können Sie auch direkt über das Startfenster aufrufen.

Die Datei `x:\Toystore_Install\Readme.txt` enthält Details zum manuellen Erstellen und zum Aufruf der Java-Beispiel-Anwendung.

Durch das Script `Install.bat` wird die TOYSTORE-Datenbank inklusive Tabellen und Inhalt erstellt.

C Literaturverzeichnis

[ADMIN1]

IBM DB2 Universal Database Systemverwaltung: Konzept bzw. IBM DB2 Universal Database Admin Guide Planning Version 8

IBM, Juli 2002

IBM Form: SC12-3057 (deutsch) bzw. SC09-4822 (engl.)

[AKUZNETSOV]

Essential Facts about Index Covering in DB2 Universal Database

Alexander Kuznetsov, Chicago, März 2003

<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0303kuznetsov/0303kuznetsov.html>

[BROK]

Der Brockhaus in einem Band

10., neu bearbeitete Auflage 2003

ISBN: 3-7653-1680-6

<http://www.brockhaus.de>

[DB2LIN]

Up and Running with DB2 for Linux

IBM ITSO Redbook, März 2003;

IBM Form: SG24-6899

ISBN 0738427675

[GUITOOLS]

IBM DB2 Universal Database GUI-Tools für die Verwaltung und Entwicklung Version 8 bzw. IBM DB2 Universal Database Guide to GUI Tools for Administration and Development

IBM September 2002

IBM Form: SC12-3041-00 (deutsch) bzw. SC09-4851 (englisch)

[MONGUIDE]

IBM DB2 Universal Database System Monitor Guide and Reference Version 8

IBM, 2002

IBM Form: SC09-4847-00

[SG246546]

DB2 UDBs High Function Business Intelligence in e-business

Nagraj Alur u.a.

IBM Corporation, International Technical Support Organisation, California/USA

IBM Form: SG24-6546-00, September 2002

ISBN: 0738424609

[SNOW]

Advanced DBA Certification Guide and Reference for

DB2 Universal Database v8 for Linux, UNIX and Windows

Dwaine R. Snow, Thomas X. Phan ; Prentice Hall, New Jersey, 2003

ISBN 0-13-046388-4

[SQLREF]

IBM DB2 Universal Database SQL Reference Version 8

IBM, Juli 2002

IBM Form: SC09-4822

[SRIELAU]

INSTEAD OF Triggers – All Views are Updatable!

Serge Rielau, IBM Toronto, Oktober 2002

<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0210rielau/0210rielau.html>

[YIP1]

Using SQL Pcedural Language for Triggers in DB2 UDB

Paul Yip, IBM 2001

<http://www.7b.boulder.ibm.com/dmdd/library/techarticle/yip/0111yip.html>

[YIP2]

How to Temporarily Disable Triggers in DB2 Universal Database

Paul Yip, Erasmo Acosta, Tony Lee, IBM, Oktober 2002

<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0211yip/0211yip.html>

[YIP3]

DB2 Basics: Fun with Dates and Times

Paul Yip, IBM Toronto Lab Nov. 2002

<http://www7b.boulder.ibm.com/dmdd/library/techarticle/0211yip/0211yip3.html>

D Glossar

Adressraum (address space)

Der von einem Programm verwendete Speicherbereich im Hauptspeicher.

ADSM (ADSTAR Distributed Storage Manager)

Ein Tool zur Durchführung und Verwaltung von Sicherungen, das inzwischen in TSM (Tivoli Storage Manager) umbenannt wurde. Eine direkte Schnittstelle in DB2 erlaubt es, Datenbanksicherungen direkt an dieses Tool zu übergeben.

Agent

Ein Prozess oder Thread, der in DB2 alle Anforderungen der Anwendung ausführt. Er ist sozusagen der Stellvertreter der Anwendung im DBMS.

Aktive Protokolldatei (active log)

Alle Protokolldateien einer Datenbank, die für eine ROLLBACK-Operation benötigt werden, d.h. in anderen Worten: alle Protokolldateien, die noch eine nicht abgeschlossene Transaktion beinhalten.

API (Application Programming Interface)

Standardisierte Programmschnittstellen. Diese stellen die Funktionalität des jeweiligen Programms für externe Programme zur Verfügung.

APPC (Advanced Program to Program Communication)

Ermöglicht die Kommunikation im Netz zwischen mehreren Programmen innerhalb der Architektur SNA.

Archivprotokolldatei (archive log)

Alle Protokolldateien, die für ein ROLLBACK nicht mehr benötigt werden, aber für eine spätere aktualisierende Wiederherstellung aufbewahrt werden müssen.

ASCII (American Standard Code for Information Interchange)

Kodierungsart für Zeichenfolgen auf Personal-Computern und Workstations, um systemübergreifend Daten austauschen zu können. Ein Zeichen wird dabei durch sieben Bits kodiert. Siehe auch **EBCDIC**.

Asynchron (asynchronous)

Zeitlich unabhängig. Bei Datenbanken ist damit meist gemeint, dass eine Aktion nicht innerhalb einer Transaktion ausgeführt wird.

Authentifizierung (authentication)

In DB2 wird dies durch das Betriebssystem (oder externe Produkte) übernommen.

Cache bzw. Caching

Ein Cache ist ein schneller Zwischenspeicher. Caching ist die Nutzung dieses Zwischenspeichers, um bei Bedarf Daten schnell zur Verfügung stellen zu können.

CLP (command line processor)

Die CLP stellt die DB2-Befehlszeile dar und wird benötigt, da das jeweilige Betriebssystem weder die SQL-Syntax noch die Syntax der Datenbankbefehle verarbeiten kann. Der CLP ist also ein Programm, dem die entsprechenden SQL- oder Datenbankbefehle übergeben werden und das diese dann nach einer einfachen Syntax-Prüfung an das DBMS weiterleitet.

Codd'sche Regeln

Mitte der 70er Jahre hat Edgar Codd in einem Forschungsprojekt der IBM die theoretischen Grundlagen für relationale Datenbanken und Datenbank-Systeme gelegt. Zusammen mit Chris Date wurden diese Regeln dann weiterentwickelt und vervollständigt.

COMMIT-Operation (commit)

Festschreibung von Änderungen in einer Datenbank. Die Commit-Anweisung stellt das Transaktionsende dar und gibt alle von der Transaktion gehaltenen Sperren auf Objekte frei.

CRDA (character data representation architecture)

Interne Darstellung der Daten, z.B. ASCII- oder EBCDIC-Codierung

CRM (customer relationship management)

Dieser Begriff steht für Methoden und auch häufig für Software-Pakete, mit denen Firmen Ihre Kundenbeziehungen organisiert managen.

Cursor

Eine Steuerstruktur, mit der Anwendungen Datenbankinhalte abrufen. Ein Cursor wird durch die abzusetzende SQL-Abfrage definiert und im Programm geöffnet, bevor die Inhalte mit FETCH-Operationen übertragen werden.

DBMS (Datenbankmanagementsystem)

Das Datenbankmanagementsystem interpretiert die eingehenden Befehle (SQL- oder Datenbankbefehle) und führt diese dann, falls sie fehlerfrei sind, aus.

DCL (Data Control Language)

DCL ist der Teil des SQL-Standards, der sich mit der Berechtigungsverwaltung, also der Vergabe und dem Entzug von Berechtigungen beschäftigt. GRANT und REVOKE gehören zu dieser Gruppe.

