IBM DB2 Universal Database

Systemverwaltung: Optimierung

Version 7

SC12-2877-01

IBM DB2 Universal Database

Systemverwaltung: Optimierung

Version 7

SC12-2877-01

Anmerkung:

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Anhang F. Bemerkungen" auf Seite 723 gelesen werden.

- Die IBM Homepage finden Sie im Internet unter: **ibm.com**
- IBM und das IBM Logo sind eingetragene Marken der International Business Machines Corporation.
- Das e-business Symbol ist eine Marke der International Business Machines Corporation
- Infoprint ist eine eingetragene Marke der IBM.
- ActionMedia, LANDesk, MMX, Pentium und ProShare sind Marken der Intel Corporation in den USA und/oder anderen Ländern.
- C-bus ist eine Marke der Corollary, Inc. in den USA und/oder anderen Ländern.
- Java und alle Java-basierenden Marken und Logos sind Marken der Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- Microsoft Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.
- PC Direct ist eine Marke der Ziff Communications Company in den USA und/oder anderen Ländern.
- SET und das SET-Logo sind Marken der SET Secure Electronic Transaction LLC.
- UNIX ist eine eingetragene Marke der Open Group in den USA und/oder anderen Ländern.
- Marken anderer Unternehmen/Hersteller werden anerkannt.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Universal Database Administration Guide: Performance,
IBM Form SC09-2945-01,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
August 2001

Inhaltsverzeichnis

Zu diesem Handbuch ix
Zielgruppe x
Aufbau dieses Handbuchs x
Kurzübersicht über die anderen Bände des Handbuchs zur Systemverwaltung xii
Systemverwaltung: Konzept xii
Systemverwaltung: Implementierung . . xiii

Teil 1. Einführung in die Optimierung 1

Kapitel 1. Elemente der Leistung 3
Richtlinien für die Optimierung 4
Plattenspeicher 5
Verfahren zur Leistungsverbesserung 6
Wie viel Optimierung verträgt ein System? . . 7
Ein weniger formaler Ansatz 8
Zusammenfassen aller Aspekte 8

Kapitel 2. Übersicht zu Architektur und Prozessen 11
Speicherarchitektur. 16
Datenbankverzeichnis 16
Tabellenbereiche 18
Datenverwaltung 23
Satz-IDs (RIDs) und Seiten 24
Speicherbereichsverwaltung 25
Indexverwaltung 28
Sperren 29
Protokollierung 31
Vorgänge bei der Aktualisierung 32
Prozessmodell 34
Speichermodell 40

Teil 2. Optimieren der Anwendungsleistung 45

Kapitel 3. Überlegungen zu Anwendungen 47
Gemeinsamer Zugriff 47
Wiederholtes Lesen (RR) 49
Lesestabilität (RS) 50
Cursorstabilität (CS) 51
Nicht festgeschriebener Lesevorgang (UR) 52
Wählen der Isolationsstufe 53
Angeben der Isolationsstufe 54
Deklarierte temporäre Tabellen und gemeinsamer Zugriff 57
Sperren 57
Attribute von Sperren 58
Sperren und die Leistung von Anwendungen 60
Faktoren mit Auswirkung auf Sperren . . 69
Deklarierte temporäre Tabellen und Sperren 73
Anweisung LOCK TABLE 73
Anweisung CLOSE CURSOR WITH RELEASE 75
Zusammenfassung der Überlegungen zu Sperren 75
Anpassen der Optimierungsklasse 76
Einstellen der Optimierungsklasse . . . 81
Wie viel Optimierung ist erforderlich? . . 82
Einschränkungen für Ergebnismengen zur Leistungsverbesserung 85
Klausel FOR UPDATE 86
Klausel FOR READ oder FETCH ONLY. . 86
Klausel OPTIMIZE FOR n ROWS 87
Klausel FETCH FIRST n ROWS ONLY . . 89
Anweisung DECLARE CURSOR WITH HOLD 89
Zeilenblockung 90
Optimieren von Abfragen 92
Verwenden einer SELECT-Anweisung . . 92
Richtlinien zur Verwendung einer SELECT-Anweisung 92
Compound-SQL-Anweisungen. 95
Dynamische Compound-Anweisungen . . . 96
Informationen zur Leistung und Zeichenumsetzung 96
Umsetzung von Codepages 97
Codepage-Unterstützung für Erweiterten UNIX-Code (EUC) 98
Gespeicherte Prozeduren. 98
Aktivieren einer Datenbank 100
Parallele Verarbeitung von Anwendungen 100

Kapitel 4. Überlegungen zur Umgebung 103

Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung 103
Auswirkung der Knotengruppe auf die Abfrageoptimierung 107
Auswirkung des Tabellenbereichs auf die Abfrageoptimierung 107
Auswirkung des Indexierens auf die Abfrageoptimierung 111
Tabellen mit und ohne Index 111
Verwenden von Index Advisor 113
Verwenden von größeren Indexschlüsseln 113
Richtlinien zur Erstellung von Indizes . . 114
Hinweise zur Leistung für die Verwaltung von Indizes 117
Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken 121

Kapitel 5. Systemkatalogstatistiken . . . 129
Erfassen statistischer Daten mit dem Dienstprogramm RUNSTATS 131
Die Datenbankpartition, auf der RUNSTATS ausgeführt wird 132
Analysieren der Statistikdaten 132
Erfassen und Verwenden von Verteilungsstatistiken 140
Beschreibung der Verteilungsstatistiken 141
Wann sollten Verteilungsstatistiken verwendet werden? 143
Wie viele Statistikdaten sollten erfasst werden? 144
Wie verwendet das Optimierungsprogramm die Verteilungsstatistiken? . . 146
Erfassen und Verwenden detaillierter Indexstatistikdaten 151
Zweck detaillierter Indexstatistikdaten 152
Wann sollten detaillierte Indexstatistiken verwendet werden? 154
Benutzeraktualisierbare Katalogstatistiken 154
Regeln zur Aktualisierung von Katalogstatistiken 156
Regeln zur Aktualisierung von Kurznamenstatistiken 157
Regeln zur Aktualisierung von Spaltenstatistiken 158
Regeln zur Aktualisierung von Verteilungsstatistiken für Spalten. . . . 159
Regeln zur Aktualisierung von Indexstatistiken 160
Aktualisieren der Statistiken für benutzerdefinierte Funktionen 162
Modellieren im Einsatz befindlicher Datenbanken 164
Unterelementstatistiken 167

Kapitel 6. Der SQL-Compiler 171
Übersicht über den SQL-Compiler 172
Umschreiben der Abfrage durch den SQL-Compiler. 176
Mischen von Operationen 176
Verschieben von Operationen 180
Übersetzen von Vergleichselementen . . 182
Behandeln von Spaltenkorrelation 184
Datenzugriffskonzepte und Optimierung . . 187
Konzepte der Indexsuche 188
Tabellensuche und Indexsuche 200
Vergleichselementterminologie 200
Verknüpfungskonzepte 203
Replizierte Übersichtstabellen 213
Verknüpfungsstrategien in einer partitionierten Datenbank 217
Einfluss des Sortierens auf das Optimierungsprogramm 225
Optimierungsstrategien für partitionsinterne Parallelität 227
Strategien für paralleles Suchen 228
Strategien für paralleles Sortieren . . . 229
Parallele temporäre Tabellen 230
Strategien für parallele Spaltenberechnung 230
Parallele Verknüpfungsstrategien. . . . 230
Automatische Übersichtstabellen. 231
Abfragecompilerphasen für zusammengeschlossene Datenbanken 234
Pushdown-Analyse 234
Generierung von fernem SQL und globale Optimierung 244

Kapitel 7. Die SQL-EXPLAIN-Einrichtung 251
Auswählen eines EXPLAIN-Programms . . 252
Verwenden der SQL-EXPLAIN-Einrichtung 254
Grundkonzepte für EXPLAIN. 257
EXPLAIN-Informationen für Datenobjekte 258
EXPLAIN-Informationen für Datenoperatoren 259
Organisation von EXPLAIN-Informationen 260
Informationen über EXPLAIN-Exemplare 261
Informationen der EXPLAIN-Momentaufnahmen 264
Informationen der EXPLAIN-Tabellen . . 265
Abrufen von EXPLAIN-Daten. 267

Erfassen von EXPLAIN-Tabelleninformationen 267
Erfassen von EXPLAIN-Momentaufnahmedaten 269
Richtlinien zur Verwendung der EXPLAIN-Ausgabe 271
Visual Explain 273
SQL-Advise-Einrichtung 274

Teil 3. Optimieren und Konfigurieren des Systems. 281

Kapitel 8. Leistung bei der Ausführung 283
Verwendung des Speichers durch DB2 . . . 283
Einstellen von Parametern mit Auswirkung auf die Speicherbelegung 290
FCM-Anforderungen. 291
Verwalten des Datenbankpufferpools . . . 292
Nutzung großer Speicher bei Windows-Systemen 293
Funktionsweise von Pufferpool-Seiten . . 295
Verwalten mehrerer Datenbankpufferpools 298
Auswählen eines oder mehrerer Pufferpools 300
Vorablesen von Daten in den Pufferpool . . 301
Sequenzieller Vorablesezugriff 301
Vorablesezugriff über Listen 303
Vorablesezugriff und partitionsinterne Parallelität 304
Konfigurieren von E/A-Servern für Vorablesezugriff und parallele E/A 304
Aktivieren paralleler E/A 306
Gleichzeitiges Zuordnen mehrerer Seiten 309
Sortieren. 309
Verschiedene Arten der Sortierung . . . 309
Optimieren der Parameter mit Auswirkung auf Sortierungen 310
Anzeichen für Probleme bei der Sortierleistung 310
Techniken zur Optimierung der Sortierleistung 312
Reorganisieren von Katalogen und Benutzertabellen 313
Online-Indexreorganisation 316
Verringern der Notwendigkeit, Tabellen zu reorganisieren 317
Überlegungen zur Leistung bei DMS-Einheiten. 318
Verwalten des Initialisierungsaufwands . . 319
Datenbankagenten 320
Verwenden des Datenbanksystemmonitors 328
Erweitern von Speicher 330

Kapitel 9. Verwenden von Governor . . . 333
Starten und Stoppen von Governor 334
Der Governor-Dämon 336
Erstellen der Governor-Konfigurationsdatei 337
Governor-Protokolldateien 346
Abfragen auf Governor-Protokolldateien . . 348
Ausführen von Governor und Leistung des Datenbankmanagers 348

Kapitel 10. Skalieren der Konfiguration über das Hinzufügen von Prozessoren . . 349
Hinzufügen von Prozessoren zu einer Maschine 350
Hinzufügen von Datenbankpartitionen zu einem System mit partitionierten Datenbanken 351
Hinzufügen von Datenbankpartitionen zu einem aktiven System 352
Hinzufügen von Datenbankpartitionen zu einem gestoppten System 354
Löschen einer Datenbankpartition von einem System 357
Probleme beim Hinzufügen von Knoten zu einer partitionierten Datenbank 359

Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen 363
Partitionierung von Daten 364
Hinzufügen und Entfernen von Datenbankpartitionen 365
Angeben einer Zielpartitionierungszuordnung 365
Verfahren der Datenumverteilung auf Datenbankpartitionen 366
Verfahren der Datenverteilung in Tabellen 367
Beheben von Fehlern bei der Umverteilung 369
Datenumverteilung und andere Operationen 370
Abschließen der Datenumverteilung . . . 370

Kapitel 12. Durchführen von Vergleichstests 371
Methoden für Vergleichstests 372
Vorbereiten von Vergleichstests 373
Erstellen eines Vergleichstestprogramms . . 375
Ausführen der Vergleichstests. 382

Kapitel 13. Konfigurieren von DB2 . . . 387
Optimieren der Konfigurationsparameter 388
Parameter des Datenbankmanagers 389
Überblick über die Konfigurationsparameter des Datenbankmanagers . . . 390
Datenbankparameter 396
Überblick über die Konfigurationsparameter der Datenbank 398
Einzelheiten zu Parametern nach Funktion 403
Kapazitätsverwaltung 404
Gemeinsam benutzter Speicher der Datenbank 404
Gemeinsamer Anwendungsspeicher. . . 420
Privater Agentenspeicher 422
Agenten-/Anwendungskommunikationsspeicher 436
Exemplarspeicher des Datenbankmanagers 443
Sperren 448
Ein-/Ausgabe und Speicher 453
Agenten 461
Gespeicherte Prozeduren (DARI). . . . 474
Protokollieren und Wiederherstellung . . . 478
Datenbankprotokolldateien 478
Datenbankprotokollierung 486
Wiederherstellung. 492
Wiederherstellung der verteilten Arbeitseinheit 499
Datenbankverwaltung 504
Query Enabler 504
Attribute. 505
DB2 Data Links Manager 508
Status 511
Compilereinstellungen 513
Übertragung 521
Konfiguration der Übertragungsprotokolle 521
Verteilte Services 526
DB2 Discovery 531
Partitionierte Datenbank 535
Übertragung 535
Parallelverarbeitung 541
Exemplarverwaltung. 544
Diagnose. 544
Parameter des Datenbanksystemmonitors 547
Systemverwaltung 548
Exemplarverwaltung. 557

Teil 4. Anhänge und Schlussteil 569

Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen 571

Anhang B. EXPLAIN-Tabellen und Definitionen 609
Die Tabelle EXPLAIN_ARGUMENT . . . 610
Die Tabelle EXPLAIN_INSTANCE 614
Die Tabelle EXPLAIN_OBJECT 617
Die Tabelle EXPLAIN_OPERATOR 620
Die Tabelle EXPLAIN_PREDICATE 622
Die Tabelle EXPLAIN_STATEMENT . . . 624
Die Tabelle EXPLAIN_STREAM 627
Die Tabelle ADVISE_INDEX 629
Die Tabelle ADVISE_WORKLOAD 633
Tabellendefinitionen für EXPLAIN-Tabellen 634
Tabellendefinition EXPLAIN_ARGUMENT 634
Tabellendefinition EXPLAIN_INSTANCE 635
Tabellendefinition EXPLAIN_OBJECT . . 636
Tabellendefinition EXPLAIN_OPERATOR 637
Tabellendefinition EXPLAIN_PREDICATE 638
Tabellendefinition EXPLAIN_STATEMENT 639
Tabellendefinition EXPLAIN_STREAM 640
Tabellendefinition ADVISE_INDEX . . . 641
Tabellendefinition ADVISE_WORKLOAD 643

Anhang C. EXPLAIN-Programme (SQL) 645
Ausführen von db2expln und dynexpln . . 646
Syntax und Parameter für db2expln. . . . 646
Hinweise zur Verwendung von db2expln 649
Syntax und Parameter für dynexpln . . . 651
Hinweise zur Verwendung von dynexpln 653
Beschreibung der Ausgabe von db2expln und dynexpln 654
Zugriff auf Tabellen 656
Temporäre Tabellen 661
Verknüpfungen 665
Datenströme 667
Einfügen, Aktualisieren und Löschen . . 668
Vorbereitung von Satzkennungen (Satz-IDs) 668
Spaltenberechnungen 669
Parallelverarbeitung 670
Verarbeitung von Anweisungen in zusammengeschlossenen Datenbanken 673
Verschiedene Angaben 674
Beispiele für die Ausgabe von db2expln und dynexpln 677
Beispiel 1: Plan ohne Parallelität 677

Beispiel 2: Zugriffsplan für Datenbank mit einer einzelnen Partition und partitionsinterner Parallelität 680
Beispiel 3: Zugriffsplan für eine Datenbank mit mehreren Partitionen und partitionsübergreifender Parallelität . . . 683
Beispiel 4: Zugriffsplan für die Datenbank mit mehreren Partitionen sowie partitionsinterner und partitionsübergreifender Parallelität 687
Beispiel 5: Zugriffsplan für eine zusammengeschlossene Datenbank 692

Anhang D. db2exfmt - EXPLAIN-Tool für Tabellenformat 695

Anhang E. Verwenden der DB2-Bibliothek 699
PDF-Dateien und gedruckte Bücher für DB2 699
Informationen zu DB2 699
Drucken der PDF-Handbücher 712
Bestellen der gedruckten Handbücher . . 713
DB2-Online-Dokumentation 715
Zugreifen auf die Online-Hilfefunktion 715
Anzeigen von Online-Informationen . . 717
Verwenden der DB2-Assistenten 719
Einrichten eines Dokument-Servers . . . 721
Suchen nach Online-Informationen . . . 722

Anhang F. Bemerkungen 723
Marken 726

Index 729

Kontaktaufnahme mit IBM 747
Produktinformationen 747

Zu diesem Handbuch

Das Handbuch zur Systemverwaltung bietet in seinen drei Bänden Informationen, die zur Verwendung und Verwaltung der Jahr 2000-konformen DB2-Produkte für Verwaltungssysteme für relationale Datenbanken (RDBMS) benötigt werden:

- Informationen zum Datenbankentwurf (im Band *Systemverwaltung: Konzept*)
- Informationen zur Implementierung und Verwaltung von Datenbanken (im Band *Systemverwaltung: Implementierung*)
- Informationen zur Konfiguration und Optimierung der Datenbankumgebung zum Zweck der Leistungsverbesserung (im Band *Systemverwaltung: Optimierung*).

Viele der in diesem Handbuch beschriebenen Operationen können mit Hilfe verschiedener Schnittstellen durchgeführt werden:

- Der **Befehlszeilenprozessor** ermöglicht Ihnen den Zugriff auf Datenbanken und deren Bearbeitung über eine grafische Schnittstelle. Von dieser Schnittstelle aus können Sie SQL-Anweisungen und DB2-Dienstprogrammfunktionen ausführen. Die Mehrzahl der Beispiele in diesem Handbuch zeigt die Verwendung dieser Schnittstelle. Weitere Informationen zur Verwendung des Befehlszeilenprozessors finden Sie im Handbuch *Command Reference*.
- Die **Anwendungsprogrammierschnittstelle** ermöglicht Ihnen die Ausführung von DB2-Dienstprogrammfunktionen innerhalb eines Anwendungsprogramms. Weitere Informationen zur Verwendung der Anwendungsprogrammierschnittstelle finden Sie im Handbuch *Administrative API Reference*.
- Die **Steuerzentrale** ermöglicht Ihnen die Ausführung von Verwaltungsaufgaben, z. B. das Konfigurieren des Systems, die Verwaltung von Verzeichnissen, das Sichern und Wiederherstellen des Systems, die zeitliche Festlegung von Jobs und die Verwaltung von Medien, über eine grafische Schnittstelle. Die Steuerzentrale enthält außerdem eine `Replikationsverwaltung`, mit der die Replikation von Daten zwischen den Systemen mit Hilfe einer grafischen Schnittstelle eingerichtet werden kann. Darüber hinaus ermöglicht die Steuerzentrale das Ausführen von DB2-Dienstprogrammfunktionen über eine grafische Benutzerschnittstelle. Ja nach Plattform gibt es unterschiedliche Möglichkeiten, die Steuerzentrale aufzurufen. Geben Sie z. B. den Befehl `db2cc` in der Befehlszeile ein, wählen Sie unter OS/2 das Symbol der Steuerzentrale im DB2-Ordner aus oder verwenden Sie bei Windows-Plattformen das Menü **Start**. Wenn Sie eine einführende Hilfe benöti-

gen, wählen Sie **Erste Schritte** im Menü **Hilfe** des Fensters der Steuerzentrale aus. **Visual Explain** und **Performance Monitor** werden von der Steuerzentrale aus aufgerufen.

Es gibt darüber hinaus noch weitere Tools, die Sie zur Durchführung von Verwaltungsaufgaben verwenden können. Dazu gehören:

- Die `Prozedurzentrale`, die zum Speichern kleiner Anwendungen dient, die als Prozeduren (Scripts) bezeichnet werden. Diese Prozeduren können sowohl SQL-Anweisungen als auch DB2-Befehle und Betriebssystembefehle enthalten.
- Die `Alert-Zentrale`, die zur Überwachung von Nachrichten dient, die sich aus anderen DB2-Operationen ergeben.
- Das Programm `Tools - Einstellungen`, mit dem Sie die Einstellungen für die Steuerzentrale, die Alert-Zentrale und die Replikation ändern können.
- Das `Journal`, das zur zeitlichen Terminierung von Jobs dient, die im nicht-überwachten Modus ausgeführt werden sollen.
- Die `Data Warehouse-Zentrale`, die zur Verwaltung von Warehouse-Objekten dient.

Zielgruppe

Dieses Handbuch ist hauptsächlich für Datenbankadministratoren, Systemadministratoren, Sicherheitsadministratoren und Systembediener gedacht, die eine Datenbank für den lokalen oder fernen Zugriff entwerfen, implementieren und pflegen müssen. Es wendet sich auch an Programmierer und andere Benutzer, die Kenntnisse über die Verwaltung und Bedienung des relationalen Datenbankverwaltungssystems von DB2 benötigen.

Aufbau dieses Handbuchs

Das vorliegende Handbuch enthält Informationen zu folgenden Hauptthemen:

Einführung in die Optimierung

- Kapitel 1. Elemente der Leistung, enthält eine Einführung in die Konzepte und Überlegungen zur Verwaltung und Optimierung der Leistung von DB2 UDB.
- Kapitel 2. Übersicht zu Architektur und Prozessen, stellt die zugrundeliegende Architektur und die Prozesse von DB2 Universal Database vor.

Optimieren der Anwendungsleistung

- Kapitel 3. Überlegungen zu Anwendungen, beschreibt einige Techniken zur Verbesserung der Datenbankleistung beim Entwurf der Anwendungen.
- Kapitel 4. Überlegungen zur Umgebung, beschreibt einige Techniken zur Verbesserung der Datenbankleistung bei der Einrichtung der Datenbankumgebung.
- Kapitel 5. Systemkatalogstatistiken, beschreibt einige Techniken zur Erfassung von Statistiken über die Daten und ihre Verwendung zur Gewährleistung einer optimalen Leistung.
- Kapitel 6. Der SQL-Compiler, beschreibt die Verarbeitung einer SQL-Anweisung durch den SQL-Compiler bei der Kompilierung.
- Kapitel 7. Die SQL-EXPLAIN-Einrichtung, beschreibt die Einrichtung EXPLAIN, die Ihnen ermöglicht, die Pfade und Methoden anzuzeigen, die der SQL-Compiler ausgewählt hat, um auf die Daten zuzugreifen.

Optimieren und Konfigurieren des Systems

- Kapitel 8. Leistung bei der Ausführung, gibt einen Überblick über die Verwendung von Speicher durch den Datenbankmanager und enthält Informationen zu weiteren Faktoren, die sich auf die Leistung zur Laufzeit auswirken.
- Kapitel 9. Verwenden von Governor, gibt eine Einführung in die Verwendung des Programms Governor, mit dem einige Aspekte der Datenbankverwaltung gesteuert werden können.
- Kapitel 10. Skalieren der Konfiguration über das Hinzufügen von Prozessoren, enthält einige Informationen und Hinweise, die bei der Erweiterung des Datenbanksystems von Bedeutung sind.
- Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen, behandelt die Punkte, die in einer Umgebung mit partitionierten Datenbanken bei der Neuverteilung von Daten auf die Partitionen zu beachten sind.
- Kapitel 12. Durchführen von Vergleichstests, gibt einen Überblick über Vergleichstests (Benchmark-Tests) und behandelt verschiedene Aspekte ihrer Durchführung.
- Kapitel 13. Konfigurieren von DB2, behandelt die Dateien zur Konfiguration des Datenbankmanagers und der Datenbanken sowie die Werte für die Konfigurationsparameter.

Anhänge

- Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen, enthält Werte für die Profilregistrierdatenbank und für Umgebungsvariablen.
- Anhang B. EXPLAIN-Tabellen und Definitionen, enthält Informationen zu den Tabellen, die von der DB2-EXPLAIN-Einrichtung verwendet werden, und beschreibt die Erstellung dieser Tabellen.
- Anhang C. EXPLAIN-Programme (SQL), enthält Informationen zur Verwendung der DB2-EXPLAIN-Programme: db2expln und dynexpln.
- Anhang D. db2exfmt - EXPLAIN-Tool für Tabellenformat, enthält Informationen dazu, wie der Inhalt von EXPLAIN-Tabellen formatiert wird.
- Anhang E. Verwenden der DB2-Bibliothek, enthält Informationen zu der Struktur der DB2-Bibliothek, einschließlich Assistenten, Online-Hilfefunktion, Nachrichten und Handbücher.

Kurzübersicht über die anderen Bände des Handbuchs zur Systemverwaltung

Systemverwaltung: Konzept

Der Band *Systemverwaltung: Konzept* behandelt den Datenbankentwurf. Er enthält Themen zum logischen und physischen Entwurf, zu verteilten Transaktionen und zur hohen Verfügbarkeit. Die einzelnen Kapitel und Anhänge des Bandes werden im Folgenden kurz vorgestellt:

DB2 Universal Database

- Das Kapitel zum Verwalten von DB2 Universal Database enthält eine Einführung sowie eine Übersicht zu DB2 Universal Database.

Datenbankkonzepte

- Das Kapitel zu allgemeinen Konzepten relationaler Datenbanken enthält eine Übersicht über Datenbankobjekte, einschließlich Wiederherstellungsobjekte, Speicherobjekte und Systemobjekte.
- Das Kapitel über Systeme zusammengeschlossener Datenbanken behandelt Datenbankverwaltungssysteme (DBMS), die Anwendungen und Benutzer unterstützen, die SQL-Anweisungen übergeben, die auf zwei oder mehr DBMSs oder Datenbanken in einer einzigen Anweisung verweisen.
- Das Kapitel über parallele Datenbanksysteme enthält eine Einführung in die Arten von Parallelität, die mit Hilfe von DB2 implementiert werden können.
- Das Kapitel über Data Warehouses enthält eine Übersicht über den Einsatz von Data Warehouses und Data Warehouse-Funktionen.
- Das Kapitel über Spatial Extender stellt Spatial Extender vor und enthält Erläuterungen zum Zweck des Produkts sowie zu den Daten, die von ihm verarbeitet werden.

Datenbankentwurf

- Das Kapitel zum Entwerfen des logischen Datenbankaufbaus behandelt die Konzepte und Richtlinien zum logischen Entwurf einer Datenbank.
- Das Kapitel zum Entwerfen der physischen Datenbank behandelt die Richtlinien zum physischen Entwurf einer Datenbank und enthält Überlegungen im Hinblick auf die Datenspeicherung.

Verteilte Transaktionsverarbeitung

- Das Kapitel zum Entwerfen verteilter Datenbanken beschreibt die Möglichkeit des Zugriffs auf mehrere Datenbanken in einer einzigen Transaktion.
- Das Kapitel zum Entwerfen für Transaktionsmanager behandelt die Verwendung von Datenbanken in einer Umgebung für verteilte Transaktionsverarbeitung wie CICS.

Systeme mit hoher Verfügbarkeit

- Die Einführung in die Unterstützung von Funktionsübernahmen zur Implementierung hoher Verfügbarkeit bietet eine Übersicht über die von DB2 bereitgestellte Unterstützung von Funktionsübernahmen zur Implementierung hoher Verfügbarkeit.

Anhänge

- Der Anhang zur Planung der Datenbankmigration enthält Informationen zur Migration von Datenbanken auf Version 7.
- Der Anhang zu Inkompatibilitäten zwischen Releases behandelt die entstandenen Inkompatibilitäten von Release zu Release bis einschließlich Version 7.
- Der Anhang zur Unterstützung von Landessprachen gibt eine Einführung in die DB2-Unterstützung von Landessprachen und enthält Informationen zu Ländern, Sprachen und Codepages.

Systemverwaltung: Implementierung

Der Band *Systemverwaltung: Implementierung* behandelt die Implementierung des entwickelten Datenbankentwurfs. Die einzelnen Kapitel und Anhänge des Bandes werden im Folgenden kurz vorgestellt:

Systemverwaltung mit der Steuerzentrale

- Das Kapitel zum Verwalten von DB2 mit GUI-Tools behandelt die Tools der grafischen Benutzerschnittstelle (GUI), die zur Verwaltung der Datenbank verwendet werden.

Implementieren des Datenbankentwurfs

- Das Kapitel über die Vorbereitungen für die Erstellung einer Datenbank behandelt die Voraussetzungen, die zur der Erstellung einer Datenbank erfüllt sein müssen.
- Das Kapitel zum Erstellen einer Datenbank behandelt die Aufgaben im Zusammenhang mit der Erstellung einer Datenbank sowie der zugehörigen Datenbankobjekte.
- Das Kapitel über das Ändern einer Datenbank behandelt, was vor der Änderung einer Datenbank zu tun ist und welche Aufgaben im Zusammenhang mit dem Ändern oder Löschen einer Datenbank oder zugehöriger Datenbankobjekte erledigt werden müssen.

Datenbanksicherheit

- Das Kapitel zur Steuerung des Datenbankzugriffs beschreibt die Möglichkeiten zur Steuerung des Zugriffs auf die Ressourcen einer Datenbank.
- Das Kapitel zur Protokollierung von DB2-Aktivitäten beschreibt Methoden zur Erkennung und Überwachung unerwünschten bzw. unvorhergesehenen Zugriffs auf Daten.

Versetzen von Daten

- Das Kapitel zu den Dienstprogrammen zum Versetzen von Daten ist eine kurze Einführung in die verschiedenen Möglichkeiten zum Versetzen von Daten, die Sie an das Handbuch *Versetzen von Daten Dienstprogramme und Referenz* verweist.

Wiederherstellung

- Das Kapitel zur Wiederherstellung bietet auf einer Seite eine Einführung in die Konzepte der Sicherung, Wiederherstellung und aktualisierenden Wiederherstellung von Datenbanken. Ausführlichere Informationen finden Sie in *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*.

Anhänge

- Der Anhang zur Verwendung der DCE-Verzeichnisservices enthält Informationen zu den Einsatzmöglichkeiten der DCE-Verzeichnisservices.
- Der Anhang über den Benutzer-Exit zur Datenbankwiederherstellung behandelt die Verwendung von Benutzer-Exit-Programmen für Datenbankprotokolldateien und beschreibt einige Beispiele von Benutzer-Exit-Programmen.

- Der Anhang über das Absetzen von Befehlen an mehrere Datenbankpartitions-Server behandelt die Verwendung der Shell-Prozeduren *db2_all* und *rah* zum Senden von Befehlen an alle Partitionen in einer Umgebung mit partitionierten Datenbanken.
- Der Anhang über die Arbeitsweise von DB2 für Windows NT mit der Windows NT-Sicherheit beschreibt, wie DB2 mit der Windows NT-Sicherheit funktioniert.
- Der Anhang zur Verwendung des Windows NT-Systemmonitors enthält Informationen über das Registrieren von DB2 beim Windows NT-Systemmonitor und über die Nutzung der Leistungsinformationen.
- Der Anhang zur Arbeit mit Datenbankpartitions-Servern unter Windows NT oder Windows 2000 enthält Informationen über die verfügbaren Dienstprogramme für die Arbeit mit Datenbankpartitions-Servern unter Windows NT oder Windows 2000.
- Der Anhang zur Konfiguration mehrerer logischer Knoten beschreibt, wie mehrere logische Knoten in einer Umgebung mit partitionierten Datenbanken konfiguriert werden.
- Der Anhang zur Hochgeschwindigkeitsübertragung zwischen Knoten beschreibt, wie VIA (Virtual Interface Architecture) zur Verwendung mit DB2 Universal Database eingerichtet wird.
- Der Anhang zu den LDAP-Verzeichnisservices enthält Informationen zu den Einsatzmöglichkeiten der LDAP-Verzeichnisservices.
- Der Anhang zur Erweiterung der Steuerzentrale enthält Informationen zur Erweiterung der Steuerzentrale durch Hinzufügen neuer Knöpfe in der Menüleiste, einschließlich neuer Aktionen, Hinzufügen neuer Objektdefinitionen und Hinzufügen neuer Aktionsdefinitionen.

Teil 1. Einführung in die Optimierung

Kapitel 1. Elemente der Leistung

Leistung ist die Art und Weise, wie ein Computersystem unter einer bestimmten Auslastung arbeitet. Die Leistung wird an einem oder mehreren Faktoren eines Systems wie Antwortzeit, Durchsatz und Verfügbarkeit gemessen. Außerdem wird die Leistung von folgenden Faktoren beeinflusst:

- Von den verfügbaren Ressourcen
- Von der Auslastung und vom Ausmaß der gemeinsamen Nutzung dieser Ressourcen

Es ist allgemein empfehlenswert, eine Leistungsoptimierung durchzuführen, wenn das Kosten-Nutzen-Verhältnis eines Systems verbessert werden soll. Folgende Optimierungsziele können dabei verfolgt werden:

- Verarbeiten einer größeren oder anspruchsvolleren Arbeitsbelastung ohne steigende Verarbeitungskosten. (Zum Beispiel Steigerung der Auslastung ohne Erwerb neuer Hardware bzw. Erhöhung des Bedarfs an Prozessorzeit.)
- Erreichen schnellerer Systemantwortzeiten bzw. eines höheren Durchsatzes ohne Steigerung der Verarbeitungskosten.
- Reduzieren von Verarbeitungskosten ohne negative Auswirkungen für Benutzer.

Die Übersetzung der Leistung vom technischen in den wirtschaftlichen Begriff ist schwierig. Eine Leistungsoptimierung kostet sicherlich Geld (durch den Zeitaufwand für Personal und Prozessorzeit), so dass vor einem Optimierungsprojekt die Kosten gegen die möglichen Vorteile abgewogen werden müssen. Einige dieser Vorteile liegen auf der Hand:

- Effizientere Ressourcennutzung
- Die Möglichkeit, weitere Benutzer dem System hinzuzufügen

Andere Vorteile, wie größere Zufriedenheit seitens der Benutzer aufgrund schnellerer Antwortzeiten sind weniger fassbar. Aber alle dieser Vorteile sollten mit bedacht werden.

DB2 verfügt über integrierte Assistenten, die Sie bei der Durchführung einiger leistungsrelevanter Verwaltungsmaßnahmen unterstützen. Diese Maßnahmen sind häufig solche, die bei nur geringem Zeitaufwand eine deutliche Leistungsverbesserung ermöglichen. Die Assistenten führen Sie schrittweise durch die jeweilige Maßnahme. Assistenten sind über die Steuerzentrale und die Komponente *Client-Konfiguration - Unterstützung* verfügbar.

Der *Assistent: Leistungskonfiguration* hilft Ihnen bei der Optimierung der Leistung einer Datenbank, indem er die Konfigurationsparameter auf Ihre Geschäftsanforderungen abstimmt. Dieser Assistent und in geringerem Maße auch der *Assistent: Datenbank erstellen*, können Sie bei der Verbesserung der Leistung einer Datenbank unterstützen. Es sind zudem noch weitere Assistenten verfügbar, die Hilfestellung bei der Leistungsverbesserung für den Zugriff auf einzelne Tabellen sowie für den allgemeinen Datenzugriff bieten. Die Assistenten auf diesem Gebiet sind: Assistent: Tabelle erstellen, Assistent: Index erstellen und Assistent: Aktualisierung auf mehreren Systemen konfigurieren. Auf die Assistenten kann über die Steuerzentrale durch Anklicken eines Objekts mit Maustaste 2 zugegriffen werden.

Richtlinien für die Optimierung

Die folgenden Richtlinien sind als Hilfe für die Entwicklung eines allgemeinen Ansatzes zur Leistungsoptimierung zu verstehen.

Vergessen Sie das Gesetz der abnehmenden Ertragsgewinne nicht: Die größten Leistungsvorteile werden in der Regel durch die ersten Maßnahmen erzielt. Weitere Änderungen erbringen im Allgemeinen immer kleinere Vorteile und erfordern immer höheren Aufwand.

Optimieren Sie nicht nur des Optimierens wegen: Optimieren Sie, um erkannten Engpässen abzuhelfen. Das Optimieren von Ressourcen, die nicht den Hauptgrund für Leistungsprobleme darstellen, hat wenig oder gar keine Wirkung auf Antwortzeiten, solange Sie nicht die wichtigeren Probleme behoben haben, und kann tatsächlich eine nachfolgende Optimierungsarbeit erschweren. Wenn es ein bedeutendes Verbesserungspotenzial gibt, liegt es in der Verbesserung der Leistung von Ressourcen, die wichtige Faktoren in der Antwortzeit bilden.

Berücksichtigen Sie das gesamte System: Ein Parameter bzw. System kann nie isoliert optimiert werden. Bevor Sie Anpassungen vornehmen, überlegen Sie, wie sich diese Anpassungen auf das System als Ganzes auswirken werden.

Ändern Sie immer nur einen Parameter gleichzeitig: Ändern Sie in einem Schritt nicht mehrere Parameter zur Leistungsoptimierung. Selbst wenn Sie sich sicher sind, dass alle Änderungen vorteilhaft sind, haben Sie hinterher keine Möglichkeit, den Beitrag jeder Änderung zu bewerten. Darüber hinaus können Sie bei gleichzeitiger Änderung mehrerer Parameter den erzielten Vorteil nicht effektiv den möglicherweise in Kauf genommenen Einbußen gegenüberstellen. Von jeder Anpassung eines Parameters zur Optimierung eines Bereichs ist fast immer auch mindestens ein anderer Bereich betroffen, der vorher vielleicht nicht bedacht wurde. Wenn Sie jeweils nur einen Parameter ändern, haben Sie einen Vergleichspunkt und können feststellen, ob die Änderung die gewünschte Wirkung hat.

Führen Sie Messungen und Neukonfigurierungen nach Ebenen durch: Aus denselben Gründen, aus denen nur jeweils ein Parameter geändert werden soll, empfiehlt sich auch, die einzelnen Ebenen des Systems getrennt zu optimieren. Die folgende Liste von Ebenen innerhalb eines Systems kann Ihnen dabei als Richtlinie dienen:

- Hardware
- Betriebssystem
- Anwendungs-Server und -Requester
- Datenbankmanager
- SQL-Anweisungen
- Anwendungsprogramme

Prüfen Sie auf Hardware- und Softwareprobleme: Einige Leistungsprobleme können vielleicht durch Wartung der Hardware oder Korrektur der Software oder durch beides behoben werden. Verwenden Sie nicht zu viel Zeit auf die Überwachung und Optimierung des Systems, wenn eine Hardwarewartung oder Softwarekorrektur dies unnötig machen könnte.

Ermitteln Sie die Ursache eines Problems, bevor Sie Ihre Hardware aufrüsten: Auch wenn es so aussieht, als könnten zusätzliche Speicher- und Prozessorkapazitäten die Leistung sofort verbessern, sollten Sie sich die Zeit nehmen, die Engpässe zu lokalisieren und zu verstehen. Sie könnten ansonsten Geld für zusätzlichen Plattenspeicher ausgeben und anschließend feststellen, dass Sie nicht über die Prozessorkapazitäten oder die Kanäle verfügen, um den Speicher vorteilhaft zu nutzen.

Bereiten Sie Zurücksetzungsprozeduren vor, bevor Sie mit der Optimierung beginnen: Wie bereits früher erwähnt, können einige Optimierungsmaßnahmen zu unerwarteten Ergebnissen führen. Wenn sich daraus eine schlechtere Leistung ergibt, sollten die Maßnahmen rückgängig gemacht und alternative Optimierungsmaßnahmen versucht werden. Wenn der vorige Stand in einer Weise gesichert wurde, dass er einfach wiederhergestellt werden kann, ist die Rücknahme der nicht korrekten Informationen wesentlich einfacher.

Plattenspeicher

Wie bereits erwähnt, kann die Hardware, aus der sich Ihr System zusammensetzt, die Leistung Ihres Systems beeinflussen. Als Beispiel für den Einfluss, den die Hardware auf die Leistung hat, werden im Folgenden einige der Auswirkungen betrachtet, die mit dem Plattenspeicher zusammenhängen.

Ihre Plattenspeicherverwaltung wirkt sich auf vier Arten auf die Leistung aus:

- **Aufteilung des Speichers**:

Wie Sie eine begrenzte Speichergröße zwischen Indizes und Daten, unter Tabellenbereichen und Pufferpools aufteilen, bestimmt in hohem Maße, welche Leistung die einzelnen Komponenten in verschiedenen Situationen erreichen.

- **Verschwendeter Speicher**:

  Verschwendeter Speicher wirkt sich vielleicht als solcher nicht auf die Leistung des Systems aus, das ihn besitzt, aber er stellt eine Ressource da, die zur Verbesserung der Leistung an anderer Stelle genutzt werden könnte.

- **Verteilen der Platten-E/A**:

  Wie ausgewogen Sie den Bedarf an Platten-E/A auf mehrere Plattenspeichereinheiten und Controller verteilen, kann sich auf die Geschwindigkeit auswirken, mit der der Datenbankmanager Informationen von Platten abrufen kann.

- **Speicherknappheit**:

  Bei Erreichen der Grenze des verfügbaren Speichers kann die allgemeine Leistung beeinträchtigt werden.

Verfahren zur Leistungsverbesserung

Gehen Sie nach folgendem Verfahren vor, um die Leistung eines Systems zu verbessern:

1. Definieren Sie Leistungsziele.
2. Legen Sie Leistungsindikatoren fest.
3. Entwerfen Sie einen Plan zur Leistungsüberwachung.
4. Führen Sie den Plan aus.
5. Analysieren Sie Ihre Messergebnisse, um festzustellen, ob die definierten Ziele erreicht wurden. Ist dies der Fall, ziehen Sie eine Verringerung der Anzahl der erhobenen Messwerte in Betracht, da die Leistungsüberwachung ihrerseits Systemressourcen beansprucht. Andernfalls fahren Sie mit dem folgenden Schritt fort.
6. Ermitteln Sie die wichtigsten Leistungsprobleme im System.
7. Entscheiden Sie, an welchen Stellen Sie sich Einbußen leisten können und welche Ressourcen eine zusätzliche Belastung verkraften können. (Zu beinahe jeder Optimierung gehören Kompromisse, die zwischen Systemressourcen und verschiedenen Aspekten der Leistung eingegangen werden müssen.)
8. Passen Sie die Konfiguration des Systems an. Wenn Ihrer Ansicht nach mehrere Optimierungsoptionen geändert werden können, implementieren Sie jeweils nur eine Änderung. Wenn auf allen Ebenen keine weiteren Optionen verbleiben, haben Sie die Grenze Ihrer Ressourcen erreicht und müssen die Hardware aufrüsten.

9. Kehren Sie zu Schritt 4 zurück, und fahren Sie mit der Überwachung des Systems fort.

Folgende Maßnahmen empfehlen sich in regelmäßigen Abständen bzw. nach einer wesentlichen Änderung im System oder in der Auslastung:

- Gehen Sie zu Schritt 1 zurück.
- Überprüfen Sie die definierten Ziele und Indikatoren.
- Verfeinern Sie Ihre Überwachungs- und Optimierungsstrategie.

Wie viel Optimierung verträgt ein System?

Die potenzielle Verbesserung der Effizienz eines Systems unterliegt bestimmten Grenzen. Berücksichtigen Sie, wie viel Zeit und Geld Sie in die Optimierung der Systemleistung investieren sollten, und wie viel Hilfe für Benutzer des Systems durch zusätzliche Investitionen an Zeit und Geld bereitgestellt werden kann.

Ihr System zeigt vielleicht bereits ohne jede Optimierung eine akzeptable Leistung, die jedoch wahrscheinlich nicht an das Leistungspotenzial heranreicht. Jede Datenbank ist individuell verschieden. Sobald Sie eine eigene Datenbank und Anwendungen zu Ihrer Nutzung entwickeln, untersuchen Sie die verfügbaren Parameter, die optimiert werden können, und machen Sie sich damit vertraut, wie Sie die Einstellungen der Parameter auf Ihre Situation anpassen können. In einigen Fällen ist der Vorteil einer Optimierung des Systems möglicherweise begrenzt. In den meisten Fällen lässt sich jedoch potenziell ein erheblicher Vorteil erzielen.

Über die Steuerzentrale stehen Assistenten zur Verfügung, die Ihnen bei der Optimierung der Datenbankparameter helfen. Der *Assistent: Leistungskonfiguration* kann durch Anklicken der zu optimierenden Datenbank mit Maustaste 2 in der Steuerzentrale aufgerufen werden.

Je näher Ihr System an einen Leistungsengpass stößt, umso wahrscheinlicher wird eine Optimierung Wirkung zeigen. Wenn Ihr System nahe an den Leistungsgrenzen arbeitet und Sie die Anzahl von Benutzern am System um zehn Prozent erhöhen, werden sich die Antwortzeiten wahrscheinlich um wesentlich mehr als zehn Prozent erhöhen. In dieser Situation müssen Sie feststellen, wie Sie dieser Beeinträchtigung der Leistung durch Optimieren des Systems entgegenwirken können. Allerdings gibt es einen Punkt, ab dem das Optimieren keine weiteren Leistungsvorteile mehr erbringen kann. An diesem Punkt sollten Sie eine Überprüfung Ihrer Ziele und Erwartungen innerhalb dieser Umgebung in Betracht ziehen. Oder Sie sollten eine Änderung Ihrer Systemumgebung durch folgende Möglichkeiten ins Auge fassen: mehr Plattenspeicher, schnellere CPU, schnellere Kommunikationsverbindungen oder eine Kombination aus diesen Änderungen.

Ein weniger formaler Ansatz

Wenn Sie nicht genügend Zeit zur Definition von Leistungszielen sowie zur Überwachung und Optimierung in umfassender Weise haben, können Sie sich mit der Leistung auseinander setzen, indem Sie Ihren Benutzern zuhören. Finden Sie heraus, ob sie Probleme hinsichtlich Leistung haben. In der Regel können Sie das Problem lokalisieren oder zumindest feststellen, wo mit der Untersuchung des Problems anzufangen ist, indem Sie eine Anzahl einfacher Fragen stellen. Sie können Ihre Benutzer zum Beispiel Folgendes fragen:

- Was meinen sie mit „langsamer Reaktion"? Heißt dies, um zehn Prozent langsamer, als Sie erwarten, oder um das Zigfache langsamer?
- Wann haben sie die Probleme festgestellt? Treten sie erst seit kurzem auf oder waren sie immer da?
- Ist ihnen bekannt, ob auch andere Benutzer über das gleiche Problem klagen? Handelt es sich bei denen, die sich beklagen, um ein oder zwei Einzelpersonen oder eine ganze Gruppe?
- (Wenn eine ganze Gruppe von Benutzern Schwierigkeiten hat, sind diese mit demselben lokalen Netz verbunden?)
- Stehen die fraglichen Probleme in Zusammenhang mit einem bestimmten Transaktions- oder Anwendungsprogramm?
- Treten die Probleme in regelmäßigen Zeiträumen, zum Beispiel in der Mittagszeit, oder permanent auf?

Zusammenfassen aller Aspekte

Die DB2 zugrundeliegende Architektur ist wichtig, da ein Verständnis der Schlüsselkonzepte und -prozesse Sie bei anderen Leistungsfragen unterstützt. Zu Themen wie Speicherarchitektur, Datenverwaltung, dem Verarbeitungsmodell und dem Speichermodell gibt es jeweils eine einführende Darstellung im nächsten Kapitel. Weitere Informationen finden Sie in „Kapitel 2. Übersicht zu Architektur und Prozessen" auf Seite 11.

Die Optimierung der Anwendungsleistung beschäftigt sich mit den Aspekten der Leistung, die für Ihre Anwendungen und deren Interaktion mit der Datenbank relevant sind. Es gibt Aspekte, die für Anwendungen spezifisch sind: gemeinsamer Zugriff, Sperren, Optimierungsklassen, Steuerung von Ergebnismengen in Abfragen, Zeilenblockung und Verwenden von Compound-SQL-Anweisungen. Zudem werden folgende Themen kurz behandelt: Zeichenumsetzung im Zusammenhang mit der Anwendungsleistung, gespeicherte Prozeduren, Aktivierung von Datenbanken und die Vorteile der Parallelverarbeitung. Weitere Informationen finden Sie in „Kapitel 3. Überlegungen zu Anwendungen" auf Seite 47.

Es gibt Aspekte, die für die Optimierung von Abfragen spezifisch sind: Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung, die Auswirkungen von Knotengruppen und Tabellenbereichen auf die Abfrageoptimierung sowie die tief greifenden Auswirkungen, die Indizes auf die Abfrageoptimierung haben können. Weitere Informationen finden Sie in „Kapitel 4. Überlegungen zur Umgebung" auf Seite 103.

Systemkatalogstatistiken können einen bedeutenden Einfluss darauf haben, wie gut Anwendungen auf Daten zugreifen können. Die folgenden Themen sind für die Statistiken relevant: das Dienstprogramm RUNSTATS, die Verteilungsstatistiken, die Indexstatistiken und die Statistiken, die von Benutzern aktualisiert werden können. Weitere Informationen finden Sie in „Kapitel 5. Systemkatalogstatistiken" auf Seite 129.

Der SQL-Compiler analysiert jede Anwendung und ermittelt den besten Zugriffsplan für sie. Jede Abfrage in der Anwendung wird ausgewertet und kann mehreren verschiedenen Operationen unterzogen werden, die dazu dienen, das Ziel der Abfrage möglichst klar zu definieren. Anschließend werden verschiedene Zugriffsmethoden (Tabellensuchen und Verknüpfungen) für jede Abfrage geprüft, um den schnellsten Weg zum Abrufen der von der Abfrage angeforderten Daten zu ermitteln. Die Auswirkungen der Parallelverarbeitung werden ebenfalls berücksichtigt. Weitere Informationen finden Sie in „Kapitel 6. Der SQL-Compiler" auf Seite 171.

Im Produkt DB2 stehen verschiedene Tools zur Verfügung, die einen Einblick in die Verarbeitung der Abfragen einer Anwendung geben. Diese Tools liefern eine Erklärung der Faktoren, die die Anwendungsleistung betreffen. Weitere Informationen finden Sie in „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251.

Neben der Optimierung einzelner Anwendungen ist es auch sinnvoll, die Leistung der Datenbank, in der die Anwendungen aktiv sind, zu überprüfen. Die Leistung der Datenbank wird großenteils durch die Nutzung des Hauptspeichers bestimmt. Es gibt zahlreiche Themen zum Hauptspeicher, die für die Leistung relevant sind: Pufferpools, der Vorablesezugriff, die parallele E/A, Sortierfunktionen, die Häufigkeit, Daten in Tabellen reorganisieren zu müssen, sowie das Konzept der Datenbankagenten. Weitere Informationen finden Sie in „Kapitel 8. Leistung bei der Ausführung" auf Seite 283.

Es steht das Programm Governor zur Verfügung, das zur Verwaltung der Datenbanknutzung durch Anwendungen eingerichtet werden kann. Weitere Informationen finden Sie in „Kapitel 9. Verwenden von Governor" auf Seite 333.

Die Anzahl der Prozessoren und die Anzahl der Datenbankpartitionen kann zur Verbesserung der Leistung der Datenbank erhöht werden. Weitere Informationen finden Sie in „Kapitel 10. Skalieren der Konfiguration über das Hinzufügen von Prozessoren" auf Seite 349.

Wenn die Anzahl der Datenbankpartitionen erhöht wurde, muss sinnvollerweise sichergestellt werden, dass die Daten in der Datenbank korrekt auf die Datenbankpartitionen verteilt bzw. umverteilt werden. Weitere Informationen finden Sie in „Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen" auf Seite 363.

Wenn Sie die Leistungsfähigkeit der Datenbank prüfen wollen, können Sie Vergleichstests (Benchmark Tests) durchführen. Wichtige Themen auf diesem Gebiet sind die Methodik der Vergleichstests, die Vorbereitung eines Vergleichstests, die Erstellung eines Vergleichstestprogramms und die Durchführung der Vergleichstests. Weitere Informationen finden Sie in „Kapitel 12. Durchführen von Vergleichstests" auf Seite 371.

Die sehr umfangreiche Gruppe der Parameter zur Konfiguration von Datenbankmanager und Datenbank wird in „Kapitel 13. Konfigurieren von DB2" auf Seite 387 im Einzelnen vorgestellt.

Darüber hinaus gibt es weitere Informationen mit Bezug auf diese Leistungsthemen. Die Anhänge enthalten folgende Informationen:

- „Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen" auf Seite 571
- „Anhang B. EXPLAIN-Tabellen und Definitionen" auf Seite 609
- „Anhang C. EXPLAIN-Programme (SQL)" auf Seite 645
- „Anhang D. db2exfmt - EXPLAIN-Tool für Tabellenformat" auf Seite 695

Kapitel 2. Übersicht zu Architektur und Prozessen

Für den Umgang mit der Leistung der Datenbankoperationen für DB2 benötigen Sie ein Verständnis für die grundlegenden Konzepte, die mit der DB2-Architektur und den DB2-Prozessen in Zusammenhang stehen. Dieses Kapitel enthält ausreichende Informationen zur Arbeitsweise von DB2 Universal Database. Nachfolgende Kapitel bieten ausführlichere Einzelheiten zu einigen der hier dargestellten Themen, wobei jedoch die in diesem Kapitel enthaltenen Informationen den Kontext für das spätere Verständnis bieten.

Die erste Abbildung zeigt eine Übersicht zur Architektur und den Prozessen für DB2 UDB.

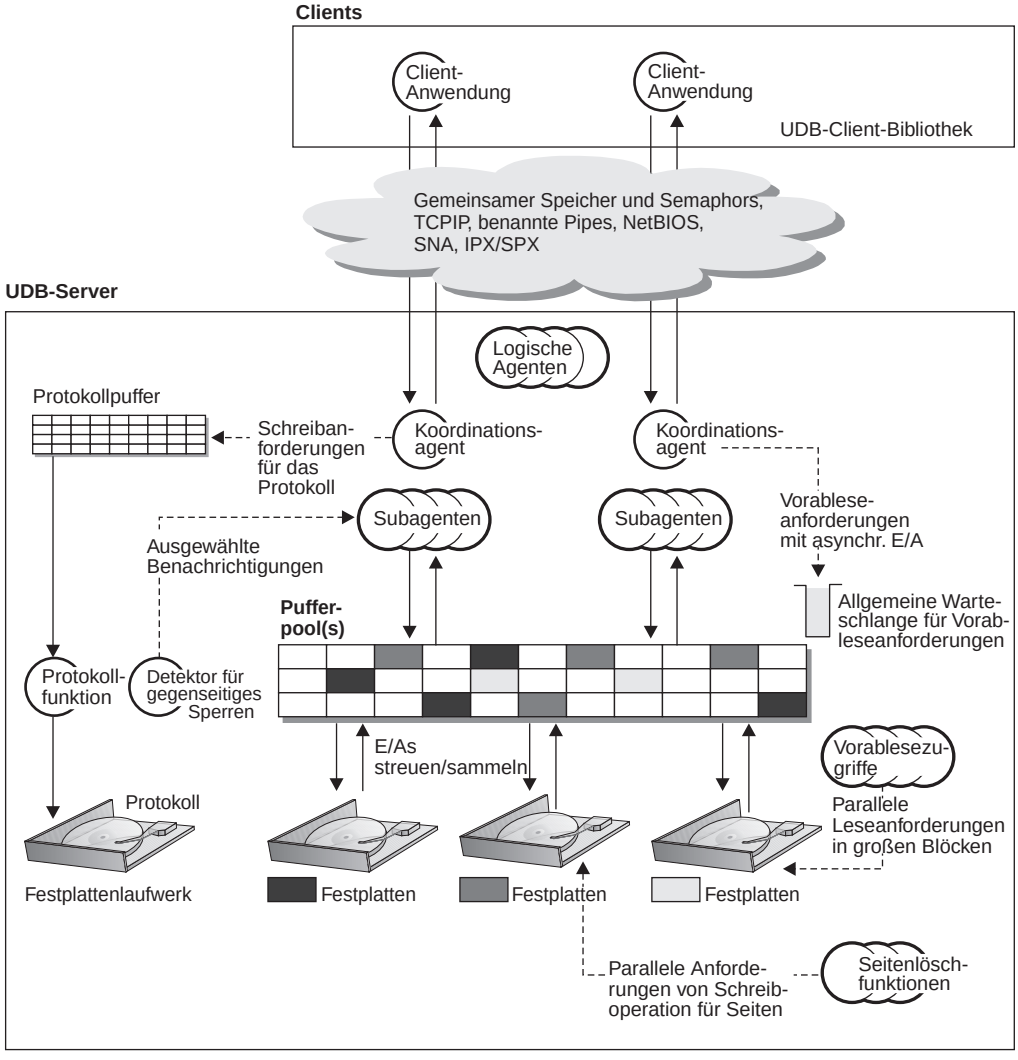


Abbildung 1. Übersicht zu Architektur und Prozessen

Auf der Client-Seite gibt es lokale und/oder ferne Anwendungen, die mit der DB2 Universal Database-Client-Bibliothek verbunden sind.

Zwischen den Clients und dem DB2 Universal Database-Server befindet sich eine „Wolke", die die Kommunikationsmittel zwischen den lokalen oder fernen Clients und dem Server darstellt. Lokale Clients kommunizieren mit Hilfe von gemeinsam benutztem Speicher und Semaphors; ferne Clients verwenden ein Protokoll wie beispielsweise "Benannte Pipes" (Named Pipes, NPIPE), TCP/IP, NetBIOS, IPX/SPX oder SNA.

Auf der Server-Seite wird die Aktivität durch **EDUs** (Engine Dispatchable Units, zuteilbare Einheiten der Steuerkomponente) gesteuert. In allen Abbildungen in diesem Kapitel werden EDUs als Kreise oder Gruppen von Kreisen dargestellt. EDUs werden auf Windows-Plattformen und unter OS/2 als Threads (alle in einem einzigen Prozess) und unter UNIX als Prozesse implementiert. Die am weitesten verbreitete Art von EDUs sind DB2-Agenten. Diese EDUs führen den Großteil der SQL-Verarbeitung für Anwendungen durch. Andere Beispiele für EDUs sind die DB2-Vorablesezugriffe und -Seitenlöschfunktionen, die für verschiedene Arten von E/A-Verarbeitung verantwortlich sind. Weitere Informationen finden Sie im Abschnitt „Datenbankagenten" auf Seite 320.

Jede Client-Anwendung wird einer eindeutigen EDU mit dem Namen „Koordinationsagent" zugeordnet, die die Verarbeitung für die betreffende Anwendung koordiniert und mit ihr kommuniziert. Es kann auch eine Gruppe von Subagenten geben, die gemeinsam dafür zugeordnet sind, die Verarbeitung von Anforderungen der Client-Anwendung zu bearbeiten. Es können mehrere Subagenten zugeordnet werden. So können, wenn die Maschine, auf der sich der Server befindet, mehrere Prozessoren hat (wie beispielsweise eine symmetrische Mehrprozessorumgebung), die Anforderungen der Client-Anwendung diese Prozessoren nutzen.

Alle Agenten und Subagenten werden unter Verwendung eines Algorithmus für eine Poolfunktion verwaltet, der die Erstellung und/oder Zerstörung von EDUs so gering wie möglich hält.

Ein Pufferpool ist ein Speicherbereich, in den Datenbankseiten mit Benutzertabellen-, Index- und Katalogdaten temporär verschoben und darin möglicherweise geändert werden. Der Pufferpool ist bei der Beeinflussung der Gesamtleistung einer Datenbank von zentraler Bedeutung, da der Zugriff auf Daten im Speicher weit schneller erfolgen kann als auf Platte. Wenn sich größere Mengen der von Anwendungen benötigten Daten im Pufferpool befinden würden, würde im Vergleich zu der Zeit, die für das Finden der Daten im Plattenspeicher aufgewendet werden muss, weniger Zeit zum Zugriff auf diese Daten benötigt werden. Weitere Informationen hierzu finden Sie im Abschnitt „Verwalten des Datenbankpufferpools" auf Seite 292.

Die Konfiguration des Pufferpools steuert gemeinsam mit EDUs für Vorablesezugriff und Seitenlöschfunktionen die Geschwindigkeit des Datenzugriffs sowie die daraus resultierende Verfügbarkeit der Daten, die von den Anwendungen benötigt werden.

Die Vorablesezugriffe sind dazu vorhanden, Daten von Platte abzurufen und in den Pufferpool zu verschieben, bevor diese Daten von Anwendungen benötigt werden. Beispielsweise würden Anwendungen, die große Volumen von Daten durchsuchen müssen, darauf warten müssen, dass Daten von Platte in den Pufferpool verschoben werden, wenn es keine Vorablesezugriffe für Daten geben würde. Agenten der Anwendung senden asynchrone Vorausleseanforderungen an eine allgemeine Bereitstellungswarteschlange. Wenn Vorablesezugriffe verfügbar werden, implementieren sie diese Anforderungen. Dabei verwenden sie Eingabeoperationen für große Blöcke (Big-Block) oder gestreutes Lesen (Scatter Read), um die angeforderten Seiten von Platte in den Pufferpool zu verschieben. Wenn mehrere Platten zur Speicherung der Datenbankdaten verfügbar sind, bedeutet dies, dass die Daten einheitenübergreifend gespeichert werden können. Mit Hilfe des einheitenübergreifenden Speicherns von Daten können die Vorablesezugriffe mehrere Platten gleichzeitig zum Abruf von Daten verwenden. Weitere Informationen finden Sie in den Abschnitten „Vorablesen von Daten in den Pufferpool" auf Seite 301 und „Konfigurieren von E/A-Servern für Vorablesezugriff und parallele E/A" auf Seite 304.

Vorablesezugriffe werden dazu verwendet, Daten in den Pufferpool zu verschieben. Seitenlöschfunktionen werden dazu verwendet, Daten vom Pufferpool zurück auf die jeweilige Platte zu verschieben.

Seitenlöschfunktionen sind Hintergrund-EDUs, die von den Anwendungsagenten unabhängig sind. Sie suchen Seiten im Pufferpool, die nicht mehr benötigt werden, und lagern diese aus. Seitenlöschfunktionen können sicherstellen, dass im Pufferpool Platz für die Seiten ist, die von den Vorabzugriffen abgerufen werden.

Wenn die unabhängigen Vorablesezugriffe und Seitenlöschfunktions-EDUs nicht vorhanden wären, würden die Anwendungsagenten alle Lese- und Schreiboperationen für Daten zwischen Pufferpool und Plattenspeicher selbst ausführen müssen.

Wenn mehrere Anwendungen mit Daten aus der Datenbank arbeiten, kann zwischen zwei oder mehreren von ihnen „gegenseitiges Sperren" auftreten. Ein solches gegenseitiges Sperren ist in der folgenden Abbildung dargestellt.

Das Konzept von gegenseitigem Sperren

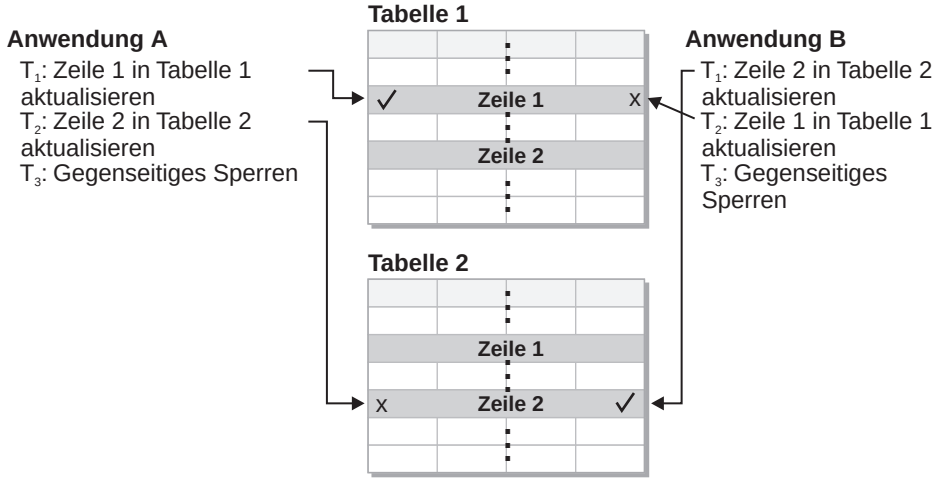


Abbildung 2. Detektor für gegenseitiges Sperren

„Gegenseitiges Sperren" bedeutet, dass mehr als eine Anwendung darauf wartet, dass eine andere Anwendung eine Sperre für Daten freigibt. Alle wartenden Anwendungen sperren Daten, die von anderen Anwendungen benötigt werden. Diese gesperrten Daten werden von einer oder mehreren Anwendungen benötigt, die ihrerseits Daten sperren, die von anderen Anwendungen benötigt werden. Dieses gegenseitige Warten darauf, dass die andere Anwendung eine Sperre für gesperrte Daten freigibt, führt zu gegenseitigem Sperren: Die Anwendungen können unbegrenzt darauf warten, dass die jeweils „andere" Anwendung die aktive Sperre für die Daten freigibt. Die anderen Anwendungen geben die Sperren für Daten, die sie benötigen, nicht freiwillig frei. Zum Beenden dieser Situationen des gegenseitigen Sperrens ist ein Prozess erforderlich.

Wie der Name schon sagt, überwacht der Detektor für gegenseitiges Sperren die Informationen zu Agenten, die an Sperren warten. Der Detektor für gegenseitiges Sperren wählt eine beliebige der Anwendungen aus, die sich gegenseitig sperren, um die Sperren freizugeben, die momentan von der „freiwilligen" Anwendung verwendet werden. Das Freigeben der Sperren dieser Anwendung führt dazu, dass die Daten wieder verfügbar gemacht werden, die von anderen wartenden Anwendungen benötigt werden. Die bis dahin wartenden Anwendungen können dann die Daten verwenden, die zum Durchführen von Aktionen in Bezug auf Daten in der Datenbank erforderlich sind.

Änderungen an Datenseiten im Pufferpool werden protokolliert. Ein Protokollpuffer ist vorhanden und einer Protokollfunktions-EDU zugeordnet. Agenten, die einen Datensatz in der Datenbank aktualisieren, aktualisieren die zugeord-

nete Seite im Pufferpool und schreiben einen Protokollsatz. Dieser Protokollsatz enthält die Informationen, die zum Wiederholen bzw. Wiederrufen der Änderung notwendig sind. Die Seite im Pufferpool und der Protokollsatz im Protokollpuffer werden nicht sofort auf Platte geschrieben, damit die Leistung optimiert wird. Die Protokollfunktions-EDU und der Manager des Pufferpools implementieren gemeinsam ein WAL-Protokoll (Write Ahead Logging, vorausschreibende Protokollierung), das sicherstellt, dass die Datenseite nicht auf Platte geschrieben wird, bevor der zugehörige Protokollsatz in das Protokoll geschrieben wird. Das WAL-Protokoll gewährleistet, dass immer genügend Informationen im Protokoll sind, um die Wiederherstellung nach einem Absturz durchzuführen und die Datenbankkonsistenz wiederherzustellen. Wenn eine nicht festgeschriebene Aktualisierung einer Seite auf Platte geschrieben wurde, werden bei der Wiederherstellung nach einem Systemabsturz zum Widerrufen der Aktualisierung Widerrufsinformationen im zugehörigen Protokollsatz verwendet. Wenn eine festgeschriebene Aktualisierung (COMMIT-Operation) nicht auf Platte gespeichert wurde, werden bei der Wiederherstellung nach einem Systemabsturz die Wiederholungsinformationen im zugehörigen Protokollsatz zum Wiederholen der Aktualisierung verwendet.

Anmerkung: Mit Hilfe einer COMMIT-Operation werden alle Protokollsätze in der Transaktion auf Platte geschrieben, wenn dies nicht bereits geschehen ist.

Speicherarchitektur

Im Verlauf der Darstellung der Speicherarchitektur werden folgende Punkte in Betracht gezogen:

- „Datenbankverzeichnis"
- „Tabellenbereiche" auf Seite 18

Datenbankverzeichnis

Beim Erstellen einer Datenbank werden zugehörige Informationen einschließlich Standardinformationen in einem Verzeichnis gespeichert. Die Verzeichnisstruktur wird an der Lokation für Sie erstellt, die im Befehl CREATE DATABASE angegeben wurde. Wenn Sie beim Erstellen der Datenbank die Lokation des Pfads oder Laufwerks nicht angeben, wird die Standardlokation verwendet.

Es empfiehlt sich, dass Sie explizit angeben, wo die Datenbank erstellt werden soll.

In dem Verzeichnis, das im Befehl CREATE DATABASE angegeben wurde, wird unter Verwendung des Exemplarnamens ein Unterverzeichnis erstellt. Dieses Unterverzeichnis stellt sicher, dass Datenbanken, die in verschiedenen Exemplaren in demselben Verzeichnis erstellt wurden, nicht denselben Pfad verwenden. Unter dem Unterverzeichnis mit dem Exemplarnamen wird ein

Unterverzeichnis mit dem Namen NODE0000 erstellt. Dieses Unterverzeichnis wird zur Unterscheidung von Partitionen in einer Umgebung mit mehreren logischen partitionierten Datenbanken verwendet. Unter dem Knotenverzeichnis (Node = Knoten) wird ein Unterverzeichnis mit dem Namen SQL00001 erstellt. Dieses Unterverzeichnis wird unter Verwendung des Datenbank-Token benannt und stellt die Datenbank dar, die gerade erstellt wird. Darüber hinaus wird es dazu verwendet, Datenbanken zu unterscheiden, die in diesem Exemplar unter dem Verzeichnis erstellt wurden, das Sie im Befehl CREATE DATABASE angegeben haben.

Die Verzeichnisstruktur würde also folgendermaßen aussehen:

```
<ihr_verzeichnis>/<ihr_exemplar>/NODE0000/SQL00001/
```

Das Datenbankverzeichnis enthält mehrere Dateien, die durch den Befehl CREATE DATABASE erstellt wurden. Informationen zum Pufferpool sind in den Dateien SQLBP.1 und SQLBP.2 enthalten. Informationen zum Tabellenbereich sind in den Dateien SQLSPCS.1 und SQLSPCS.2 enthalten. Von diesen Dateien sind jeweils zwei Exemplare vorhanden, damit die Informationen darin gesichert werden können.

Informationen zur Datenbankkonfiguration sind in der Datei SQLDBCON enthalten. Die Protokolldatei DB2RHIST.ASC und ihre Sicherungsdatei DB2RHIST.BAK können von Ihnen gelesen werden und enthalten Protokolldaten zu Sicherungen, zu Wiederherstellungen, zum Laden von Tabellen, zur Reorganisation von Tabellen, zum Ändern eines Tabellenbereichs und zu anderen Änderungen an einer Datenbank.

Die Protokollsteuerdatei SQLOGCTL.LFH enthält Informationen zu den aktiven Protokollen. Die Wiederherstellungsverarbeitung verwendet Informationen aus dieser Datei um festzustellen, an welcher Stelle in den Protokollen die Wiederherstellung beginnen soll. Das Unterverzeichnis SQLOGDIR enthält die eigentlichen Protokolldateien.

Anmerkung: Sie sollten sicherstellen, dass das Unterverzeichnis für die Protokolle anderen Platten zugeordnet ist als denen, die für Ihre Daten verwendet werden. Ein Plattenproblem kann in diesem Fall auf Ihre Daten bzw. Ihre Protokolle beschränkt werden, so dass nicht beide gleichzeitig davon betroffen sind. Dies kann zudem einen deutlichen Vorteil für die Leistung mit sich bringen, da die Protokolldateien und die Datenbankbehälter nicht um das Verschieben derselben Plattenkopfsätze konkurrieren. Sie können die Lokation des Protokollunterverzeichnisses mit Hilfe des Konfigurationsparameters *newlogpath* der Datenbank ändern.

Die SQLT*-Unterverzeichnisse werden erstellt und enthalten die standardmäßigen SMS-Tabellenbereiche (SMS - System Managed Space, vom System verwalteter Speicher), die für eine Datenbank in Betrieb erforderlich sind. Die folgenden drei standardmäßigen Tabellenbereiche werden erstellt:

- Das Unterverzeichnis SQLT0000.0 enthält den Katalogtabellenbereich mit den Systemkatalogtabellen.
- Das Unterverzeichnis SQLT0001.0 enthält den standardmäßigen temporären Tabellenbereich.
- Das Unterverzeichnis SQLT0002.0 enthält den standardmäßigen Tabellenbereich für Benutzerdaten.

In Zusammenhang mit Tabellenbereichen werden Sie auch über „Behälter" lesen. Bei SMS-Tabellenbereichen sind Behälter Betriebssystemverzeichnisse.

In jedem Unterverzeichnis bzw. jedem Behälter wird eine Datei mit dem Namen SQLTAG.NAM erstellt. Diese Datei markiert das betreffende Unterverzeichnis als in Gebrauch, so dass bei späteren Erstellungsoperationen für Tabellenbereiche nicht versucht wird, solche Unterverzeichnisse zu verwenden. Es gibt auch weitere Dateien, die in den Behälterunterverzeichnissen erstellt werden und über verschiedene Erweiterungen für ihre Namen verfügen, wodurch zwischen den darin gespeicherten Datentypen unterschieden werden kann. Es gibt folgende Erweiterungen:

- SQL*.DAT (enthalten nicht lange (Non-Long) Tabellendaten)
- SQL*.LF (enthalten LONG VARCHAR- oder LONG VARGRAPHIC-Daten)
- SQL*.LB (enthalten BLOB-, CLOB- oder DBCLOB-Daten)
- SQL*.LBA (enthalten Informationen zu Zuordnung und freiem Speicherbereich für SQL*.LB-Dateien)
- SQL*.INX (enthalten Indextabellendaten)
- SQL*.DTR (enthalten temporäre Daten für eine Reorganisation einer SQL*.DAT-Datei)
- SQL*.LFR (enthalten temporäre Daten für eine Reorganisation einer SQL*.LF-Datei)
- SQL*.RLB (enthalten temporäre Daten für eine Reorganisation einer SQL*.LB-Datei)
- SQL*.RBA (enthalten temporäre Daten für eine Reorganisation einer SQL*.LBA-Datei)

Tabellenbereiche

Es gibt zwei unterstützte Arten von Tabellenbereichen: System Managed Space (SMS, vom System verwalteter Tabellenbereich) und Database Managed Space (DMS, vom Datenbankmanager verwalteter Tabellenbereich). Beide haben eigene Kenndaten, wodurch sie für verschiedene Umgebungen geeignet sind.

Im Band *Systemverwaltung: Konzept* finden Sie weitere Informationen zum Entwerfen und Auswählen von Tabellenbereichen.

SMS-Tabellenbereiche

SMS-Tabellenbereiche (SMS - System Managed Space, vom System verwalteter Speicher) speichern Daten in Betriebssystemdateien. Die Daten in den Tabellenbereichen werden einheitenübergreifend in Speicherbereichen in allen Behältern des Systems gespeichert. Ein **Speicherbereich** ist eine Gruppe von aufeinander folgenden Seiten, die für die Datenbank definiert sind. Jede Tabelle in einem Tabellenbereich erhält ihren eigenen Dateinamen, der von allen Behältern verwendet wird. Die Dateierweiterung bezeichnet den Datentyp, der in der betreffenden Datei gespeichert wird. Der Startspeicherbereich für alle Tabellen wird „reihum" in allen Behältern platziert. Dadurch wird der Platzbedarf auf alle Behälter im Tabellenbereich gleichmäßig verteilt. Dies ist sehr wichtig, wenn es eine große Zahl kleiner Tabellen gibt.

Die Zuordnung von Speicherbereich erfolgt, wenn Bedarf für zusätzlichen Speicherbereich besteht. Standardmäßig wird Speicherbereich jeweils Seite für Seite zugeordnet.

DMS-Tabellenbereiche

Bei einem DMS-Tabellenbereich (DMS - Database Managed Space, vom Datenbankmanager verwalteter Tabellenbereich) steuert der Datenbankmanager den Speicherbereich. Wenn der DMS-Tabellenbereich definiert wird, wird eine Liste mit Einheiten oder Dateien ausgewählt, die zu diesem gehören sollen. Der Speicherbereich in diesen Einheiten oder Dateien wird durch den DB2-Datenbankmanager verwaltet. Wie auch SMS-Tabellenbereiche und Behälter verwenden DMS-Tabellenbereiche und der Datenbankmanager einheitenübergreifendes Lesen und Schreiben von Daten für Speicherbereiche, um eine gleichmäßige Verteilung von Daten auf alle Behälter sicherzustellen.

DMS-Tabellenbereiche unterscheiden sich von SMS-Tabellenbereichen dadurch, dass bei ersteren Speicherbereich zugeordnet wird, wenn der Tabellenbereich erstellt wird, und nicht erst, wenn er benötigt wird.

Darüber hinaus kann sich die Platzierung von Daten bei beiden Arten von Tabellenbereichen unterscheiden. Betrachten Sie beispielsweise den Bedarf nach effizienten Tabellensuchoperationen: Es ist wichtig, dass die Seiten in einem Speicherbereich physisch aneinander grenzen. Bei SMS entscheidet das Dateisystem des Betriebssystems, wo die logischen Dateiseiten physisch abgelegt werden. Die Seiten können aneinander grenzend zugeordnet werden, müssen aber nicht. Dies hängt von der Stufe der anderen Aktivitäten im Dateisystem und dem Algorithmus ab, der zum Bestimmen der Platzierung verwendet wird. Bei DMS jedoch kann der Datenbankmanager sicherstellen, dass die Seiten physisch aneinander grenzen, da er eine direkte Schnittstelle zur Platte aufweist.

Zu dieser allgemeinen Feststellung bezüglich der aneinander grenzenden Platzierung von Seiten im Speicher gibt es eine Ausnahme. Es gibt beim Arbeiten mit DMS-Tabellenbereichen die zwei folgenden Behälteroptionen: unformatierte Einheiten und Dateien. Beim Arbeiten mit Dateibehältern ordnet der Datenbankmanager den gesamten Behälter während der Erstellung des Tabellenbereichs zu. Ein Ergebnis dieser ersten Zuordnung des gesamten Tabellenbereichs besteht darin, dass die physische Zuordnung normalerweise direkt aneinander grenzend erfolgt, obwohl das Dateisystem die Zuordnung vornimmt. Dies ist allerdings nicht garantiert der Fall. Beim Arbeiten mit Behältern aus unformatierten Einheiten übernimmt der Datenbankmanager die Steuerung der gesamten Einheit und stellt immer sicher, dass die Seiten in einem Speicherbereich direkt aneinander grenzend sind.

Im Unterschied zu SMS-Tabellenbereichen müssen die Behälter, aus denen ein DMS-Tabellenbereich besteht, in Bezug auf ihre Kapazität nicht annähernd gleich sein. Es empfiehlt sich jedoch, dass die Behälter in Bezug auf ihre Kapazität gleich oder annähernd gleich sind. Darüber hinaus kann in einem DMS-Tabellenbereich jeder verfügbare freie Speicherbereich verwendet werden, wenn ein bestimmter Behälter voll ist.

Beim Arbeiten mit DMS-Tabellenbereichen sollten Sie in Betracht ziehen, jeden Behälter einer anderen Platte zuzuordnen. Dadurch wird die Tabellenbereichskapazität vergrößert und die Möglichkeit geschaffen, parallele E/A-Operationen zu nutzen.

Die Anweisung CREATE TABLESPACE erstellt einen neuen Tabellenbereich in einer Datenbank, ordnet diesem Tabellenbereich Behälter zu und trägt Tabellenbereichsdefinition und -attribute in den Katalog ein. Einer der Werte, der beim Erstellen des Tabellenbereichs definiert wird, ist die Größe des Speicherbereichs. Ein Speicherbereich ist die Einheit der Bereichszuordnung in einem Tabellenbereich. Er ist einfach eine Reihe direkt aufeinander folgender Seiten. Die Größe des Speicherbereichs ist die Zahl direkt aufeinander folgender Seiten. Nur eine einzige Tabelle (oder ein anderes Objekt, wie z. B. ein Index) kann die Seiten in einem einzelnen Speicherbereich verwenden. Alle Objekte (Tabellen, Indizes und andere), die in dem betreffenden Tabellenbereich erstellt wurden, werden Speicherbereichen in einer Adressenzuordnung für einen logischen Tabellenbereich zugeordnet. Ein Speicherbereich gehört zu einem bestimmten Zeitpunkt jeweils nur zu einem Objekt. Die Zuordnung von Speicherbereich wird über SMP verwaltet (SMP - Space Map Pages, Speicherzuordnungsseiten).