DDL (Data Definition Language)

Teil des SQL-Standards, der die Befehle für die Erstellung und das Löschen der (Datenbank-)Objekte zusammenfasst. Darunter fallen z.B. ein CREATE TABLE... oder DROP VIEW...

DEL (delimited ASCII)

Ein ASCII-Format mit Zeichenbegrenzer, d.h. die Spalten sind z.B. durch Kommata getrennt. Siehe auch **ASCII**.

DML (Data Manipulation Language)

Teil des SQL-Standards, der die Befehle für die Abfrage der Daten beinhaltet, also INSERT, UPDATE, DELETE und SELECT-Anweisungen.

DMS-Tabellenbereich (database managed space – table space)

Tabellenbereich, dessen Verwaltung die Datenbank übernimmt.

DRDA (Distributed Relational Database Architecture)

Ein von IBM entwickelter Standard, durch den Zugriffe und Formate auf ferne Datenbanken definiert werden. Dabei kommuniziert ein DRDA AR (application requestor) mit der Serverkomponente DRDA AS (application server).

DUOW (distributed unit of work)

Diese verteilte Arbeitseinheit verändert Daten auf mehreren Datenbanken in einer einzelnen UOW. Um die Konsistenz in diesem Fall gewährleisten zu können, sind spezielle Mechanismen erforderlich (die zweiphasige Festschreibung – two-phase commit). Siehe auch **UOW**.

E/A (Eingabe/Ausgabe) oder I/O (input/output)

Beschreibt den Lese- und Schreibzugriff auf z.B. die Festplatten in einem System. Dies ist im Vergleich zu einem Hauptspeicherzugriff (dieser wird nicht als E/A-Zugriff bezeichnet) sehr viel langsamer.

EBCDIC (Extended Binary-Coded Decimal Interchange Code)

Zeichenkodierung, die hauptsächlich auf Großrechnern Verwendung findet. Dabei stehen in einem Zeichensatz 256 Zeichen zur Verfügung, die jeweils durch acht Bit kodiert sind.

EDU (engine dispatchable unit)

Allgemeiner Oberbegriff für Threads und Prozesse. Threads werden im Windows-Umfeld, Prozesse im Unix-Umfeld benutzt.

ER-Modell (Entity-Relationship-Modell)

Um zu wissen, welche Tabellen, Sichten etc. in einer Datenbank erstellt werden müssen, muss ein Datenmodell existieren. Dieses Datenmodell wird aus den Vorgaben einer Geschäftsanforderung erstellt. Die ER-Modellierung ist eine Technik, die Regeln und Notationen für diese Datenmodellierung vorgibt.

ERP (enterprise resource planning)

Diese Abkürzung steht für eine integrierte, betriebswirtschaftliche Standardsoftware, wie z.B. SAP. Diese Software integriert u.a. Komponenten für das Rechnungswesen und Personalmanagement.

ERwin

Der *AllFusion ERwin Data Modeler* der Firma Computer Associates ist ein Programm, das die (ER-)Modellierung für Datenbanken unterstützt. Mit diesem Tool ist es möglich, aus einem Modell die Datenbanktabellen generieren zu lassen oder auch das Datenmodell aus einer bestehenden Datenbank per „reverse-engineering“ aufbauen zu lassen.

ESE (Enterprise Server Edition)

DB2 Server Version, mit der eine Verbindung zu DB2 am Großsystem, also z.B. zu DB2 für z/OS oder DB2 für AS/400 möglich ist. Dazu ist die DB2-Produktkomponente DB2 Connect integriert. Diese Version wird auf Prozessorbasis lizenziert.

IMS (Information Management System)

Hierarchisches Datenbankprodukt von IBM.

Information Integrator

Der **Information Integrator** ermöglicht die Einbindung von Datenbanktabellen anderer Hersteller in eine DB2-Datenbank mit vollen Lese- und Schreibmöglichkeiten. Der DB2 Information Integrator ist dabei ein separates Produkt, das die Produkte *DB2 Relational Connect* und *DB2 DataJoiner* ersetzt.

IXF (Integrated Exchange Format)

Binäres Speicherformat für Datenbankinhalte, das neben den eigentlichen Daten auch die Daten- bzw. Spaltendefinitionen speichert.

Kardinalität (cardinality)

Anzahl der Zeilen einer Tabelle.

Mining/Data Mining

Unter dem Oberbegriff Data Mining sind eine Reihe komplexer Techniken und Algorithmen für die Datenanalyse zusammengefasst. Dabei können große Datenmengen auf vorher unbekannte Zusammenhänge überprüft werden.

Null

Der NULL-Wert bedeutet, dass ein Feld leer ist. Er stellt also nicht die klassische „0“ dar, sondern bedeutet: „nicht vorhanden“.

ODBC (open database connectivity)

Standardschnittstelle für Datenbanken. ODBC-Programme ermöglichen so einen datenbankunabhängigen Einsatz.

OLAP (online analytical processing)

Analyse-orientierte Abfragen finden häufig in Data Warehouse-Umgebungen statt und sind eher komplexer Natur. Um diese teilweise komplexen Analysen performant (online) durchzuführen, ist neben speziellen Analysewerkzeugen auch eine spezielle Aufbereitung und Speicherung der Daten nötig.

OLTP (online transactional processing)

Diese Abkürzung steht für die transaktionsorientierte Verarbeitung, bei der typischerweise Änderungen in viel kleineren (und relativ einfachen) Transaktionen durchgeführt werden. Diese Art der Datenbankverarbeitung findet man in operativen Datenbanken für die tägliche Verarbeitung von Aufträgen etc.

PE (Personal Edition)

DB2-Produkt für den lokalen Einsatz. Diese Version hat keine Server-Funktionalität, und somit können keine entfernten Clients angeschlossen werden. Lokal können Verbindungen aufgebaut werden und Anwendungen ohne Einschränkungen laufen.

RC bzw. RTC (run-time client)

Der RTC simuliert sozusagen das auf dem Client nicht vorhandene DBMS. Er nimmt den CONNECT-Befehl entgegen und schaut in den entsprechenden Katalogeinträgen nach, auf welchem Server sich die Datenbank befindet.

RDB (relational database)

Relationale Datenbank – diese Datenbank folgt den Codd'schen Regeln.

RDBMS (relational database management system)

Relationales Datenbankmanagementsystem – siehe auch DBMS.

RID (record identifier) bzw. Satzkennung

Die Satzkennung ist ein Verweis auf die Speicherseite und -den Speicherort eines Datensatzes. Diese RIDs kommen zum Einsatz, um aus der Indexstruktur auf die zugehörigen Daten zu verweisen.

SMS-Tabellenbereich (system managed space – table space)

Bei dieser Tabellenbereichsart werden die Daten vom Betriebssystem in Dateien verwaltet. SMS ist die Standard-Tabellenbereichsart in DB2.

SNA (Systems Network Architecture)

SNA ist eine von IBM entwickelte Architektur für die Kommunikation im Netz. Dazu gehören Beschreibungen von logischen Strukturen, Formaten, Protokollen etc. für die Datenübertragung sowie für die Konfiguration und den Betrieb des Netzes.

SQL (Structured Query Language)

Abfragesprache für relationale Datenbanken.