Der erste Speicherbereich in der Adressenzuordnung für den logischen Tabellenbereich besteht aus den Kopfdaten für den Tabellenbereich, die interne Steuerdaten enthalten. Der zweite Speicherbereich ist der erste Speicherbereich der SMP für den Tabellenbereich. SMP-Speicherbereiche sind in regelmäßigen Abständen über den gesamten Tabellenbereich verteilt. Jeder

SMP-Speicherbereich besteht einfach aus einer Bitzuordnung der Speicherbereiche vom aktuellen SMP-Speicherbereich bis zum nächsten. Die Bitzuordnung wird dazu verwendet, zu überprüfen, welche der dazwischenliegenden Speicherbereiche in Gebrauch sind und welche nicht.

Der auf die SMP folgende Speicherbereich ist die Objekttabelle für den Tabellenbereich. Die Objekttabelle ist eine interne Tabelle, die protokolliert, welche Benutzerobjekte im Tabellenbereich vorhanden sind und wo sich deren erster EMP-Speicherbereich (EMP - Extent Map Page, Speicherbereichszuordnungsseite) befindet. Jedes Objekt verfügt über eigene EMPs, die eine Zuordnung zu allen Seiten des Objekts darstellen, das in der Adressenzuordnung für den logischen Speicherbereich gespeichert ist.

Die folgende Abbildung zeigt die logische Adressenzuordnung für einen DMS-Tabellenbereich.

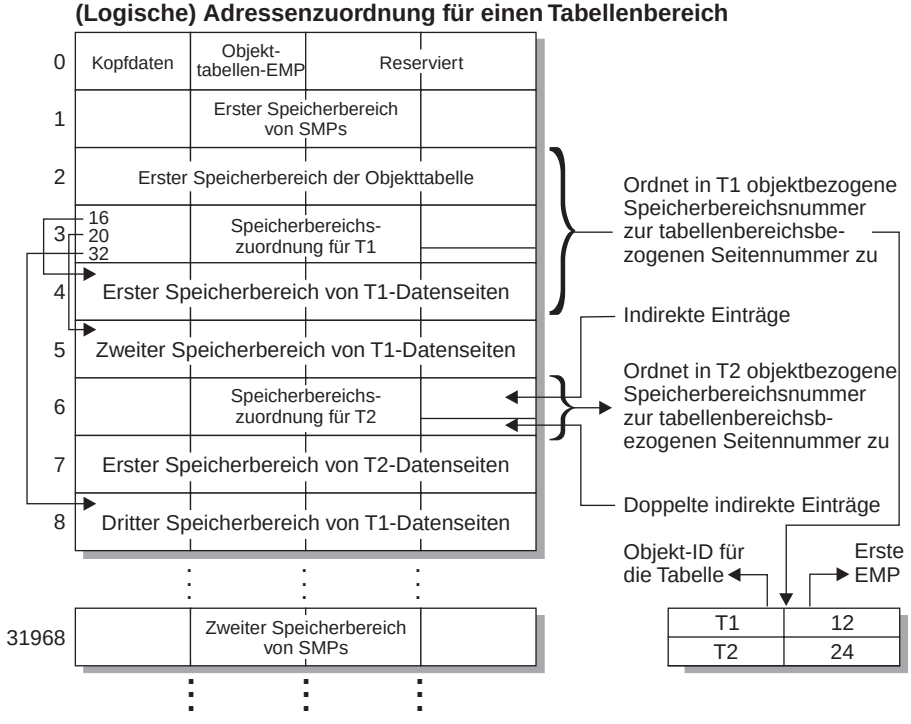


Abbildung 3. DMS-Tabellenbereiche

Die Objekttabelle ist eine interne relationale Tabelle, die eine Objektkennung zur Lokation des ersten EMP-Speicherbereichs der Tabelle zuordnet. Dieser EMP-Speicherbereich stellt, direkt oder indirekt, alle Speicherbereiche im betreffenden Objekt dar. Jede EMP enthält eine Reihe von Einträgen. Jeder Eintrag ordnet eine zum Objekt relative Speicherbereichsnummer einer zum Tabellenbereich relativen Seitennummer zu, an der sich der Objektspeicherbereich befindet. Direkte EMP-Einträge ordnen Adressen, die zum Objekt relativ sind, direkt Adressen zu, die zum Tabellenbereich relativ sind. Die letzte EMP-Seite im ersten EMP-Speicherbereich enthält indirekte Einträge. Indirekte EMP-Einträge ordnen EMP-Seiten zu, die anschließend Zuordnungen zu Objektseiten herstellen. Die letzten 16 Einträge in der letzten EMP-Seite im ersten EMP-Speicherbereich enthalten doppelt indirekte Einträge.

Die Speicherbereiche der Adressenzuordnung für den logischen Tabellenbereich werden "reihum" einheitenübergreifend in den Behältern gespeichert, die dem Tabellenbereich zugeordnet sind.

Vergleich von SMS- und DMS-Tabellenbereichen

Vergleicht man SMS- und DMS-Tabellenbereiche, so stellt man fest, dass SMS-Tabellenbereiche eine hervorragende Wahl für allgemeine Zwecke sind. SMS-Tabellenbereiche bieten eine sehr gute Leistung bei geringem Verwaltungsaufwand.

DMS-Tabellenbereiche sind die beste Wahl, wenn Sie eine Spitzenleistung erreichen möchten. Einheitenbehälter bieten die beste Leistung, da beim Verschieben von Daten unter Verwendung von Dateibehältern oder SMS-Tabellenbereichen doppelte Speicherung im Puffer auftreten kann. (Doppelte Speicherung im Puffer kann auftreten, wenn die Daten erst auf Datenbankmanagerebene und anschließend auf Dateisystemebene im Puffer zwischengespeichert werden.)

Datenverwaltung

Nachdem eine Datenbank, ein Tabellenbereich und eine Tabelle erstellt sowie Daten in dieser Tabelle platziert wurden, ist es interessant zu erfahren, wie die Tabelle organisiert ist und wie Indizes zum Abrufen dieser Tabellendaten verwendet werden.

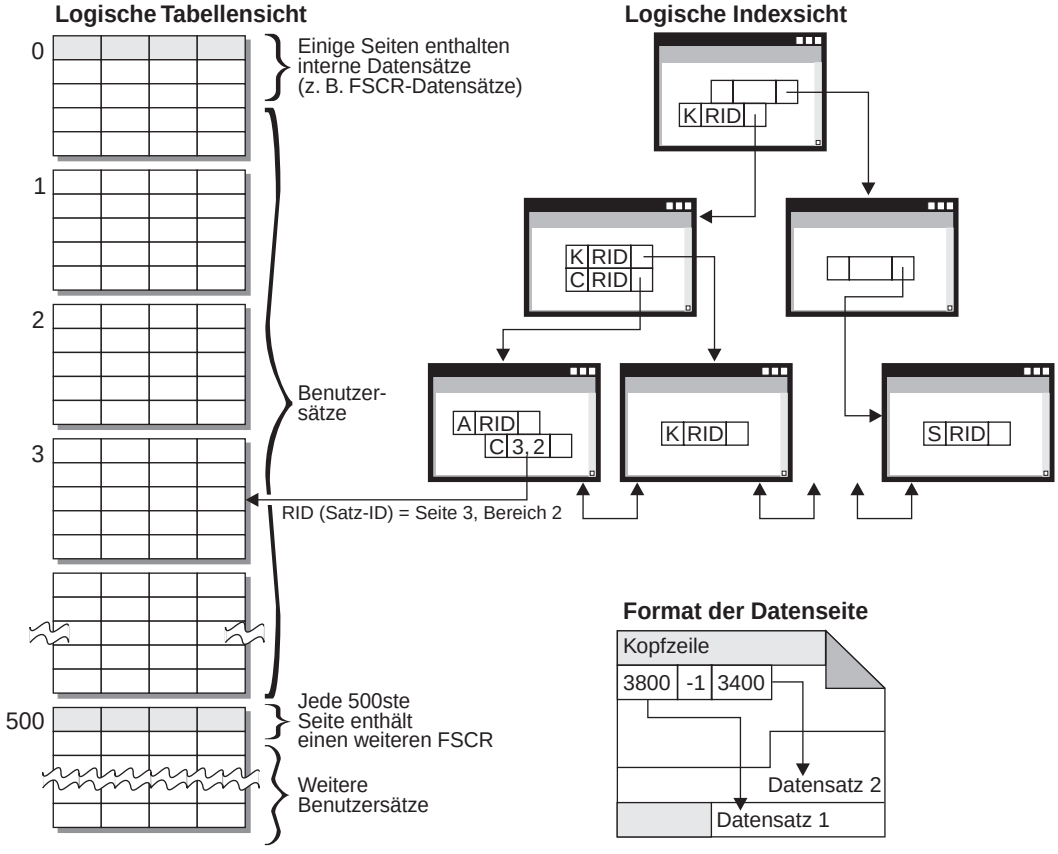


Abbildung 4. Tabellen, Datensätze und Indizes

Logisch gesehen sind Tabellendaten als Liste von Datenseiten organisiert. Diese Datenseiten werden logisch zu Gruppen zusammengestellt, wobei die Größe des Speicherbereichs im Tabellenbereich die Grundlage ist. Wenn die Speicherbereichsgröße beispielsweise vier beträgt, sind die Seiten null bis drei Teil des ersten Speicherbereichs, die Seiten vier bis sieben sind Teil des zweiten usw.

Die Zahl der Datensätze, die in den einzelnen Datenseiten enthalten sind, kann auf der Basis der Größe der Datenseite und der Größe der Datensätze variieren. Maximal können 255 Datensätze auf einer Seite untergebracht werden. Die meisten Seiten enthalten nur Benutzersätze. Eine kleine Zahl von Seiten enthält jedoch spezielle interne Datensätze, die von DB2 zur Verwaltung der Tabelle verwendet werden. Auf jeder 500sten Seite befindet sich beispielsweise ein FSCR (Free Space Control Record, Steuersatz für freien Speicherbereich). Diese Datensätze halten fest, wie viel freier Speicherbereich für neue Datensätze auf jeder einzelnen der folgenden 500 Datenseiten (bis zum nächsten FSCR) vorhanden ist. Dieser verfügbare freie Speicherbereich wird verwendet, wenn Datensätze in die Tabelle eingefügt werden.

Logisch gesehen sind Indexseiten als B-Baumstruktur organisiert, durch die Datensätze in den Tabellendaten, die über einen gegebenen Schlüsselwert verfügen, auf effiziente Weise lokalisiert werden können. Die Zahl der Entitäten auf einer Indexseite ist nicht festgelegt, sondern hängt von der Größe des Schlüssels ab. Bei Tabellen in DMS-Tabellenbereichen verwenden RIDs (Record Identifiers, Satz-IDs) auf den Indexseiten Seitennummern, die nicht zum Objekt, sondern zum Tabellenbereich relativ sind. Dadurch kann bei einer **Indexsuche** direkt auf die Datenseiten zugegriffen werden, ohne dass eine EMP zur Zuordnung erforderlich ist.

Alle Datenseiten haben das folgende Format: Sie beginnen jeweils mit einer Kopfzeile. Nach der Kopfzeile folgt ein Bereichsverzeichnis. Jeder Eintrag im Bereichsverzeichnis entspricht einem anderen Datensatz auf der Seite. Der Eintrag selbst ist die relative Byteadresse auf der Datenseite, an der der Datensatz beginnt. Einträge von minus eins (-1) entsprechen gelöschten Datensätzen.

Satz-IDs (RIDs) und Seiten

Satz-IDs (Record Identifier, RID) sind eine aus drei Byte bestehende Seitennummer gefolgt von einer Bereichsnummer aus einem Byte. Wenn der Index zur Identifizierung einer Satz-ID verwendet wird, wird diese Satz-ID dazu verwendet, zur richtigen Datenseite und Bereichsnummer auf dieser Seite zu gelangen. Der Inhalt des Bereichs besteht aus der relativen Byteadresse auf der Seite bis zum Anfang des gesuchten Datensatzes. Wenn dem Datensatz eine Satz-ID zugeordnet ist, wird diese erst wieder bei einer Reorganisation der Tabelle geändert.

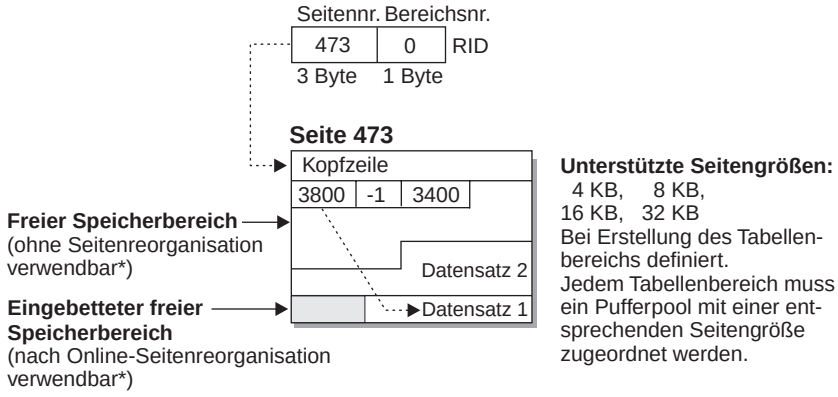


Abbildung 5. Datenseiten und RID-Format

Wenn eine Tabelle reorganisiert wird, wird eingebetteter freier Speicherbereich, der nach dem Löschen eines Datensatzes auf der Datenseite verbleibt, in verwendbaren freien Speicherbereich konvertiert. Satz-IDs werden basierend auf dem Verschieben von Datensätzen auf einer Datenseite erneut definiert, damit der verwendbare freie Speicherbereich genutzt werden kann.

DB2 unterstützt verschiedene Seitengrößen. Verwenden Sie umfangreichere Seitengrößen für Auslastungen, bei denen auf Zeilen normalerweise sequenziell zugegriffen wird. Sequenzieller Zugriff wird beispielweise für Anwendungen zur Entscheidungshilfe oder in Fällen verwendet, in denen ein starker Gebrauch von temporären Tabellen gemacht wird. Verwenden Sie geringere Seitengrößen für Auslastungen, deren Zugriff normalerweise wahlfrei erfolgt. Wahlfreier Zugriff wird beispielsweise in OLTP-Umgebungen verwendet.

Weitere Informationen zur Reorganisation einer Tabelle finden Sie im Abschnitt „Reorganisieren von Katalogen und Benutzertabellen" auf Seite 313.

Speicherbereichsverwaltung

Sie verwenden die SQL-Anweisung INSERT, um neue Informationen in eine Tabelle einzufügen. Wenn Sie dies tun, wird ein INSERT-Suchalgorithmus verwendet, um diese Arbeit durchzuführen. Zuerst werden die FSCRs (Free Space Control Records, Steuersätze für freien Speicherbereich) dazu verwendet, eine Seite mit genügend Speicherbereich zu finden. Selbst wenn der FSCR angibt, dass auf einer bestimmten Seite genügend freier Speicherbereich vorhanden ist, kann dieser jedoch möglicherweise nicht verwendet werden, da er durch eine nicht festgeschriebene Anweisung DELETE von einer anderen Transaktion „reserviert" ist. Daher sollten Sie sicherstellen, dass alle Transakti-

onen häufig COMMIT-Operationen zum Festschreiben ausführen, da andernfalls nicht festgeschriebener freigewordener Speicherbereich nicht verwendet werden kann.

Nicht alle FSCRs in einer Tabelle werden durchsucht. Die Registrierungsvariable DB2MAXFSCRSEARCH begrenzt die Zahl der FSCRs, die bei der Durchführung einer Anweisung INSERT in Betracht gezogen werden. Der Standardwert für diese Registrierungsvariable ist fünf. Wenn innerhalb von fünf FSCRs kein Speicherbereich gefunden wird, wird der einzufügende Datensatz an das Ende der Tabelle angehängt. Darüber hinaus werden zur Optimierung der Geschwindigkeit der Anweisung INSERT nachfolgende Datensätze ebenfalls am Ende der Tabelle angehängt, bis zwei Speicherbereiche gefüllt sind. Wenn diese beiden Speicherbereiche gefüllt sind, setzt die nächste Anweisung INSERT die Suche bei dem FSCR fort, bei dem die letzte Suche beendet wurde.

Anmerkung: Der Wert der Variablen DB2MAXFSCRSEARCH ist wichtig. Legen Sie diese Registrierungsvariable auf einen kleinen Wert fest, um die Geschwindigkeit von INSERT (möglicherweise mit der negativen Auswirkung eines rascheren Anwachsens der Tabelle) zu optimieren. Legen Sie diese Registrierungsvariable auf einen großen Wert fest, um die erneute Verwendung von Speicherbereich (möglicherweise mit der negativen Auswirkung einer geringeren Verarbeitungsgeschwindigkeit der Anweisung INSERT) zu optimieren.

Wenn die gesamte Tabelle durchsucht wurde, wird der einzufügende Datensatz ohne weitere Suchaktionen angehängt. Suchaktionen unter Verwendung der FSCRs werden erst dann wieder durchgeführt, wenn an einer Stelle in der Tabelle Speicherbereich frei wird (beispielsweise als Folge einer DELETE-Anweisung).

Es gibt zwei weitere Optionen für Suchalgorithmen. Die erste ist APPEND MODE. In diesem Modus werden neue Zeilen immer an das Ende der Tabelle angehängt. Hierbei finden weder Such- noch Pflegeoperationen für FSCRs statt. Diese Option wird unter Verwendung der Anweisung ALTER TABLE APPEND ON aktiviert und kann die Leistung für ausschließlich wachsende Tabellen wie Journale verbessern. Die zweite Wahlmöglichkeit besteht im Definieren eines Clusterungsindex für die betreffende Tabelle. In diesem Fall versucht der Datenbankmanager, Datensätze auf derselben Seite wie andere Datensätze mit ähnliche Indexschlüsselwerten einzufügen. Wenn auf der betreffenden Seite kein Speicherbereich verfügbar ist, wird versucht, den Datensatz auf den umgebenden Seiten unterzubringen.

Wenn die Operation danach noch immer nicht erfolgreich war, wird der oben beschriebene FSCR-Suchalgorithmus verwendet – mit einem kleinen Unterschied: diesmal wird der am schlechtesten passende Speicherbereich gesucht, nicht der erste passende. Durch diesen Ansatz werden normalerweise Seiten ausgewählt, die mehr freien Speicherbereich enthalten. Dadurch kann ein neuer Clusterungsbereich für Zeilen mit dem betreffenden Schlüsselwert geschaffen werden.

Wenn Sie einen Clusterungsindex für eine Tabelle definieren, verwenden Sie ALTER TABLE... PCTFREE, bevor Sie die betreffende Tabelle laden oder reorganisieren. Die Klausel PCTFREE belässt den Prozentsatz, der als freier Speicherbereich angegeben wurde, nach dem Laden und Reorganisieren auf der Datenseite dieser Tabelle. Dadurch steigt die Wahrscheinlichkeit, dass bei der Clusterungsindexoperation auf der gewünschten Seite freier Speicherbereich gefunden wird.

Datenseiten und Überlaufsätze

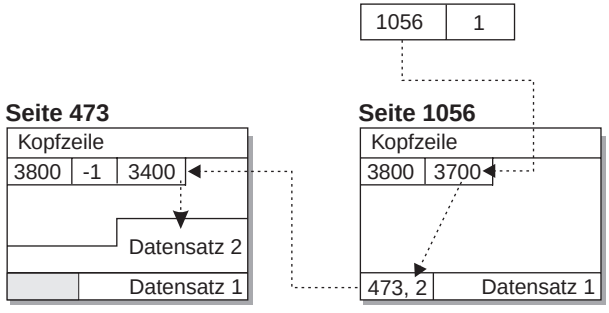


Abbildung 6. Datenseite und Überlaufsätze

Überlaufsätze sind möglich, wenn eine Aktualisierungsanforderung einen bestehenden Datensatz so vergrößert, dass dieser nicht mehr auf die aktuelle Seite passt. Der vergrößerte Datensatz wird als Überlaufsatz auf einer anderen Seite eingefügt, auf der genügend Platz vorhanden ist. Die ursprüngliche Satz-ID wird in einen Zeigerdatensatz konvertiert, der die neue Satz-ID des Überlaufsatzes enthält. In den Indizes für die Tabelle bleibt die ursprüngliche Satz-ID gespeichert. Ein zusätzlicher Lesevorgang für eine Seite ist erforderlich, um zum angeforderten Datensatz zu gelangen. Viele Überlaufsätze bedeuten viele zusätzliche Lesevorgänge für Seiten und geringere Leistung beim Zugriff auf die Tabelle. Durch die Reorganisation der Tabelle werden Überlaufsätze beseitigt. Wenn möglich, sollten Sie jedoch Aktualisierungsanforderungen, die Datensätze vergrößern, und dadurch Überlaufsätze vermeiden.

Indexverwaltung

DB2-Indizes sind eine optimierte B-Baumstrukurimplementierung auf der Basis einer effizienten Indexverwaltungsmethode mit einer hohen Zahl gemeinsamer Zugriffe unter Verwendung von WAL (Write Ahead Logging, vorausschreibende Protokollierung).

Die optimierte B-Baumstrukturimplementierung verfügt über bidirektionale Zeiger auf den Blattseiten, mit denen ein einzelner Index sowohl vorwärts als auch rückwärts gerichtete Suchoperationen unterstützen kann (jedoch nicht in beide Richtungen gleichzeitig). Teilungen von Indexseiten befinden sich normalerweise genau in der Mitte, wobei die HIGHKEY-Seite eine Ausnahme bildet, bei der eine 90/10-Teilung verwendet wird. Das heißt, dass die hohen zehn Prozent der Indexschlüssel auf der neuen Seite Platz finden. Diese Art der Indexseitenteilung ist bei Auslastungen nützlich, bei denen INSERT-Anforderungen oft mit neuen HIGHKEY-Werten abgeschlossen werden.

Seiten im Index werden freigegeben, wenn der letzte Indexschlüssel auf der betreffenden Seite entfernt wird. Eine Ausnahme zu dieser Regel tritt auf, wenn beim Erstellen des Index die Klausel MINPCTUSED ausgewählt wird. Die Verwendung dieser Klausel gibt an, dass der Index online reorganisiert werden kann; darüber hinaus dient der in dieser Klausel angegebene Wert als Schwellenwert für den Mindestprozentsatz an Speicherbereich, der auf den Indexseiten verwendet wird. Wenn nach dem Löschen eines Schlüssels in einer Indexseite der Prozentsatz an auf der Seite verwendetem Speicherbereich bei oder unter dem angegebenen Wert liegt, wird versucht, die verbleibenden Schlüssel mit denen von einer benachbarten Seite zusammenzufügen. Wenn genügend Platz vorhanden ist, wird die Operation zum Zusammenfügen ausgeführt und eine Indexseite gelöscht. Die Verwendung dieser Klausel kann die erneute Verwendung von Speicherbereich verbessern; wenn der verwendete Wert jedoch zu hoch ist, steigt die zum Zusammenfügen nötige Zeit, und die Erfolgsaussichten für diese Operation werden gleichzeitig geringer. Es empfiehlt sich, dass der Wert für diese Klausel immer geringer als 50 Prozent ist.

Die Klausel INCLUDE der Anweisung CREATE INDEX ermöglicht das Einfügen der angegebenen Spalte(n) auf den Indexseiten zusätzlich zu den Schlüsselspalten. Dadurch kann die Zahl der Abfragen steigen, bei denen reiner Indexzugriff möglich ist. Gleichzeitig können jedoch die Erfordernisse für Indexbereich und möglicherweise auch die Wartungskosten für den betreffenden Index steigen, wenn die eingefügten Spalten häufig aktualisiert werden. Das Ordnen der Index-B-Baumstruktur geschieht nur unter Verwendung der Schlüsselspalten, nicht der eingeschlossenen Spalten.

Sperren

Der Datenbankmanager bietet mit Hilfe von Sperren Steuerung des gemeinsamen Zugriffs und verhindert damit ungesteuerten Zugriff auf Ressourcen und Daten. Eine **Sperre** ordnet eine Anwendung einer Datenbankmanagerressource oder einem Datenbankmanagerdatensatz zu. Die Sperre steuert die Art und Weise, in der andere Anwendungen auf dieselbe Ressource bzw. denselben Datensatz zugreifen können.

Der Datenbankmanager verwendet Sperren entweder auf Datensatz- oder auf Tabellenebene, wobei folgende Faktoren als Basis für die Auswahl der jeweils geeigneten Sperre dienen:

- Die Isolationsstufe, die bei der Vorkompilierung oder beim Binden einer Anwendung an die Datenbank angegeben wird. Die Isolationsstufe kann Folgendes sein: Nicht festgeschriebener Lesevorgang (Uncommitted Read, UR), Cursorstabilität (Cursor Stability, CS), Lesestabilität (Read Stability, RS) oder wiederholtes Lesen (Repeatable Read, RR). Die unterschiedlichen Isolationsstufen werden zur Steuerung des Zugriffs auf nicht festgeschriebene Daten, zur Verhinderung verlorener Aktualisierungen, zum Ermöglichen nicht wiederholter Lesevorgänge für Daten und zur Verhinderung von Phantomzeilen verwendet. Verwenden Sie die minimale Isolationsstufe, die den Bedürfnissen Ihrer jeweiligen Anwendung entspricht.
- Der vom Optimierungsprogramm ausgewählte Zugriffsplan. Tabellensuchvorgänge, Indexsuchvorgänge und andere Methoden zum Datenzugriff erfordern jeweils verschiedene Arten des Zugriffs auf die Daten.
- Das Attribut LOCKSIZE der Tabelle. Die Klausel LOCKSIZE in der Anweisung ALTER TABLE gibt die Unterteilung (Granularität) der Sperren an, die beim Zugriff auf die Tabellen verwendet werden. Ausgewählt werden können ROW für Zeilensperren oder TABLE für Tabellensperren. Verwenden Sie ALTER TABLE... LOCKSIZE TABLE für Tabellen, auf die lediglich Lesezugriff besteht. Dadurch wird die Zahl der Sperren verringert, die durch die Datenbankaktivität erforderlich sind.
- Die Speicherkapazität, die für das Sperren aufgewendet wird. Die Speicherkapazität, die für das Sperren aufgewendet wird, wird durch die den Konfigurationsparameter *locklist* (Sperrenliste) der Datenbank gesteuert. Wenn die Sperrenliste sich füllt, kann sich die Leistung aufgrund von Sperreneskalationen und geringerem gemeinsamen Zugriff auf gemeinsam verwendete Objekte in der Datenbank verringern. Steigern Sie den Wert von *locklist* und/oder *maxlocks*, wenn bei Ihnen häufig Sperreneskalationen auftreten.

Stellen Sie sicher, dass für alle Transaktionen häufig COMMIT-Operationen durchgeführt und so aktive Sperren freigegeben werden.

Im allgemeinen werden Sperren auf Datensatzebene verwendet, wenn nicht einer der folgenden Fälle auftritt:

- Die ausgewählte Isolationsstufe ist ein nicht festgeschriebener Lesevorgang (Uncommitted Read, UR).
- Die ausgewählte Isolationsstufe ist wiederholtes Lesen (Repeatable Read, RR) und der Zugriffsplan erfordert eine Suche ohne Prädikate.
- Das Attribut LOCKSIZE der Tabelle ist „TABLE".
- Die Sperrenliste wird gefüllt und Sperreneskalation tritt auf.
- Über die Anweisung LOCK TABLE erfolgt eine explizite Sperrung einer Tabelle. Die Anweisung LOCK TABLE hindert gleichzeitig ablaufende Anwendungsprozesse daran, eine Tabelle zu ändern bzw. zu verwenden.

Eine **Sperreneskalation** ist die Umwandlung einer oder mehrerer Satzsperren in eine Tabellensperre. Eine exklusive Sperrerweiterung ist eine Sperreneskalation, bei der die aufgetretene Tabellensperre eine exklusive Sperre ist. Sperreneskalationen verringern den gemeinsamen Zugriff und sollten vermieden werden.

Die Dauer der Satzsperre ist abhängig von der verwendeten Isolationsstufe:

- UR-Suche (Uncommitted Read, nicht festgeschriebener Lesevorgang): Keine Satzsperren sind aktiv, außer wenn sich ein Satz ändert.
- CS-Suche (Cursor Stability, Cursorstabilität): Satzsperren sind nur dann aktiv, während sich der Cursor auf dem Satz befindet.
- RS-Suche (Read Stability, Lesestabilität): Nur qualifizierende Satzsperren sind für die Dauer der Transaktion aktiv.
- RR-Suche (Repeatable Read, Wiederholtes Lesen): Alle Satzsperren sind für die Dauer der Transaktion aktiv. Wenn Sie sich in einer Umgebung befinden, bei der diese Isolationsstufe nicht gewünscht oder nicht benötigt wird, verwenden Sie die Registrierungsvariable DB2_RR_TO_RS. Dadurch wird der Datenbankmanager angewiesen, die zum Aktivieren der RR-Semantik erforderlichen zusätzlichen Sperren zu vermeiden, was zu einer Leistungssteigerung führt.

Weitere Informationen zu diesem Thema finden Sie im Abschnitt „Sperren" auf Seite 57.

Protokollierung

Zwei Protokollierungsstrategien stehen zur Auswahl:

- Umlaufprotokollierung, wobei die Protokollsätze die Protokolldateien füllen und anschließend die ersten Protokollsätze in der ersten Protokolldatei überschreiben. Die überschriebenen Protokollsätze können nicht wiederhergestellt werden.
- Protokollspeicherung, wobei eine Protokolldatei archiviert wird, wenn sie mit Protokollsätzen gefüllt ist. Für weitere Protokollsätze werden neue Protokolldateien verfügbar gemacht. Die Protokollspeicherung aktiviert die **aktualisierende Wiederherstellung**. Die aktualisierende Wiederherstellung wendet Änderungen an der Datenbank auf Basis von abgeschlossenen Arbeitseinheiten (Transaktionen) erneut an, die im Protokoll aufgezeichnet wurden. Es kann angegeben werden, dass die aktualisierende Wiederherstellung bis zum Ende der Protokolle oder bis zu einem bestimmten Zeitpunkt vor dem Ende der Protokolle durchgeführt wird.

Unabhängig von der getroffenen Auswahl werden alle Änderungen an regulären Daten- und Indexseiten in den Protokollpuffer geschrieben. Die Speicherung der Daten im Protokollpuffer auf Platte wird nur in den folgenden Fällen erzwungen:

- Bevor die Speicherung der entsprechenden Datenseiten auf Platte erzwungen wird. Dies wird als „vorausschreibende Protokollierung" (Write Ahead Logging, WAL) bezeichnet.
- Nach einer COMMIT-Operation oder nachdem der Wert des Datenbankkonfigurationsparameters Anzahl der Gruppenfestschreibungen (*mincommit*) erreicht wurde.
- Wenn der Protokollpuffer fast voll ist. Der Protokollierungsprozess schreibt Protokolldaten auf Platte, um die Bedingung „Protokollpuffer voll" zu vermeiden.

Anmerkung: Zu dem Zeitpunkt, an dem die Transaktion unter Verwendung der Anweisung COMMIT abgeschlossen wird, werden auch alle geänderten Seiten auf Platte geschrieben, um so die Wiederherstellbarkeit sicherzustellen.

Wenn die Transaktionen kurz sind, kann die Protokoll-E/A aufgrund der Häufigkeit der Speicherung des Protokolls bei COMMIT-Operationen ein „Engpass" werden. In solchen Umgebungen kann der „Engpass" durch das Festlegen des Konfigurationsparameters *mincommit* auf einen Wert größer als eins entfernt werden.

Wenn ein Wert verwendet wird, der größer als eins ist, werden die COMMIT-Operationen für mehrere Transaktionen angehalten oder „gestapelt". Die erste Transaktion, für die eine COMMIT-Operation durchgeführt werden soll, wartet bis (*mincommit* - 1) dies für mehrere Transaktionen der Fall ist; anschließend wird die Speicherung des Protokolls auf Platte erzwungen und alle Transaktionen antworten ihren Anwendungen. Dies führt zu einer einzigen Protokoll-E/A anstatt von mehreren einzelnen Protokoll-E/As.

Um die übermäßige Verschlechterung der Antwortzeit zu vermeiden, warten alle Transaktionen nur maximal eine Sekunde auf (*mincommit* - 1) COMMIT-Operationen anderer Transaktionen. Wenn diese Sekunde verstrichen ist, erzwingt die betreffende wartende Transaktion das Protokoll selbst und antwortet auf ihre Anwendung. Dadurch kann der Parameter *mincommit* festgelegt werden, wobei Sie sich andererseits aber nicht zu viele Gedanken über die Leistung in Zeiträumen machen müssen, in denen weniger Transaktionen verarbeitet werden.

Änderungen an großen Objekten (LOBs) und LONG VARCHARs werden mit Hilfe des Erstellens einer Spiegelkopie für Speicherseiten (Shadow Paging) protokolliert. Änderungen an LOB-Spalten werden nur dann protokolliert, wenn Protokollspeicherung verwendet wird und die betreffende LOB-Spalte in der Anweisung CREATE TABLE so definiert wurde, dass sie die Klausel NOT LOGGED nicht verwendet. Änderungen an den Zuordnungsseiten für LONG- oder LOB-Datentypen werden wie reguläre Datenseiten protokolliert.

Vorgänge bei der Aktualisierung

Was geschieht mit dem Protokoll und mit der Datenseite, wenn ein Agent eine Seite aktualisiert? Das hier beschriebene Protokoll minimiert die E/A, die durch die Transaktion erforderlich ist, und stellt zudem die Wiederherstellbarkeit sicher.

Zuerst wird die zu aktualisierende Seite mit einer exklusiven Sperre verriegelt. In den Protokollpuffer wird ein Protokollsatz geschrieben, der beschreibt, wie die Änderung wiederholt und widerrufen werden kann. Während dieser Aktion wird auch eine Protokollfolgenummer (Log Sequence Number, LSN) empfangen und in der Kopfzeile der gerade aktualisierten Seite gespeichert. Anschließend wird die Änderung an der Seite vorgenommen. Zuletzt wird die Seite entsperrt. Die Seite gilt als „benutzt", da es darin Änderungen gibt, die nicht auf Platte gespeichert wurden. Der Protokollpuffer wurde ebenfalls aktualisiert.

Sowohl die Speicherung der Daten im Protokollpuffer als auch der „benutzten" Datenseite auf Platte muss anschließend erzwungen werden. Damit die Leistung nicht beeinträchtigt wird, werden diese E/As bis zu einem geeigneten Zeitpunkt (z. B. während einer Phase mit niedriger Systembelastung), bis zu einem Zeitpunkt, zu dem ihre Ausführung zum Sicherstellen der Wiederherstellbarkeit erforderlich ist, oder bis zum geplanten Wiederherstellungszeitpunkt verzögert. Genauer gesagt, wird die Speicherung einer „benutzten" Seite auf Platte in folgenden Fällen durchgeführt:

- Wenn ein anderer Agent sie auswählt.
- Wenn eine Seitenlöschfunktion die Seite aus einem der folgenden Gründe bearbeitet:
  - Ein anderer Agent wählt sie aus.
  - Der Prozentsatz des Konfigurationsparameters *chngpgs_thresh* der Datenbank wird überschritten. Wenn dies der Fall ist, werden asynchrone Seitenlöschfunktionen „aktiv" und schreiben geänderte Seiten auf Platte.
  - Der Prozentsatz des Konfigurationsparameters *softmax* der Datenbank wird überschritten. Wenn dies der Fall ist, werden asynchrone Seitenlöschfunktionen „aktiv" und schreiben geänderte Seiten auf Platte.
- Wenn die Seite als Teil einer Tabelle aktualisiert wurde, für die die Klausel NOT LOGGED INITIALLY aufgerufen wird, und eine COMMIT-Operation ausgegeben wird. Zum Zeitpunkt der COMMIT-Operation werden alle geänderten Seiten auf Platte geschrieben, um die Wiederherstellbarkeit sicherzustellen.

Die Speicherung eines Protokollpuffers auf Platte wird in den folgenden Fällen durch die Protokollfunktions-EDU (Engine Dispatchable Unit, zuteilbare Einheit der Steuerkomponente) erzwungen:

- Bevor die Speicherung der entsprechenden Datenseiten auf Platte erzwungen wird. Dies wird als „vorausschreibende Protokollierung" (Write Ahead Logging, WAL) bezeichnet.
- Nach einer COMMIT-Operation oder nachdem der Wert des Datenbankkonfigurationsparameters Anzahl der Gruppenfestschreibungen (*mincommit*) erreicht wurde.
- Wenn der Protokollpuffer fast voll ist. Der Protokollierungsprozess schreibt Protokolldaten auf Platte, um die Bedingung „Protokollpuffer voll" zu vermeiden.

Prozessmodell

Lokale und ferne Anwendungsprozesse können mit derselben Datenbank arbeiten. Eine ferne Datenbank initiiert eine Datenbankaktion von einer Maschine aus, die sich von der Datenbankmaschine entfernt befindet. Lokale Anwendungen sind an der Server-Maschine direkt mit der Datenbank verbunden.

Alle Kreise in der folgenden Abbildung stellen EDUs (Engine Dispatchable Units, zuteilbare Einheiten der Steuerkomponente) dar, die auf UNIX-Plattformen als „Prozesse" und auf Windows NT- und OS/2-Plattformen als „Threads" bekannt sind.

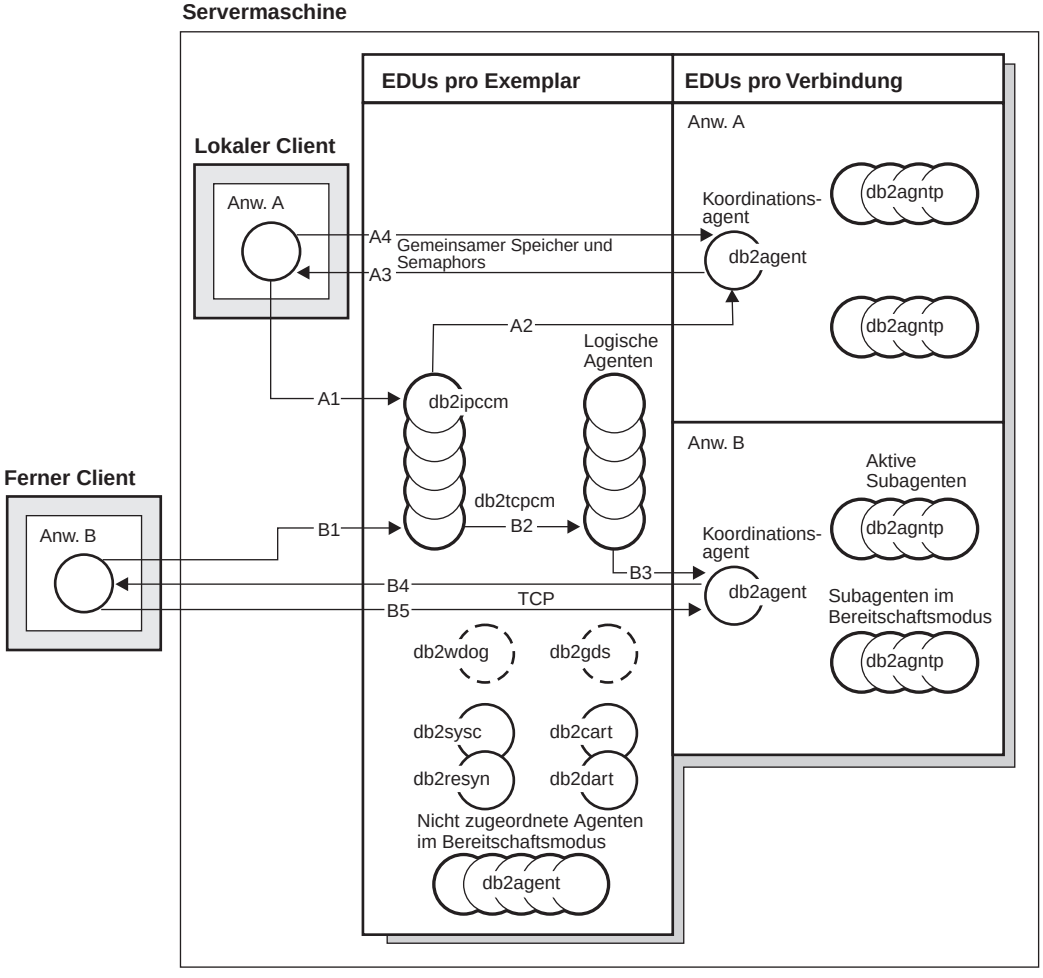


Abbildung 7. Prozessmodell - Übersicht

Bevor die Arbeit ausgeführt werden kann, die für die Anwendung in der Datenbank durchgeführt werden soll, muss zwischen einer Anwendung und dem Datenbankmanager ein Kommunikationsmittel eingerichtet werden.