SQLCA (SQL communication area)

Die *SQL communication area* dient zur Übermittlung der Kommunikationsdaten wie Fehlermeldungen, Warnhinweisen und weiteren Informationen.

SQLDA (SQL data area)

Die *SQL data area* beinhaltet sowohl die Beschreibung der Daten (Datentyp, Länge, Pointer auf den Dateninhalt) als auch den eigentlichen Dateninhalt.

Systemkatalog (catalog)

Der Systemkatalog einer Datenbank besteht aus mehreren Tabellen und Sichten, die die Struktur der Datenbank beschreiben. Die Systemkatalog-Tabellen sind mit dem reservierten Schema SYSIBM, die Systemkatalog-Sichten mit den Schemata SYSCAT und SYSSTAT gekennzeichnet.

TCP/IP (Transmission Control Protocol/Internet Protocol)

Ein sehr verbreitetes Netzwerk-Protokoll, mit dem u.a. das Internet arbeitet und heute auch die meisten Verbindungen von Clients zu Datenbank-Servern.

Toolskatalog (DB2 tools catalog)

Verschiedene Tabellen, die typischerweise in einer separaten Tools-Datenbank gespeichert sind und Informationen für die Verwaltung von Scripts und Prozessen speichern. Ein Beispiel dafür sind die Zeitpläne zur zeitlich gesteuerten Ausführung von Scripts.

TSM (Tivoli Storage Manager)

IBM-Tool zur Verwaltung von Sicherungen. Eine direkte Schnittstelle im DB2 unterstützt die Übergabe von Datenbank- oder Tabellenbereichssicherungen an diese Software.

UOW (unit of work)

Ein anderer Begriff für UOW ist Transaktion. Beide Begriffe beschreiben eine „Arbeitseinheit“ in einer Datenbank, die aus mehreren Operationen (SELECT, INSERT, UPDATE oder DELETE) bestehen kann und nach dem Alles-oder-nichts-Prinzip ausgeführt wird. Das heißt, entweder wird die UOW komplett ausgeführt und festgeschrieben (COMMIT) oder – im Fehlerfall – komplett rückgängig gemacht (ROLLBACK).

WSE (Workgroup Server Edition)

DB2-Produkt, das als Server für DB2-Clients in nicht-partitionierten Umgebungen eingesetzt werden kann. Diese Version wird auf Benutzerbasis lizenziert.

WSF (Work Sheet Format)

Ein Lotus-Dateiformat, das für den Export und Import in DB2 unterstützt wird.

XML (Extensible Markup Language)

Ähnlich wie bei HTML ist XML ein „Ableger“ von SGML. In XML werden so genannte „Tags“ verwendet, um den Inhalt zu beschreiben, während die Tags in HTML die Darstellung beschreiben. Dieses Format kann in DB2 gespeichert werden bzw. von DB2 generiert werden. XML eignet sich besonders zum Austausch von Daten, z.B. zum Export von Metadaten im Data Warehouse-Umfeld.

Zugriffspfad oder Zugriffsweg

Die vom DBMS bzw. Optimizer gewählte Methode des Datenzugriffs. Beispiele dafür sind ein Indexzugriff oder wahlweise ein Tabellen-Scan, also das sequenzielle Lesen der Tabelle.

Stichwortverzeichnis

► Numerisch

64-Bit-Exemplare 50

► A

Abfragediagramm modell 599
Abfragesprache SQL 457
Ablaufplan 458
Admin Node Directory
 siehe Knotenverzeichnis
Administration 28
ADO-Schnittstelle 497
ADVISE_INDEX 323
ADVISE_WORKLOAD 323
AIX 4.3 51
Aktive Protokolldateien 372
Aktualisierende Wiederherstellung 388
 anstehend 426
Algorithmen 61
Aliasnamen
 CREATE ALIAS Befehl 284
Aliasnamen (Alias) 284
Allozieren 480
Ambiguous Cursor 465
Änderbarer Cursor 464
Anwendungsserver 52
Anzahl Puffer 399
API (Application Programming Interface) 37
API-Funktion 455, 471
API-Schnittstelle 455
API-Schnittstellenfunktionen 455
Applikationsserver 55
ARCHIVE_PATH siehe Userexit
AS (Application Server) 42
ASC 326
Assistent
 Datenbank erstellen 166
 Datenbank hinzufügen 254
 Tabelle erstellen 271
 Tabellenbereich erstellen 195
ATTACH 243
Audio Extender 57
Auslöser (Trigger) 300
 CREATE TRIGGER Befehl 302
AUTHENTICATION
 siehe Authentifizierung
Authentifizierung 219
 Client 221
 Kerberos 223
 Konfiguration am Client 224
 Server 220
 SERVER_ENCRYPT 221
AUTOCONFIGURE 181

Automatic Summary Table (AST) 277

Autorisierung 437

► B

BACKUP DATABASE 389, 401
BACKUP DATABASE ONLINE 393
Backup Mode 405
Batch-Files 461
BCLMSSQC.BAT 472
bcls.bat 472
Befehl con.rollback() 480
Befehlstypen 454
Befehlszentrale siehe GUI, Befehlszentrale
BEGIN DECLARE SECTION 463
Behälter 185, 188
 Aufbau 186
Beispielprogramm für dynamisches SQL 492
Benachrichtigungsprotokoll 144
 Beispiel 543
Benannter Cursor 476
Benannter Iterator 476
Benutzerbasierende Lizenzierung 52
Benutzerdefinierte Funktion (UDF) 307
 Skalar-Funktion 308
 Spalten-Funktion 308
 Tabellen-Funktion 308
Benutzerdefinierter Datentyp (UDT) 305
 Benutzerdefinierter einzigartiger
 Datentyp 305
 Benutzerdefinierter Strukturtyp 306
 CREATE DISTINCT TYPE-Befehl 305
Benutzerexit siehe Userexit
Benutzerzuordnung 260
Berechtigungsstufe 437, 448
 DBADM 450
 SYSADM 449
 SYSCTRL 450
 SYSMAINT 450
 Überblick 451
BIND 541, 603
BINDFILE 470–471
BLK_LOG_DSK_FUL 383, 430
BLOCKING 473
Break-Even-Point 55
BuchDB_S_SQC.BND 472
BuchDB_S_SQC.EXE 467
BuchDB_S_SQC.obj 471
BuchDB_S_SQC.SQC 467
BuchDB_S_SQLJ.cl 480
buchdb_s_sqlj.java 481
BuchDB_S_SQLJ.SQLJ 461, 480–481
Bufferpool siehe Pufferpool

buffpage Parameter 316
Business-Intelligence 64

► C

C 456, 460–461
C_dclgn.bat 464
CAE 55
CANCEL 423
CATALOG Befehl 242
CATALOG DATABASE 244
CATALOG DCS DATABASE 255
CATALOG TCPIP NODE 242
CHARDEL 326, 330
CHNGPGS_THRESH 371
cl 471
Clientprofil 246
 erstellen 250
 importieren 252
CLI-Schnittstelle 497
Close 476
CLOSE Cursor1 467
CLP 33
Clustering 289, 531
COBOL 456, 461
Cobol 456
Codd, Edgar 23
Codd'sche Regeln 23
Codepage 329
Codepage-Konvertierungsproblem 465
Codierter Zeichensatz (codeset)
 siehe Datenbankpartitionsgruppe
COLDEL 326, 330
con.close() 480
concurrent User 51
Concurrent-Benutzer 52
CONNECT 214
CONNECT RESET 467
Connect zur Datenbank 478
CONNECT-Befehl 265
Connection-Context 478
Consistency Token 469
Container 185
Context 478
CPE 54
crash recovery 388
CRDA (Character Data Representation
 Architecture) 44
CREATE EVENT MONITOR 568
 WRITE TO FILE 576
CREATE NICKNAME 263
CREATE SERVER 262
CREATE TOOLS CATALOG 113, 626
CREATE USER MAPPING 263
CREATE WRAPPER 261
CREATE_EXTERNAL_ROUTINE 627
CS 509–510
CUE 55
CURSOR 353

Cursor 464
CURSOR FOR UPDATE 515
Cursor Stability 509–510
Cursor-Deklaration 464
Cursor-Klasse 475
Cursor-Typen 464

► D

DAS (Database Administration Server) 69,
 238
DAS (DB2 Administration Server)
 Aktualisierung 628
dasrct 113
dasupdt 121
Data Warehouse-Umgebung 149
 ETL-Prozess 151
Data Warehouse-Zentrale 86
Data Warehouse-Zentrale siehe GUI,
 Data Warehouse-Zentrale
Database Partitioning Feature 53
database seed 408
DATABASE_MEMORY 618
DATEISO 330
Datenbank 34
 aktivieren 172
 Aliasnamen 166, 214
 Anlegen einer Datenbank 169
 CONNECT 172
 CREATE DATABASE
 Interne Abläufe 169
 CREATE DATABASE-Syntax 169
 deaktivieren 172
 Definition 31
 Grundlagen 23
 Integritätsprüfung 545
 Katalog 162
 Komponenten 162
 Konfiguration 176
 ändern 178
 Konfigurationsadvisor 180
 Objekte 36
 Parameter 178
 Pflege 525
 Rechte 439
 SAMPLE-Datenbank 319
 Sichten (Views) 282
 Verbindung 214
 entfernt 224, 245
 Fehlerdiagnose 264
 Konfiguration 240
 lokal 216, 218
 Profile 246
 vorbereiten 234
 Verwaltungsdateien 171
 Verzeichnisstruktur 170
 Wartungsmodus 209
Datenbankgebiet (territory) 164
Datenbankgebietscode (country) 164

- Datenbankobjekte 267
 - Aliasnamen (Alias) 284
 - Sichten (Views) 282
- Datenbankpartitionsgruppe 163
- Datenbankschnittstellen 457
- Datenbank-Ursprungspunkt 408
- Datenbewegungen 325
- Datenformate 325
 - ASC 325
 - IXF 325
 - WSF 325
- DB2
 - Berechtigungen 437
 - Einzelplatz-Einsatz 37
 - Exemplar 67
 - heterogener Netzwerkbetrieb 39
 - homogener Netzwerkbetrieb 38
 - Installation 67
 - per Response-File 88
 - Response-File 71
 - Installationsmöglichkeiten 70
 - Installationspfad 625
 - Installationsverzeichnis 628
 - Internetbetrieb 40
 - Monitore 553
 - Systemmonitor 553
 - Überwachung (Monitoring) 553
- db2
 - Syntaxdiagramm 104
- db2 bind 472
- DB2 Connect 54, 213, 255
- DB2 Connect Application Server Edition 55
- DB2 Connect EE 53
- DB2 Connect Enterprise Edition 55
- DB2 Connect Gateway 55
- DB2 Connect Personal Edition 54
- DB2 Connect Unlimited Edition 55
- DB2 Data Links Manager 59
- DB2 Dokumentation 88
- DB2 ESE 53
- DB2 High Performance Unload 65
- DB2 Information Integrator 62, 259
- DB2 Intelligent Miner for Data 63
- DB2 Intelligent Miner Modelling 63
- DB2 Intelligent Miner Scoring 64
- DB2 Intelligent Miner Visualization 64
- DB2 Net Search Extender 57
- DB2 NSE 57
- DB2 OLAP Integration Server 61
- DB2 OLAP Server 61
- DB2 PE 49
- DB2 Performance Expert 64
- DB2 Personal Development Edition 56
- db2 prep 470
- DB2 Recovery Expert 64
- DB2 Satellite Edition 50
- DB2 SQLJ Profile Customizer 483
- DB2 Table Editor 64
- DB2 TE 57
- DB2 Text Extender 57
- DB2 TIE 57
- DB2 UDB Enterprise Server Edition 53
- DB2 UDB ESE 53
- DB2 UDB Everyplace Edition 47
- DB2 UDB Express Edition 50
- DB2 UDB Personal Edition 49
- DB2 UDB Workgroup Server Edition 51
- DB2 UDB WSE 53
- DB2 Universal Developer Edition 56
- DB2 Warehouse Manager 60
- DB2 Web Query Tool 64
- DB2 WSE 50–51
- DB2 XML Extender 57
- db2_install 72
- DB2_PARALLEL_IO 211
- db2admin create 112
- db2advis Tool 323
- db2api.lib 471
- db2batch 595, 604
- DB2-Befehlszeile (CLP) 102
 - interaktive 103
 - nicht interaktive 103
 - Optionen 107
 - Optionen ändern 108
- db2cfexp 250
- db2cidmg 627
- db2ckbkb 431
- db2ckmig 624
- db2cli.ini 603
- DB2COMM 228, 238
- DB2CTLSV 82
- db2dart 547
- DB2DBDFT 173
- db2dclgn 464, 477
- db2diag.log 543
- db2empfa 211
- db2empfa-Tool 315
- db2ese.rsp 71
- DB2EVA 572
- DB2EVMON-Tool 576
- db2evtbl 569
- db2explain 603
- db2fmd 593
- db2icrt 95, 98–99
 - Unix-Syntax 99
 - Windows-Syntax 100
- db2idrop 95
- db2ilist 95, 116
- db2imigr 626
- db2inspf 547
- DB2INSTANCE 111, 115, 214
- db2iupdt 95, 119, 628
- db2level 118, 549
- db2licm 73
- db2look 367
- db2look-Tool 319
- db2memvis 590
- DB2MOVE 365