In der oben stehenden Abbildung richtet ein lokaler Client bei A1 die Kommunikation ein, indem er zuerst mit der EDU ″db2ipccm″ (Engine Dispatchable Unit, zuteilbare Einheit der Steuerkomponente) arbeitet. Diese EDU arbeitet bei A2 mit einer EDU ″db2agent″, die zum Koordinationsagenten für die Anwendungsanforderungen vom lokalen Client wird. Der Koordinationsagent nimmt bei A3 mit der Client-Anwendung Kontakt auf und richtet bei A4 zwischen der Client-Anwendung und der Datenbank eine Kommunikation mit gemeinsam benutztem Speicher und Semaphors ein. Zwischen der Anwendung auf dem lokalen Client und der Datenbank wird eine Verbindung hergestellt.

In der oben stehenden Abbildung richtet ein ferner Client bei B1 die Kommunikation ein, indem er zuerst mit der EDU ″db2tcpcm″ arbeitet. Wenn ein anderes Kommunikationsprotokoll ausgewählt worden wäre, würde die entsprechende EDU verwendet werden. Die EDU ″db2tcpcm″ arbeitet bei B2 mit einem logischen Agenten. Diese EDU arbeitet bei B3 mit einer EDU ″db2agent″, die zum Koordinationsagenten für die Anwendungsanforderungen vom fernen Client wird. Der Koordinationsagent nimmt bei B4 mit der Client-Anwendung Kontakt auf und richtet bei B5 zwischen der Client-Anwendung und der Datenbank eine TCP/IP-Kommunikation ein. Zwischen der Anwendung auf dem fernen Client und der Datenbank wird eine Verbindung hergestellt.

Im folgenden sind weitere wichtige Details dieser Abbildung aufgeführt:

- Es gibt zwei Agentenklassen: einen logischen Agenten und einen Verarbeitungsagenten. Ein logischer Agent stellt eine mit dem Datenbankmanager verbundene Anwendung dar. Ein Verarbeitungsagent führt Anwendungsanforderungen aus, ist aber nicht dauerhaft mit einer bestimmten Anwendung verbunden.
- Es gibt vier Arten von Verarbeitungsagenten: aktive Koordinationsagenten (Active Coordinator Agents), Subagenten (Subagents), inaktive Agenten (Inactive Agents) und Agenten im Bereitschaftsmodus (Idle Agents).
- Jeder Prozess oder Thread einer Client-Anwendung, die durch einen logischen Agenten repräsentiert wird, wird mit einem aktiven Koordinationsagenten verbunden.

- In einer Umgebung mit partitionierten Datenbanken sowie in Umgebungen mit aktivierter partitionsinterner Parallelität verteilen die Koordinationsagenten Datenbankanforderungen an Subagenten (db2agntp). Die Subagenten führen die Anforderungen für die jeweilige Anwendung aus.
- Es gibt einen Agentenpool ("db2agent"), wo Agenten im Bereitschaftsmodus auf neue Arbeit warten.
- Es gibt andere EDUs, die Client-Verbindungen, Protokolle, zweiphasige COMMIT-Operationen, Sicherungs- und Wiederherstellungsaufgaben sowie andere Aufgaben verwalten.

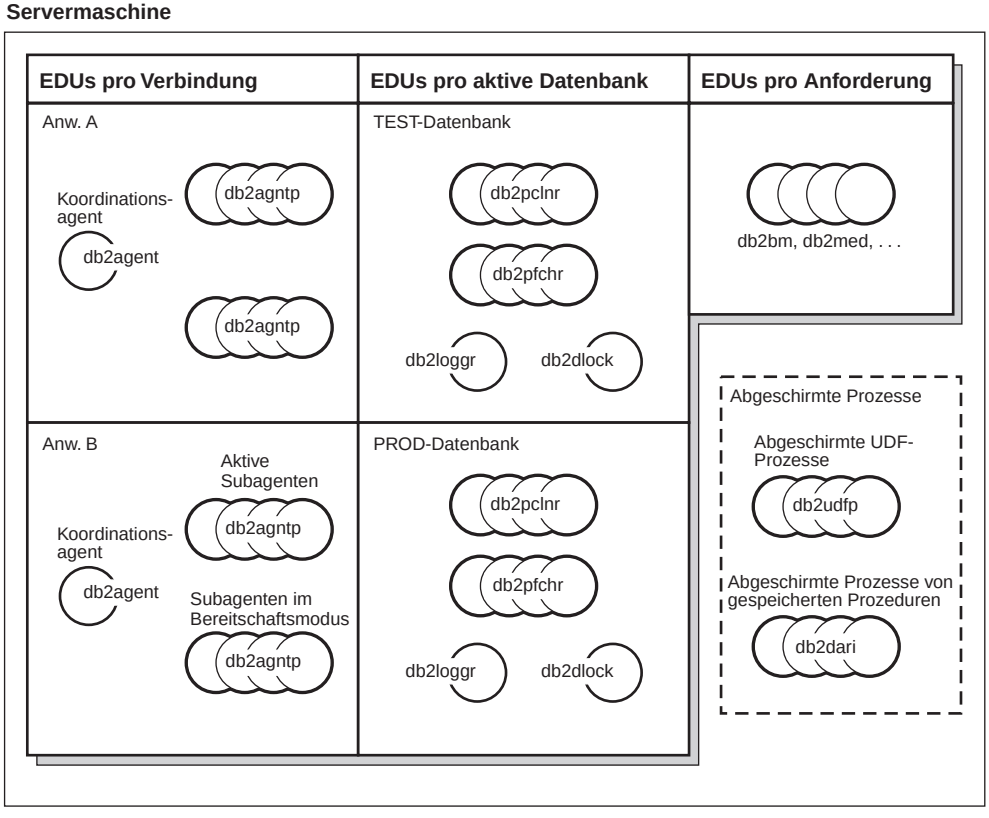


Abbildung 8. Prozessmodell Teil 2

In dieser Abbildung werden zusätzliche EDUs gezeigt (Engine Dispatchable Units, zuteilbare Einheiten der Steuerkomponente), die Teil der Server-Maschinenumgebung sind. Alle aktiven Datenbanken verfügen über eigene gemeinsam benutzte Pools von Vorablesezugriffen (″db2pfchr″) und Seitenlöschfunktionen (″db2pclnr″) sowie eigene Protokollfunktionen (″db2loggr″) und Detektoren für gegenseitiges Sperren (″db2dlock″).

Die Kreise mit den Namen „db2udfp″ und „db2dari″ unten rechts in der Abbildung stellen Prozesse dar, die in DB2 Universal Database als abgeschirmte UDFs (User-Defined Functions, benutzerdefinierte Funktionen) bzw. gespeicherte Prozeduren ausgeführt werden. Diese Prozesse werden verwaltet, um den Aufwand in Zusammenhang mit ihrer Erstellung und Zerstörung zu minimieren. Die Standardeinstellung für den Konfigurationsparameter *keepdari* des Datenbankmanagers ist „YES″ (Ja), wodurch der Prozess der gespeicherten Prozedur zur erneuten Verwendung beim nächsten Aufruf einer gespeicherten Prozedur verfügbar bleibt.

Anmerkung: Es gibt auch nicht abgeschirmte UDFs und gespeicherte Prozeduren, die direkt im Adressraum eines Agenten ausgeführt werden. Diese Vorgehensweise führt zu einer besseren Leistung. Da diese UDFs und gespeicherten Prozeduren jedoch über unbeschränkten Zugriff auf den Adressraum des Agenten verfügen, müssen sie vor ihrer Verwendung umfassend getestet werden.

Weiter Informationen hierzu finden Sie im Kapitel über gespeicherte Prozeduren im Handbuch *Application Development Guide*.

Das Prozessmodell mit mehreren Partitionen ist eine logische Erweiterung des Prozessmodells mit einer einzigen Partition. Beide Betriebsarten werden durch eine einzige allgemeine Codebasis unterstützt. Die folgende Abbildung stellt die Ähnlichkeiten und Unterschiede zwischen dem Prozessmodell mit einer einzigen Partition, wie in den vorangehenden zwei Abbildungen gezeigt, und dem Prozessmodell mit mehreren Partitionen dar.

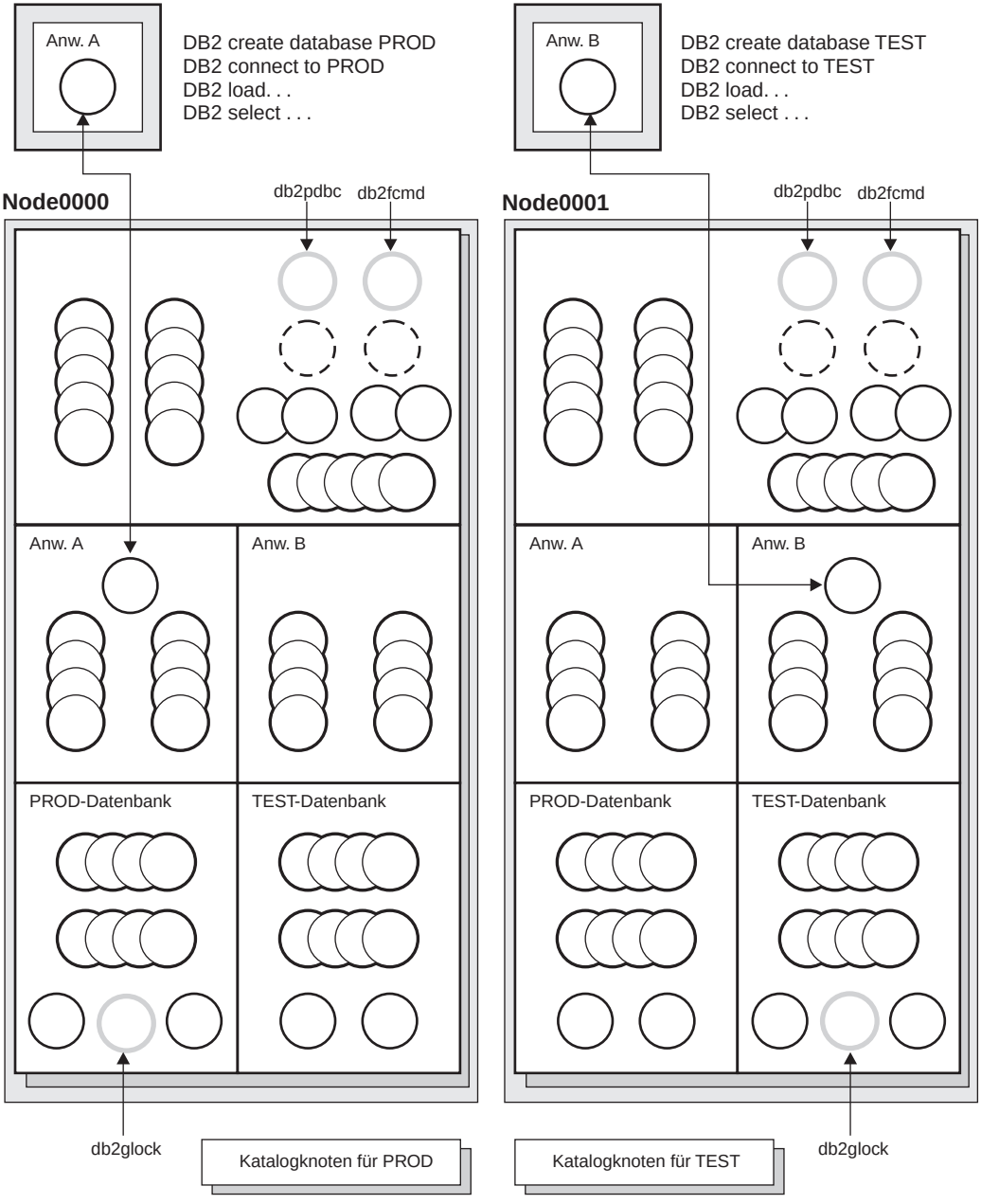


Abbildung 9. Prozessmodell und mehrere Partitionen

Die Mehrzahl der EDUs entsprechen beim Prozessmodell mit einer einzigen Partition denen beim Prozessmodell mit mehreren Partitionen.

In einer Umgebung mit mehreren Partitionen (bzw. in einer Knotenumgebung) ist eine der Partitionen der Katalogknoten. Der Katalog speichert alle Information in Bezug auf Objekte in der Datenbank.

Wie in der oben stehenden Abbildung gezeigt, wird der Katalog der PROD-Datenbank im Knoten Node0000 erstellt, weil Anwendung A die PROD-Datenbank in diesem Knoten erstellt. Ebenso wird der Katalog für die TEST-Datenbank im Knoten Node0001 erstellt, weil Anwendung B die TEST-Datenbank in diesem Knoten erstellt. Die Datenbanken sollten in verschiedenen Knoten erstellt werden, um so die zusätzliche Aktivität in Zusammenhang mit den Katalogen für jede Datenbank gleichmäßig auf die Knoten in der Systemumgebung zu verteilen.

Dem Exemplar sind zusätzliche EDUs (″db2pdbc″ und ″db2fcmd″) zugeordnet, die sich in jedem Knoten in einer Datenbankumgebung mit mehreren Partitionen befinden. Diese EDUs werden für die Koordination von Anforderungen zwischen Datenbankpartitionen und zum Aktivieren des FCM (Fast Communications Manager) benötigt.

Dem Katalogknoten der Datenbank ist ebenfalls eine zusätzliche EDU (″db2glock″) zugeordnet. Diese EDU steuert globale Sperren für die Knoten, in denen sich die aktive Datenbank befindet.

Alle CONNECT-Operationen von Anweisungen werden durch einen logischen Agenten in der Datenbank repräsentiert und führen zu einem einzigen Koordinationsagenten. Der Koordinationsagent ist in der Partition vorhanden, zu der eine Verbindung von der Anwendung besteht. Diese Partition wird dann zum „Koordinatorknoten” für die betreffende Anwendung. Der Koordinatorknoten kann auch unter Verwendung des Befehls *SET CLIENT CONNECT_NODE* definiert werden. Teile der Datenbankanforderungen von der Anwendung werden vom Koordinatorknoten an Subagenten der anderen Partitionen gesendet; zudem werden alle Ergebnisse der anderen Partitionen am Koordinatorknoten konsolidiert, bevor sie zurück zur Anwendung gesendet werden.

Die Datenbankpartition, in der der Befehl CREATE DATABASE ausgegeben wurde, wird „Katalogknoten” der Datenbank genannt. In dieser Datenbankpartition werden die Katalogtabellen gespeichert. Normalerweise werden alle Benutzertabellen über eine Reihe von Knoten hinweg partitioniert.

Anmerkung: Eine beliebige Anzahl von Partitionen kann zur Ausführung auf einer einzigen Maschine konfiguriert werden. Dies ist als Konfiguration mit „mehreren logischen Partitionen" oder mit „mehreren logischen Knoten" bekannt. Eine solche Konfiguration ist bei großen SMP-Maschinen (SMP - Symmetric Multiprocessor, symmetrischer Mehrprozessor) mit einem sehr großen Hauptspeicher sehr nützlich. In einer derartigen Umgebung kann die Kommunikation zwischen Partitionen durch die Verwendung von gemeinsam benutztem Speicher und Semaphors optimiert werden.

Speichermodell

Speicher ist wichtig, da er eine starke Auswirkung darauf hat, wie Arbeit in der Datenbank verrichtet wird. Die Aufteilung des verfügbaren Speichers auf die Bereiche in der Datenbank stellt eine primäre Methode dar, die Leistung der Datenbank zu beeinflussen. Die Steuerung dieser Aufteilung des Speichers auf verschiedene Zwischenspeicher erfolgt über Konfigurationsparameter. Die zentralen Konfigurationsparameter und die von ihnen gesteuerten Teile des Speichers werden in diesem Abschnitt behandelt. Weitere Informationen zu diesem Thema finden Sie im Abschnitt „Verwendung des Speichers durch DB2" auf Seite 283.

Alle EDUs (Engine Dispatchable Unit, zuteilbare Einheit der Steuerkomponente) in einer Partition sind mit dem gemeinsam benutzten Speicher des Exemplars verbunden. Alle in einer Datenbank arbeitenden EDUs sind mit dem gemeinsam benutzten Speicher dieser Datenbank verbunden. Alle für eine bestimmte Anwendung arbeitenden EDUs sind mit einer Region eines gemeinsam benutzten Speichers dieser Anwendung verbunden. Diese Art von gemeinsam benutztem Speicher wird nur zugeordnet, wenn partitionsinterne oder partitionsübergreifende Parallelität aktiviert ist. Zuletzt verfügen alle EDUs über einen eigenen privaten Speicher.

Gemeinsam benutzter Speicher des Exemplars (auch als gemeinsam benutzter Speicher des Datenbankmanagers bekannt) wird beim Start der Datenbank zugeordnet. Vom gemeinsam benutzten Speicher des Exemplars wird der gesamte weitere Speicher verbunden/zugeordnet. Wenn der FCM (Fast Communications Manager) verwendet wird, werden diesem Speicher Puffer entnommen. Der FCM wird für die interne Kommunikation, in erster Linie Nachrichten, zwischen und innerhalb der Datenbank-Server in einer bestimmten Datenbankumgebung verwendet. Wenn die erste Anwendung die Verbindung zu einer Datenbank herstellt, werden gemeinsam benutzte Speicherbereiche der Datenbank, gemeinsame Anwendungs- und private Agentenspeicherbereiche zugeordnet.

Gemeinsam benutzter Speicher der Datenbank (auch als globaler Datenbankspeicher bekannt) wird zugeordnet, wenn eine Datenbank aktiviert oder zum ersten Mal eine Verbindung zu ihr hergestellt wird. Dieser Speicher wird für alle Anwendungen verwendet, die möglicherweise eine Verbindung zur Datenbank herstellen. In einem gemeinsam benutzten Speicher einer Datenbank sind viele verschiedene Speicherbereiche enthalten. Unter anderem sind dies:

- Pufferpools
- Sperrenliste
- Datenbankzwischenspeicher – dies umfasst auch den Protokollpuffer und den Katalog-Cache.
- Zwischenspeicher für Dienstprogramme
- Paket-Cache

Der Konfigurationsparameter *numdb* des Datenbankmanagers gibt die Zahl der lokalen Datenbanken an, die gleichzeitig aktiv sein können. In einer Umgebung mit partitionierten Datenbanken beschränkt dieser Parameter die Anzahl aktiver Datenbankpartitionen auf einem Datenbankpartitions-Server. Der Wert des Parameters *numdb* kann Auswirkungen auf die Gesamtmenge an Speicher haben, die zugeordnet wird.

Gemeinsamer Anwendungsspeicher (auch als globaler Anwendungsspeicher bekannt) wird zugeordnet, wenn eine Verbindung zwischen einer Anwendung und einer Datenbank hergestellt wird. Diese Zuordnung wird nur in einer Umgebung mit partitionierten Datenbanken oder bei einem aktivierten Konfigurationsparameter *intra_parallel* des Datenbankmanagers durchgeführt. Dieser Speicher wird von Agenten verwendet, die im Auftrag der Anwendung für die gemeinsame Verwendung von Daten sorgen und untereinander Aktivitäten koordinieren.

Der Konfigurationsparameter *maxappls* der Datenbank legt eine obere Begrenzung für die Zahl der Anwendungen fest, von denen aus eine Verbindung zu einer Datenbank hergestellt wird. Da für jede Anwendung, die die Verbindung zu einer Datenbank herstellt, privater Speicher zugeordnet wird, entsteht durch Erhöhung dieses Parameters für gleichzeitig ablaufende Anwendungen möglicherweise ein größerer Speicherbedarf.

Bis zu einem gewissen Grad wird die maximale Anzahl von Anwendungen auch vom Konfigurationsparameter *maxagents* des Datenbankmanagers (bzw. *max_coordagents* bei Umgebungen mit partitionierten Datenbanken) bestimmt. Der Parameter *maxagents* legt eine obere Begrenzung für die Gesamtzahl an Datenbankmanageragenten in einer Partition fest. Diese Datenbankmanageragenten umfassen aktive Koordinationsagenten (Active Coordinator Agents), Subagenten (Subagents), inaktive Agenten (Inactive Agents) sowie Agenten im Bereitschaftsmodus (Idle Agents).

Ein privater Agentenspeicher wird für einen Agenten zugeordnet, wenn der betreffende Agent zur Arbeit für eine bestimmte Anwendung zugeordnet wird. Der private Agentenspeicher wird für den Agenten zugeordnet und enthält Speicherzuordnungen, die nur von diesem bestimmten Agenten verwendet werden, wie beispielsweise den Sortierspeicher und den Zwischenspeicher für Anwendungen.

Es gibt ein paar spezielle Arten von gemeinsam benutztem Speicher:

- Gemeinsam benutzter Speicher von Agent/lokaler Anwendung. Dieser Speicher wird zur Kommunikation von SQL-Anforderungen und -Antworten zwischen einem Agenten und der zugehörigen Client-Anwendung verwendet.
- Gemeinsam benutzter Speicher von UDF/Agent. Zu diesem Speicher wird von Agenten eine Verbindung hergestellt, die eine abgeschirmte UDF oder gespeicherte Prozedur ausführen. Er wird als Kommunikationsbereich verwendet.
- Erweiterter Speicher. Ein normalerweise sehr großer (größer als 4 GB) Bereich von gemeinsam benutztem Speicher, der als erweiterter Pufferpool verwendet wird. Agenten/Vorablesezugriffe/Seitenlöschfunktionen sind nicht permanent damit verbunden, sondern stellen je nach Bedarf eine Verbindung zu einzelnen Segmenten darin her.

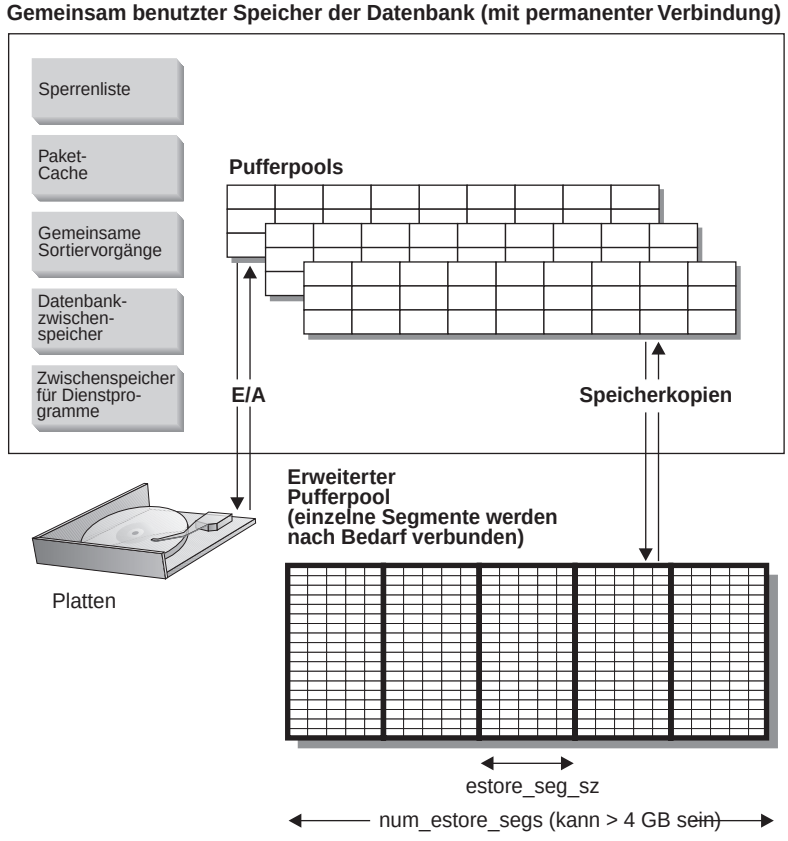


Abbildung 10. Pufferpools und erweiterter Speicher

Erweiterter Speicher dient als erweiterter Lookaside-Puffer für den/die Hauptpufferpool(s). Weitere Informationen zu diesem Thema finden Sie im Abschnitt „Erweitern von Speicher" auf Seite 330. Der erweiterte Speicher kann viel größer als 4 GB sein und stellt eine hervorragende Methode dazu dar, Maschinen mit umfangreichem Hauptspeicher zu nutzen. Der Erweiterungsspeicher-Cache wird in Speichersegmenten definiert.

Wenn Sie einen Teil des adressierbaren Realspeichers als Erweiterungsspeicher-Cache verwenden, sollte Ihnen bewusst sein, dass dieser Speicher dann nicht mehr für andere Zwecke auf der Maschine, wie beispielsweise als JFS-Cache (JFS, Journaled File System) oder als privater Adressraum für Prozesse, verwendet werden kann. Das Zuordnen von zusätzlichem adressierbarem Realspeicher zum Erweiterungsspeicher-Cache führt möglicherweise zu höherer Systemauslagerung.

Die folgenden Konfigurationsparameter der Datenbank beeinflussen die Menge und die Größe des für den erweiterten Speicher verfügbaren Speichers:

- *num_estore_segs* definiert die Zahl der Speichersegmente des erweiterten Speichers.
- *estore_seg_sz* definiert die Größe der einzelnen Segmente des erweiterten Speichers.

Jeder Tabellenbereich wird einem Pufferpool zugeordnet. Ein erweiterter Speicher-Cache muss immer einem oder mehreren bestimmten Pufferpools zugeordnet werden. Die Seitengröße des erweiterten Speicher-Cache muss mit der Seitengröße des Pufferpools übereinstimmen, dem er zugeordnet ist.

Weitere Informationen zum erweiterten Speicher-Cache finden Sie im Abschnitt „Erweitern von Speicher” auf Seite 330.

Teil 2. Optimieren der Anwendungsleistung

Kapitel 3. Überlegungen zu Anwendungen

Es gibt eine Reihe von Faktoren, die die Leistung von Anwendungen zur Laufzeit beeinflussen können. In diesem Kapitel werden folgende Themen behandelt, die beim Entwurf und bei der Codierung einer Anwendung zu berücksichtigen sind:

- Gemeinsamer Zugriff
- Sperren
- Anpassen der Optimierungsklasse
- Einschränkungen für Ergebnismengen zur Leistungsverbesserung
- Zeilenblockung
- Optimieren von Abfragen
- Compound-SQL-Anweisungen
- Informationen zur Leistung und Zeichenumsetzung
- Gespeicherte Prozeduren
- Aktivieren einer Datenbank
- Parallele Verarbeitung von Anwendungen.

Sie sollten darüber hinaus in den Handbüchern *Application Development Guide* und *CLI Guide and Reference* die zusätzlichen Informationen zu den Themen lesen, die Auswirkungen auf die Leistung der Anwendungen haben können, wie zum Beispiel:

- Schreiben von Programmen mit eingebetteten statischen SQL-Anweisungen
- Schreiben von Programmen mit eingebetteten dynamischen SQL-Anweisungen
- Schreiben von Programmen unter Verwendung von DB2 Call Level Interface (CLI)

Gemeinsamer Zugriff

Die Integrität der Daten in einer relationalen Datenbank muss gewährleistet werden, auch wenn mehrere Benutzer auf die Daten zugreifen und sie ändern. Der Begriff *gemeinsamer Zugriff* bezeichnet die Möglichkeit, dass mehrere interaktive Benutzer oder Anwendungsprogramme die Ressourcen zur gleichen Zeit verwenden können. Der Datenbankmanager steuert diesen Zugriff, um unerwünschte Folgen zu vermeiden. Solche Folgen sind zum Beispiel:

- *Verlorene Aktualisierungen*. Zwei Anwendungen A und B lesen beispielsweise dieselbe Zeile aus einer Datenbank und berechnen aufgrund der Daten, die von den Anwendungen gelesen werden, neue Werte für eine ihrer Spalten.

Die Anwendung A aktualisiert die Zeile mit ihrem neuen Wert und anschließend aktualisiert B die Zeile ebenfalls, so dass die von A durchgeführte Aktualisierung verloren geht.

- *Zugriff auf nicht festgeschriebene Daten*. Anwendung A könnte zum Beispiel einen Wert in der Datenbank aktualisieren, den Anwendung B liest, bevor er festgeschrieben wurde. Wenn dann der Wert von der Anwendung A später nicht festgeschrieben, sondern rückgängig gemacht wird, basieren die Berechnungen der Anwendung B auf nicht festgeschriebenen (und daher wahrscheinlich ungültigen) Daten.
- *Nichtwiederholbare Lesevorgänge*. Einige Anwendungen bewirken die folgende Reihenfolge von Ereignissen: Anwendung A liest eine Zeile aus der Datenbank und verarbeitet anschließend zunächst andere SQL-Anforderungen. In der Zwischenzeit ändert oder löscht Anwendung B die Zeile und schreibt diese Aktion fest. Später, wenn Anwendung A versucht, die ursprüngliche Zeile erneut zu lesen, empfängt sie die geänderte Zeile oder stellt fest, dass die ursprüngliche Zeile gelöscht wurde.
- *Das Phänomen der Phantomzeilen*. Phantomzeilen treten unter folgenden Umständen auf:
  1. Eine Anwendung führt eine Abfrage aus, die eine Menge von Zeilen nach einem Suchkriterium liest.
  2. Eine andere Anwendung fügt neue Daten ein oder aktualisiert vorhandene Daten, die die Abfragekriterien der ersten Anwendung erfüllen würden.
  3. Die erste Anwendung wiederholt die Abfrage aus Schritt 1 (innerhalb derselben Arbeitseinheit).

  Bei der Wiederholung der Abfrage (Schritt 3) werden einige zusätzliche Zeilen („Phantomzeilen") als Teil des Ergebnisses zurückgeliefert, die bei der Erstausführung der Abfrage (Schritt 1) noch nicht zurückgeliefert wurden.

Eine *Isolationsstufe* legt fest, wie Daten gesperrt oder von anderen Prozessen während des Zugriffs auf die Daten isoliert werden. Die Isolationsstufe ist während der Dauer der Arbeitseinheit in Kraft. Anwendungen, die einen Cursor verwenden, der mit der Anweisung DECLARE CURSOR und mit der Klausel WITH HOLD deklariert wurde, behalten die gewählte Isolationsstufe für die Dauer der Arbeitseinheit bei, in der die Anweisung OPEN CURSOR durchgeführt wurde. (Weitere Informationen finden Sie im Handbuch *SQL Reference*.) Lesen Sie die Informationen zur Angabe der Isolationsstufe im Abschnitt „Angeben der Isolationsstufe" auf Seite 54.

DB2 unterstützt die folgenden Isolationsstufen:

- Wiederholtes Lesen (RR)
- Lesestabilität (RS)
- Cursorstabilität (CS)
- Nicht festgeschriebener Lesevorgang (UR).

(Beachten Sie, dass einige DRDA-Datenbank-Server die Isolationsstufe *Kein Festschreiben* (No Commit) unterstützen. In anderen Datenbanken verhält sich diese Stufe wie die Isolationsstufe *Nicht festgeschriebener Lesevorgang* (Uncommited Read). Informationen zu dieser Isolationsstufe finden Sie in *SQL Reference*.)

Siehe auch:
- „Wählen der Isolationsstufe" auf Seite 53
- „Angeben der Isolationsstufe" auf Seite 54.

Sie arbeiten eventuell in einem *System mit zusammengeschlossenen Datenbanken*, das das Übergeben von SQL-Anweisungen durch Anwendungen und Benutzer unterstützt, in denen auf mehrere Datenbankverwaltungssysteme oder Datenbanken in einer einzelnen Anweisung verwiesen wird.In einem DB2-System mit zusammengeschlossenen Datenbanken besteht für Datenbankobjekte *Positionstransparenz*.Wenn zum Beispiel Informationen zu Tabellen und Sichten versetzt werden, können Verweise auf diese Informationen (so genannte *Kurznamen*) ohne Änderungen an den Anwendungen, die die Informationen anfordern, aktualisiert werden.Wenn eine Anwendung auf Kurznamen zugreift, verlässt sich DB2 auf die Protokolle zur Steuerung des gemeinsamen Zugriffs von Datenbankmanagern der Datenquellen, um Isolationsstufen sicherzustellen. (Eine *Datenquelle* besteht aus einem Datenbankverwaltungssystem und Daten.)DB2 versucht, die angeforderte Isolationsstufe an der Datenquelle mit einem logischen Äquivalent abzugleichen. Die Ergebnisse sind abhängig vom Leistungsspektrum der Datenquelle jedoch eventuell unterschiedlich. Im Handbuch *Application Development Guide* finden Sie Informationen zum Schreiben von Anwendungen, die auf Kurznamen zugreifen.

Im folgenden sind detaillierte Erläuterungen zu den Isolationsstufen aufgeführt. Diese sind in absteigender Reihenfolge bezüglich der Auswirkungen auf die Leistung, aber in aufsteigender Reihenfolge bezüglich der Vorsicht geordnet, die beim Zugriff auf und Aktualisieren von Daten erforderlich ist.

Wiederholtes Lesen (RR)

Die Isolationsstufe *Wiederholtes Lesen* (RR - Repeatable Read) sperrt alle Zeilen innerhalb einer Arbeitseinheit, auf die eine Anwendung verweist. Bei Verwendung der Isolationsstufe RR liefert eine Anweisung SELECT, die von einer Anwendung zweimal innerhalb derselben Arbeitseinheit, in der der Cursor geöffnet wurde, ausgegeben wird, jedes Mal dasselbe Ergebnis. Bei der Isolationsstufe RR können verlorene Aktualisierungen, Zugriffe auf nicht festgeschriebene Daten und Phantomzeilen nicht auftreten.

Eine Anwendung mit der Isolationsstufe RR kann, bis die Arbeitseinheit beendet wird, so oft wie erforderlich Zeilen abrufen und Aktionen an ihnen vornehmen. Andere Anwendungen können jedoch keine Zeile aktualisieren,

löschen oder einfügen, die sich auf die Ergebnistabelle auswirken würde, bis die Arbeitseinheit beendet ist. Für Anwendungen mit der Isolationsstufe RR sind nicht festgeschriebene Änderungen anderer Anwendungen nicht sichtbar.

Durch die Isolationsstufe RR werden alle Zeilen, auf die verwiesen wird, und nicht nur die Zeilen, die abgerufen werden, gesperrt. Die Sperrung wird so ausgeführt, dass eine andere Anwendung keine Zeile einfügen oder aktualisieren kann, die der Liste von Zeilen, auf die in der Abfrage verwiesen wird, hinzugefügt würde, wenn die Abfrage erneut ausgeführt würde. Dadurch wird das Auftreten von Phantomzeilen verhindert. Das bedeutet, wenn Sie zum Beispiel 10000 Zeilen unter Verwendung von Vergleichselementen durchsuchen, werden alle 10000 Zeilen gesperrt, auch wenn letzten Endes nur 10 Zeilen den Vergleichselementen entsprechen.

Anmerkung: Durch die Isolationsstufe RR wird sichergestellt, dass alle zurückgelieferten Daten bis zu dem Zeitpunkt, zu dem die Daten für die Anwendung *sichtbar* werden, unverändert bleiben, auch wenn temporäre Tabellen oder Zeilenblockung verwendet werden.

Da die Isolationsstufe RR eine beträchtliche Anzahl an Sperren erfordern kann, können diese Sperren die Anzahl von Sperren, die durch die Konfigurationsparameter *locklist* und *maxlocks* festgelegt wird, überschreiten. (Siehe „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450 und „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415.) Um eine Sperreneskalation zu vermeiden, kann das Optimierungsprogramm eventuell von vornherein eine einzige Sperre auf Tabellenebene für eine Indexsuche anfordern, wenn das Auftreten einer Sperreneskalation wahrscheinlich ist. (Informationen zur Sperreneskalation finden Sie in „Sperreneskalation" auf Seite 64.) Dies funktioniert so, als würde der Datenbankmanager für Sie eine Anweisung LOCK TABLE ausgeben. Wenn Sie keine Sperre auf Tabellenebene aktivieren lassen möchten, stellen Sie sicher, dass für die Transaktion ausreichend Sperren verfügbar sind, oder verwenden Sie die Isolationsstufe Lesestabilität (RS).

Lesestabilität (RS)

Die Isolationsstufe *Lesestabilität* (RS - Read Stability) sperrt nur die Zeilen, die von einer Anwendung innerhalb einer Arbeitseinheit abgerufen werden. Sie stellt sicher, dass jede den Vergleichselementen entsprechende gelesene Zeile während einer Arbeitseinheit nicht von Prozessen anderer Anwendungen geändert wird und dass keine, von einem Prozess einer anderen Anwendung geänderte Zeile gelesen wird, bis die Änderung von dem Prozess festgeschrieben wurde. Das heißt, „nichtwiederholbare Lesevorgänge" sind **nicht** möglich.

Anders als bei der Isolationsstufe RR (Wiederholtes Lesen) können bei der Isolationsstufe RS (Lesestabilität), wenn die Anwendung dieselbe Abfrage

mehr als einmal absetzt, zusätzliche *Phantomzeilen* auftreten (*Phänomen der Phantomzeilen*). In dem bereits angeführten Beispiel der Suche über 10000 Zeilen werden durch die Isolationsstufe RS nur die den Vergleichselementen entsprechenden Zeilen gesperrt. Da durch die Abfrage nur 10 Zeilen abgerufen werden, werden unter der Isolationsstufe RS nur für diese 10 Zeilen Sperren aktiviert. Unter der Isolationsstufe RR (Wiederholtes Lesen) werden in diesem Beispiel hingegen sämtliche 10000 Zeilen gesperrt. Bei den aktivierten Sperren kann es sich um Sperren der Modi Share, Next Share, Update oder Exclusive handeln. (Weitere Informationen zu den Sperrenattributen finden Sie in „Attribute von Sperren" auf Seite 58.)

Anmerkung: Durch die Isolationsstufe RS wird sichergestellt, dass alle zurückgelieferten Daten bis zu dem Zeitpunkt, zu dem die Daten für die Anwendung *sichtbar* werden, unverändert bleiben, auch wenn temporäre Tabellen oder Zeilenblockung verwendet werden.

Ein Zweck der Isolationsstufe RS besteht darin, sowohl einen hohen Grad des gemeinsamen Zugriffs als auch eine stabile Sicht der Daten zur Verfügung zu stellen. Um diesen Zweck zu erfüllen, stellt das Optimierungsprogramm sicher, dass Sperren auf Tabellenebene erst aktiviert werden, wenn eine Sperreneskalation auftritt. (Weitere Informationen zur Sperreneskalation finden Sie in „Sperreneskalation" auf Seite 64.)

Die Isolationsstufe RS eignet sich am besten für Anwendungen, die alle folgenden Merkmale aufweisen:
- Sie arbeiten in einer Umgebung mit gemeinsamem Zugriff.
- Sie fordern Zeilen an, die für die Dauer der Arbeitseinheit stabil bleiben müssen.
- Sie verwenden dieselbe Abfrage in einer Arbeitseinheit nur einmal, oder es ist nicht notwendig, dass bei mehrmaliger Verwendung der Abfrage innerhalb derselben Arbeitseinheit stets dieselben Abfrageergebnisse erzielt werden.

Cursorstabilität (CS)

Die Isolationsstufe *Cursorstabilität* (CS - Cursor Stability) sperrt jede Zeile, auf die von einer Transaktion einer Anwendung zugegriffen wird, während sich der Cursor auf der Zeile befindet. Diese Sperre bleibt aktiv, bis die nächste Zeile abgerufen oder die Transaktion beendet wird. Wenn jedoch Daten in einer Zeile geändert werden, muss die Sperre aktiv bleiben, bis die Änderung in der Datenbank festgeschrieben wurde.