- db2mtrk 591
 - DB2OPTIMIZATION 603
 - DB2-Optimizer 599
 - db2profile 102, 111
 - DB2-Profilregistrierdatenbank (DB2-Registry) 115
 - DB2-Prozesse
 - beim Sichern einer Datenbank 616
 - in Windows 615
 - DB2-Prozessmodell 612
 - db2rbind 627
 - db2relocate 212
 - db2relocatedb siehe Userexit
 - db2rspgn 72
 - db2rtcl.rsp 72
 - DB2-Scripts 109
 - ausführen 110
 - db2set 115–116
 - Syntaxdiagramm 117
 - db2setup 71–72, 92
 - DB2-Speichermodell 618
 - db2sqljcustomize 483
 - db2start 95, 101
 - db2stop 95, 101
 - DB2-Support 548
 - db2support 549
 - Syntaxdiagramm 550
 - DB2-Systemmonitor 555
 - DB2-Toolskatalog
 - erstellen 113
 - db2trc 551
 - Anleitung 551
 - DB2-Umgebungsvariablen 115
 - Auflösung 117
 - db2updv8 629
 - DB2-Verwaltungsserver (DAS)
 - erstellen 112
 - db2wse.rsp 72
 - DBADM 450
 - DBDFTPATH 166
 - dbheap Parameter 316
 - DBMS 33
 - Einsatzgebiete 37
 - DCS (database connection services) 255
 - DCS-Verzeichnis 255
 - DDL (Data Definition Language) 267
 - DEACTIVATE DATABASE 174
 - Deadlock 516
 - decompose 57
 - DECPLUSBLANK 329
 - DECPT 331
 - Default-Context 478
 - Deklaration
 - der Host-Variablen 477
 - des Cursors 479
 - und Öffnen des Cursors 493
 - DEL 326
 - Delimiter 461
 - Delta-Sicherung 387
 - DESCRIBE TABLE Befehl 280
 - Designadvisor 321
 - Dezimalzeichen siehe DECPT
 - DFT_QUERYOPT 603
 - DIAGLEVEL 544
 - Diagnosemonitor 580
 - Diagnoseprotokoll (db2diag.log) 543
 - Diagnosezentrale 553
 - Diagnosezentrale (Health Center) 580
 - Diagnosezentrale siehe GUI, Diagnosezentrale
 - DIAGPATH 542, 544
 - Dimension 61
 - Dirty Read 510
 - DISCONNECT 174
 - DISCOVER 234
 - DLCHKTIME 519, 522
 - DMS 190
 - Behälter 195
 - DPF 53
 - DRDA 213, 623
 - Application Requestor (AR) 213
 - Application Server (AS) 213
 - AR (Application Requestor) 42
 - AS (Application Server) 44
 - Schichten-Architektur 43
 - Verbindung 213
 - drill-through 61
 - DUMPFIL 352
 - DUOW (Distributed Unit Of Work) 214
 - siehe Zweiphasige Festschreibung
 - Dynamisches SQL 458, 489, 603
 - mit bekannter Spaltenliste 490
 - mit unbekannter Spaltenliste 489
- **E**
- E-Commerce 64
 - EDU (Engine Despaching Unit) 613
 - Embedded-SQL 456
 - Embedded-SQL-Methode 457
 - END COMPLETE/STOP 423
 - END DECLARE SECTION 463
 - END OF LOGS 423
 - Endbenutzer 30
 - ENDE EXEC 461
 - Enterprise-Management-Information 61
 - Entwicklungszentrale siehe GUI, Entwicklungszentrale
 - Ereignismonitor 566
 - aktivieren 571
 - Datenanalyse 572
 - event types 567
 - Optionen 568
 - stoppen 572
 - ergebnis.close() 480
 - Erste aktive Protokolldatei 372
 - Eskalation 523
 - Essbase 61

Exception 477
 Exclusive 514–515
 EXEC SQL 461
 EXECUTE-Berechtigung 472
 Execution-Context 478
 Exemplar
 Aktualisierung 628
 anlegen 97
 Anzahl 68
 Befehle 95
 Eigner 70
 Exemplarliste 116
 Fenced-Benutzer 70
 Konfiguration 110
 Konfiguration ändern 111
 Starten und stoppen 101
 Exemplar (instance) 67
 anlegen 95
 Definition 68
 Exklusiver Connect 505
 EXPLAIN 473
 EXPLAIN MODE YES 604
 EXPLAIN SNAPSHOT EXPLAIN 604
 EXPLAIN.DDL 607
 EXPLSNAP 473
 EXPORT
 LONG-Daten 332
 Syntaxdiagramm 332
 Export 327
 mit der Steuerzentrale 328
 eXtensible Markup Language 57
 Extent siehe Speicherbereich

► F

FDOCA (Formatted Data Object Content Architecture) 44
 FEDERATED 261
 Federated Database 259
 Fehlermonitor 554, 593
 FETCH 466, 475
 FETCH CURSOR 466, 493, 507
 FETCH-Anweisung 479
 FileNet Panagon 63
 FLUSH PACKAGE CACHE 541
 FOR FETCH ONLY 465
 FOR READ ONLY 465
 FOR SNAPSHOT 604
 FOR UPDATE 465
 FORCE APPLICATION 175, 520
 FORTAN 456, 461
 Funktionsweisen 453

► G

Geschäftsprozesse 62
 Gespeicherte Abfragetabelle (MQT) 277
 Gespeicherte Prozedur
 CREATE PROCEDURE-Befehl 311

Gespeicherte Prozeduren
 (Stored Procedures) 309
 get admin cfg 114
 get dbm cfg 110
 get health snapshot 586
 get monitor switches 556
 get snapshot 559
 getUpdateCount() 479
 GRANT 445–446, 450
 GUI
 Ansprechpartner 130
 Assistenten 128
 Befehlszentrale 134
 Interaktiv 134
 Prozedur 136
 Reiter Ergebnisse 135
 Data Warehouse-Zentrale 149
 Anmeldung 151
 Klickstartleiste 151
 Lernprogramm 152
 Prozessmodell 152
 Steuerungsdatenbank-
 verwaltung 150
 vorbereitende Schritte 150
 Diagnosezentrale 146
 Entwicklungszentrale 153
 Assistent 154
 Projekt 153
 Icons 124
 Information, Unterstützung 155
 Browserfenster 157
 Suchen 156
 Informationskatalogzentrale 153
 Journal 142
 Benachrichtigungsprotokoll 144
 Datenbankprotokoll 143
 Nachrichten 144
 Taskprotokoll 142
 Konfigurationsassistent 158
 Lizenzzentrale 155
 Replikationszentrale 146
 Klickstartleiste 148
 Satellitensynchronisator 149
 Satellitenverwaltungszentrale 149
 Steuerzentrale 125
 Angepassten Ordner erstellen 132
 Filter 126
 Kontextmenü 128
 Optionen 133
 Sichten 127
 Taskzentrale 137
 Ergebnisse anzeigen 141
 Statistiken 141
 Task aufnehmen 138
 Taskkategorien 140
 Tools-Einstellungen 130
 Webtools 160

►H

Hierarchiestufe 61
 Historie von DB2 25
 hostname 234, 241
 hosts-Datei 227, 235
 Hostsprache 456
 Host-Variable 463, 465
 HP-UX 52
 HWM 428
 hybrides 61
 Hyperion 61

►I

IBM Content Manager 63
 IBM DB2 Information Integrator 62
 IBM DB2 Information Integrator for Content 62–63
 IBM DB2 OLAP 61
 IBM DB2 OLAP Server Analyser 61
 IBM Enterprise Information Portal 63
 IBM Informix IDS und XPS 62
 IDS 64
 Image Extender 57
 Image Plus 63
 IMPORT
 COMMITCOUNT 336
 COMPOUND 338
 Message-Datei 337
 mit der Steuerzentrale 338
 Syntaxdiagramm 333
 Vergleich mit LOAD 365
 Wiederaufsetzpunkt 336
 Import 332, 473
 Verfahren 335
 IMS 23
 IN 513
 Index
 Typ-2 538
 INDEXREC 364
 Indikator-Variable 465
 Indizes 284
 Clustered Index 289
 CLUSTERFACTOR 292
 CLUSTERRATIO 292
 CREATE INDEX Befehl 286
 DESCRIBE INDEXES Befehl 293
 Eindeutiger Index (unique index) 289
 RENAME INDEX Befehl 286
 Satzkennung (RID) 285, 290
 Industriestandard Predictive Model Markup Language 63
 Information Catalog 60–61
 Information Catalog Center 61
 Information, Unterstützung siehe GUI, Information, Unterstützung
 Informationskatalogzentrale siehe GUI, Informationskatalogzentrale

Informative Parameter 383
 Informix 454, 459
 Informix Dynamic Server 64
 Inkonsistenzen 505
 Inkrementelle Sicherung 387
 IN-Lock 513
 INSERT
 CREATE 336
 INSERT 335
 INSERT_UPDATE 335
 REPLACE 335
 REPLACE_CREATE 335
 INSPECT 545
 Ausgabe 548
 Beispiele 547
 Installation 88
 Benutzer für DB2 einrichten 102
 Multifixpack-Installation 118
 Vorgehensweise 120
 Verzeichnisse unter Linux 94
 Verzeichnisse unter Windows 94
 Instance siehe Exemplar
 INSTANCE_MEMORY 618
 INTENT 515
 Intent Exclusive 514
 Intent None 513
 Intent Share 513
 INTO
 Host-Variable 475
 IP-Adresse 227, 235, 237
 IS 513, 515
 IS-Lock 513
 ISOLATION 473
 Isolation-Levels 509
 Iterator 475, 479
 Iterator-Deklaration 476
 Iterator-Typen 475
 IX 514–515
 IXF 327

►J

JAVA 456, 461
 Java 460
 java BuchDB_S_SQLJ 484
 java.util.Date 475
 javac 483
 Java-Precompiler-Fehler 475
 Java-Runtime-Fehlermeldung 484
 JDBC-Schnittstelle 498
 JDBC-Treiber 477
 JDBC-Type-4-Treiber 477
 Journal siehe GUI, Journal

►K

Knoten siehe Datenbankpartitionsgruppe
 Knotenname 225
 Knotenverzeichnis 225, 242
 administratives Knotenverzeichnis 230

Konfigurationsassistent 231, 251
 siehe GUI, Konfigurationsassistent
Konkurrierende Verarbeitung 505
Konnektoren 60
Kontrolldateien 402

► L

Lade-Tool siehe Load
LDAP (Lightweight Directory Access Protocol) 266
Linguistische Suche 57
Linux (zSeries) 53
LIST ACTIVE DATABASES 173
LIST ADMIN NODE DIRECTORY 230
LIST APPLICATIONS 173
LIST DATABASE DIRECTORY 215, 217
LIST HISTORY 403, 410
LIST INDOUBT TRANSACTIONS 258
LIST NODE DIRECTORY 225, 243
 SHOW DETAIL 264
list packages for all 502
LIST TABLES Befehl 280
LIST TABLESPACES 200
Lizenzmodell 45
Lizenzzentrale siehe GUI, Lizenzzentrale
LOAD 450
 Backup 362
 Fehler 363
 INSERT 353
 MESSAGES 353
 mit der Steuerzentrale 344
 Phasen des Ladevorgangs 341
 REPLACE 353
 RESTART INTO 363
 Syntaxdiagramm 352
 TERMINATE 364
 Vergleich mit IMPORT 365
 Vorbereitung 342
Load 341
LOAD QUERY
 Syntaxdiagramm 354
LOAD QUERY TABLE 355
Local Database Directory siehe lokales
 Datenbankverzeichnis
LOCK_LIST 522
Lock-Eskalation 523
LOCKTIMEOUT 520, 522
Lock-Timeout 519
LOGBUFSZ 382
logfiles 369
LOGFILSIZ 382
LOGPATH 372
logpath 377
LOGPRIMARY 374, 381
LOGRETAIN 380
Logretain 373
LOGRETAIN=ON 421
LOGSECOND 381

Lokales Datenbankverzeichnis 217
Lost Update 506, 508
Lotus Domino.Doc 63

► M

materialized query table (MQT) 277
MAXLOCKS 522–523
Media-Header 402
Mehrdimensionale Cluster-Tabelle
 (MDC) 311
 Dimensionen 314
Memory Visualizer 554, 590
METHOD L 334
METHOD N 331, 334
METHOD P 334
Microsoft SQL Server 62
Microsoft Visual C++ 6.0 461
migrate 626
Migration 623
 Ablauf 623
 Durchführung 625
 Kompatibilität 623
 Migrationsreihenfolge 624
 Nachbereitung 627
 Vorbereitung 623
MINCOMMIT 382
Mining-Algorithmen 63
Mini-RDBMS 48
MIRRORLOGPATH 382
mirrorlogpath 377
Mobile Computing 47
Modellierung 26
Mögliche Anomalien 510
Momentaufnahmen-Monitor 566
mon_heap_sz 556
Monitor für die Ereignisverfolgung
 (Event-Monitor) 554
Monitor für Momentaufnahmen (Snapshot-
 Monitor) 554
MSA (Management Services Architecture) 43
MSU 55
multidimensional 61
Multiplattform-Tools 64
Multiplexing-Programm 52

► N

Nächste zu lesende Protokolldatei 425
Name des Servers 477
Namenskonvention 390
Net Search Extender 50, 56
NEWLOGPATH 382, 420
Next 476
Next Key Share 515
Next Key Weak Exclusive 515
Nicht aktive Protokolldateien 372
Nicht wiederholbare Lesevorgänge 506–507
Nicht-Embedded-Schnittstellen 495
Nicht-Embedded-SQL 456–457

Nicht-Select-Anweisung 490
 NO ACTION
 describe indexes 296
 Node Directory siehe Knotenverzeichnis
 NOLINEMACRO 471
 Non-Select 490–491
 NOROWWARNINGS
 NOT LOGGED INITIALLY 371
 notification log 541
 siehe Benachrichtigungsprotokoll
 NOTIFYLEVEL 541
 NS 515
 NS-Lock 515
 NULL 465, 467
 NW 515
 NW-Lock 515

► O

Obere Grenze 428
 ODBC (Open Database Connectivity) 236
 ODBC-Schnittstelle 496
 Offline-Sicherung 386
 OLAP-Metadaten-Katalog 61
 OLAP-Mining 61
 OLAP-Technologie 61
 OLAP-Würfel 61
 OLE DB-Schnittstelle 497
 Online-Sicherung 386
 OPEN Cursor1 466
 Optimizer 458
 Abfrage umschreiben 600
 geschätzter Aufwand 609
 Optimierungsklassen 602
 Optimierungsklassen einstellen 602
 Zugriffsplan optimieren 601
 Oracle 454, 459
 OVERFLOWLOGPATH 382

► P

PACKAGE 473
 package 458
 Pakete (packages) 318
 LIST PACKAGES Befehl 318
 Parallele Verarbeitung 505
 Parallelität 399
 Parametermarker 493
 Partitioning Feature 53
 Partitionsgruppen 182
 Passwortänderung 265
 PATH 461
 pckcachesz 601
 PCTFREE 538
 PDA 46–47
 PDE 56
 Performance 459, 599
 Personal Digital Assistant 46–47
 Pervasive Computing 47
 Phantom Reads 506