Keine anderen Anwendungen können eine Zeile aktualisieren oder löschen, die von einer Anwendung mit der Isolationsstufe CS abgerufen wurde, wäh-

rend sich ein Aktualisierungscursor auf der Zeile befindet. Für Anwendungen mit der Isolationsstufe CS sind nicht festgeschriebene Änderungen anderer Anwendungen nicht sichtbar.

In dem bereits angeführten Beispiel einer Suche über 10000 Zeilen heißt dies, dass unter der Isolationsstufe CS lediglich eine Sperre für die Zeile, die unter der aktuellen Cursorposition liegt, aktiv ist. Die Sperre wird entfernt, wenn der Cursor von der Zeile fortbewegt wird (sofern die Zeile nicht aktualisiert wurde).

Bei der Isolationsstufe CS können sowohl unterschiedliche Abfrageergebnisse innerhalb derselben Arbeitseinheit (d. h. nicht wiederholbare Abfragevorgänge) als auch Phantomzeilen auftreten. Die Isolationsstufe CS ist die Standardisolationsstufe, die verwendet werden sollte, wenn es auf den höchstmöglichen Grad des gemeinsamen Zugriffs ankommt und nur festgeschriebene Zeilen anderer Anwendungen sichtbar sein sollen.

Nicht festgeschriebener Lesevorgang (UR)

Die Isolationsstufe *Nicht festgeschriebener Lesevorgang* (UR - Uncommitted Read) erlaubt einer Anwendung den Zugriff auf nicht festgeschriebene Änderungen anderer Transaktionen. Die Anwendung sperrt die Zeile, die sie liest, auch für keine andere Anwendung, sofern die andere Anwendung nicht versucht, die Tabelle zu löschen oder zu ändern. Die Isolationsstufe UR (Uncommitted Read) wirkt sich auf Aktualisierungscursor und Lesecursor unterschiedlich aus.

Cursor mit Lesezugriff können auf die meisten nicht festgeschriebenen Änderungen anderer Transaktionen zugreifen. Tabellen, Sichten und Indizes, die momentan von anderen Transaktionen erstellt oder gelöscht werden, sind während der Verarbeitung der Transaktion jedoch nicht verfügbar. Alle anderen Änderungen durch andere Transaktionen können gelesen werden, bevor sie festgeschrieben oder rückgängig gemacht wurden.

Anmerkung: Aktualisierungscursor, die unter der Isolationsstufe UR arbeiten, verhalten sich genauso wie unter der Isolationsstufe CS (Cursorstabilität).

Wenn Sie ein Anwendungsprogramm unter der Isolationsstufe UR ausführen, kann auch eine Anwendung unter Isolationsstufe CS ausgeführt werden. Dies liegt an der Mehrdeutigkeit der Cursor, die in den Anwendungsprogrammen verwendet werden. Mehrdeutige Cursor können im Rahmen einer BLOCKING-Option auf die Isolationsstufe CS eskaliert werden. Der Standardwert der BLOCKING-Option ist UNAMBIG. Das bedeutet, dass mehrdeutige Cursor als aktualisierbar behandelt werden und die Eskalation der Isolationsstufe auf CS durchgeführt wird. Es gibt zwei Möglichkeiten, diese Eskalation zu verhindern. Zum einen können die Cursor im Anwendungsprogramm geän-

dert und ihre Mehrdeutigkeit beseitigt werden. Dazu muss die Klausel FOR READ ONLY in die SELECT-Anweisungen eingefügt werden. Wenn die Mehrdeutigkeit der Cursor im Anwendungsprogramm beibehalten wird, können Sie zum anderen das Programm unter Angabe der Option BLOCKING ALL erneut vorkompilieren oder binden. Hierauf können alle mehrdeutigen Cursor bei der Programmausführung als Nur-Lese-Cursor behandelt werden.

Im genannten Beispiel einer Suche über 10000 Zeilen sind bei der Isolationsstufe UR keine Zeilensperren erforderlich.

Bei der Isolationsstufe UR können sowohl unterschiedliche Abfrageergebnisse innerhalb derselben Arbeitseinheit (d. h. nicht wiederholbare Abfragevorgänge) als auch Phantomzeilen auftreten.

Die Isolationsstufe UR wird in der Regel für Abfragen auf Tabellen verwendet, die sich im Lesezugriff befinden, oder wenn Anweisungen SELECT ausgeführt werden und es dabei keine Rolle spielt, ob nicht festgeschriebene Daten anderer Anwendungen angezeigt werden.

Wählen der Isolationsstufe

Tabelle 1 fasst die verschiedenen Isolationsstufen im Hinblick auf die im Handbuch *Application Development Guide* beschriebenen, unerwünschten Nebeneffekte zusammen.

Tabelle 1. Zusammenfassung der Isolationsstufen

Isolationsstufe	Zugriff auf nicht festgeschriebene Daten	Nichtwiederholbare Lesevorgänge	Phänomen der Phantomzeilen
Wiederholtes Lesen (RR)	Nicht möglich	Nicht möglich	Nicht möglich
Lesestabilität (RS)	Nicht möglich	Nicht möglich	Möglich
Cursorstabilität (CS)	Nicht möglich	Möglich	Möglich
Nicht festgeschriebener Lesevorgang (UR)	Möglich	Möglich	Möglich

Tabelle 2 auf Seite 54 gibt einfache Anhaltspunkte, die Ihnen bei der Wahl der ersten Isolationsstufe für Ihre Anwendungen helfen können. Betrachten Sie die Angaben dieser Tabelle als Ausgangspunkt, und lesen Sie die Beschreibung der verschiedenen Isolationsstufen in den vorigen Abschnitten, um Hinweise zu erhalten, die vielleicht einen geeigneteren Wert für Ihre Anforderungen empfehlen.

Tabelle 2. Hinweise zur Wahl einer Isolationsstufe

Anwendungsart	Hohe Datenstabilität erforderlich	Hohe Datenstabilität nicht erforderlich
Schreib-/Lese-Transaktionen	RS	CS
Transaktionen mit Lesezugriff	RR oder RS	UR

Die Wahl der geeigneten Isolationsstufe für eine Anwendung ist zur Vermeidung von Erscheinungen, die für die Anwendung nicht tolerierbar sind, äußerst wichtig. Von der Isolationsstufe sind nicht nur die verschiedenen Grade der Isolation zwischen Anwendungen, sondern auch die Leistungsmerkmale einer einzelnen Anwendung betroffen, da die CPU- und Speicherressourcen, die zur Aktivierung und Freigabe von Sperren erforderlich sind, mit der Isolationsstufe variieren. Die Möglichkeit, dass gegenseitige Sperren auftreten, ist ebenfalls je nach Isolationsstufe unterschiedlich.

Angeben der Isolationsstufe

Die Isolationsstufe wird beim Vorkompilieren oder beim Binden einer Anwendung an die Datenbank angegeben. Für eine Anwendung, die in einer unterstützten kompilierten Sprache geschrieben ist, wird die Option ISOLATION der Befehle PREP oder BIND des Befehlszeilenprozessors verwendet. Die Isolationsstufe kann außerdem mit den Anwendungsprogrammierschnittstellen PREP oder BIND angegeben werden.

Wenn eine Bindedatei beim Vorkompilieren erstellt wird, wird die Isolationsstufe in der Bindedatei gespeichert. Wenn beim Binden keine Isolationsstufe angegeben wird, wird beim Vorkompilieren die Standardisolationsstufe verwendet.

Wenn keine Isolationsstufe angegeben wird, wird die Standardisolationsstufe CS (Cursorstabilität) verwendet.

Die Isolationsstufe eines Pakets kann mit Hilfe der folgenden Abfrage festgestellt werden:

```
SELECT ISOLATION FROM SYSCAT.PACKAGES
  WHERE PKGNAME = 'XXXXXXXX'
  AND PKGSCHEMA = 'YYYYYYYY'
```

Hier steht *XXXXXXXX* für den Namen des Pakets und *YYYYYYYY* für den Schemennamen des Pakets. Beide Namen müssen vollständig in Großbuchstaben angegeben werden.

Wenn eine Datenbank erstellt wird, werden mehrere Bindedateien, die zur Unterstützung der verschiedenen Isolationsstufen für SQL in REXX dienen, an

die Datenbank (auf den Servern, die REXX unterstützen) gebunden. Andere Befehlszeilenprozessorpakete werden auch an die Datenbank gebunden, wenn die Datenbank erstellt wird. Weitere Informationen zu Bindedateien finden Sie im Handbuch *Application Development Guide*.

REXX und der Befehlszeilenprozessor stellen die Verbindung zu einer Datenbank mit der Standardisolationsstufe CS (Cursorstabilität) her. Durch die Änderung in eine andere Isolationsstufe wird der Status der Verbindung nicht geändert. Dies muss im Status CONNECTABLE AND UNCONNECTED oder IMPLICITLY CONNECTABLE ausgeführt werden. (Genauere Informationen zu den Verbindungsstatus finden Sie im Abschnitt zur Anweisung CONNECT TO im Handbuch *SQL Reference*.)

Die momentan verwendete Isolationsstufe kann von einer REXX-Anwendung mit Hilfe des Werts der REXX-Variablen SQLISL überprüft werden. Dieser Wert wird jedes Mal aktualisiert, wenn der Befehl CHANGE SQLISL ausgeführt wird.

Mit der Profilregistrierungsvariablen DB2_RR_TO_RS können Sie Next Key-Sperren minimieren und auf diese Weise den gleichzeitigen Zugriff fördern und die Leistung verbessern. Allerdings verhindern Sie mit dieser Variablen die Verwendung der Isolationsstufe Wiederholtes Lesen (RR). Es wird daher empfohlen, diese Variable auf "Yes" zu setzen, wenn Sie in keiner der Anwendungen, die auf die Datenbank zugreifen, RR-Semantik benötigen. Die Variable wird erst wirksam, nachdem Sie den Datenbankmanager gestoppt und erneut gestartet haben. Nach `db2start` wirkt sich diese Änderung auf das gesamte Exemplar aus. Wenn dann eine Anforderung zum Zugriff auf eine Benutzertabelle unter Verwendung der Isolationsstufe RR empfangen wird, wird die Anforderung intern so geändert, dass die Isolationsstufe RS (Lesestabilität) verwendet wird. In diesem Fall wird keine Warnung ausgegeben.

Neben der Einstellung der Isolationsstufe auf der Paketebene bei der Vorbereitung oder beim Binden einer Anwendung können Sie eine Isolationsstufe auch auf Anweisungsebene festlegen. Eine Isolationsstufe auf Anweisungsebene wird mit Hilfe der WITH-Klausel angegeben.

Die folgenden SQL-Anweisungen unterstützen Isolationsstufen auf Anweisungsebene:

- SELECT-Anweisung
- SELECT INTO
- Gezielte DELETE-Anweisung
- INSERT
- UPDATE-Anweisung mit Suche
- DECLARE CURSOR

An die Verwendung von Isolationsstufen auf Anweisungsebene sind einige Bedingungen geknüpft:

- Die WITH-Klausel kann bei Unterabfragen nicht verwendet werden.
- Die Option WITH UR gilt nur für Operationen mit Lesezugriff. Wenn die Option in anderen Situationen verwendet wird, ändert sich die Anweisung automatisch von „UR" in „CS".
- Die Standardisolationsstufe einer Anweisung ist die Isolationsstufe des Pakets, in dem die Anweisung gebunden ist.
- Die Isolationsstufe auf Anweisungsebene setzt die Isolationsstufe des Pakets außer Kraft, in dem die Anweisung auftritt.

Wenn Sie den Befehlszeilenprozessor verwenden, können Sie die Isolationsstufe mit Hilfe des Befehls CHANGE ISOLATION LEVEL ändern. Im Handbuch *Command Reference* finden Sie weitere Informationen.

Für DB2 Call Level Interface (CLI) können Sie die Isolationsstufe bei der Konfiguration von CLI ändern. In CLI verwenden Sie bei der Laufzeit die Funktion "SQLSetConnectAttr" mit dem Attribut SQL_ATTR_TXN_ISOLATION. Dadurch wird die Isolationsstufe der Transaktion für die aktuelle Verbindung festgelegt, auf die durch *ConnectionHandle* verwiesen wird. In der Datei "db2cli.ini" können Sie auch das Schlüsselwort TXNISOLATION verwenden.

Anmerkung: JDBC und SQLJ sind in DB2 mit CLI implementiert. Dies bedeutet, dass die Einstellungen der Datei "db2cli.ini" Einfluss auf die Projekte haben, die unter Verwendung von JDBC und SQLJ geschrieben und ausgeführt werden. Im Handbuch *CLI Guide and Reference* finden Sie weitere Informationen hierzu.

Wenn Sie zur Laufzeit mit JDBC oder SQLJ arbeiten, können Sie die Methode "setTransactionIsolation" in der java.sql-Schnittstellenverbindung verwenden, um die Isolationsstufe einzurichten. Weitere Informationen hierzu finden Sie im Kapitel über die Java-Programmierung im Handbuch *Application Development Guide*.

Während der Arbeit mit SQLJ wird beim Ausführen des SQLJ-Optimierungsprogramms "db2profc" ein Paket erstellt. Die letzten Optionen für dieses Paket können die zu verwendende Isolationsstufe enthalten. Weitere Informationen hierzu finden Sie im Kapitel über die Java-Programmierung im Handbuch *Application Development Guide*.

Außerdem bieten zahlreiche gewerblich geschriebene Anwendungen eine Möglichkeit an, die Isolationsstufe zu wählen. Im Handbuch *CLI Guide and Reference* finden Sie weitere Informationen.

Deklarierte temporäre Tabellen und gemeinsamer Zugriff

Bei deklarierten temporären Tabellen gibt es keine Probleme bezüglich des gemeinsamen Zugriffs, da sie nur für die Anwendung verfügbar sind, die sie deklariert hat. Diese Art von Tabelle besteht nur vom Zeitpunkt ihrer Deklaration durch die Anwendung bis zu deren Beendigung bzw. bis zum Unterbrechen der Verbindung durch diese.

Sperren

Der Datenbankmanager ermöglicht die Steuerung des gemeinsamen Zugriffs und verhindert den unkontrollierten Zugriff mit Hilfe von Sperren. Eine *Sperre* ist eine Methode, eine Ressource des Datenbankmanagers einer Anwendung zuzuordnen, um zu steuern, wie andere Anwendungen auf diese Ressource zugreifen können. Eine Anwendung, der eine Ressource zugeordnet ist, aktiviert die Sperre oder ist Eigner der Sperre.

Der Datenbankmanager richtet Sperren ein, um Anwendungen daran zu hindern, auf nicht festgeschriebene Daten zuzugreifen, die von anderen Anwendungen geschrieben wurden (außer bei Verwendung der Isolationsstufe für nicht festgeschriebenes Lesen). Durch dieses Prinzip wird die Datenintegrität geschützt (d. h. die Konsistenz und die Sicherheit von Daten). Sperren können auch die Aktualisierung von Zeilen (zum Beispiel für eine Anwendung mit der Isolationsstufe RR) verhindern.

Zur Gewährleistung der Datenintegrität aktiviert der Datenbankmanager implizit Sperren unter der Steuerung des Datenbankmanagers. Außer für die Isolationsstufe UR (Nicht festgeschriebener Lesevorgang) muss eine Anwendung nie explizit eine Sperre anfordern, um sicherzustellen, dass nicht festgeschriebene Daten für andere Prozesse unsichtbar gemacht werden.

Aufgrund dieses Grundkonzepts der Sperren ist es in den meisten Fällen nicht nötig, zur Steuerung von Sperren explizite Maßnahmen zu ergreifen. Dennoch werden Sperren für Anwendungen auf der Grundlage bestimmter allgemeiner Parameter aktiviert. Die Kenntnis der Verhältnisse in Ihrer lokalen Umgebung kann dazu beitragen, dass Sie die Systemressourcen durch Ändern der Parameter effektiver nutzen können. Die folgenden Themen enthalten eingehendere Informationen zu Sperren:

- Attribute von Sperren
- Sperren und die Leistung von Anwendungen
- Faktoren mit Auswirkung auf Sperren
- Anweisung LOCK TABLE
- Anweisung CLOSE CURSOR WITH RELEASE
- Zusammenfassung der Überlegungen zu Sperren.

Attribute von Sperren

Sperren des Datenbankmanagers verfügen über folgende Basisattribute:

Modus
Die Art des Zugriffs, die dem Sperreneigner gewährt wird, sowie die Art des Zugriffs, die Benutzern gewährt wird, die das gesperrte Objekt gleichzeitig verwenden. Dies wird manchmal auch als *Status* der Sperre bezeichnet.

Objekt
Die Ressource, die gesperrt ist. Die einzige Objektart, die explizit gesperrt werden kann, ist die Tabelle. Der Datenbankmanager implementiert Sperren auch für andere Arten von Ressourcen, wie Zeilen, Tabellen und Tabellenbereiche. Das Objekt, das gesperrt wird, stellt die *Granularität* der Sperre an.

Dauer Der Zeitraum, für den eine Sperre aktiv ist. Die Dauer der Sperren wird durch Isolationsstufen beeinflusst, die in „Gemeinsamer Zugriff" auf Seite 47 beschrieben werden.

In der folgenden Tabelle werden Modi und ihre Auswirkungen in der Reihenfolge zunehmender Ressourcenbeschränkung angegeben:

Tabelle 3. Zusammenfassung der Sperrmodi

Sperrmodus	Gültige Objektart	Beschreibung
IN (Intent None)	Tabellenbereiche, Tabellen	Der Sperreneigner kann alle Daten in der Tabelle lesen, einschließlich der nicht festgeschriebenen Daten, aber er kann keine Daten aktualisieren. Vom Sperreneigner werden keine Zeilensperren aktiviert. Andere, gleichzeitig ausgeführte Anwendungen können die Tabelle lesen oder aktualisieren.
IS (Intent Share)	Tabellenbereiche, Tabellen	Der Sperreneigner kann die Daten in der gesperrten Tabelle lesen, aber nicht aktualisieren. Wenn für eine Anwendung der Modus IS der Tabellensperre aktiv ist, aktiviert die Anwendung eine Sperre des Modus S oder NS für jede gelesene Zeile. In beiden Fällen können andere Anwendungen die Tabelle lesen oder aktualisieren.
NS (Next Key Share)	Zeilen	Der Sperreneigner und jede gleichzeitig ausgeführte Anwendung kann die gesperrten Zeilen lesen, aber nicht aktualisieren. Diese Sperre wird anstatt einer Sperre im Modus S für Zeilen einer Tabelle aktiviert, wenn die Isolationsstufe für die Anwendung entweder RS oder CS ist.
S (Share)	Zeilen, Tabellen	Der Sperreneigner und jede gleichzeitig ausgeführte Anwendung kann die gesperrten Daten lesen, aber nicht aktualisieren. Einzelne Zeilen einer Tabelle können mit einer Sperre im Modus S versehen werden. Wenn eine Tabelle mit einer solchen Sperre im Modus S versehen ist, sind keine Zeilensperren notwendig.

Tabelle 3. Zusammenfassung der Sperrmodi (Forts.)

Sperrmodus	Gültige Objektart	Beschreibung
IX (Intent Exclusive)	Tabellenbereiche, Tabellen	Der Sperreneigner und gleichzeitig ausgeführte Anwendungen können Daten in der Tabelle lesen und aktualisieren. Wenn der Sperreneigner Daten liest, wird für jede gelesene Zeile ein Sperre im Modus S, NS, X oder U aktiviert. Daneben wird für alle Zeiten, die der Sperreneigner aktualisiert, eine Sperre im Modus X aktiviert. Andere, gleichzeitig ausgeführte Anwendungen können die Tabelle sowohl lesen als auch aktualisieren.
SIX (Share with Intent Exclusive)	Tabellen	Der Sperreneigner kann Daten in der Tabelle lesen und aktualisieren. Für den Sperreneigner werden Sperren im Modus X für die Zeilen aktiviert, die er aktualisiert, aber keine Sperren für die Zeilen, die er liest. Andere, gleichzeitig ausgeführte Anwendungen können die Tabelle lesen.
U (Update)	Zeilen, Tabellen	Der Sperreneigner kann Daten in der gesperrten Zeile oder Tabelle aktualisieren. Für den Sperreneigner werden Sperren im Modus X für die Zeilen aktiviert, bevor diese aktualisiert werden. Andere Arbeitseinheiten können die Daten lesen, aber nicht versuchen, sie zu aktualisieren.
NX (Next Key Exclusive)	Zeilen	Der Sperreneigner kann die gesperrte Zeile lesen, aber nicht aktualisieren. Dieser Modus entspricht einer Sperre im Modus X, die jedoch mit einer Sperre im Modus NS kompatibel ist.
NW (Next Key Weak Exclusive)	Zeilen	Diese Sperre wird für die nächste Zeile aktiviert, wenn eine Zeile in den Index einer Nicht-Katalog-Tabelle eingefügt wird.a Der Sperreneigner kann die gesperrte Zeile lesen, aber nicht aktualisieren. Dieser Modus entspricht einer Sperre im Modus X, die jedoch mit den Sperren im Modus W und NS kompatibel ist.
X (Exclusive)	Zeilen, Tabellen	Der Sperreneigner kann Daten in der gesperrten Zeile oder Tabelle lesen und auch aktualisieren. Mit dem Modus Exclusive können Tabellen gesperrt werden, d. h., für Zeilen in diesen Tabellen werden keine Zeilensperren aktiviert. Nur Anwendungen mit der Isolationsstufe UR (Nicht festgeschriebener Lesevorgang) können auf die gesperrte Tabelle zugreifen.
W (Weak Exclusive)	Zeilen	Diese Sperre wird für die Zeile aktiviert, wenn eine Zeile in eine Nicht-Katalog-Tabelle eingefügt wird. Der Sperreneigner kann die gesperrte Zeile ändern. Diese Sperre entspricht einer Sperre im Modus X, die jedoch mit der Sperre im Modus NW kompatibel ist. Nur Anwendungen mit der Isolationsstufe UR (Nicht festgeschriebener Lesevorgang) können auf die gesperrte Zeile zugreifen.

Tabelle 3. Zusammenfassung der Sperrmodi (Forts.)

Sperrmodus	Gültige Objektart	Beschreibung
Z (Super Exclusive)	Tabellenbereiche, Tabellen	Diese Sperre wird für eine Tabelle unter bestimmten Umständen aktiviert, zum Beispiel, wenn die Tabelle geändert (ALTER) oder gelöscht (DROP) wird, ein Index für die Tabelle erstellt oder gelöscht wird oder eine Tabelle reorganisiert wird. Keine anderen, gleichzeitig ausgeführten Anwendungen können die Tabelle lesen oder aktualisieren.

Anmerkung: Nur für Tabellen und Tabellenbereiche werden die Sperren in den „Intent-Modi" aktiviert. Das heißt, für Zeilen werden Intent-Sperren nicht aktiviert.

Sperren und die Leistung von Anwendungen

Anwendungsprogrammierer sollten sich über einige Faktoren hinsichtlich der Verwendung von Sperren und ihrer Auswirkungen auf die Leistung von Anwendungen im Klaren sein. Zu diesen Faktoren zählen die folgenden:

- Gemeinsamer Zugriff und Granularität
- Sperrenkompatibilität
- Sperrenumwandlung
- Sperreneskalation
- Warten auf Sperren und Zeitlimits
- Gegenseitige Sperren.

Gemeinsamer Zugriff und Granularität

Eine Sperre, die für eine Anwendung aktiv ist, kann den Zugriff durch eine andere Anwendung verhindern. Daher ist zur Erhaltung eines möglichst hohen Grades des gemeinsamen Zugriffs eine Sperre auf Zeilenebene besser als eine auf Tabellenebene. Jedoch werden für Sperren Speicher und Verarbeitungszeit zur Verwaltung benötigt. Daher ist zur Minimierung der benötigten Speicherressourcen und der Verarbeitungszeit eine einzige Sperre auf Tabellenebene besser als eine große Anzahl von Zeilensperren.

Sie können die Größe (Granularität) von Sperren auf Zeilen- oder Tabellenebene mit der Klausel LOCKSIZE der Anweisung ALTER TABLE definieren. Standardmäßig werden Zeilensperren verwendet. Bei permanenten Tabellensperren, wie durch ALTER TABLE definiert, werden nur S- und X-Tabellensperren verwendet. Der Durchsatz wird gesteigert, weil die Anwendung nicht so viele Zeilensperren aktivieren und freigeben muss.

In den folgenden Fällen ist eine permanente Tabellensperre durch die Anweisung ALTER TABLE anstelle einer Tabellensperre für eine einzelne Transaktion durch die Anweisung LOCK TABLE vorzuziehen:

- Die Tabelle ist schreibgeschützt, und Sie benötigen immer S-Sperren. Eine Sperre auf Tabellenebene steigert den Durchsatz und ermöglicht anderen Benutzern die Aktivierung von S-Sperren für die Tabelle.
- Ein einzelner Benutzer greift zu Wartungszwecken auf die Tabelle zu und benötigt für eine begrenzte Zeit eine X-Sperre. Durch das Ändern einer Sperre auf Tabellenebene durch ALTER TABLE für die Tabelle wird dem Benutzer eine X-Sperre auf Tabellenebene bereitgestellt. Nach Abschluss der Arbeit kann der Benutzer die Tabelle mit ALTER TABLE auf Sperren auf Zeilenebene zurückstellen.

Die Verwendung der Anweisung ALTER TABLE verhindert normale Sperreneskalation nicht.

Beachten Sie zudem, dass das Aktivieren von Sperren auf Tabellenebene durch ALTER TABLE eine globale Vorgehensweise ist, die sich auf alle Anwendungen und Benutzer auswirkt, die auf diese Tabelle zugreifen. Alternativ können die einzelnen Anwendungen die Anweisung LOCK TABLE verwenden. Dies ermöglicht Ihnen den Aufruf von Tabellensperren auf Anwendungsebene, nicht auf Datenbankebene (wie im zweiten Punkt oben erwähnt).

Sperrenkompatibilität

Tabelle 4 zeigt, ob eine Sperrenanforderung erfüllt wird, wenn ein anderer Prozess für dieselbe Ressource bereits eine Sperre aktiviert hat oder anfordert. Ein **Nein** bedeutet, dass der Anforderer warten muss, bis alle inkompatiblen Sperren von anderen Prozessen inaktiviert werden. Beachten Sie, dass es zu einer Zeitlimitüberschreitung beim Warten auf eine Sperre kommen kann. Ein **Ja** bedeutet, dass die Sperre aktiviert wird (es sei denn, ein anderer Benutzer wartet auf die Ressource).

Tabelle 4. Kompatibilität der Sperrmodi

	Status der aktiven Sperre												
Angeforderter Status	**NONE**	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**NX**	**X**	**Z**	**NW**	**W**
NONE	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
IN	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Ja	Ja
IS	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein
NS	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Ja	Ja	Nein	Nein	Ja	Nein
S	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein
IX	Ja	Ja	Ja	Nein	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein
SIX	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
U	Ja	Ja	Ja	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
NX	Ja	Ja	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
X	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
Z	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein
NW	Ja	Ja	Nein	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja
W	Ja	Ja	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein

Anmerkung:

I Intent
N None
NS Next Key Share
S Share
NX Next Key Exclusive
X Exclusive
U Update
Z Super Exclusive
NW Next Key Weak Exclusive
W Weak Exclusive

Einzelheiten zu diesen Sperrenarten finden Sie in der Beschreibung der „Attribute von Sperren” auf Seite 58.

Anmerkung:

- Ja - angeforderte Sperre sofort **erteilen**
- Nein - **warten**, bis aktive Sperre freigegeben oder Zeitlimit überschritten wird

Nehmen Sie an, dass für die Anwendung A eine Sperre für eine Tabelle aktiv ist, auf die die Anwendung B ebenfalls zugreifen möchte. Der Datenbankmanager fordert für die Anwendung B eine Sperre eines bestimmten Modus an. Wenn der Modus der für A aktiven Sperre die für B angeforderte Sperre zulässt, werden diese beiden Sperren (bzw. Modi) als kompatibel bezeichnet.

Wenn der für Anwendung B angeforderte Sperrmodus nicht mit der für Anwendung A aktiven Sperre kompatibel ist, kann Anwendung B nicht fortgesetzt werden. Statt dessen muss Anwendung B warten, bis Anwendung A die Sperre freigegeben hat und bis *alle* weiteren bestehenden inkompatiblen Sperren freigegeben wurden.

Sperrenumwandlung

Die Sperrenumwandlung erfolgt, wenn ein Prozess auf ein Datenobjekt zugreift, für das er bereits eine Sperre aktiviert hat, und der Zugriffsmodus eine noch stärker begrenzende Sperre erfordert. Für einen Prozess kann immer nur eine Sperre für ein Datenobjekt aktiv sein, obwohl er mehrfach eine Sperre für dasselbe Datenobjekt (indirekt durch eine Abfrage) anfordern kann. Das Ändern des Modus einer bereits bestehenden Sperre wird als *Umwandlung* bezeichnet.

Die Sperrenumwandlung im Fall von Zeilensperren ist einfach: Eine Umwandlung findet beispielsweise nur statt, wenn eine Sperre des Modus X benötigt wird und eine Sperre des Modus S oder U bereits aktiv ist.

Für Tabellen und Zeilen gibt es verschiedene Sperrmodi. Sperren der Modi IX (Intent Exclusive) und S (Shared) stellen jedoch in Bezug auf Sperrenumwandlung einen Sonderfall dar. Weder der Modus S noch der Modus IX wird als stärker einschränkend angesehen als der jeweils andere Modus, d. h. wenn einer dieser Modi aktiv ist und der andere benötigt wird, erfolgt die Sperrenumwandlung in den Modus SIX (Share with Intent Exclusive). Alle anderen Umwandlungen werden so ausgeführt, dass der angeforderte Sperrmodus zum Modus der aktiven Sperre wird, wenn der angeforderte Modus einen niedrigeren Grad der Einschränkung bewirkt.

Eine Abfrage zur Aktualisierung einer Zeile kann auch zu einer doppelten Umwandlung führen. Nehmen Sie an, die Zeile wurde mit Hilfe eines Indexzugriffs gelesen und im Modus S gesperrt. Die Tabelle mit der Zeile besäße eine Intent-Tabellensperre. Nehmen Sie an, dass dies eine Sperre im Modus IS und nicht im Modus IX ist. Wenn in diesem Fall die Zeile geändert wird, wird die Tabellensperre in den Modus IX und die Zeilensperre in den Modus X umgewandelt.

Beachten Sie, dass die Aktivierung von Sperren in der Regel implizit während der Ausführung einer Abfrage stattfindet. Das Verständnis der verschiedenen Arten von Sperren für verschiedene Abfragen sowie Tabellen- und Indexkombinationen kann beim Entwurf und bei der Optimierung von Anwendungen sehr hilfreich sein. Weitere Informationen zu diesem Thema finden Sie in „Faktoren mit Auswirkung auf Sperren” auf Seite 69.

Sperreneskalation

Sperreneskalation ist ein interner Mechanismus zur Verringerung der Anzahl der aktivierten Sperren. Eskaliert wird von vielen Zeilensperren (in einer einzigen Tabelle) auf eine einzige Tabellensperre.

Eine Sperreneskalation tritt auf, wenn zu viele Sperren (unabhängig von der Art der Sperren) zu einem Zeitpunkt aktiv sind.

Eine Sperreneskalation kann für einen bestimmten Datenbankagenten auftreten, wenn der Agent die Zuordnung der Sperrenliste überschreitet (siehe „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450).

Eine Sperreneskalation wird intern durchgeführt. Das einzige, extern bemerkbare Ergebnis könnte sein, dass der gemeinsame Zugriff auf eine oder mehrere Tabellen eingeschränkt wird. In einer zweckmäßig konfigurierten Datenbank tritt die Sperreneskalation in der Regel recht selten auf.

Eine Sperreneskalation kann beispielsweise auftreten, wenn ein Anwendungsprogrammierer einen Index für eine sehr umfangreiche Tabelle verwendet, um die Leistung und den gemeinsamen Zugriff zu verbessern, aber die Anwendung auf einen großen Prozentsatz der Datensätze in der Tabelle zugreift. Der Datenbankmanager kann (in diesem Fall) nicht vorausberechnen, dass die Tabelle in einem solchen Ausmaß gesperrt wird, und sperrt daher jeden Datensatz einzeln, anstatt die Tabelle mit dem Modus S oder X zu sperren. Als Lösung für diesen Fall kann der Datenbankentwickler nach Absprache mit dem Anwendungsprogrammierer empfehlen, für diese Transaktion eine Anweisung LOCK TABLE zu verwenden.

Es kann vorkommen, dass der Prozess, der die Eskalationsanforderung erhält (intern), wenig oder gar keine Datensätze in irgendeiner Tabelle gesperrt hat. Der Grund für diese Eskalation ist, dass für einen Prozess (oder mehrere Prozesse) mehrere Sperren aktiviert sein können (obwohl diese Anzahl unter dem Wert des Konfigurationsparameters für Sperren pro Prozess der Datenbank liegt), aber die Anzahl nicht ausreicht, um die Eskalationsanforderung auszulösen. Der Prozess fordert eventuell keine weitere Sperre an oder greift auf die Datenbank nur zu, um die Transaktion zu beenden. Wenn in diesem Fall ein anderer Prozess eine Sperre oder Sperren anfordert, können diese Sperren die Eskalationsanforderung auslösen.

Wenn die Sperreneskalation den gemeinsamen Zugriff auf ein nicht akzeptables Maß reduziert, können Sie Folgendes ausführen:

- Überprüfen Sie den Inhalt von *db2diag.log* auf Informationen zu Eskalationen. Für jede eskalierte Tabelle werden Informationen aufgezeichnet. Zu den aufgezeichneten Arten von Informationen zählen folgende:
  - Die Anzahl der momentan aktiven Sperren

- Die Anzahl der bis zum Ende der Sperreneskalation erforderlichen Sperren
- Die Tabellenkennung und der Tabellenname jeder eskalierten Tabelle
- Die Anzahl der momentan aktiven Sperren für andere Elemente als Tabellen
- Die neue Sperre auf Tabellenebene, die als Teil der Eskalation aktiviert werden soll. Dabei handelt es sich in der Regel um die Sperre „S" (Share) oder „X" (eXclusive).
- Der interne Rückkehrcode für das Ergebnis des Aktivierens der neuen Tabellenebene

Die aktuelle dynamische SQL-Anweisung wird eventuell auch aufgezeichnet. In diesem Fall enthalten die aufgezeichneten Informationen die aktuelle SQL-Anweisung vor der Eskalation von Tabellensperren, **wenn** der Konfigurationsparameter DIAGLEVEL des Datenbankmanagers auf 4 gesetzt ist. Wenn die Sperreneskalation fehlschlägt, enthalten die aufgezeichneten Informationen die Tabelle, für die die Eskalation fehlgeschlagen ist, und die aktuelle SQL-Anweisung (falls sie verfügbar ist und nicht zuvor geschrieben wurde), **wenn** DIAGLEVEL auf 2 oder höher gesetzt wurde.

Mit Hilfe dieser Informationen können Sie eine geeignete Aktion auf Grundlage der anderen unten erwähnten Punkte ausführen.

Sie müssen den Konfigurationsparameter DIAGLEVEL des Datenbankmanagers auf 3 (Standardwert) oder auf 4 setzen, um diese Art der Informationsaufzeichnung zu starten.

- Erhöhen der Anzahl der zulässigen Sperren durch Heraufsetzen des Werts des Parameters *maxlocks* und/oder *locklist* in der Konfigurationsdatei der Datenbank. (Siehe „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450 und „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415.) Dies kann die Maßnahme der Wahl sein, wenn der gleichzeitige Zugriff auf die Tabelle durch andere Prozesse von höchster Wichtigkeit ist. Jedoch kann der Systemaufwand, der mit der Aktivierung von Sperren auf Datensatzebene verbunden ist, zu größeren Verzögerungen für andere Prozesse führen, als an Zeit durch den gleichzeitigen Zugriff auf eine Tabelle an Zeit eingespart werden kann. (Stellen Sie beim Ändern dieser Parameter in einer partitionierten Datenbank sicher, dass die Parameter in allen Partitionen aktualisiert werden.)
- Feststellen und Anpassen des Prozesses (oder der Prozesse), bei dessen Ausführung die Sperreneskalation auftrat. Dieser Prozess (oder Prozesse) kann die Ursache für die Eskalation oder das Zurücksetzen sein, muss es aber nicht. Geben Sie Anweisungen LOCK TABLE explizit an.
- Ändern der Isolationsstufe. Beachten Sie, dass diese Maßnahme zu einer Verringerung des gemeinsamen Zugriffs führen kann.

- Erhöhen der Häufigkeit von COMMIT-Operationen. Diese Maßnahme verringert meist die Anzahl der zu einem bestimmten Zeitpunkt vorhandenen Sperren. Weitere Informationen zu Isolationsstufen und gemeinsamen Zugriff finden Sie in „Gemeinsamer Zugriff" auf Seite 47.

Warten auf Sperren und Zeitlimits

Ohne Überwachung eines Zeitlimits für Sperren muss eine Anwendung in abnormalen Situationen möglicherweise unbegrenzt darauf warten, dass eine Sperre wieder freigegeben wird. Dies kann zum Beispiel geschehen, wenn eine Transaktion auf eine Sperre wartet, die für die Anwendung eines anderen Benutzers aktiviert wurde, und dieser andere Benutzer seinen Arbeitsplatz verlassen hat, ohne seine Anwendung durch weitere Eingaben die durchgeführten Transaktionen festschreiben und dadurch die Sperre freigeben zu lassen. Dies führt natürlich zu einer Beeinträchtigung der Anwendung. Um diese *Blockierung* des Programms in einem solchen Fall zu vermeiden, kann der Konfigurationsparameter *locktimeout* zur Einstellung der maximalen Zeitdauer, die eine Anwendung auf eine Sperre wartet, verwendet werden. (Siehe „Zeitlimit für Sperren (locktimeout)" auf Seite 451.)

Mit Hilfe dieses Parameters können globale gegenseitige Sperren besser vermieden werden, besonders in Anwendungen mit verteilten Arbeitseinheiten (DUOW). Wenn das Zeitlimit für die Sperre überschritten wird, das heißt, wenn die Zeit, die die Sperrenanforderung bereits ansteht, größer ist als der Wert des Parameters *locktimeout*, empfängt die Anwendung einen Fehler und die Transaktion wird rückgängig gemacht. Wenn zum Beispiel *Programm 1* versucht, eine Sperre zu erhalten, die bereits für *Programm 2* aktiv ist, liefert *Programm 1* den SQLCODE -911 (SQLSTATE 40001) mit dem Ursachencode 68 zurück, wenn das Zeitlimit abgelaufen ist.

Wenn der Konfigurationsparameter des Datenbankmanagers *diaglevel* auf vier gesetzt ist und das Zeitlimit für eine Sperrenanforderung überschritten wird, können Sie der Datei "db2diag.log" weitere Informationen entnehmen. Die Informationen darin umfassen das Objekt, den Sperrmodus und die Anwendung, die die Sperre für das Objekt aktiviert. Der Name der aktuellen dynamischen SQL-Anweisung oder des statischen Pakets kann auch darin enthalten sein.

Gegenseitige Sperren

Im Datenbankmanager können Konkurrenzsituationen bei der Anforderung von Sperren für Prozesse, die die Datenbank verwenden, zu gegenseitigen Sperren führen. Zum Beispiel sperrt der Prozess 1 die Tabelle A im Modus X (Exclusive), und Prozess 2 sperrt die Tabelle B im Modus X. Wenn nun der Prozess 1 versucht, die Tabelle B im Modus X zu sperren, und Prozess 2 versucht, die Tabelle A im Modus X zu sperren, sperren sich die Prozesse gegenseitig. Bei einer gegenseitigen Sperre sind beide Prozesse so lange blockiert, bis die jeweils zweite Sperrenanforderung erfüllt wird, und keine der beiden

Anforderungen wird ausgeführt, sofern nicht einer der Prozesse eine COMMIT- oder ROLLBACK-Operation ausführt. Dieser Zustand dauert an, bis ein externer Agent einen der Prozesse aktiviert und dazu zwingt, eine ROLLBACK-Operation auszuführen.