Phantom-Datensätze 506
 PING 242
 ping 235
 PLI 456
 PMML 63
 portNumber 481
 Portnummer 227, 477
 Positionierter Cursor 475
 Positionierter Iterator 475
 Prädikatausbildung 460
 Prädikaten 460
 Präfix 463
 Precompile 469, 481
 Precompiler 456, 461
 Optionen 470
 PREP 603
 prep 470
 Prepare 493
 Produktpalette 45
 Programme mit SQL-Anweisungen 454
 Programmende 467
 Programmierschnittstellen 453
 Programmierung 27, 453
 Hauptaufgabengebiete 453
 von dynamischem SQL 455
 von statischem SQL 455
 Protokolldatei
 siehe Datenbankpartitionsgruppe
 Protokollierung 369
 ausschalten siehe Userexit
 Prozessorbasierende Lizenzierung 52
 Prüfung auf Integritätsbedingung
 (check constraint) 293
 PUBLIC 437, 439, 476
 Pufferpool (Bufferpool) 315, 317, 619
 Anzahl 317
 CREATE BUFFERPOOL Befehl 316
 Größe 317
 IBMDEFAULTBP 315
 Seitengröße 316
 siehe Datenbankpartitionsgruppe

► Q

Quantil 528
 QUERY CLIENT 258
 query graph model 599
 Query Patroller 60
 QUERY STATUS 423
 QUERYOPT 473, 603
 query-rewrite 458
 QUIESCE DATABASE 209, 380, 398

► R

RAID-Systeme 209
 Read Only Cursor 465
 Read Stability 509
 realisiert 61
 REBIND 541

- RECOVERY 380
- redirected restore 427
- Referenzielle Integrität (referential integrity) 295
 - Kochbuch RI 299
- Registrierte Benutzer 51–52
- Relationale Extender 56
- Relationale Quelldaten 61
- reorg 534
 - inplace 534, 537
 - Syntaxdiagramm reorgindexes 539
 - Syntaxdiagramm reorgtable 536
- REORG INDEX 627
- Reorganisation
 - Indizes 537
 - offline 536
 - online 536
 - Tabelle 534
- REORGCHK 531, 540
 - Ausgabe 533
 - Syntaxdiagramm 532
- Repeatable Read 509
- REPLACE EXISTING 420
- Replikation 146
- Replikationszentrale siehe GUI, Replikationszentrale
- Replizieren 48
- Reporting-Lösungen 61
- reset monitor 562
- Response-File (Antwortdatei) 71
 - Beispiel 91
- RESTORE DATABASE 417, 419
- resync_interval 258
- REVOKE 447
- ROLLBACK 518
- ROLLFORWARD 408
- Row Lock Modes 515
- RR 509
- RS 509
- RUNSTATS 525–526, 540
 - Syntaxdiagramm 530
- Run-Time Client 42
- **S**
- SAP/3 60
- Satellitenverwaltungszentrale siehe GUI, Satellitenverwaltungszentrale
- Schema 267
 - CREATE SCHEMA Befehl 268
 - explizit 268
 - implizit 268
 - NULLID 267
 - SESSION 277
 - SYSCAT 267
 - SYSIBM 267
 - SYSSTAT 267
- Schreibkonflikte 521
- Seiten (Speicherseiten) 185
- Select
 - mit bekannter Spaltenliste 491
 - mit unbekannter Spaltenliste 491
- SELECT PUBLIC 476
- Self Managing And Resource Tuning 64
- SERVER_ENCRYPT 221
- Serverprofil 247
 - erstellen 249
 - importieren 249
- Servicename 227
- Services-Datei 227, 229, 241, 247
- SET CLIENT 258
- SET CURRENT QUERY OPTIMIZATION 603
- SET EVENT MONITOR 571
- SET INTEGRITY 359
- SET TABLESPACE CONTAINERS 430
- setDefaultContext (ctx) 478
- SETJAVA.BAT 461
- setup 72
- Share 514–515
- Share with Intent Exclusive 514
- Sicherheit 459, 498
 - bei Verwendung von Embedded Statischen SQL 499
- Sicherungsassistent 395
- Sicherungskonzepte 385
- Sicherungstyp und Verfügbarkeit 398
- Sichten (Views) 282
 - CREATE VIEW Befehl 283
 - unbrauchbar Markierung 283
- SIX 514–515
- S-Lock 465, 513–515
- SMART-Funktionen 64
- SMS 189
 - Behälter 194
- Snapshot-Monitor 556
 - Informationsgruppen 562
 - mit Tabellen 565
 - Schalter (switches) 556
 - Schalter für Monitoraufzeichnungen 557
 - über Tabellenfunktionen 563
- SOFTMAX 371
- Solaris 52
- SORTHEAP 573
- Sortierreihenfolge (collating sequence) 164
- Spaltenbegrenzung siehe COLDEL
- Speicherbereich 184
- Speicherseiten, Größe 186
- Sperrflags 513
- SQL 502
- SQL communication area 466
- SQL Fehler -805 502
- SQL Fehler -818 502
- SQL3-Standard 453
- SQL92 599

SQL-Befehl
 DCL 454
 DDL 454
 DML 454
 SQLCA 42, 463, 486
 sqlca.h 463
 SQLCODE 465
 SQLCODE == 100 466
 SQL-Compiler 599–600
 SQLDA 42, 486
 sqlda 463
 sqlda.h 463
 SQL-Ergebnis 460
 SQL-Fehler -805 502
 SQL-Fehler Code_818 469
 SQL-Header-Files 461
 sqlj 482
 SQL-Java-Klassen 473, 475
 SQLJ-Precompiler 481, 483
 SQLJ-Profiles 456
 SQLJ-Übersetzer 456
 SQLLIB 79
 SQL-Performance 621
 SQL-Schnittstelle 454
 SQL-Server 454, 459
 SQL-Statement 458
 Star Schema 61
 stat_heap_sz 529
 Statisch 460
 Statisches SQL 459, 603
 Statisches SQL-Programm 462
 Steuerzentrale 125
 Stored Procedure
 siehe Gespeicherte Prozeduren
 Structured Query Language 457
 Super Exclusive 514
 SVCENAME 229, 241
 Sybase 62, 454, 459
 Syntax des Bind-Befehls 472
 Syntaxdiagramm des DB2 SQLJ Profile
 Customizer 484
 SYSADM 441, 449
 SYSCAT.BUFFERPOOLS 316
 SYSCAT.COLUMNS 270, 281
 SYSCAT.INDEXES 286, 292
 SYSCAT.ROUTINES 309
 SYSCAT.TABLES 270, 280
 SYSCAT.VIEWS 282
 SYSCtrl 450
 SYSFUN 267
 SYSIBM.SYSPLAN 601
 SYSMAINT 450
 SYSPROC 267
 SYSSTAT 267
 System R 25
 Systemdatenbankverzeichnis 215, 224
 Systemkatalog
 Statistiktabellen 525

► T

Tabelle (Table) 269
 ALTER TABLE Befehl 275
 ALTER TABLE-Befehl 298
 CREATE TABLE Befehl 312
 CREATE TABLE Syntax 274
 DECLARE GLOBAL TEMPORARY
 TABLE 276
 Fremdschlüssel 297
 gespeicherte Abfragetabelle (MQT) 277
 LIKE Option 275
 PCTFREE Option 291
 RENAME TABLE Befehl 276
 Temporäre Tabellen 276
 Tabellenbereich 183
 Administration 203
 ALTER TABLESPACE Befehl 203
 Aufbau 183
 DMS 190
 LARGE 192
 REGULAR 192
 siehe Datenbankpartitionsgruppe
 SMS 189
 Standard-Tabellenbereich 207
 SYSCATSPACE 183
 SYSTEM TEMPORARY 193
 TEMPSPACE1 183
 Umgekehrter Neuausgleich 205
 USER TEMPORARY 193
 USERSPACE1 183
 Verteilung 191
 Vorwärtsneuausgleich 205
 Tabellenbereichssicherung 387, 406
 Tabellenstatus 355
 TAKEN AT 419
 Taskzentrale siehe GUI, Taskzentrale
 TCP/IP 213, 227
 Ports 228
 Verbindung testen 242
 Teilsicherung 387
 TEMPFILES PATH siehe Load
 Teradata 62
 TERMINATE 174, 364
 Text Information Extender 57
 Threads 613
 Timeron 609
 Timestamp Error 502
 tm_database 258
 Toolskatalog 84
 Transaktionsorientierung 24
 Transaktionsserver 52
 Trends bei Programmiersprachen 455
 TRUST_ALLCLNTS 221–222
 TRUST_CLNTAUTH 221–222
 two-phase commit
 siehe Zweiphasige Festschreibung

►U

Überwachungs-Tools 553
UDE 56
U-Lock 464, 514–515
Umgeleitete Wiederherstellung 427
Umlaufprotokollierung 372
UNCATALOG-Befehl 245
Uncommitted Read 509–510
Unscharfe Suche 56
Updatable Cursor 464
Update 514–515
update adm cfg 114
UPDATE ALERT CFG 586
UPDATE DB CFG 179
update dbm cfg 111
update monitor switches 557
UPDATE-Anweisung 467
UR 509–510
url 477
User Defined Function
 siehe Benutzerdefinierte Funktion
User Defined Type
 siehe Benutzerdefinierter Datentyp
USEREXIT 381
Userexit 431
USEREXIT=ON 384, 421
USING LOCAL TIME 423

►V

Verschiedene Sperrsituationen 516
versions recovery 388
Verteilungstatistiken 528
Verwaltungsserver (DAS) 69
 Eigenschaften 69
 Tools-Datenbank 80
Video Extender 57
Visual Explain 603
Vollsicherung 386
Volltext 56
Vorkompilieren 456

►W

WARNINGCOUNT siehe Load
Was 458
Weak Exclusive 515
WebSphere MQ 62
Weitergabeklasse 382
Welche Datenbankobjekte werden
 gesperrt 512
Wert NULL 466–467
Wertzuweisung
 für die Select-Anweisung 491
 für die Update-Anweisung 493
while-Schleife 466
Wie 458
Wiederherstellung 407

Wiederherstellungskonzepte 388
Windows
 Startmenü 123
Windows 2000 49
Windows 9x 51
Windows ME 51
Windows NT 51
Windows XP 49, 51
Windows.Net Server 51
Wirtsprogramm 471
Wirtssprache 456
 C 461–462, 464
 Java 473
Wirtsvariablen 463
WITH SNAPSHOT 604
WITHOUT PROMPTING 421
WITHOUT ROLLING FORWARD 407, 420
W-Lock 515
Workflow-Designer 63
Workgroup Server Unlimited Edition 52
Wrapper 260
WSF 327
WSUE 51–52

►X

X 514
X-Lock 512, 514
XML-Extender 50

►Z

Z 514
Zeichenfolgebegrenzung
 siehe CHARDEL
Zeit in ISO 422
Zeitaufwand 458
Zugriff auf nicht bestätigte Daten 508
Zugriffsplan 458
Zugriffsrechte 437
 CONTROL-Recht 440
 für Datenbanken 438
 für Datenbankobjekte 440
 Indexrecht 441
 Paketrechte 441
 Schemarechte 440
 Sichtenrechte 441
 Systemkatalogsichten 448
 Tabellenbereichrecht 440
 Tabellenrechte 441
 Verwaltung mit der Steuerzentrale 442
 Verwaltung über Befehle 444
Zusammengeschlossene Datenbanken
 siehe Federated Database
Zweiphasige Festschreibung (two-phase
 commit) 256
Aktivierung 258



Das Forum für kollaborative IT-Systeme in der Praxis

Leistungen und Angebote für Mitglieder

- ⇒ Organisation von Konferenzen mit internationalen Top-Referenten, zahlreichen Workshops und begleitender Fachausstellung
- ⇒ Special Interest Groups mit Fachvorträgen, Diskussion und Erfahrungsaustausch zu den Themen:
 - Anwendungsentwicklung
 - System-Management
 - Methoden und Strategien
 - IBM Business Partner
 - Knowledge Management
 - E-Learning
 - Mobile Datenkommunikation
 - Enterprise Integration
 - WebSphere
 - Kunden und interne Dienstleister
 - Regionale Koordinatoren
 - Business Design
- ⇒ Informations- und Gesprächsangebote im Internet u.a. mit den Präsentationen und Protokollen der Veranstaltungen unter <http://DNUG.DE> bzw. <http://DNUG.DE/Forum>
- ⇒ Hochschulinitiative zur Förderung des Notes- und WebSphere-Nachwuchses: Veranstaltungen in Kooperation mit Hochschulen und IBM; Diplomarbeiten-Wettbewerb zu den Themen Lotus Domino/Notes, WebSphere und begleitenden Technologien
- ⇒ Halbjährliche Ausschreibung des DNUG-AWARDS für beispielhafte Anwendungen/Projekte auf der Basis von Lotus Domino/Notes bzw. IBM WebSphere
- ⇒ Wirksame Vertretung der Mitgliederinteressen und Adressierung aktueller Anwenderforderungen durch kontinuierliche Kontakte zur IBM Software Group

DNUG-Mitgliedschaft

In der DNUG bestehen Mitgliedschaften für Institutionen, Hochschulen und Privatpersonen. Für Studenten wird die Personenmitgliedschaft zu besonders günstigen Konditionen angeboten. Es sind ca. 500 Mitglieder und Mitgliedsunternehmen in der DNUG organisiert; insgesamt etwa 1.200 aktive Groupware-Spezialisten (Stand Januar 2004).

Informationen

DNUG e. V., Berggasse 1, D-07745 Jena

Tel.: +49 (0)3641 / 4569 - 0, Fax: +49 (0)3641 / 4569 - 15

E-Mail: Info@DNUG.DE, Internet: <http://DNUG.DE>



Copyright

Daten, Texte, Design und Grafiken dieses eBooks, sowie die eventuell angebotenen eBook-Zusatzdaten sind urheberrechtlich geschützt.

Dieses eBook stellen wir lediglich als **Einzelplatz-Lizenz** zur Verfügung!

Jede andere Verwendung dieses eBooks oder zugehöriger Materialien und Informationen, einschliesslich der Reproduktion, der Weitergabe, des Weitervertriebs, der Platzierung im Internet, in Intranets, in Extranets anderen Websites, der Veränderung, des Weiterverkaufs und der Veröffentlichung bedarf der schriftlichen Genehmigung des Verlags.

Bei Fragen zu diesem Thema wenden Sie sich bitte an:

<mailto:info@pearson.de>

Zusatzdaten

Möglicherweise liegt dem gedruckten Buch eine CD-ROM mit Zusatzdaten bei. Die Zurverfügungstellung dieser Daten auf der Website ist eine freiwillige Leistung des Verlags. Der Rechtsweg ist ausgeschlossen.

Hinweis

Dieses und andere eBooks können Sie rund um die Uhr und legal auf unserer Website



(<http://www.informit.de>)

herunterladen