Gegenseitige Sperren im Sperrensystem werden im Datenbankmanager von einem asynchronen Hintergrundprozess des Systems, dem Detektor für gegenseitiges Sperren, behandelt. Der Detektor für gegenseitiges Sperren wird regelmäßig in den vom Konfigurationsparameter *dlchktime* bestimmten Abständen aktiv (siehe „Intervall für Prüfung gegenseitiger Sperren (dlchktime)" auf Seite 448). Wenn der Detektor für gegenseitiges Sperren aktiv wird, wird das Sperrensystem auf gegenseitige Sperren hin untersucht. Wenn die Datenbank partitioniert ist, sendet jede Partition *Sperrendiagramme* an die Datenbankpartition, die über die Systemkatalogsichten verfügt und in der die globale Erkennung gegenseitiger Sperren stattfindet.

Wenn eine gegenseitige Sperre erkannt wird, wählt der Detektor für gegenseitiges Sperren einen durch diese Sperre blockierten Prozess aus, um ihn rückgängig machen zu lassen. Der ausgewählte Prozess wird wieder aktiviert und kehrt zur aufrufenden Anwendung mit dem SQLCODE -911 (SQLSTATE 40001) und dem Ursachencode 2 zurück. Der Datenbankmanager setzt den ausgewählten Prozess automatisch zurück. Wenn die ROLLBACK-Operation beendet ist, werden die Sperren, die zu dem ausgewählten Prozess gehörten, freigegeben, und die anderen Prozesse, die ebenfalls in der gegenseitigen Sperre blockiert waren, können die Verarbeitung fortsetzen.

Die Auswahl eines geeigneten Intervalls für den Detektor für gegenseitiges Sperren ist wesentlich, um eine gute Leistung zu gewährleisten. Ein zu kurzes Intervall würde einen unnötigen Systemaufwand verursachen, ein zu langes Intervall würde die Verzögerung eines Prozesses durch eine gegenseitige Sperre auf ein nicht tolerierbares Maß erhöhen. Zum Beispiel könnte ein Aktivierungsintervall von 30 Minuten zulassen, dass eine gegenseitige Sperre über annähernd 30 Minuten bestehen kann. Der Anwendungsentwickler muss den geeigneten Mittelweg zwischen den möglichen Verzögerungen bei der Auflösung gegenseitiger Sperren und dem Systemaufwand für die Erkennung dieser Sperren finden.

In einer partitionierten Datenbank sollte das Intervall für alle Partitionen gleich sein (der Konfigurationsparameter *dlchktime* muss auf allen Partitionen auf den gleichen Wert aktualisiert werden). Ist der Wert auf dem Katalogknoten kleiner als in anderen Partitionen, werden möglicherweise scheinbare gegenseitige Sperren festgestellt. Ist der Wert auf dem Katalogknoten größer als auf anderen Partitionen, können scheinbar mehr als zwei Intervalle vergehen, bis eine gegenseitige Sperre festgestellt wird. Werden in einer partitio-

nierten Datenbank zahlreiche gegenseitige Sperren festgestellt, sollten Sie den Wert des Parameters *dlchktime* erhöhen, um Wartezeiten für Sperren und für die Übertragung zu berücksichtigen.

Ein anderes Problem kann auftreten, wenn eine Anwendung mit mehr als einem unabhängigen Prozess, der auf die Datenbank zugreift, so strukturiert ist, dass die Entstehung gegenseitiger Sperren wahrscheinlich ist. Ein Beispiel wäre eine Anwendung, in der mehrere Prozesse auf dieselbe Tabelle zuerst zu Leseoperationen und anschließend zu Schreiboperationen zugreifen. Wenn die Prozesse zuerst SQL-Abfragen im Lesezugriff durchführen und anschließend SQL-Aktualisierungsanweisungen für dieselbe Tabelle verarbeiten, steigt die Wahrscheinlichkeit gegenseitiger Sperren, weil es zwischen den Prozessen zu Konkurrenzsituationen beim Zugriff auf dieselben Daten kommen kann. Wenn zum Beispiel zwei Prozesse die Tabelle lesen und anschließend aktualisieren, können sie in eine Situation geraten, in der Prozess A versucht, eine Sperre des Modus X für eine Zeile zu erhalten, für die Prozess B eine Sperre des Modus S hat, und umgekehrt. Daraus kann sich eine gegenseitige Sperre ergeben. Zur Vermeidung dieser gegenseitigen Sperren, sollten Anwendungen, die auf Daten zugreifen, um diese zu ändern, beim Auswählen die Klausel FOR UPDATE OF verwenden. Durch diese Klausel wird sichergestellt, dass eine Sperre des Modus U aktiviert wird, wenn der Prozess A versucht, die Daten zu lesen.

Anmerkung: Erwägen Sie das Definieren einer Überwachung, die das Auftreten gegenseitiger Sperren aufzeichnet. Erstellen Sie die Überwachung mit der im Handbuch *SQL Reference* beschriebenen Anweisung CREATE EVENT.

In einer Systemumgebung mit zusammengeschlossenen Datenbanken kann es beim Zugriff einer Anwendung auf Kurznamen sein, dass die von der Anwendung angeforderten Daten nicht verfügbar sind, weil es an einer Datenquelle zu gegenseitigen Sperren gekommen ist. In diesem Fall ist DB2 davon abhängig, dass die Einrichtungen zur Behandlung gegenseitiger Sperren an der Datenquelle die gegenseitige Sperrung auflösen. Fall es an mehreren Datenquellen zu gegenseitigen Sperren gekommen ist, ist DB2 für das Aufheben der gegenseitigen Sperren auf die Zeitlimitmechanismen der Datenquellen angewiesen.

Wenn der Konfigurationsparameter des Datenbankmanagers *diaglevel* auf vier gesetzt ist und eine Sperrenanforderung wegen einer gegenseitigen Sperre fehlschlägt, sind in der Datei "db2diag.log" weitere Informationen enthalten. Die Informationen darin umfassen das Objekt, den Sperrmodus und die Anwendung, die die Sperre für das Objekt aktiviert. Der Name der aktuellen dynamischen SQL-Anweisung oder des statischen Pakets kann auch darin enthalten sein.

Faktoren mit Auswirkung auf Sperren

Der Modus und die Unterteilung (Granularität) der Sperren des Datenbankmanagers werden durch eine Kombination von Faktoren bestimmt: die Art der Verarbeitung, die eine Anwendung durchführt, die Art des Zugriffs auf Daten und verschiedene Parameter, die Sie angeben.

Anwendungsverarbeitung

Zur Feststellung der Sperrenattribute kann die Verarbeitung einer der folgenden vier Kategorien zugeordnet werden:

Lesezugriff
Dieser Typ beinhaltet alle SELECT-Anweisungen, die an sich nur Lesezugriff haben (Informationen zu Cursorn finden Sie in *SQL Reference*), über eine explizite Klausel FOR READ ONLY verfügen, oder mehrdeutig sind, die vom SQL-Compiler jedoch aufgrund der im Befehl PREP oder BIND angegebenen Option BLOCKING für Anweisungen mit Lesezugriff gehalten werden. Für diesen Typ sind nur Sperren des Modus Share (S bzw. IS) erforderlich.

Änderungsabsicht
Dieser Typ umfasst alle Anweisungen SELECT mit der Klausel FOR UPDATE oder Anweisungen SELECT, bei denen der SQL-Compiler als Ergebnis der Interpretation der mehrdeutigen Anweisung annimmt, dass sie zum Zweck der Änderung von Daten ausgeführt werden. Für diesen Typ werden Sperren der Modi Share und Update (S, U und X für Zeilen sowie IX, U und X für Tabellen) verwendet.

Änderung
Dieser Typ umfasst Anweisungen UPDATE, INSERT und DELETE, aber keine Anweisungen UPDATE WHERE CURRENT OF oder DELETE WHERE CURRENT OF. Für diesen Typ sind Sperren des Modus Exclusive (X oder IX) erforderlich.

Cursorgesteuert
Dieser Typ umfasst die Anweisungen UPDATE WHERE CURRENT OF und DELETE WHERE CURRENT OF. Für diesen Typ sind ebenfalls Sperren des Modus Exclusive (X oder IX) erforderlich.

Eine Anweisung, die an einer Zieltabelle Einfüge-, Aktualisierungs- oder Löschoperationen (INSERT, UPDATE oder DELETE) auf der Grundlage des Ergebnisses einer Unterauswahl vornimmt, führt zwei Typen der Verarbeitung aus. Die Sperren für die Tabellen, die durch die Unterauswahl geliefert werden, werden durch die Regeln für den Typ Lesezugriff bestimmt, die Sperren für die Zieltabelle durch die Regeln für den Typ Änderung.

Zugriffspfade

Ein *Zugriffspfad* ist die Methode, die das Optimierungsprogramm auswählt, um Daten aufgrund eines bestimmten Tabellenverweises abzurufen. (Siehe

„Datenzugriffskonzepte und Optimierung" auf Seite 187.) Der vom Optimierungsprogramm gewählte Zugriffspfad kann bedeutende Auswirkungen auf die Sperrmodi haben. Wenn zum Beispiel eine Indexsuche zum Auffinden einer bestimmten Zeile verwendet wird, wählt das Optimierungsprogramm sehr wahrscheinlich Sperren auf Zeilenebene (IS) für die Tabelle aus. Mit dieser Zugriffsart können mit einer Anweisung wie der folgenden Informationen für einen bestimmten Mitarbeiter aus der Tabelle EMPLOYEE ausgewählt werden, die einen Index für die Personalnummer (EMPNO) hat:

```
SELECT *
  FROM EMPLOYEE
  WHERE EMPNO = '000310';
```

Wenn kein Index verwendet wird, muss die gesamte Tabelle der Reihe nach durchsucht werden, um die ausgewählten Zeilen zu finden. Dafür ist möglicherweise eine einzige Sperre auf Tabellenebene (S) erforderlich. Mit dieser Zugriffsart könnten zum Beispiel mit einer Anweisung wie der folgenden alle männlichen Mitarbeiter ausgewählt werden, wobei es für die Spalte SEX keinen Index gibt:

```
SELECT *
  FROM EMPLOYEE
  WHERE SEX = 'M';
```

Die folgenden Tabellen enthalten eine Übersicht, welche Sperren für welche Art des Zugriffs aktiviert werden. Definitionen der Spaltenüberschriften finden Sie in „Anwendungsverarbeitung" auf Seite 69. Definitionen der Zugriffsmethode finden Sie auch in „Datenzugriffskonzepte und Optimierung" auf Seite 187. Beachten Sie, dass der Verarbeitungstyp *Cursorgesteuert* den Sperrmodus des zugrundeliegenden Cursors verwendet, bis die Anwendung eine Zeile findet, die zu aktualisieren oder zu löschen ist. Für diesen Typ der Verarbeitung wird unabhängig vom Sperrmodus eines Cursors immer eine Sperre des Modus Exclusive aktiviert, um die Aktualisierung oder Löschung durchzuführen.

Für die nachfolgenden Tabellen gelten folgende Konventionen: Wenn nur ein Sperrmodus aufgeführt ist, ist dies ein Sperrmodus auf Tabellenebene. Wenn zwei Sperrmodi aufgeführt sind, ist der erste der Sperrmodus auf Tabellenebene und der zweite der Sperrmodus auf Zeilenebene.

Tabelle 5. Sperrmodi für Tabellensuchen

Isolationsstufe	Lesezugriff	Änderungsabsicht	Änderung
Zugriffsmethode: ***Tabellensuche ohne Vergleichselemente***			
RR	S	U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X

Tabelle 5. Sperrmodi für Tabellensuchen (Forts.)

Isolationsstufe	Lesezugriff	Änderungsabsicht	Änderung
UR	IN	IX / U	IX / X
Zugriffsmethode: ***Tabellensuche mit Vergleichselementen***			
RR	S	U	U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

Tabelle 6. Sperrmodi für Indexsuchen

Isolationsstufe	Lesezugriff	Änderungsabsicht	Änderung
Zugriffsmethode: ***Indexsuche ohne Vergleichselemente***			
RR	S	IX / U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Zugriffsmethode: ***Indexsuche nach einer zu findenden Zeile***			
RR	IS / S	IX / U	IX / X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Zugriffsmethode: ***Indexsuche nur mit Start- und Stoppvergleichselementen***			
RR	IS / S	IX / S	IX / X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Zugriffsmethode: ***Indexsuche mit Vergleichselementen***			
RR	IS / S	IX / S	IX / U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

Tabelle 7 auf Seite 72 zeigt die Sperrmodi für Fälle, in denen der Lesezugriff auf die Datenseiten verzögert erfolgt, um in der Zeilenliste Folgendes zu ermöglichen:

- Die weitere Qualifizierung mit Hilfe mehrerer Indizes. Weitere Informationen finden Sie im Abschnitt „Zugriff über mehrere Indizes" auf Seite 196.
- Die Sortierung für einen effizienten Vorablesezugriff. Weitere Informationen finden Sie im Abschnitt „Vorablesezugriff über Listen" auf Seite 303.

Beim verzögerten Zugriff auf die Datenseiten erfolgt der Zugriff auf die Zeile in zwei Schritten, wodurch komplexere Sperrsituationen auftreten. Es gibt zwei Hauptkategorien, die von der Isolationsstufe abhängen. Da bei der Isolationsstufe RR (*Wiederholtes Lesen*) alle Sperren bis zum Ende der Transaktion aktiviert bleiben, bleiben die im ersten Schritt aktivierten Sperren weiterhin gültig, und im zweiten Schritt müssen keine weiteren Sperren aktiviert werden. Für die Isolationsstufen Lese- und Cursorstabilität müssen Sperren im zweiten Schritt aktiviert werden. Um den gemeinsamen Zugriff zu maximieren, werden im ersten Schritt keine Sperren aktiviert und alle Vergleichselemente erneut angewandt, so dass nur den Auswahlkriterien entsprechende Zeilen zurückgegeben werden.

Tabelle 7. Sperrmodi für Indexsuchen bei verzögertem Zugriff auf Datenseiten

Isolationsstufe	Lesezugriff	Änderungsabsicht	Änderung
Zugriffsmethode: ***Indexsuche ohne Vergleichselemente***			
RR	IS / S	IX / S	X
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
Zugriffsmethode: ***Verzögerter Zugriff auf Datenseiten nach Indexsuche ohne Vergleichselemente***			
RR	IN	IX / S	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
Zugriffsmethode: ***Indexsuche mit Vergleichselementen***			
RR	IS / S	IX / S	IX / S
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
Zugriffsmethode: ***Indexsuche nur mit Start- und Stoppvergleichselementen***			
RR	IS / S	IX / S	IX / X
RS	IN	IN	IN
CS	IN	IN	IN

Tabelle 7. Sperrmodi für Indexsuchen bei verzögertem Zugriff auf Datenseiten (Forts.)

Isolationsstufe	Lesezugriff	Änderungsabsicht	Änderung
UR	IN	IN	IN
Zugriffsmethode: ***Verzögerter Zugriff auf Datenseiten nach Indexsuche mit Vergleichselementen***			
RR	IN	IX / S	IX / S
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

Der Zugriffspfad unterliegt nicht der Steuerung durch den Benutzer, sondern wird vom Optimierungsprogramm gewählt.

Der verwendete Zugriffspfad kann sich auf den Modus und die Granularität einer Sperre auswirken. Zum Beispiel in einer Anwendung mit der Isolationsstufe RR (Wiederholtes Lesen) würde für eine Abfrage mit UPDATE-Operationen, die über eine Tabellensuche ohne Vergleichselemente auf die Zeilen zugreift, eine Sperre des Modus X für die Tabelle verwendet. Wenn die Zeilen über einen Index lokalisiert würden, sperrt der Datenbankmanager möglicherweise eher einzelne Zeilen der Tabelle.

Deklarierte temporäre Tabellen und Sperren

Deklarierte temporäre Tabellen werden nicht gesperrt, da sie nur für die Anwendung verfügbar sind, die sie deklariert hat. Diese Art von Tabelle besteht nur vom Zeitpunkt ihrer Deklaration durch die Anwendung bis zu deren Beendigung bzw. bis zum Unterbrechen der Verbindung durch diese.

Anweisung LOCK TABLE

Sie können die Regeln zum Anfordern anfänglicher Sperrmodi durch Verwenden der Anweisung LOCK TABLE in einer Anwendung außer Kraft setzen.

Diese Anweisung sperrt eine ganze Tabelle. Es wird nur die in der Anweisung LOCK TABLE angegebene Tabelle gesperrt. Übergeordnete und abhängige Tabellen der angegebenen Tabelle werden nicht gesperrt. Sie müssen selbst entscheiden, ob das Sperren anderer Tabellen, auf die möglicherweise zugegriffen wird, erforderlich ist, um das gewünschte Ergebnis hinsichtlich des gemeinsamen Zugriffs und der Leistung zu erzielen. Die Sperre wird erst freigegeben, wenn die Arbeitseinheit festgeschrieben oder rückgängig gemacht wurde.

Wenn eine Tabelle normalerweise von mehreren Benutzern gemeinsam benutzt wird, könnten Sie sie z. B. aus folgenden Gründen sperren:

LOCK TABLE IN SHARE MODE

Sie wollen auf Daten zugreifen, die *zeitlich konsistent* sind, d. h. Daten, die für eine Tabelle zu einem bestimmten Zeitpunkt aktuell sind. Wenn auf die Tabelle häufig zugegriffen wird, liegt die einzige Möglichkeit, die Tabelle in einem stabilen Zustand zu erhalten, darin, die Tabelle zu sperren. Die Anwendung soll zum Beispiel eine Momentaufnahme der Tabelle machen. Aber während der Zeit, die die Anwendung zur Verarbeitung einiger Zeilen der Tabelle benötigt, aktualisieren andere Anwendungen Zeilen, die die Anwendung noch nicht verarbeitet hat. Dies ist unter der Isolationsstufe RR (Wiederholtes Lesen) zulässig, jedoch nicht beabsichtigt.

Als alternative Methode kann Ihre Anwendung die Anweisung LOCK TABLE IN SHARE MODE absetzen: Es können keine Zeilen geändert werden, egal ob sie von Ihnen abgerufen wurden oder nicht. Anschließend kann die Anwendung die benötigte Anzahl von Zeilen abrufen und gleichzeitig davon ausgehen, dass die abgerufenen Zeilen nicht kurz vor dem Abrufen geändert wurden.

Nach Ausführung der Anweisung LOCK TABLE IN SHARE MODE können andere Benutzer Daten aus der Tabelle abrufen, aber sie können keine Zeilen in der Tabelle aktualisieren, löschen oder einfügen.

LOCK TABLE IN EXCLUSIVE MODE

Sie beabsichtigen, einen großen Teil der Tabelle zu aktualisieren. Es ist weniger aufwendig und daher effektiver, alle anderen Benutzer vom Zugriff auf die Tabelle auszuschließen, als jede Zeile zur Aktualisierung zu sperren und anschließend die Sperren wieder freizugeben, wenn alle Änderungen festgeschrieben sind.

Durch LOCK TABLE IN EXCLUSIVE MODE werden alle anderen Benutzer ausgeschlossen, d. h. keine anderen Anwendungen können auf die Tabelle zugreifen, außer wenn es sich um Anwendungen mit der Isolationsstufe UR (für nicht festgeschriebener Lesevorgang) handelt.

Weitere Einzelheiten zur Anweisung LOCK TABLE finden Sie im Handbuch *SQL Reference*.

Eine Alternative zur Verwendung der Anweisung LOCK TABLE ist die Anweisung ALTER TABLE mit dem Parameter LOCKSIZE. Der Parameter LOCKSIZE ermöglicht die Auswahl von Zeilensperren oder Tabellensperren. Die vorgenommene Auswahl wird zur Granularität (Unterteilung) der gewählten Sperren beim nächsten Zugriff auf die Tabelle. Die Auswahl von Zeilensperren entspricht der Auswahl der Standardsperrenunterteilung bei der Erstellung einer Tabelle. Die Auswahl von Tabellensperren verbessert eventuell die Leistung von Abfragen durch Begrenzung der Anzahl benötigter Sperren. Der gemeinsame Zugriff wird jedoch eventuell verringert, da alle

Sperren in der gesamten Tabelle aktiv sind. Die vorgenommene Auswahl verhindert jedoch nicht, dass eine normale Sperreneskalation auftreten kann. Weitere Einzelheiten zur Anweisung ALTER TABLE finden Sie im Handbuch *SQL Reference*.

Anweisung CLOSE CURSOR WITH RELEASE

Wenn Sie einen Cursor mit der Anweisung CLOSE CURSOR schließen, die die Klausel WITH RELEASE enthält, versucht der Datenbankmanager alle ggf. vorhandenen Lesesperren freizugeben, die für diesen Cursor aktiviert waren. Lesesperren sind Tabellensperren im Modus IS, S und U sowie Zeilensperren im Modus S, NS und U. Weitere Informationen zu den Sperrmodi finden Sie in „Attribute von Sperren" auf Seite 58.

Die Klausel WITH RELEASE hat keine Auswirkung auf Cursor, die unter den Isolationsstufen CS oder UR verwendet werden. Wenn die Klausel WITH RELEASE für Cursor angegeben wird, die unter der Isolationsstufe RS oder RR verwendet werden, hebt sie einige Merkmale dieser Isolationsstufen auf. Das heißt im Einzelnen, dass beim einem RS-Cursor *nichtwiederholbare Lesevorgänge* und bei einem RR-Cursor *nichtwiederholbare Lesevorgänge* oder *Phantomzeilen* auftreten können.

Wird ein Cursor, der ursprünglich ein RR- oder RS-Cursor ist, nachdem er mit der Klausel WITH RELEASE geschlossen wurde, erneut geöffnet, werden neue Lesesperren aktiviert.

Einen Vergleich zur anderen primären Klausel der Anweisung CLOSE CURSOR finden Sie im Abschnitt „Anweisung DECLARE CURSOR WITH HOLD" auf Seite 89.

Bei Verwendung des DB2-CLI-Verbindungsattributs SQL_ATTR_CLOSE_BEHAVIOR in CLI-Anwendungen können Sie die gleichen Ergebnisse erzielen wie mit CLOSE CURSOR WITH RELEASE. Weitere Informationen hierzu finden Sie im Abschnitt `SQLSetConnectAttr()` des Handbuchs *CLI Guide and Reference*.

Zusammenfassung der Überlegungen zu Sperren

Beachten Sie bitte im Hinblick auf Sperren folgende Punkte:

- Kleine Arbeitseinheiten (häufige Anweisungen COMMIT) fördern den gemeinsamen Zugriff auf Daten durch zahlreiche Benutzer. Nehmen Sie Anweisungen COMMIT in die Anwendung an den Stellen auf, an denen sich die Anwendung logisch in einem Konsistenzzustand befindet, das heißt, wenn die geänderten Daten konsistent sind. Wenn eine Anweisung COMMIT abgesetzt wird, werden Sperren freigegeben (außer bei Tabellensperren, die mit der Klausel WITH HOLD deklarierten Cursorn zugeordnet sind).

- Sperren werden aktiviert, auch wenn die Anwendung Zeilen lediglich liest, so dass auch Arbeitseinheiten im Lesezugriff festgeschrieben (COMMIT) werden müssen. Der Grund dafür ist der, dass in den Isolationsstufen RR, RS und CS in Anwendungen mit Lesezugriff gemeinsame Sperren aktiviert werden. Bei den Isolationsstufen RR und RS werden alle Sperren beibehalten, bis eine Anweisung COMMIT ausgeführt wird, so dass keine anderen Prozesse die gesperrten Daten aktualisieren können, es sei denn, Sie schließen den Cursor mit der Klausel WITH RELEASE. Außerdem werden auch Katalogsperren in Anwendungen mit dynamischem SQL unter der Isolationsstufe UR aktiviert.
- Der Datenbankmanager stellt sicher, dass die Anwendung keine noch nicht festgeschriebenen Daten (Zeilen, die von anderen Anwendungen aktualisiert, aber noch nicht mit der Anweisung COMMIT festgeschrieben wurden) abruft, sofern Sie nicht mit der Isolationsstufe UR (Uncommitted Read) arbeiten.
- Sie können die gesamte Tabelle, die Sie schützen wollen, durch Absetzen einer Anweisung LOCK TABLE sperren:
  - Eine Tabelle kann so gesperrt werden, dass andere Anwendungen Zeilen abrufen, aber nicht aktualisieren, löschen oder einfügen können.
  - Eine Tabelle kann gesperrt werden, um anderen Anwendungen (außer Anwendungen mit der Isolationsstufe UR) den Zugriff auf die Zeilen einer Tabelle zu verwehren.
- Wenn Sie einen Cursor mit der Anweisung CLOSE CURSOR schließen, die die Klausel WITH RELEASE enthält, versucht der Datenbankmanager alle ggf. vorhandenen Lesesperren freizugeben, die für diesen Cursor aktiviert waren.
- Stellen Sie beim Ändern der Konfigurationsparameter, die sich auf Sperren auswirken, in einer partitionierten Datenbank sicher, dass die Änderungen in allen Partitionen der Datenbank vorgenommen werden.

Anpassen der Optimierungsklasse

Bei der Kompilierung einer SQL-Abfrage kann eine Reihe von Optimierungstechniken verwendet werden, um den effizientesten Zugriffsplan für diese Abfrage zu ermitteln. Die Verwendung weiterer Optimierungsverfahren hat folgende Konsequenzen:

1. Verbesserungen der Leistung bei der Ausführung.
2. Höherer Zeitaufwand für die Abfragekompilierung.
3. Höherer Bedarf an Systemressourcen.

Aus diesem Grund kann es sinnvoll sein, die Anzahl der Techniken, die zur Optimierung der Abfrage verwendet werden, durch Festlegung der Optimierungsklasse zu beschränken. Dies ist besonders unter folgenden Umständen hilfreich:

- Sie haben sehr kleine Datenbanken oder sehr einfache dynamische Abfragen.
- Sie verfügen zum Zeitpunkt der Kompilierung nur über eingeschränkte Speicherressourcen auf dem Datenbank-Server.
- Sie wollen die Kompilierzeit (z. B. PREPARE) für die Abfrage verringern.

Sie können jede der unten beschriebenen Abfrageoptimierungsklassen auswählen, die Klassen 0 und 9 sollten jedoch nur in speziellen Situationen verwendet werden. Klasse 5 ist der Standardwert. Die Klassen 0, 1 und 2 verwenden den Algorithmus für schnelle Verknüpfungsaufzählung. Bei komplexen Abfragen berücksichtigt dieser Algorithmus wesentlich weniger alternative Pläne und benötigt deutlich weniger Kompilierungszeit als die Klassen ab Klasse 3. Die Klassen ab 3 aufwärts verwenden den Algorithmus für dynamisch programmierte Verknüpfungsaufzählung. Dieser Algorithmus berücksichtigt wesentlich mehr alternative Pläne und benötigt deutlich mehr Kompilierungszeit als die Klassen 0, 1 und 2 bei steigender Tabellenzahl.

0 - Diese Klasse weist das Optimierungsprogramm an, nur eine Minimaloptimierung zur Generierung eines Zugriffsplans durchzuführen. Beispiele:
- Statistiken über ungleichmäßige Verteilungen von Daten werden vom Optimierungsprogramm nicht berücksichtigt.
- Nur Grundregeln für das Umschreiben der Abfragen werden beachtet. (Informationen über das Umschreiben von Abfragen finden Sie in „Umschreiben der Abfrage durch den SQL-Compiler" auf Seite 176).
- Schnelle Verknüpfungsaufzählung tritt auf (siehe „Suchstrategien zur Auswahl der optimalen Verknüpfungsmethode" auf Seite 210).
- Nur die Zugriffsmethoden der Verknüpfung über Verschachtelungsschleife und der Indexsuche sind aktiviert (siehe „Verknüpfungskonzepte" auf Seite 203 und „Konzepte der Indexsuche" auf Seite 188).
- Der Vorablesezugriff über Listen und das logische Verknüpfen von Indizes über AND sind inaktiviert, damit sie nicht in generierten Zugriffsmethoden verwendet werden.
- Die Strategie der Sternverknüpfung (Star Join) wird nicht berücksichtigt.

Diese Klasse sollte nur unter speziellen Umständen verwendet werden, wenn der Systemaufwand zur Kompilierung der Abfrage so gering wie möglich gehalten werden muss. Eine Anwendung, die insgesamt aus sehr einfachen dynamischen SQL-Anweisungen besteht, die auf Tabellen mit sehr zweckmäßigen Indizes zugreifen, ist für die Kompilierung mit der Optimierungsklasse 0 zum Beispiel geeignet.

1 - Diese Klasse weist das Optimierungsprogramm an, einen

Optimierungsgrad zu verwenden, der ungefähr mit DB2/6000 Version 1 vergleichbar ist, sowie einige zusätzliche Funktionen mit geringem Aufwand, die in Version 1 nicht enthalten sind. Im Einzelnen sind dies die folgenden Funktionen:

- Statistiken über ungleichmäßige Verteilungen von Daten werden vom Optimierungsprogramm nicht berücksichtigt.
- Nur ein Teilsatz der Regeln für das Umschreiben von Abfragen wird verwendet, einschließlich der von DB2/6000 Version 1 bereitgestellten Regeln.
- Schnelle Verknüpfungsaufzählung tritt auf (siehe „Suchstrategien zur Auswahl der optimalen Verknüpfungsmethode" auf Seite 210.)
- Der Vorablesezugriff über Listen und das logische Verknüpfen von Indizes über AND sind inaktiviert, damit sie nicht in generierten Zugriffsmethoden verwendet werden.

  Anmerkung: Logisches Verknüpfen von Indizes über AND (Index ANDing) wird noch immer bei der Arbeit mit einfachen Gleichheitsverknüpfungen (Semi-Joins) verwendet, die bei Sternverknüpfungen (Star Joins) vorgefunden werden.

Die Optimierungsklasse 1 ist der Klasse 0 sehr ähnlich, abgesehen davon, dass Mischverknüpfungen und Tabellensuchen ebenfalls verfügbar sind.

2 - Diese Klasse weist das Optimierungsprogramm an, einen Optimierungsgrad zu verwenden, der den der Klasse 1 deutlich übertrifft, und hält den Kompilierungsaufwand für komplexe Abfragen wesentlich geringer als die Klassen ab 3 aufwärts. Im Einzelnen sind dies:

- Alle verfügbaren Statistiken, einschließlich der Statistiken zur Häufigkeit und zu Quantilen ungleichmäßiger Verteilungen, werden verwendet.
- Alle Regeln für das Umschreiben von Abfragen, einschließlich der Weiterleitung von Abfragen an Übersichtstabellen, werden berücksichtigt, außer den Regeln, die sehr CPU-intensiv sind und nur in seltenen Fällen zum Einsatz kommen.
- Schnelle Verknüpfungsaufzählung wird verwendet.
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen und der Weiterleitung an Übersichtstabellen.
- Die Strategie der Sternverknüpfung (Star Join) wird gegebenenfalls berücksichtigt.

Die Optimierungsklasse 2 ist der Klasse 5 sehr ähnlich, sie verwendet jedoch schnelle Verknüpfungsaufzählung und nicht dynamische pro-

grammierte Verknüpfungsaufzählung. Diese Klasse hat den höchsten Optimierungsgrad aller Klassen, die mit dem Algorithmus für schnelle Verknüpfungsaufzählung arbeiten, der für komplexe Abfragen weniger Alternativen berücksichtigt und dadurch einen geringeren Kompilierungsaufwand erfordert als die Klassen ab 3 aufwärts. Sie ist daher empfehlenswert für sehr komplexe Abfragen in einer Umgebung für Entscheidungshilfe oder analytische Online-Verarbeitung (OLAP). In solchen Fällen besteht eine hohe Wahrscheinlichkeit, dass dieselbe Abfrage nur selten ausgeführt wird, d. h. der dazugehörige Zugriffsplan bleibt sehr wahrscheinlich nicht bis zur nächsten Ausführung der Abfrage im Cache erhalten.

3 - Diese Klasse gibt an, dass ein mittlerer Grad an Optimierung zur Generierung des Zugriffsplans durchgeführt wird. Diese Klasse kommt den Merkmalen der Abfrageoptimierung von DB2 für MVS/ESA oder OS/390 am nächsten. Diese Optimierungsklasse ist durch folgende Merkmale gekennzeichnet:

- Statistiken über ungleichmäßige Verteilungen von Daten, die häufig auftretende Werte festhalten, werden verwendet, wenn sie verfügbar sind.
- Die meisten Regeln zum Umschreiben von Abfragen, einschließlich der Umsetzungen von Unterabfragen in Verknüpfungen, werden berücksichtigt.
- Dynamisch programmierte Verknüpfungsaufzählung (Dynamic Programming Join Enumeration) wird durchgeführt (siehe „Suchstrategien zur Auswahl der optimalen Verknüpfungsmethode" auf Seite 210):
  - Zusammengesetzte innere Tabellen werden in eingeschränktem Maß verwendet (siehe „Zusammengesetzte Tabellen" auf Seite 213)
  - Kartesische Produkte für Sternschemen, für die „Suchtabellen" erforderlich sind, werden in eingeschränktem Maß verwendet (siehe „Suchstrategien für eine Sternverknüpfung (Star Join)" auf Seite 211)
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen, logisches Verknüpfen von Indizes über AND (Index ANDing) und Star Joins.

Diese Klasse eignet sich für eine große Bandbreite von Anwendungen. Durch Verwendung dieser Klasse erhält das Optimierungsprogramm bessere Möglichkeiten, einen günstigen Zugriffsplan für Abfragen mit vier oder mehr Verknüpfungen auszuwählen. Es ist jedoch möglich, dass der vom Optimierungsprogramm gewählte Plan nicht so gut ist wie einer, der mit der standardmäßigen Optimierungsklasse gewählt worden wäre.

5 - Diese Klasse weist das Optimierungsprogramm an, einen bedeutenden Grad an Optimierung bei der Generierung eines Zugriffsplans durchzuführen. Klasse 5 hat zum Beispiel die folgenden Merkmale:

- Alle verfügbaren Statistiken, einschließlich der Statistiken zur Häufigkeit und zu Quantilen ungleichmäßiger Verteilungen.
- Alle Regeln für das Umschreiben von Abfragen, einschließlich der Weiterleitung von Abfragen an Übersichtstabellen, werden berücksichtigt, außer den Regeln, die sehr CPU-intensiv sind und nur in seltenen Fällen zum Einsatz kommen.
- Dynamisch programmierte Verknüpfungsaufzählung (Dynamic Programming Join Enumeration) wird durchgeführt (siehe „Suchstrategien zur Auswahl der optimalen Verknüpfungsmethode" auf Seite 210):
  - Zusammengesetzte innere Tabellen werden in eingeschränktem Maß verwendet (siehe „Zusammengesetzte Tabellen" auf Seite 213)
  - Kartesische Produkte für Sternschemen, für die „Suchtabellen" erforderlich sind, werden in eingeschränktem Maß verwendet (siehe „Suchstrategien für eine Sternverknüpfung (Star Join)" auf Seite 211)
- Viele verschiedene Zugriffsmethoden werden berücksichtigt, einschließlich des Vorablesezugriffs über Listen, des logischen Verknüpfens von Indizes über AND und der Weiterleitung an Übersichtstabellen.

Stellt das Optimierungsprogramm fest, dass die zusätzlichen Ressourcen und die Verarbeitungszeit für komplexe dynamische SQL-Abfragen nicht gewährleistet sind, wird die Optimierung reduziert. Das Ausmaß oder der Umfang der Reduzierung hängt von der Maschinengröße und der Anzahl der Vergleichselemente ab.

Reduziert das Abfrageoptimierungsprogramm den Grad an ausgeführter Abfrageoptimierung, verwendet es weiterhin alle Regeln für das Umschreiben von Abfragen, die normalerweise angewandt würden. Es verwendet jedoch die schnelle Verknüpfungszählung und reduziert die Anzahl der Zugriffsplankombinationen, die in Erwägung gezogen werden.

Die Abfrageoptimierungsklasse 5 ist hervorragend für eine gemischte Umgebung geeignet, in der sowohl Transaktionen als auch komplexe Abfragen ausgeführt werden. Diese Optimierungsklasse wurde zur Verwendung der meisten wertvollen Abfragetransformationen und anderer Optimierungstechniken für Abfragen in einer effizienten Weise entwickelt.

7 - Diese Klasse weist das Optimierungsprogramm an, einen bedeutenden Grad an Optimierung bei der Generierung eines Zugriffsplans durchzuführen. Sie entspricht der Abfrageoptimierungsklasse 5, sie reduziert jedoch nicht den Grad der Abfrageoptimierung für komplexe dynamische SQL-Abfragen.

9 - Diese Klasse weist das Optimierungsprogramm an, alle verfügbaren Optimierungstechniken anzuwenden. Dazu gehören:

- Alle verfügbaren Statistiken
- Alle Regeln für das Umschreiben von Abfragen
- Alle Möglichkeiten für Verknüpfungsaufzählungen, einschließlich kartesischer Produkte und uneingeschränkter zusammengesetzter innerer Tabellen
- Alle Zugriffsmethoden

Diese Klasse kann die Anzahl der möglichen Zugriffspläne, die vom Optimierungsprogramm ausgewertet werden, erheblich vergrößern. Diese Klasse sollte verwendet werden, um festzustellen, ob eine umfassendere Optimierung zur Generierung eines besseren Zugriffsplans für sehr komplexe und zeitintensive Abfragen auf große Tabellen führt. Die Einrichtung EXPLAIN sowie Leistungsmessungen sollten zur Überprüfung, ob ein besserer Plan gefunden wurde, herangezogen werden.

Einstellen der Optimierungsklasse

Die Art, wie eine bestimmte Optimierungsklasse für eine Abfrage angefordert wird, hängt davon ab, ob statisches oder dynamisches SQL verwendet wird.

- *Statische SQL-Anweisungen* verwenden die in den Befehlen PREP und BIND angegebene Optimierungsklasse. In der Spalte QUERYOPT in der Katalogtabelle SYSCAT.PACKAGES wird die zum Binden des Pakets verwendete Optimierungsklasse aufgezeichnet. Wenn das Paket erneut implizit oder mit dem Befehl REBIND PACKAGE gebunden wird, wird dieselbe Optimierungsklasse für die statischen SQL-Anweisungen verwendet. Wenn Sie die Optimierungsklasse für diese statischen SQL-Anweisungen ändern wollen, müssen Sie den Befehl BIND verwenden. Wenn Sie keine Optimierungsklasse angeben, verwendet DB2 die vom Datenbankkonfigurationsparameter *dft_queryopt* angegebene Standardoptimierungsklasse.
- *Dynamische SQL-Anweisungen* verwenden die Optimierungsklasse, die in dem Sonderregister CURRENT QUERY OPTIMIZATION angegeben wird, das mit der SQL-Anweisung SET gesetzt wird. Beispielsweise setzt die folgende Anweisung die Optimierungsklasse auf 1:

```
SET CURRENT QUERY OPTIMIZATION = 1
```

Um sicherzustellen, dass eine dynamische SQL-Anweisung immer dieselbe Optimierungsklasse verwendet, kann diese Anweisung SET in das Anwendungsprogramm mit aufgenommen werden. Weitere Informationen finden Sie im Handbuch *SQL Reference*.

Wenn das Register CURRENT QUERY OPTIMIZATION nicht gesetzt wird, werden die dynamischen Anweisungen mit der Standardoptimierungsklasse für Abfragen gebunden. Der Standardwert für dynamisches und statisches SQL wird durch den Wert des konfigurierbaren Datenbankparameters *dft_queryopt* festgelegt. Klasse 5 ist die standardmäßige Optimierungsklasse für Abfragen, wenn Sie den Standardwert nicht geändert haben. (Weitere Informationen zu diesem Parameter finden Sie in „Standardabfrageoptimierungsklasse (dft_queryopt)" auf Seite 517.) Die Standardwerte für die Bindeoption und die Sonderregister werden vom Datenbankkonfigurationsparameter *dft_queryopt* übernommen.

Wie viel Optimierung ist erforderlich?

Die meisten Anweisungen werden in angemessener Weise bei einem sinnvollen Einsatz von Ressourcen unter Verwendung der Standardoptimierungsklasse optimiert. Die Kompilierzeit und der Bedarf an Ressourcen für eine Abfrage werden bei einer bestimmten Optimierungsklasse hauptsächlich durch die Komplexität der Abfrage, besonders durch die Anzahl der Verknüpfungen und Unterabfragen, beeinflusst. Allerdings werden die Kompilierzeit und der Ressourcenbedarf auch vom Grad der durchgeführten Optimierung für verschiedene Optimierungsklassen beeinflusst. Für jede Optimierungsklasse können bei einer sehr komplexen Abfrage größere Unterschiede in der Kompilierzeit und im Ressourcenbedarf erwartet werden als bei einer einfachen Abfrage.

Die folgenden Regeln können Ihnen beim Auswählen der geeigneten Optimierungsklasse behilflich sein:

- Beginnen Sie mit der Standardoptimierungsklasse für Abfragen.
- Wenn Sie eine andere als die Standardklasse verwenden möchten, versuchen Sie zunächst die Klasse 1, 2 oder 3.
- Verwenden Sie eine niedrige Optimierungsklasse (0 oder 1) für Abfragen, die sehr kurze Laufzeiten haben, d. h. deren Ausführung weniger als eine Sekunde dauert. (Lesen Sie in den folgenden Erläuterungen die Informationen zu den weiteren Kriterien, wann eine niedrige Optimierungsklasse ausgewählt werden kann.)
- Verwenden Sie die Optimierungsklasse 1 oder 2, wenn Sie viele Tabellen mit vielen Verknüpfungsvergleichselementen für dieselbe Spalte haben und die Dauer der Kompilierung von Bedeutung ist.
- Verwenden Sie eine höhere Optimierungsklasse (3, 5 oder 7) für Abfragen mit langer Laufzeit, d. h. für Abfragen, deren Ausführung länger als 30 Sekunden dauert.

- Unter normalen Umständen sollten Sie die Optimierungsklasse 9 nicht verwenden.
- Führen Sie für Abfragen mit langer Laufzeit die Abfrage mit db2batch aus, um die Kompilierungszeit und die Ausführungszeit zu bestimmen.
  - Wenn die Kompilierung die meiste Zeit beansprucht, verwenden Sie eine niedrigere Optimierungsklasse.
  - Wenn die Ausführungszeit die meiste Zeit beansprucht, kommt eventuell eine höhere Optimierungsklasse in Frage.

Beachten Sie, dass die Abfrageoptimierungsklassen 1, 3, 5 und 7 alle für allgemeine Zwecke geeignet sind.

Nur wenn Sie die Dauer der Abfragekompilierung weiter verringern müssen und Sie die Art von SQL (z. B. extrem einfache Anweisungen) kennen, die ausgeführt wird, sollten Sie die Klasse 0 in Betracht ziehen. Dieses SQL hat normalerweise die folgenden Kenndaten:
- Zugriff auf eine oder nur einige wenige Tabellen
- Abrufen nur einer oder einiger weniger Zeilen
- Verwenden vollständig qualifizierter eindeutiger Indizes

Transaktionen der Online-Transaktionserarbeitung (OLTP) sind gute Beispiele für diese Art von SQL.

Für komplexe Abfragen können unterschiedliche Grade an Optimierung erforderlich sein, um den besten Zugriffsplan auszuwählen. Für Abfragen mit folgenden Merkmalen kommt möglicherweise eine höhere Optimierungsklasse in Frage:
- Zugriff auf große Tabellen
- Eine große Anzahl von Vergleichselementen
- Zahlreiche Unterabfragen
- Zahlreiche Verknüpfungen
- Zahlreiche Gruppenoperatoren wie UNION und INTERSECT
- Zahlreiche die Vergleichselemente erfüllende Zeilen
- Operationen GROUP BY und HAVING
- Verschachtelte Tabellenausdrücke
- Eine große Anzahl von Sichten

Abfragen zur Entscheidungshilfe oder Abfragen für Monatsberichte aus vollständig normalisierten Datenbanken sind gute Beispiele für komplexe Abfragen, für die zumindest die Standardoptimierungsklasse verwendet werden sollte.

Ein anderer Grund für die Verwendung höherer Optimierungsklassen ist dann gegeben, wenn das SQL von einem Abfragegenerator erstellt wurde. Viele Abfragegeneratoren erstellen SQL, das nicht effizient ist. Ineffizient geschriebene Abfragen, einschließlich der von einem Abfragegenerator erstellten, kön-

nen eine zusätzliche Optimierung erforderlich machen, um einen guten Zugriffsplan auswählen zu können. Durch Verwendung der Abfrageoptimierungsklasse 2 oder höher können schlecht geschriebene SQL-Abfragen verbessert werden.

Ein weiterer wichtiger Aspekt ist die Frage, ob statisches oder dynamisches SQL verwendet wird und ob dasselbe dynamische SQL wiederholt ausgeführt wird oder nicht. Für statisches SQL tritt der Aufwand an Kompilierzeit und Ressourcen nur einmal auf, und der ausgewählte Zugriffsplan kann mehrfach verwendet werden. Im allgemeinen gilt, dass für statisches SQL stets die Standardoptimierungsklasse verwendet werden sollte. Dynamische Anweisungen werden zur Laufzeit gebunden und ausgeführt. Daher ist hier zu überlegen, ob der Aufwand für eine zusätzliche Optimierung der dynamischen Anweisungen die allgemeine Leistung verbessert. Wenn dieselbe dynamische SQL-Anweisung jedoch wiederholt ausgeführt wird, wird der ausgewählte Zugriffsplan im Cache zwischengespeichert. Zum Auswählen einer Abfrageoptimierungsklasse kann die Anweisung wie eine statische SQL-Anweisung behandelt werden.

(Weitere Informationen zur Verwendung von statischem und dynamischem SQL finden Sie in *Application Development Guide*.)

Wenn Sie annehmen, dass für eine Abfrage eine weitere Optimierung von Vorteil wäre, Sie aber nicht sicher sind oder Bedenken hinsichtlich der Kompilierzeit oder des Ressourcenbedarfs haben, können Sie einige Vergleichstests (Benchmarktests) durchführen. Diese Tests können Anhaltspunkte über den Nutzen liefern, der sich durch verschiedene Optimierungsklassen erzielen lässt. Informationen zu allgemeinen Verfahren und zur spezifischen Verwendung des Tools `db2batch` finden Sie in „Kapitel 12. Durchführen von Vergleichstests" auf Seite 371. Beim Entwerfen und Durchführen des Vergleichstests ist zu berücksichtigen, ob die SQL-Anweisungen in Ihrer Anwendung statisch oder dynamisch sind:

- Für **dynamische** SQL-Anweisungen sollte beim Testen die durchschnittliche Laufzeit für die Anweisung verglichen werden. Mit Hilfe der folgenden Formel lässt sich die durchschnittliche Laufzeit berechnen:

  ```
      Kompilierzeit + Summe der Ausführungszeiten aller Iterationen
  ------------------------------------------------------
                 Anzahl der Iterationen
  ```

  Die `Anzahl der Iterationen` ist die Häufigkeit, mit der die SQL-Anweisung Ihrer Schätzung nach jedes Mal, wenn sie kompiliert wird, ausgeführt wird.

  Anmerkung: Nach der Erstkompilierung werden dynamische SQL-Anweisungen erneut kompiliert, wenn eine Änderung an der Umgebung eine erneute Kompilierung der Anweisung erforderlich macht. Eine zwischengespeicherte SQL-Anweisung braucht

nicht erneut kompiliert zu werden, da nachfolgende PREPARE-Anweisungen (in der Annahme, dass sich die Umgebung nicht ändert) auf die zwischengespeicherte Anweisung zugreifen. (Informationen zu einem Cache, der die Leistung bei der Verarbeitung dynamischer SQL-Anweisungen verbessern kann, finden Sie in „Katalog-Cache-Größe (catalogcache_sz)" auf Seite 409 und in „Größe des Paket-Cache (pckcachesz)" auf Seite 418.)

- Für **statische** SQL-Anweisungen sollten beim Testen die Laufzeiten der Anweisungen verglichen werden.

  Anmerkung: Obwohl es vielleicht auch interessant ist, die Kompilierzeit statischer SQL-Anweisungen zu kennen, ist die Gesamtzeit (Kompilier- und Laufzeit) der Anweisung in einem sinnvollen Kontext nur wenig aussagekräftig. Beim Vergleich der Gesamtzeit wird die Tatsache außer Acht gelassen, dass eine statische SQL-Anweisung nach jedem Binden viele Male ausgeführt werden kann und dass sie in der Regel nicht während der Laufzeit gebunden wird.

Einschränkungen für Ergebnismengen zur Leistungsverbesserung

Durch eine Anweisung SELECT wird eine Menge von Zeilen definiert, die die angegebenen Suchkriterien erfüllen. Das DB2-Optimierungsprogramm geht davon aus, dass die Anwendung alle die Kriterien erfüllenden Zeilen abrufen wird. Diese Annahme ist in OLTP-Umgebungen und bei Stapelbetrieb in der Regel zutreffend. Bei reinen Such- bzw. Anzeigeanwendungen hingegen werden durch die Abfragen häufig potenziell sehr umfangreiche Antwortmengen definiert, von denen aber normalerweise nur so viele Zeilen abgerufen werden, wie auf eine Bildschirmanzeige passen.

Die vom Optimierungsprogramm standardmäßig zugrundegelegte Annahme, dass alle den Kriterien entsprechenden Zeilen abgerufen werden, ist für Anwendungen, die Informationen in den gespeicherten Daten weder aktualisieren noch löschen, vielleicht nicht die günstigste.

Es gibt fünf Möglichkeiten, die Anweisung SELECT zur Begrenzung bzw. Änderung der Ergebnistabelle zur Leistungsverbesserung zu ändern. Diese sind:

- Klausel FOR UPDATE
- Klausel FOR READ/FETCH ONLY
- Klausel OPTIMIZE FOR n ROWS
- Klausel FETCH FIRST n ROWS ONLY
- Anweisung DECLARE CURSOR WITH HOLD

Klausel FOR UPDATE

Mit der Klausel FOR UPDATE werden die Spalten angegeben, die von einer nachfolgenden positionierten Anweisung UPDATE aktualisiert werden können. Bei der Angabe der Klausel FOR UPDATE ohne Spaltennamen werden alle Spalten, die aktualisiert werden können, in der Tabelle oder Sicht mit eingeschlossen. Bei Angabe von Spaltennamen muss jeder Name unqualifiziert angegeben werden und eine Spalte der Tabelle oder Sicht bezeichnen.

Die Klausel FOR UPDATE kann nicht verwendet werden, wenn eine der folgenden Bedingungen zutrifft:

- Der der Anweisung SELECT zugeordnete Cursor kann nicht gelöscht werden.
- Mindestens eine der durch SELECT ausgewählten Spalten ist eine Spalte, die nicht in einer Katalogtabelle aktualisiert werden kann und in der Klausel FOR UPDATE nicht ausgeschlossen wurde.

Durch die Verwendung des DB2-CLI-Verbindungsattributs SQL_ATTR_ACCESS_MODE in CLI-Anwendungen können Sie die gleichen Ergebnisse erzielen. Weitere Informationen finden Sie im Abschnitt `SQLSetConnectAttr()` des Handbuchs *CLI Guide and Reference*.

Klausel FOR READ oder FETCH ONLY

Mit der Klausel FOR READ ONLY wird sichergestellt, dass die Ergebnistabelle schreibgeschützt (Nur-Lesezugriff) ist. Die Klausel FOR FETCH ONLY hat die gleiche Funktion.

Einige Ergebnistabellen lassen per Definition nur einen Lesezugriff zu. Dies ist zum Beispiel bei einer Ergebnistabelle nach einer Anweisung SELECT auf eine Sicht, die als schreibgeschützt definiert ist, der Fall. In diesem Fall ist die Angabe der Klausel FOR READ ONLY zwar zulässig, jedoch hat die Klausel keine Wirkung.

Bei Tabellen, für die Aktualisierungen und Löschungen zulässig sind, kann die Angabe der Klausel FOR READ ONLY die Leistung von Abrufoperationen (FETCH) verbessern. Diese mögliche Verbesserung der Leistung tritt ein, wenn der Datenbankmanager Datenblockungen vornehmen kann, anstatt exklusive Sperren für die Daten zu aktivieren. Die Klausel FOR READ ONLY sollte zur Verbesserung der Leistung verwendet werden außer in Fällen, in denen Abfragen in Anweisungen für positionierte Aktualisierungen (UPDATE) bzw. Löschungen (DELETE) verwendet werden.

Durch die Verwendung des DB2-CLI-Verbindungsattributs SQL_ATTR_ACCESS_MODE in CLI-Anwendungen können Sie die gleichen Ergebnisse erzielen. Weitere Informationen finden Sie im Abschnitt `SQLSetConnectAttr()` des Handbuchs *CLI Guide and Reference*.

Klausel OPTIMIZE FOR n ROWS

Die Klausel OPTIMIZE FOR gibt Anwendungen die Möglichkeit, die Absicht zu erklären, nur eine Untermenge des Ergebnisses abzurufen bzw. den Abruf der ersten paar Zeilen mit Priorität zu behandeln. Wenn diese Absicht bekannt ist, kann das Optimierungsprogramm solchen Zugriffsplänen Priorität einräumen, die die Antwortzeit zum Abruf der ersten paar Zeilen minimieren. Darüber hinaus wird die Anzahl der Zeilen, die als ein Block an den Client gesendet werden (siehe „Zeilenblockung" auf Seite 90), durch den Wert „n" der Klausel OPTIMIZE FOR begrenzt. Daher beeinflusst die Klausel OPTIMIZE FOR sowohl die Art und Weise, wie die den Kriterien entsprechenden Zeilen aus der Datenbank vom Server abgerufen werden, als auch die Art und Weise, wie diese Zeilen an den Client zurückgegeben werden.

Nehmen Sie zum Beispiel an, Sie führen regelmäßig eine Abfrage nach den Mitarbeitern mit den höchsten Gehältern für die Tabelle EMPLOYEE aus.

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
```

Sie haben einen absteigenden Index für die Spalte SALARY definiert. Da jedoch die Mitarbeiter nach der Personalnummer (Spalte EMPNO) geordnet sind, weist der Index für die Spalte SALARY wahrscheinlich eine geringe Clusterbildung auf. In dem Bemühen, zahlreiche wahlfreie synchrone E/A-Operationen zu vermeiden, wählt das Optimierungsprogramm wahrscheinlich die Zugriffsmethode des Vorablesezugriffs über Listen (siehe „Vorablesezugriff über Listen" auf Seite 303), für die die Zeilenkennungen (RIDs, Satz-IDs) aller den Suchkriterien entsprechenden Zeilen sortiert sein müssen. Dies kann dazu führen, dass die ersten Ergebniszeilen mit einer Verzögerung an die Anwendung zurückgegeben werden. In einem solchen Fall könnte die Klausel OPTIMIZE FOR der Anweisung folgendermaßen hinzugefügt werden:

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
OPTIMIZE FOR 20 ROWS
```

Das Optimierungsprogramm wählt nun wahrscheinlich direkt den Index für die Spalte SALARY, da nun bekannt ist, dass aller Wahrscheinlichkeit nach nur die ersten zwanzig Mitarbeiter mit den höchsten Gehältern abgerufen werden. Unabhängig davon, wie viele Zeilen geblockt werden könnten, wird nun alle zwanzig Zeilen ein Zeilenblock an den Client zurückgegeben. Die Klausel OPTIMIZE FOR veranlasst das Optimierungsprogramm, solche Zugriffspläne zu bevorzugen, die Operationen an umfangreichen Datenmengen bzw. Operationen, die den Zeilenfluss unterbrechen (z. B. Sortierungen), vermeiden. Am ehesten wird ein Zugriffspfad durch die Klausel OPTIMIZE FOR 1 ROW beeinflusst. Die Verwendung dieser Klausel könnte folgende Auswirkungen haben:

- Tabellenverknüpfungssequenzen mit zusammengesetzten inneren Tabellen treten mit geringerer Wahrscheinlichkeit auf, da für sie eine temporäre Tabelle erforderlich ist.
- Die Verknüpfungsmethode könnte sich ändern. Eine Verknüpfung über Verschachtelungsschleife (Nested Loop Join) wird mit größter Wahrscheinlichkeit gewählt, da diese Methode relativ geringen Aufwand verursacht und in der Regel effizienter ist, wenn nur einige wenige Zeilen abgerufen werden sollen.
- Ein Index, der der Klausel ORDER BY entspricht, wird mit größerer Wahrscheinlichkeit ausgewählt. Dies geschieht deshalb, weil in diesem Fall keine Sortierung für die Klausel ORDER BY erforderlich wird.
- Der Vorablesezugriff über Listen wird mit geringerer Wahrscheinlichkeit verwendet, da diese Zugriffsmethode eine Sortierung erforderlich macht.
- Der sequenzielle Vorablesezugriff wird von DB2 mit geringerer Wahrscheinlichkeit angefordert, da er davon ausgeht, dass nur eine kleine Anzahl von Zeilen angezeigt werden soll.
- Bei einer Verknüpfungsabfrage wird wahrscheinlich die Tabelle mit den Spalten in der Klausel ORDER BY als äußere Tabelle gewählt, wenn es einen Index für diese äußere Tabelle gibt, der die für die Klausel ORDER BY benötigte Ordnung der Zeilen enthält.

Obwohl die Klausel OPTIMIZE FOR in allen Optimierungsklassen gültig ist (siehe „Anpassen der Optimierungsklasse" auf Seite 76), zeigt sie für Klassen ab Optimierungsklasse 3 die besten Ergebnisse. Die Verwendung der Methode der schnellen Verknüpfungsaufzählung (Greedy Join Enumeration) (siehe „Suchstrategien zur Auswahl der optimalen Verknüpfungsmethode" auf Seite 210) in den Optimierungsklassen unter 3 führt manchmal zu Zugriffsplänen für Verknüpfungen mehrerer Tabellen, die für ein schnelles Abrufen der ersten Zeilen nicht geeignet sind.

Die Klausel OPTIMIZE FOR bewirkt nicht, dass das Abrufen aller Ergebniszeilen unmöglich wird. Aber das Abrufen aller Ergebniszeilen kann wesentlich länger dauern, als wenn das Optimierungsprogramm zur Optimierung der gesamten Anwortmenge angewiesen worden wäre. Wenn Sie eine Paketanwendung haben, die Call Level Interface (DB2 CLI oder ODBC) verwendet, ist es möglich, DB2 CLI automatisch unter Verwendung des Schlüsselworts OPTIMIZEFORNROWS in der Konfigurationsdatei `db2cli.ini` eine Klausel OPTIMIZE FOR an das Ende jeder Abfrageanweisung anfügen zu lassen. Weitere Informationen dazu finden Sie im Handbuch *CLI Guide and Reference*.

Beim Auswählen von Daten aus Kurznamen können die Ergebnisse abhängig von der Unterstützung durch die Datenquelle unterschiedlich sein. Wenn die durch den Kurznamen angegebene Datenquelle die Klausel OPTIMIZE FOR unterstützt und das Optimierungsprogramm die gesamte Abfrage, die die Klausel enthält, an die Datenquelle verschiebt (Pushdown), wird die Klausel im fernen SQL generiert, das an die Datenquelle gesendet wird. Wenn die

Datenquelle diese Klausel nicht unterstützt oder das Optimierungsprogramm beschließt, die Klausel lokal auszuführen (Plan des geringsten Aufwands), wird die Klausel OPTIMIZE FOR lokal in DB2 angewandt. In diesem Fall räumt das DB2-Optimierungsprogramm auch weiterhin Zugriffsplänen Priorität ein, die die Antwortzeit für das Abrufen der ersten paar Zeilen einer Abfrage minimieren, aber die dem Optimierungsprogramm zur Verfügung stehenden Optionen zur Generierung von Plänen sind etwas beschränkt, und die Leistungsgewinne aus der Klausel OPTIMIZE FOR können möglicherweise vernachlässigt werden. Wenn sowohl die Klausel FETCH FIRST als auch die Klausel OPTIMIZE FOR angegeben werden, wird der niedrigere der beiden Werte zur Beeinflussung der Größe des Kommunikationspuffers herangezogen. Die beiden Werte werden zu Optimierungszwecken als unabhängig voneinander betrachtet. Im Abschnitt „Verwenden einer SELECT-Anweisung” auf Seite 92 finden Sie weitere Informationen über das Zusammenwirken dieser beiden Klauseln.

Klausel FETCH FIRST n ROWS ONLY

Die Klausel OPTIMIZE FOR *n* ROWS bewirkt nicht, dass das Abrufen aller Ergebniszeilen unmöglich wird. (Das Abrufen aller Ergebniszeilen kann wesentlich länger dauern, als wenn das Optimierungsprogramm zur Optimierung der gesamten Anwortmenge angewiesen worden wäre.)

Die Klausel FETCH FIRST *n* ROWS ONLY legt die maximale Anzahl von Zeilen fest, die in einer SELECT-Anweisung abgerufen werden können. Die Beschränkung der Ergebnistabelle auf die ersten paar Zeilen kann die Leistung erhöhen. Es werden nur n Zeilen abgerufen, ungeachtet der Anzahl von Zeilen, die sich in der nach einer SELECT-Anweisung ohne diese Klausel erstellten Ergebnistabelle befänden. Wenn sowohl die Klausel FETCH FIRST als auch die Klausel OPTIMIZE FOR angegeben werden, wird der niedrigere der beiden Werte zur Beeinflussung der Größe des Kommunikationspuffers herangezogen. Die beiden Werte werden zu Optimierungszwecken als unabhängig voneinander betrachtet. Im Abschnitt „Verwenden einer SELECT-Anweisung” auf Seite 92 finden Sie weitere Informationen über das Zusammenwirken dieser beiden Klauseln.

Anweisung DECLARE CURSOR WITH HOLD

Wenn Sie einen Cursor mit der Anweisung DECLARE CURSOR deklarieren, die die Klausel WITH HOLD enthält, bleiben alle geöffneten Cursor nach dem Festschreiben der Transaktion geöffnet. Darüber hinaus werden alle Sperren freigegeben, mit Ausnahme solcher Sperren, die die aktuelle Cursorposition geöffneter, mit WITH HOLD deklarierter Cursor schützen.

Wenn Sie einen Cursor mit der Anweisung DECLARE CURSOR deklarieren, die die Klausel WITH HOLD enthält, werden alle geöffneten Cursor geschlos-

sen, wenn die Transaktion mit einer ROLLBACK-Operation beendet wird. Darüber hinaus werden alle Sperren freigegeben und LOB-Querverweise gelöscht.

Einen Vergleich zur anderen primären Klausel der Anweisung CLOSE CURSOR finden Sie im Abschnitt „Anweisung CLOSE CURSOR WITH RELEASE" auf Seite 75.

Durch die Verwendung des DB2-CLI-Verbindungsattributs SQL_ATTR_CURSOR_HOLD in CLI-Anwendungen können Sie die gleichen Ergebnisse erzielen. Weitere Informationen hierzu finden Sie im Abschnitt „SQLSetStmtAttr - Set Options Related to a Statement" des Handbuchs *CLI Guide and Reference*. Wenn Sie eine Paketanwendung haben, die Call Level Interface (DB2 CLI oder ODBC) verwendet, kann über das Schlüsselwort CURSORHOLD in der Konfigurationsdatei `db2cli.ini` angegeben werden, dass DB2 CLI annehmen soll, dass für jeden deklarierten Cursor die Klausel WITH HOLD angegeben wurde. Weitere Informationen hierzu finden Sie im Abschnitt zu Schlüsselwörtern der Transaktionskonfiguration im Handbuch *CLI Guide and Reference*.

Zeilenblockung

Zeilenblockung ist eine Technik, die den Systemaufwand des Datenbankmanagers durch Abrufen eines *Blocks* von Zeilen in einer einzigen Operation reduziert. Diese Zeilen werden in einem Cache gespeichert. Jede Abrufanforderung (FETCH) in der Anwendung empfängt die nächste Zeile aus dem Cache. Wenn alle Zeilen eines Blocks verarbeitet wurden, wird vom Datenbankmanager ein anderer Block von Zeilen abgerufen.

Der Cache wird zugeordnet, wenn eine Anwendung eine Anforderung OPEN CURSOR absetzt, und wird wieder freigegeben, wenn der Cursor geschlossen wird. Die Größe des Cache wird durch einen Konfigurationsparameter festgelegt, der zur Zuordnung von Speicher für den E/A-Block verwendet wird. Der verwendete Parameter ist davon abhängig, ob der Client lokal oder fern ist:

- Für *lokale Anwendungen* wird der Parameter *aslheapsz* verwendet, um den Cache für die Zeilenblockung zuzuordnen. (Informationen zu diesem Parameter finden Sie in „Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)" auf Seite 436.)
- Für *ferne Anwendungen* wird der Parameter *rqrioblk* auf der Client-Workstation verwendet, um den *Cache* für die Zeilenblockung zuzuordnen. Der Cache wird auf dem Datenbank-Client zugeordnet. (Informationen zu diesem Parameter finden Sie in „E/A-Blockgröße für Clients (rqrioblk)" auf Seite 440.)

Für *lokale* Anwendungen können Sie mit der folgenden Formel abschätzen, wie viele Zeilen pro Block zurückgegeben werden. Dabei gilt folgendes:
- *aslheapsz* wird in Speicherseiten angegeben
- 4 096 ist die Anzahl der Byte pro Seite
- *orl* ist die Ausgabezeilenlänge in Byte:

```
Zeilen pro Block = aslheapsz * 4096 / azl
```

Für *ferne* Anwendungen können Sie mit der folgenden Formel abschätzen, wie viele Zeilen pro Block zurückgegeben werden. Dabei gilt folgendes:
- *rqrioblk* wird in Speicherbyte angegeben
- *orl* ist die Ausgabezeilenlänge in Byte:

```
Zeilen pro Block = rqrioblk / azl
```

Beachten Sie, dass bei Verwendung der Klausel FETCH FIRST *n* ROWS ONLY oder OPTIMIZE FOR *n* ROWS in einer Anweisung SELECT die Anzahl der Zeilen pro Block dem kleinsten der folgenden Werte entspricht:
- Der mit der oben angegebenen Formel berechnete Wert
- Der Wert von *n* in der Klausel FETCH FIRST
- Der Wert von *n* in der Klausel OPTIMIZE FOR

Verwenden Sie die Option BLOCKING für die Befehle PREP und BIND, um eine der folgenden Zeilenblockungsarten anzugeben:

UNAMBIG
: Die Blockung wird für Nur-Lese-Cursor und für Cursor, die nicht als „FOR UPDATE OF" definiert sind, verwendet. Mehrdeutige Cursor werden als aktualisierbar betrachtet.

ALL
: Die Blockung wird für Nur-Lese-Cursor und für Cursor, die nicht als „FOR UPDATE OF" definiert sind, verwendet. Mehrdeutige Cursor werden wie Nur-Lese-Cursor betrachtet.

NO
: Es wird keine Blockung für Cursor verwendet. Mehrdeutige Cursor werden wie Nur-Lese-Cursor betrachtet.

Einzelheiten zu diesen Arten der Zeilenblockung finden Sie in den Beschreibungen der Befehle PREP und BIND im Handbuch *Command Reference*. Wenn in den Befehlen PREP und BIND keine Option angegeben wird, ist die Standardoption für Zeilenblockung UNAMBIG. Bei Verwendung des Befehlszeilenprozessors und von Call Level Interface ist die Standardoption für Zeilenblockung ALL.

Weitere Informationen zu Cursorn finden Sie im Handbuch *SQL Reference*.

Optimieren von Abfragen

Dieser Abschnitt enthält bestimmte Aspekte und Richtlinien zur Feinabstimmung der SQL-Anweisungen in einem Anwendungsprogramm. Allgemein gilt, dass diese Richtlinien Sie beim Entwurf eines Programms unterstützen können, mit dem der Bedarf an Systemressourcen und die benötigte Zeit zum Zugriff auf Daten in einer sehr umfangreichen Tabelle minimiert werden. Abhängig vom Grad der Optimierung bei der Kompilierung der SQL-Anweisung ist eine Feinabstimmung der SQL-Anweisungen eventuell nicht erforderlich. Der SQL-Compiler kann das SQL in eine effizientere Form umschreiben. Siehe hierzu „Umschreiben der Abfrage durch den SQL-Compiler" auf Seite 176 und „Anpassen der Optimierungsklasse" auf Seite 76.

Außerdem ist zu berücksichtigen, dass der vom Optimierungsprogramm gewählte Zugriffsplan auch den Auswirkungen anderer Faktoren unterliegt, zu denen auch Umgebungsaspekte und Systemkatalogstatistiken gehören. Bei der Durchführung von Vergleichstests für Ihre Anwendungen können Sie Anpassungen vornehmen, um den Zugriffsplan zu verbessern.

Verwenden einer SELECT-Anweisung

Die Sprache SQL ist eine sehr flexible höhere Programmiersprache. Sie ermöglicht das Erstellen unterschiedlicher *SELECT-Anweisungen*, mit denen dieselben Daten abgerufen werden können. Die Leistung kann jedoch für die verschiedenen Formen und Optimierungsklassen unterschiedlich sein.

Der SQL-Compiler (einschließlich der Phasen des Umschreibens und der Optimierung von Abfragen) wählt einen Zugriffsplan, der die Ergebnismenge für die codierte Abfrage liefert. Daher, wie auch in vielen der folgenden Richtlinien vermerkt, sollten Sie die Abfrage so codieren, dass nur die benötigten Daten abgerufen werden.

Richtlinien zur Verwendung einer SELECT-Anweisung

Für die Verwendung von *SELECT-Anweisungen* gelten folgende Richtlinien:

- Geben Sie in der SELECT-Liste nur die Spalten an, die Sie benötigen. Obwohl es einfacher ist, alle Spalten mit einem Stern (*) anzugeben, kann dies zu unnötigem Verarbeitungsaufwand und der Rückgabe unerwünschter Spalten führen.
- Begrenzen Sie die Anzahl der ausgewählten Zeilen, indem Sie Vergleichselemente zur Einschränkung der Ergebnismenge auf die benötigten Zeilen verwenden. (Weitere Informationen zu verschiedenen Arten von Vergleichselementen und ihren relativen Auswirkungen auf die Leistung finden Sie in „Vergleichselementterminologie" auf Seite 200.)
- Wenn die Anzahl der Zeilen, die Sie verwenden möchten, bedeutend kleiner ist als die Gesamtanzahl der Zeilen, die abgerufen werden könnten, geben Sie die Klausel OPTIMIZE FOR in der *SELECT-Anweisung* an. Diese Klausel wirkt sich auf die Auswahl der Zugriffspläne und die Anzahl der

im Kommunikationspuffer gesperrten Zeilen aus. (Weitere Informationen finden Sie in „Zeilenblockung" auf Seite 90.)

- Wenn die Anzahl der abzurufenden Zeilen klein ist, brauchen Sie die Klausel OPTIMIZE FOR k ROWS zusätzlich zur Klausel FETCH FIRST n ROWS ONLY nicht anzugeben. Wenn n jedoch groß ist und Sie den Vorgang optimieren wollen, indem Sie die ersten k Zeilen schnell und die nachfolgenden k Zeilen mit einer möglichen Verzögerung abrufen, geben Sie beide Klauseln an. Die Größe der Kommunikationspuffer wird in Abhängigkeit vom kleineren Wert für n oder k geändert.

  ```
  SELECT EMPNAME, SALARY FROM EMPLOYEE
     ORDER BY SALARY DESC
     FETCH FIRST 100 ROWS ONLY
     OPTIMIZE FOR 20 ROWS
  ```

- Durch die Angabe der Klausel FOR READ ONLY (oder FOR FETCH ONLY) kann die Leistung verbessert werden, da sie der Abfrage erlaubt, die Zeilenblockung zu nutzen. Diese Klausel kann auch den gemeinsamen Zugriff auf Daten verbessern, da niemals exklusive Sperren (Exclusive) für Zeilen aktiviert werden, die durch eine Abfrage mit dieser Klausel abgerufen werden. Darüber hinaus ermöglicht diese Klausel ein weiter gehendes Umschreiben der Abfrage. Durch die Angabe der Klausel FOR READ ONLY (oder FOR FETCH ONLY) zusammen mit BLOCKING ALL BIND kann die Leistung von Abfragen für Kurznamen in einem System mit zusammengeschlossenen Datenbanken verbessert werden.
- Durch die Angabe der Klausel FOR UPDATE kann auch die Leistung für Cursor, die aktualisiert werden, verbessert werden, da sie dem Datenbankmanager erlaubt, zu Beginn geeignetere Sperrmodi zu wählen, um potenzielle gegenseitige Sperren (siehe „Gegenseitige Sperren" auf Seite 66) und Sperrenumwandlungen (siehe „Sperrenumwandlung" auf Seite 63) zu vermeiden.
- Vermeiden Sie nach Möglichkeit Umsetzungen numerischer Datentypen. Beim Vergleichen von Werten ist es in der Regel effizienter, zwei Werte mit demselben Datentyp miteinander zu vergleichen. Wenn Umwandlungen erforderlich sind, kann es zu Ungenauigkeiten aufgrund eingeschränkter Präzision und zu Leistungseinbußen aufgrund von Umwandlungen, die zur Laufzeit ausgeführt werden müssen, kommen.

  Verwenden Sie, falls möglich, die folgenden Datentypen:
  - Zeichen (CHAR) anstatt Zeichen variierender Länge (VARCHAR) für kurze Spalten
  - Ganze Zahlen (INTEGER) anstatt Gleitkommazahlen (FLOAT) oder Dezimalzahlen (DECIMAL)
  - Datum/Uhrzeit (DATETIME) anstatt Zeichen (CHAR)
  - Numerische Datentypen anstatt Zeichen (CHAR)
- SQL-Anweisungen mit Klauseln oder Operationen wie DISTINCT oder ORDER BY verlangen, dass die Daten sortiert werden, um die Operation ausführen zu können. Wenn Sie die Wahrscheinlichkeit, dass eine Sortiero-

peration ausgeführt wird, verringern möchten, geben Sie diese Klauseln nicht an, wenn sie nicht unbedingt erforderlich sind.

- Zur Überprüfung, ob Zeilen in einer Tabelle enthalten sind, ist folgende Anweisung

  ```
  SELECT COUNT(*) FROM  TABLENAME
  ```

  mit anschließender Überprüfung auf einen Wert ungleich null nur dann geeignet, wenn es sich um eine sehr kleine Tabelle handelt. Mit dem Anwachsen der Tabelle kann das Zählen aller Zeilen nach und nach die Leistung beeinträchtigen. Als Alternative bietet sich der Versuch an, eine einzelne Zeile auszuwählen. Dies kann entweder durch Öffnen eines Cursors und Abrufen einer Zeile oder durch eine Anweisung SELECT INTO für eine einzige Zeile geschehen. (Vergessen Sie nicht, nach dem Fehler SQLCODE -811 zu suchen, wenn von der *SELECT-Anweisung* mehr als eine Zeile gefunden wird.)
- Wenn das Aktualisierungsaufkommen gering ist und die Tabellen umfangreich sind, sollten Sie Indizes für Spalten definieren, die häufig in Vergleichselementen verwendet werden.
- Ziehen Sie die Verwendung einer IN-Liste in Betracht, wenn dieselbe Spalte in mehreren PREDICATE-Klauseln erscheint.
- Bei großen IN-Listen, die in Verbindung mit Host-Variablen verwendet werden, können Schleifen in einer Untergruppe der Host-Variablen die Leistung verbessern.

Die folgenden Vorschläge gelten insbesondere für *SELECT-Anweisungen*, die auf mehrere Tabellen zugreifen.

- Verwenden Sie Verknüpfungsvergleichselemente, wenn Sie Tabellen verknüpfen. (Ein Verknüpfungsvergleichselement ist ein Vergleich zwischen zwei Spalten verschiedener Tabellen in einer Verknüpfung.)
- Definieren Sie Indizes für die Spalten in dem Verknüpfungsvergleichselement, um eine effizientere Verarbeitung der Verknüpfung zu ermöglichen. Davon profitieren auch UPDATE- und DELETE-Anweisungen, die SELECT-Anweisungen enthalten, die auf mehrere Tabellen zugreifen.
- Vermeiden Sie nach Möglichkeit Ausdrücke oder Klauseln OR mit Verknüpfungsvergleichselementen. In einem solchen Fall können einige Verknüpfungsmethoden vom Datenbankmanager nicht verwendet werden, was dazu führen kann, dass nicht die effizienteste Verknüpfungsmethode ausgewählt wird.
- Stellen Sie sicher (falls möglich), dass in einer Umgebung mit partitionierten Datenbanken die verknüpften Tabellen beide über die Verknüpfungsspalte partitioniert sind.

Weitere Informationen finden Sie im Abschnitt „Verknüpfungskonzepte” auf Seite 203.

Weitere Informationen zur Codierung von SQL-Anweisungen mit Verknüpfungen und Unterabfragen finden Sie auch im Handbuch *Application Development Guide*.

Compound-SQL-Anweisungen

Compound-SQL-Anweisungen ermöglichen es, mehrere SQL-Anweisungen zu einem einzigen ausführbaren Anweisungsblock zusammenzufassen. Die SQL-Anweisungen innerhalb des Blocks (*Unteranweisungen*) könnten auch einzeln ausgeführt werden. Jedoch wird durch die Erstellung und Ausführung eines Anweisungsblocks der Systemaufwand seitens des Datenbankmanagers verringert. Für ferne Clients reduzieren die Compound-SQL-Anweisungen zudem die Anzahl der Anforderungen, die über das Netz übertragen werden müssen.

Es gibt zwei Arten von Compound-SQL-Anweisungen:

- **Ganzheitliche Compound-SQL-Anweisungen**

  Die Anwendung empfängt eine Antwort vom Datenbankmanager, wenn alle Unteranweisungen erfolgreich ausgeführt wurden oder wenn eine Unteranweisung mit einem Fehler beendet wurde. Wenn eine Unteranweisung mit einem Fehler beendet wurde, wird der gesamte Block als fehlerhaft angesehen, und alle Änderungen, die innerhalb des Blocks an der Datenbank vorgenommen wurden, werden rückgängig gemacht.
- **Nicht ganzheitliche Compound-SQL-Anweisungen**

  Die Anwendung empfängt eine Antwort vom Datenbankmanager, wenn alle Unteranweisungen ausgeführt wurden. Es werden alle Unteranweisungen innerhalb eines Blocks ausgeführt ohne Rücksicht darauf, ob die zuvor ausgeführte Unteranweisung erfolgreich beendet wurde oder nicht. Der Block von Anweisungen kann nur rückgängig gemacht werden, wenn die Arbeitseinheit, die diese nicht ganzheitliche Compound-SQL-Anweisung enthält, rückgängig gemacht wird.
- Ganzheitliche Compound-SQL-Anweisungen (Atomic Compound SQL) werden von DB2 Connect nicht unterstützt.
- Compound-SQL-Anweisungen werden in gespeicherten Prozeduren (auch als DARI-Routinen bezeichnet) unterstützt.
- Compound-SQL-Anweisungen werden unterstützt durch:
  - Eingebettetes statisches SQL (siehe Handbuch *SQL Reference*)
  - DB2 Call Level Interface (siehe Handbuch *CLI Guide and Reference*)
  - JDBC (siehe Handbuch *Application Development Guide*).

Dynamische Compound-Anweisungen

Eine dynamische Compound-Anweisung fasst andere SQL-Anweisungen zu einem ausführbaren Anwendungsblock zusammen. Innerhalb der dynamischen Compound-Anweisung können Sie SQL-Variablen deklarieren und Bedingungen im Zusammenhang mit SQLSTATEs deklarieren. Außerdem kann die Anweisung eine oder mehrere prozedurale SQL-Anweisungen enthalten. Wenn in der dynamischen Compound-Anweisung ein Fehler auftritt, werden alle vorhergehenden SQL-Anweisungen rückgängig gemacht und die verbleibenden SQL-Anweisungen in der dynamischen Compound-Anweisung werden nicht verarbeitet.

Die dynamische Compound-Anweisung kann in einen Auslöser, eine SQL-Funktion oder eine SQL-Methode eingebettet oder mit Hilfe dynamischer SQL-Anweisungen abgesetzt werden. Diese ausführbare Anweisung kann dynamisch vorbereitet werden. Der Aufruf der Anweisung erfordert keine Zugriffsrechte, allerdings muss die der Anweisung zugeordnete Berechtigungs-ID über die erforderlichen Zugriffsrechte für den Aufruf der eingebetteten SQL-Anweisungen innerhalb der Compound-Anweisung verfügen.

Die Unteranweisungen der Variablendeklaration enthalten Variablen. Die Unteranweisungen enthalten Bedingungen auf der Basis der SQLSTATE-Werte der Bedingungsdeklaration. Dynamische Compound-Anweisungen werden von DB2 als einzelne Anweisung kompiliert. Diese Anweisungen lassen sich effektiv für kurze Prozeduren einsetzen, die nur wenig Steuerungsflusslogik aber einen umfangreichen Datenfluss beinhalten. Für größere Konstrukte mit komplexem verschachteltem Steuerungsfluss sollten Sie die Verwendung von SQL-Prozeduren in Erwägung ziehen.

Es gibt mehrere Steuerungsflusslogik-Anweisungen, die innerhalb der dynamischen Compound-Anweisung verwendet werden können. Dazu gehören: die Anweisung FOR, die Anweisung IF, die Anweisung ITERATE und die Anweisung WHILE. Detaillierte Informationen zu diesen Anweisungen und den anderen unterstützten Anweisungen finden Sie im Handbuch *SQL Reference*.

Informationen zur Leistung und Zeichenumsetzung

Wenn die Anwendung und die Datenbank nicht dieselbe Codepage verwenden, werden die Daten aus der einen Codepage mit der anderen Codepage abgeglichen, wenn dies möglich ist. Um die Daten zwischen den Codepages der Anwendung und der Datenbank ordnungsgemäß zuordnen zu können, muss eventuell eine Datenumsetzung durchgeführt werden.

Diese Zuordnung und diese Datenumsetzung verursachen eine gewisse Erhöhung der Verarbeitungszeit für Anwendungen, die mit einer anderen Code-

page arbeiten als die Datenbank. Die Leistung der Anwendung kann verbessert werden, wenn die Anwendung und die Datenbank dieselbe Codepage oder Identitätssortierfolge verwenden.

Umsetzung von Codepages

Eine Zeichenumsetzung kann in folgenden Fällen erfolgen:

- Wenn ein Client oder eine Anwendung beim Zugriff auf eine Datenbank mit einer anderen Codepage arbeitet als die Codepage der Datenbank.

  Die Datenbankumsetzung erfolgt auf der Datenbank-Server-Maschine: aus der Codepage der Anwendung in die Codepage der Datenbank und aus der Codepage der Datenbank in die Codepage der Anwendung.
- Wenn ein Client oder eine Anwendung beim Importieren (oder Laden) einer Datei mit einer anderen Codepage arbeitet als die Datei, die importiert (oder geladen) wird.
- Wenn DB2 Connect zum Zugriff auf Daten auf einem DRDA-Server verwendet wird.

Eine Zeichenumsetzung findet in folgenden Fällen **nicht** statt:

- Bei Dateinamen.
- Bei Daten, die aus einer Spalte mit dem Attribut FOR BIT DATA stammen oder dafür vorgesehen sind, oder Daten, die in einer SQL-Operation verwendet werden, deren Ergebnis Daten mit dem Attribut FOR BIT bzw. BLOB-Daten sind.
- Bei einem DB2-Produkt oder einer Plattform, das bzw. die über keine unterstützte Umsetzungsfunktion für EUC oder UCS-2 verfügt. Bei der Ausführung Ihrer Anwendung empfangen Sie einen SQLCODE -332 (SQLSTATE 57017).

Weitere Informationen zur Unterstützung von EUC-Codepages und zur Unterstützung nationaler Zeichensätze (NLS) finden Sie im Band *Systemverwaltung: Konzept*.

Je nach Betriebssystemumgebung verwenden DB2-Datenbankmanager eine Umsetzungsfunktion und Umsetzungstabellen oder DBCS-Umsetzungs-APIs für die Umsetzung von Mehrbyte-Codepages.

Anmerkung: Zeichenfolgenumsetzungen zwischen Mehrbyte-Codepages, wie z. B. zwischen DBCS und EUC, können zu einer Verkürzung bzw. zu einer Verlängerung der Zeichenfolgen führen.

Codepunkte, die verschiedenen Zeichen in den codierten PC-Zeichensätzen für DBCS, EUC und UCS-2 zugeordnet sind, können zu unterschiedlichen Ergebnissen bei der Sortierung derselben Zeichen führen. Wenn Sortierungen

über codierte Zeichensätze für verschiedene Länder erforderlich sind, lesen Sie die Informationen im Band *Systemverwaltung: Konzept*.

Codepage-Unterstützung für Erweiterten UNIX-Code (EUC)

Die Verwendung von Host-Variablen, die Grafikdaten in C- oder C++-Anwendungen verwenden, bedarf einiger besonderer Überlegungen, zu denen Aspekte eines speziellen Precompilers, der Leistung von Anwendungen und des Anwendungsentwurfs gehören.

Für die Entwicklung von Anwendungen, die codierte EUC-Zeichensätze erfordern, sollten Sie die Informationen im Handbuch *Administrative API Reference* lesen.

Die Unterstützung von Datenbank- und Client-Anwendungen für Grafikdaten (d. h. Doppelbytezeichen) muss bei der Behandlung vieler Zeichen in den EUC-Codepages sowohl für Japanisch als auch für traditionelles Chinesisch die Längenbeschränkung von zwei Byte überwinden. Grafikdaten aus diesen EUC-Codepages werden mit Hilfe des codierten UCS-2-Zeichensatzes gespeichert und modifiziert.

Gespeicherte Prozeduren

In der Umgebung einer Datenbankanwendung wiederholen sich zahlreiche Situationen routinemäßig. Beispiele sind der Empfang einer festen Menge von Daten, Durchführung derselben Folge von Anforderungen für eine Datenbank oder das Zurückgeben einer festen Menge von Daten. Gespeicherte Prozeduren erlauben die Ausführung einer vorprogrammierten Prozedur durch einen einmaligen Aufruf an eine ferne Datenbank. Innerhalb dieses einen Aufrufs können mehrere Zugriffe auf die Datenbank durchgeführt werden.

Für die Verarbeitung einer einzigen SQL-Anweisung für eine ferne Datenbank sind zwei Übertragungen erforderlich: eine Anforderungsübertragung und eine Empfangsübertragung. Jedoch kann eine Anwendung zahlreiche SQL-Anweisungen enthalten. Ohne die gespeicherten Prozeduren wären viele Übertragungsoperationen für die Durchführung der Arbeit einer einzigen Anwendung erforderlich.

Wenn ein Datenbank-Client eine gespeicherte Prozedur verwendet, sind für den gesamten Prozess lediglich zwei Übertragungen erforderlich, wodurch die Anzahl der Übertragungen im Netzwerk verringert wird. Um eine gespeicherte Prozedur aufrufen zu können, muss die anfordernde Anwendung vor dem Aufruf die Verbindung zu der Datenbank herstellen, in der die Prozedur gespeichert ist.

In der Regel werden diese gespeicherten Prozeduren in Prozessen ausgeführt, die von den Datenbankagenten getrennt sind. Diese Trennung macht es

erforderlich, dass die Prozesse der gespeicherten Prozeduren und die Agentenprozesse über einen Router miteinander kommunizieren. Um die höchstmögliche Leistung für eine gespeicherte Prozedur zu erzielen, können Sie eine gespeicherte Prozedur als „akzeptiert" oder "nicht abgeschirmt" identifizieren und so die Prozedur direkt im Prozess des Datenbankagenten ausführen. Was bedeuten die Begriffe „akzeptiert" und „nicht abgeschirmt"?

- *Nicht abgeschirmt* (Not Fenced) bezeichnet den Umstand, dass es keinerlei Trennung der gespeicherten Prozedur von den Steuerstrukturen der Datenbank gibt, die vom Datenbankagenten verwendet werden.
- *Akzeptiert* (Trusted) gibt an, dass Sie als Administrator überzeugt sind, dass die gespeicherte Prozedur nicht versehentlich oder vorsätzlich die Steuerstrukturen der Datenbank beschädigt. Das heißt, Sie haben die Prozedur akzeptiert und verlassen sich darauf, dass sie die Integrität der Datenbank nicht gefährdet.

Beide Begriffe bedeuten dasselbe, d. h., wenn Ihre gespeicherte Prozedur "nicht abgeschirmt" wird, dann wird ihre gespeicherte Prozedur "akzeptiert". Aufgrund des vorhandenen Risikos einer Datenbankbeschädigung sollten Sie nicht abgeschirmte gespeicherte Prozeduren **nur dann** ausführen, wenn maximale Leistungsanforderungen unabdingbar sind. Darüber hinaus müssen Sie sich vergewissern, dass die Prozedur einwandfrei codiert ist und gründlich getestet wurde, bevor Sie sie für die Ausführung als nicht abgeschirmte gespeicherte Prozedur zulassen. Wenn es doch zu einem schwerwiegenden Fehler während der Ausführung einer dieser nicht abgeschirmten Prozeduren kommt, stellt der Datenbankmanager fest, ob der Fehler in der Anwendung oder im Code des Datenbankmanagers auftrat, und führt die entsprechenden Wiederherstellungsmaßnahmen aus.

Eine nicht abgeschirmte gespeicherte Prozedur kann den Datenbankmanager so beschädigen, dass dieser nicht wiederhergestellt werden kann. Dies kann zu Datenverlust und einer beschädigten Datenbank führen. Beim Ausführen akzeptierter gespeicherter Prozeduren sollte mit großer Vorsicht vorgegangen werden. In nahezu allen Fällen führt die richtige Leistungsanalyse einer Anwendung zur gewünschten Leistung, ohne dass diese Option verwendet werden muss. Die Leistung kann beispielsweise durch die Verwendung von Auslösern verbessert werden.

Es gibt zwei Methoden, eine gespeicherte Prozedur als nicht abgeschirmte Prozedur zu erstellen:

- Verwenden des Befehls CREATE PROCEDURE mit Angabe der Klausel NOT FENCED
- Speichern der Prozedur in einem speziellen Verzeichnis, wie im Handbuch *Einstieg* für Ihre Plattform beschrieben (Diese Methode funktioniert nicht für gespeicherte Java-Prozeduren.)

Zur Ausführung einer gespeicherten Prozedur muss der Endbenutzer, der die Anwendung ausführt, die die Prozedur aufruft, eine der folgenden Berechtigungen zur Laufzeit haben:

- Das Zugriffsrecht EXECUTE oder CONTROL für das Paket, dem die gespeicherte Prozedur zugeordnet ist
- Berechtigung SYSADM oder DBADM

Informationen zum Schreiben von Programmen, die mit gespeicherten Prozeduren arbeiten, finden Sie im Handbuch *Application Development Guide*.

Aktivieren einer Datenbank

Wenn eine Datenbank gestartet wird, werden verschiedene Arten von Daten im Cache zwischengespeichert. Zum Beispiel werden Datenpuffer im Pufferpool und Pakete sowie dynamische SQL-Anweisungen im Paket-Cache zwischengespeichert.

Bei häufigen Startvorgängen, d. h., wenn es kurze Perioden gibt, in denen kein Benutzer mit der Datenbank verbunden ist, und zwischen diesen Perioden andere Zeitabschnitte auftreten, in denen nur wenige Benutzer mit der Datenbank verbunden sind, gehen die Vorteile des Caching verloren, da der Cache häufig gelöscht wird. Um dies zu vermeiden, können Sie das Aktivieren der Datenbank durch die Eingabe des folgenden Befehls in Betracht ziehen:

```
DB2 ACTIVATE DATABASE datenbank
```

Mit diesem Befehl werden die angegebene Datenbank aktiviert und alle erforderlichen Services gestartet, so dass die Datenbank für die Verbindung und Verwendung durch eine beliebige Anwendung verfügbar ist. Datenbanken, die mit dem Befehl ACTIVATE DATABASE initialisiert wurden, können mit dem Befehl DEACTIVATE DATABASE oder *db2stop* inaktiviert werden. Weitere Informationen zu diesen Befehlen finden Sie im Handbuch *Command Reference*.

Parallele Verarbeitung von Anwendungen

Eine Art der Umgebung mit Parallelverarbeitung, die von DB2 unterstützt wird, ist eine Umgebung mit symmetrischen Mehrprozessormaschinen (SMP-Maschinen). In dieser Umgebung nutzen mehrere Prozessoren den Zugriff auf die Datenbank gemeinsam. Dies ermöglicht die parallele Ausführung komplexer SQL-Anforderungen, die unter den Prozessoren aufgeteilt werden können.

Sie können den Grad der zu implementierenden Parallelität bei der Kompilierung Ihrer Anwendung angeben, indem Sie das Sonderregister CURRENT DEGREE oder die Bindeoption DEGREE verwenden. Das Schlüsselwort DEGREE (Grad) bezieht sich hier einfach auf die Anzahl der gleichzeitig (par-

allel) ausgeführten Teile einer Abfrage. Es gibt keine strenge Beziehung zwischen der Anzahl der Prozessoren und dem Wert, der für den Grad der Parallelität ausgewählt wird. Während der Ausführung Ihrer Anwendung muss nicht die gesamte Anzahl der Prozessoren, die zur Verwendung in der Hardwareplattform verfügbar sind, angefordert werden. Sie können sowohl mehr als auch weniger als diese Anzahl auswählen.

Jeder weitere Grad von Parallelität erhöht den Hauptspeicherbedarf und die CPU-Auslastung im System.

Wenn Sie die Parallelität nutzen, sollten Sie sich dessen bewusst sein, dass einige Konfigurationsparameter geändert werden müssen, um die Leistung zu optimieren. Konfigurationsparameter, die die Menge des gemeinsam benutzten Speichers und den Umfang des Vorablesezugriffs steuern, sollten überprüft und entsprechend der Anforderungen einer Umgebung mit einem hohen Grad an Parallelität geändert werden. Im Abschnitt „Partitionierte Datenbank" auf Seite 535 finden Sie eine Liste von Parametern, die sich auf parallele Operationen und Umgebungen mit partitionierten Datenbanken beziehen.

Es gibt drei Konfigurationsparameter, die Sie zur Steuerung und Verwaltung der partitionsinternen Parallelität verwenden können. Mit Hilfe des Datenbankkonfigurationsparameters *intra_parallel* wird die Unterstützung für die Parallelität aktiviert oder inaktiviert. Unter Verwendung des Datenbankkonfigurationsparameters *max_querydegree* wird eine obere Begrenzung hinsichtlich des Grads an Parallelität für alle Abfragen in der Datenbank festgelegt. Durch diesen Wert werden das Sonderregister CURRENT DEGREE und die Bindeoption DEGREE außer Kraft gesetzt. Der dritte dieser Datenbankkonfigurationsparameter ist *dft_degree*. Mit diesem Parameter werden die Standardwerte für das Sonderregister CURRENT DEGREE und für die Bindeoption DEGREE festgelegt.

Weitere Informationen zur Verwendung durch Anwendungen und zur Auswirkung der Nutzung mehr als eines Grades der Parallelität finden Sie im Handbuch *Application Development Guide*.

Wenn eine Abfrage mit der Definition DEGREE = ANY ausgeführt wird, wählt der Datenbankmanager den Grad der partitionsinternen Parallelität anhand einer Reihe von Faktoren aus, zu denen die Anzahl der Prozessoren und die Merkmale der Abfrage gehören. Der tatsächlich bei der Ausführung verwendete Grad kann aufgrund dieser Faktoren niedriger als die Anzahl der Prozessoren sein.

Der Grad der Parallelität wird durch das SQL-Optimierungsprogramm beim Kompilieren der Anweisung festgelegt und kann je nach Datenbankaktivität vor der Ausführung der Abfrage angepasst werden. Der Grad der Parallelität kann niedriger als der vom SQL-Optimierungsprogramm gewählte sein, wenn

das System hoch ausgelastet ist. Der Grund hierfür liegt in der intensiven Nutzung der Systemressourcen durch die partitionsinterne Parallelität, die einerseits die abgelaufene Zeit für eine Abfrage verringert, sich aber andererseits auch negativ auf die Leistung für andere Datenbankbenutzer auswirken kann.

Der vom SQL-Optimierungsprogramm ausgewählte Grad der Parallelität kann mit Hilfe der SQL-Explain-Einrichtung zum Anzeigen des Zugriffsplans ermittelt werden. Der Grad der Parallelität, der zur Laufzeit verwendet wird, lässt sich mit dem Datenbanksystemmonitor ermitteln. Weitere Informationen zur SQL-EXPLAIN-Einrichtung und zugehörigen Dienstprogrammen finden Sie in „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251 und „Anhang C. EXPLAIN-Programme (SQL)" auf Seite 645. Weitere Informationen zum Monitor finden Sie im Handbuch *System Monitor Guide and Reference*.

Anmerkung: Der Grad der Parallelität kann unabhängig von der Hardwareumgebung festgelegt werden. Dies bedeutet, dass Sie einen Grad der Parallelität verwenden können, auch wenn Sie keine SMP-Maschine haben. Zum Beispiel können ″ein-/ausgabegebundene″ Abfragen von der Deklaration eines Grades von ″2″ oder höher profitieren. In diesem Fall braucht der Einzelprozessor vielleicht nicht auf die Beendigung von Eingabe- bzw. Ausgabefunktionen zu warten, bevor er eine neue Abfrage bearbeitet. Die Deklaration eines Grades von ″2″ oder höher steuert die E/A-Parallelität auf einer Einzelprozessormaschine nicht direkt. Dienstprogramme wie Load können die E/A-Parallelität unabhängig von einer solchen Deklaration steuern. Das Schlüsselwort `ANY` kann beim Ändern des Parameters *dft_degree* ebenfalls verwendet werden. Das Schlüsselwort `ANY` bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität bestimmt.

In vielen Fällen werden *Datenbankagenten* zur Koordinierung der parallelen Ausführung verwendet. Im Abschnitt „Datenbankagenten" auf Seite 320 finden Sie weitere Informationen und eine Liste der verschiedenen Konfigurationsparameter des Datenbankmanagers, die sich auf Datenbankagenten auswirken.

Kapitel 4. Überlegungen zur Umgebung

Neben den Faktoren, die beim Entwurf und bei der Codierung einer Anwendung zu berücksichtigen sind (siehe „Kapitel 3. Überlegungen zu Anwendungen" auf Seite 47), gibt es auch umgebungsbedingte Faktoren, die die Auswahl des Zugriffsplans für die Anwendung beeinflussen können:

- Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung
- Auswirkung der Knotengruppe auf die Abfrageoptimierung
- Auswirkung des Tabellenbereichs auf die Abfrageoptimierung
- Auswirkung des Indexierens auf die Abfrageoptimierung
- Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken.

Lesen Sie dazu auch die Informationen zu den Faktoren, die sich auf das SQL-Optimierungsprogramm auswirken, in „Kapitel 5. Systemkatalogstatistiken" auf Seite 129.

Binden Sie bei der Optimierung der Anwendungen und der Umgebung die Anwendungen erneut, nachdem Sie Änderungen in einem der oben genannten Bereiche vorgenommen haben. Dadurch wird sichergestellt, dass der optimale Zugriffsplan verwendet wird.

Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung

Verschiedene Konfigurationsparameter wirken sich auf die Auswahl des Zugriffsplans durch den SQL-Compiler aus. Viele von ihnen gelten für eine Datenbank mit Einzelpartition, während einige nur für eine partitionierte Datenbank gelten. Wenn Sie Konfigurationsparameter in einer partitionierten Datenbank ändern, empfiehlt es sich, in allen Partitionen gleiche Werte für die jeweiligen Parameter zu verwenden.

Wenn bei der Arbeit in einem System zusammengeschlossener Datenbanken die Mehrzahl der Abfragen auf Kurznamen zugreift, sollten Sie die Art der Abfrage, die Sie senden, berücksichtigen, bevor Sie Ihre Umgebung ändern. Zum Beispiel speichert der Pufferpool keine Seiten aus Datenquellen im Cache. Durch die Erhöhung des Werts für den Parameter *buffpage* allein wird also nicht sichergestellt, dass das Optimierungsprogramm weitere Alternativen bei der Erstellung eines Zugriffsplans für Abfragen, die Kurznamen enthalten, in Betracht zieht. (Als `Datenquellen` werden Datenbankverwaltungssysteme (DBMSs) und Daten innerhalb eines Systems zusammengeschlossener Datenbanken bezeichnet.) Zudem kann das Optimierungsprogramm feststellen, dass

die lokale Anlage von Datenquellentabellen der Weg mit dem geringsten Aufwand oder ein erforderlicher Schritt für eine Sortieroperation ist. In diesem Fall kann die Vergrößerung der für DB2 Universal Database verfügbaren Ressourcen die Geschwindigkeit erhöhen. Weitere Informationen finden Sie in den Abschnitten „Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken" auf Seite 121 und „Gemeinsam benutzter Speicher der Datenbank" auf Seite 404.

Es folgt eine Liste von Konfigurationsparametern, die Einfluss auf die Auswahl des Zugriffsplan durch den SQL-Compiler haben:

- „Größe des Pufferpools (buffpage)" auf Seite 404.

  Bei der Auswahl des Zugriffsplans bezieht das Optimierungsprogramm den Ein-/Ausgabeaufwand für das Laden von Seiten von der Platte in den Pufferpool in die Kalkulation mit ein. In den Berechnungen schätzt das Optimierungsprogramm die Anzahl der E/A-Operationen ab, die zur Erfüllung einer Abfrage erforderlich sind. Diese Schätzung schließt eine Voraussage über den Bedarf an Pufferpool mit ein, da zum Lesen von Zeilen einer Seite, die sich bereits im Pufferpool befindet, keine weiteren physischen E/A-Operationen anfallen. Das Optimierungsprogramm berücksichtigt den Wert der Spalte *npages* in den Systemkatalogtabellen BUFFERPOOLS bei der Abschätzung, ob eine Seite im Pufferpool gefunden wird.

  Der Ein-/Ausgabeaufwand für das Lesen der Tabellen kann sich auf folgende Bereiche auswirken:
  - Wie zwei Tabellen verknüpft werden. Eine Beschreibung finden Sie in „Festlegen von äußerer und innerer Tabelle" auf Seite 208.
  - Ob ein Index ohne Clusterbildung zum Lesen der Daten verwendet wird (siehe „Geclusterte Indizes" auf Seite 197).

  Sie können über mehr als einen Pufferpool in einer Datenbank verfügen. Ebenso können Sie mehr als einen Pufferpool in einer partitionierten Datenbank haben. Ein neuer Pufferpool kann selektiv jeder der Partitionen in der Datenbank oder allen Partitionen hinzugefügt werden. Die Spalte *npages* in den Systemkatalogtabellen BUFFERPOOLS und BUFFERPOOLSNODE wird vom Optimierungsprogramm zur Abschätzung in einer partitionierten Datenbank verwendet.

- „Grad der Parallelität (dft_degree)" auf Seite 516.

  Der Konfigurationsparameter *dft_degree* gibt den Standardwert für das Sonderregister CURRENT DEGREE und Bindeoption DEGREE an. Ein Wert von eins (1) bedeutet keine partitionsinterne Parallelität. Ein Wert von minus eins (-1) bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität anhand der Anzahl von Prozessoren und der Art der Abfrage bestimmt.

- „Standardabfrageoptimierungsklasse (dft_queryopt)" auf Seite 517.

  Beim Kompilieren von SQL-Abfragen können Sie die Abfrageoptimierungsklasse definieren, um das Optimierungsprogramm anzuweisen, verschiedene Grade der Optimierung zu verwenden. Weitere Informationen zur Auswahl einer geeigneten Abfrageoptimierungsklasse finden Sie in „Anpassen der Optimierungsklasse" auf Seite 76.
- „Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)" auf Seite 463.

  Mit Hilfe des Parameters *avg_appls* versucht das SQL-Optimierungsprogramm zu ermitteln, wie viel vom Pufferpool zur Laufzeit für den ausgewählten Zugriffsplan verfügbar ist. Höhere Werte für diesen Parameter können sich so auswirken, dass das Optimierungsprogramm einen Zugriffsplan für Abfragen auswählt, der mit dem Pufferpool etwas sparsamer umgeht. Ist der Wert für diesen Parameter 1, behandelt das Optimierungsprogramm den gesamten Pufferpool als für die Anwendung verfügbar.
- „Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422.

  Eine Sortierung wird als „über eine Pipe geleitet (piped)" betrachtet, wenn sie keine temporäre Tabelle zur Speicherung der endgültigen, sortierten Liste von Daten erforderlich macht. Das heißt, dass die Ergebnisse der Sortierung in einem einzigen sequenziellen Zugriff gelesen werden können. Über eine Pipe geleitete Sortierungen führen zu einer besseren Leistung als nicht über eine Pipe geleitete und werden daher, wenn möglich, verwendet. (Eine Definition von Sortierungen ohne Pipe im Vergleich zu Sortierungen mit Pipe finden Sie in „Einfluss des Sortierens auf das Optimierungsprogramm" auf Seite 225.)

  Bei der Auswahl eines Zugriffsplans schätzt das Optimierungsprogramm den Aufwand der Sortieroperationen, einschließlich der Möglichkeiten, eine Sortierung über eine Pipe zu leiten, folgendermaßen ab:
  - Abschätzen der Menge der zu sortierenden Daten
  - Bestimmen mit Hilfe des Parameters *sortheap*, ob genügend Speicherbereich für die Sortierung mit Pipe zur Verfügung steht.
- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415 und „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450.

  Wenn die Isolationsstufe (siehe „Gemeinsamer Zugriff" auf Seite 47) **RR** (Wiederholtes Lesen) verwendet wird, berücksichtigt das Optimierungsprogramm die Werte der Parameter *locklist* und *maxlocks*, um zu bestimmen, ob es wahrscheinlich ist, dass Sperren auf Zeilenebene durch Sperreneskalation in eine Sperre auf Tabellenebene umgewandelt werden. Wenn das Optimierungsprogramm eine Sperreneskalation für einen Tabellenzugriff voraussagt, wählt es für den Zugriffsplan eine Sperre auf Tabellenebene und vermeidet den Systemaufwand, der mit einer Sperreneskalation während der Ausführung der Abfrage verbunden wäre.

- „CPU-Geschwindigkeit (cpuspeed)" auf Seite 549.

  Der Parameter für die CPU-Geschwindigkeit wird vom SQL-Optimierungsprogramm zur Abschätzung des Aufwands für bestimmte Operationen verwendet. Das Optimierungsprogramm verwendet diese Schätzungen zum CPU-Aufwand in Verbindung mit verschiedenen Schätzungen zum Ein-/Ausgabeaufwand, um den besten Zugriffsplan für eine Abfrage auszuwählen. Die CPU-Geschwindigkeit eines Systems kann die Auswahl des Zugriffsplans wesentlich beeinflussen. Dieser Konfigurationsparameter wird bei der Installation oder Migration der Datenbank automatisch auf einen geeigneten Wert gesetzt. Sie sollten diesen Parameter **nur** anpassen, wenn Sie eine Produktionsumgebung auf einem Testsystem modellieren oder die Auswirkungen einer Änderung der Hardware testen möchten. Wenn dieser Parameter zur Modellierung einer anderen Hardwareumgebung verwendet wird, können Sie beobachten, welcher Zugriffsplan für diese andere Umgebung ausgewählt würde.
- „SQL-Anweisungszwischenspeicher (stmtheap)" auf Seite 425.

  Die Größe des Anweisungszwischenspeichers hat keinen Einfluss darauf, welchen Zugriffspfad das Optimierungsprogramm auswählt. Sie kann sich jedoch auf den Grad der Optimierung, der für komplexe SQL-Anweisungen ausgeführt wird, auswirken. Wenn der Wert für den Parameter *stmtheap* nicht groß genug ist, empfangen Sie eventuell eine SQL-Warnung, die angibt, dass nicht genügend Speicher zur Verarbeitung der Anweisung zur Verfügung steht. Zum Beispiel kann der SQLCODE +437 (SQLSTATE 01602) angeben, dass der Optimierungsgrad, der zur Kompilierung einer Anweisung verwendet wurde, geringer war als der Grad, den Sie durch die Angabe der Optimierungsklasse angefordert haben. (Der Abschnitt „Anpassen der Optimierungsklasse" auf Seite 76 enthält weitere Informationen.)
- „Maximaler Grad der Parallelität bei Abfragen (max_querydegree)" auf Seite 541.

  Wenn dieser Parameter den Wert "ANY" hat, wählt das Optimierungsprogramm den Grad der Parallelität für die Anwendung aus. Ist ein anderer Wert als "ANY" definiert, wird der vom Benutzer angegebene Wert verwendet, um den Grad der Parallelität für die Anwendung festzulegen.
- „Übertragungsbandbreite (comm_bandwidth)" auf Seite 549.

  Die Übertragungsbandbreite wird vom Optimierungsprogramm verwendet, um den Zugriffsplan zu bestimmen. Das Optimierungsprogramm verwendet den Wert dieses Parameters, um den Aufwand für bestimmte Operationen zwischen den Datenbankpartitions-Servern einer partitionierten Datenbank abzuschätzen.

Weitere Informationen enthält der Abschnitt „Optimieren der Konfigurationsparameter" auf Seite 388.

Auswirkung der Knotengruppe auf die Abfrageoptimierung

In partitionierten Datenbanken erkennt das Optimierungsprogramm die Kollokation von Tabellen und berücksichtigt sie bei der Bestimmung des besten Zugriffsplans für eine Abfrage. Im Idealfall sollten die Zeilen häufig an Verknüpfungsabfragen beteiligter Tabellen, die innerhalb einer partitionierten Datenbank auf die Partitionen verteilt sind, jeweils so verteilt sein, dass die zu verknüpfenden Zeilen jeder Tabelle in derselben Datenbankpartition gespeichert werden. Während der Verknüpfungsoperation wird durch die Kollokation (Zusammenfassung) der Daten aus beiden an der Verknüpfung beteiligten Tabellen die sonst nötige Verschiebung der Daten von einer Partition in die andere vermieden. Durch Speichern beider Tabellen in derselben Knotengruppe wird sichergestellt, dass die Daten aus den Tabellen durch Kollokation zusammengefasst werden.

Weitere Informationen zur Zusammenfassung von Tabellen finden Sie im Band *Systemverwaltung: Konzept*.

Darüber hinaus wird durch das Verteilen der Daten auf mehrere Partitionen innerhalb einer partitionierten Datenbank und je nach Größe der Tabelle die geschätzte Dauer (bzw. der Aufwand) der Ausführung einer Abfrage verringert. Die Anzahl der Tabellen, die Größe der Tabellen, die Speicherposition der Daten in diesen Tabellen und die Art der Abfrage (d. h., ob eine Verknüpfung erforderlich ist, wie oben beschrieben) haben alle ihren Einfluss auf den Systemaufwand für die Abfrage.

Auswirkung des Tabellenbereichs auf die Abfrageoptimierung

Bestimmte Merkmale der verwendeten Tabellenbereiche können die Auswahl des Zugriffsplans durch den SQL-Compiler beeinflussen:

- Kenndaten der Behälter

  Behälterkenndaten können sich wesentlich auf den Ein-/Ausgabeaufwand, der mit der Ausführung einer Abfrage verbunden ist, auswirken. Bei der Auswahl eines Zugriffsplans berücksichtigt das SQL-Optimierungsprogramm diesen Ein-/Ausgabeaufwand, einschließlich aller Unterschiede in den Aufwänden für die Zugriffe auf Daten verschiedener Tabellenbereiche. Zwei Spalten im Systemkatalog SYSCAT.TABLESPACES werden vom Optimierungsprogramm zur Abschätzung der E/A-Aufwände für den Zugriff auf Daten in einem Tabellenbereich herangezogen:

  – Die Spalte OVERHEAD, die einen Schätzwert (in Millisekunden) für die Zeit enthält, die der Behälter benötigt, bevor irgendwelche Daten in den Speicher gelesen werden. In diesen Wert fließen der Aufwand für den E/A-Controller des Behälters und die Latenzzeit der Platte, zur der auch die Suchzeit der Platte gehört, mit ein.

Mit Hilfe der folgenden Formel lässt sich der Systemaufwand (OVERHEAD) abschätzen:

```
OVERHEAD = durchschnittliche Suchzeit in Millisekunden
           + (0,5 * rotationsbedingte Latenzzeit)
```

Dabei gilt folgendes:
- 0,5 stellt den durchschnittlichen Aufwand für eine halbe Umdrehung (Rotation) dar.
- Die rotationsbedingte Latenzzeit wird für jede vollständige Umdrehung wie folgt in Millisekunden berechnet:

  ```
  (1 / U pro Minute) * 60 * 1000
  ```

  Dabei erfolgen folgende Berechnungsschritte:
  - Dividieren durch die Umdrehungen pro Minute, um die Minuten pro Umdrehung zu erhalten
  - Multiplizieren mit 60, um die Sekunden pro Umdrehung zu erhalten
  - Multiplizieren mit 1000, um die Millisekunden pro Umdrehung zu erhalten

Als Beispiel sei eine Plattengeschwindigkeit von 7 200 Umdrehungen pro Minute angenommen. Dies ergäbe folgende Formel für die rotationsbedingte Latenzzeit:

```
(1 / 7200) * 60 * 1000 = 8,328 Millisekunden
```

Das Ergebnis kann anschließend in der Berechnung des Schätzwerts für OVERHEAD bei einer angenommenen durchschnittlichen Suchzeit von 11 Millisekunden verwendet werden:

```
OVERHEAD = 11 + (0,5 * 8,328)
         = 15,164
```

Der geschätzte OVERHEAD-Wert läge in diesem Fall bei ca. 15 Millisekunden.

– Die Spalte TRANSFERRATE, die einen Schätzwert (in Millisekunden) für die Zeit enthält, die zum Einlesen einer Datenseite in den Speicher benötigt wird.

  Wenn jeder Tabellenbereichsbehälter eine einzelne physische Platte ist, können Sie mit Hilfe der folgenden Formel den Übertragungsaufwand pro Seite in Millisekunden abschätzen:

  ```
  TRANSFERRATE = (1 / spec_rate) * 1000 / 1 024 000 * seitengröße
  ```

  Dabei gilt folgendes:
  - `spec_rate` ist die Übertragungsgeschwindigkeit in MB pro Sekunde, die für die Platte angegeben wird.
  - Dividieren durch `spec_rate`, um die Sekunden pro MB zu erhalten
  - Multiplizieren mit 1000, um die Millisekunden pro MB zu erhalten

- Dividieren durch 1 024 000, um die Millisekunden pro Byte zu erhalten
- Multiplizieren mit der Seitengröße in Byte (z. B. 4 096 Byte für eine 4-KB-Seite), um die Millisekunden pro Seite zu erhalten

Als Beispiel sei eine angegebene Übertragungsgeschwindigkeit (spec_rate) von 3 MB pro Sekunde angenommen. Daraus ergäbe sich folgende Berechnung:

```
TRANSFERRATE = (1 / 3) * 1000 / 1024000 * 4096
             = 1,333248
```

Der geschätzte TRANSFERRATE-Wert läge hier bei ca. 1,3 Millisekunden pro Seite.

Wenn es sich bei den Tabellenbereichsbehältern nicht um einzelne physische Platten, sondern um Platteneinheiten (Disk-Arrays, z. B. RAID) handelt, sind zusätzliche Punkte bei der Abschätzung des zu verwendenden Werts von TRANSFERRATE beachten. Wenn die Platteneinheit relativ klein ist, können Sie den Wert für `spec_rate` mit der Anzahl Platten multiplizieren, da anzunehmen ist, dass der Engpass auf Plattenebene liegt. Ist die Anzahl der Platten in der Einheit, die den Behälter bildet, jedoch groß, liegt der Engpass vielleicht nicht auf Plattenebene, sondern bei einer der E/A-Subsystemkomponenten wie Einheiten-Controller, E/A-Busse oder dem Systembus. In diesem Fall kann nicht angenommen werden, dass der E/A-Durchsatz das Produkt aus `spec_rate` und der Anzahl der Platten ist. Statt dessen muss die tatsächliche E/A-Geschwindigkeit (in MB) während einer sequenziellen Tabellensuche gemessen werden. Zum Beispiel könnte eine Tabellensuche mit einer Anweisung wie `select count(*) from große_tabelle` durchgeführt werden, die mehrere MB umfasst. Dividieren Sie das Ergebnis durch die Anzahl der Behälter, die den Tabellenbereich bilden, in dem `große_tabelle` gespeichert ist. Setzen Sie das Ergebnis für `spec_rate` in die oben angegebene Formel ein. Zum Beispiel würde eine gemessene sequenzielle E/A-Geschwindigkeit von 100 MB beim Durchsuchen einer Tabelle in einem Tabellenbereich mit vier Behältern einen Wert von 25 MB pro Behälter bzw. einen TRANSFERRATE-Wert von (1/25) * 1000 / 1024000 * 4096 = 0,16 Millisekunden pro Seite bedeuten.

Jeder der einem Tabellenbereich zugeordneten Behälter kann sich auf einer anderen physischen Platte befinden. Um die besten Ergebnisse erzielen zu können, sollten alle physischen Platten, die für einen bestimmten Tabellenbereich verwendet werden, über die gleichen Werte für OVERHEAD und TRANSFERRATE verfügen. Wenn diese Merkmale nicht übereinstimmen, sollten Sie bei der Einstellung der Werte für OVERHEAD und TRANSFERRATE jeweils den Mittelwert verwenden.

Medienspezifische Werte für diese Spalten können Sie den technischen Daten zur Hardware entnehmen oder durch Experimentieren ermitteln. Diese Werte können in den Anweisungen CREATE TABLESPACE und ALTER TABLESPACE angegeben werden.

Experimentieren wird in der oben erwähnten Umgebung, in der eine Platteneinheit als Behälter eingesetzt wird, besonders wichtig. Es empfiehlt sich, eine einfache Abfrage, die Daten versetzt, zu erstellen und sie in Verbindung mit einem plattformspezifischen Messprogramm auszuführen. Anschließend können Sie die Abfrage mit anderen Behälterkonfigurationen innerhalb des Tabellenbereichs wiederholen. Mit Hilfe der Anweisungen CREATE und ALTER TABLESPACE können Sie die Art und Weise ändern, wie Daten in der Umgebung übertragen werden.

Die über diese beiden Werte verfügbaren Informationen zum Ein-/Ausgabeaufwand können das Optimierungsprogramm in mehrfacher Weise beeinflussen, zum Beispiel dahingehend, ob ein Index für den Zugriff auf die Daten zu verwenden ist und welche Tabellen als innere und äußere Tabellen in einer Verknüpfung auszuwählen sind.

- Vorablesezugriff

  Bei der Kalkulation des Ein-/Ausgabeaufwands für den Zugriff auf Daten in einem Tabellenbereich berücksichtigt das Optimierungsprogramm auch die potenziellen Auswirkungen, die das Vorablesen von Daten- und Indexseiten von Platten auf die Leistung der Abfrage haben kann. Der Vorablesezugriff auf Daten- und Indexseiten kann den Aufwand und die Wartezeit verringern, die mit dem Einlesen der Daten in den Pufferpool verbunden sind. Weitere Informationen finden Sie in „Vorablesen von Daten in den Pufferpool" auf Seite 301.

  Das Optimierungsprogramm verwendet die Informationen der Spalten PREFETCHSIZE und EXTENTSIZE im Systemkatalog SYSCAT.TABLESPACES, um die Menge der durch den Vorablesezugriff gelesenen Daten für einen Tabellenbereich abzuschätzen.

  – Der Wert für EXTENTSIZE kann nur bei der Erstellung eines Tabellenbereichs (z. B. mit der Anweisung CREATE TABLESPACE) festgelegt werden. Der Standardwert für EXTENTSIZE beträgt 32 Seiten (von jeweils 4 KB) und ist in der Regel ausreichend.
  – Der Wert für PREFETCHSIZE kann sowohl bei der Erstellung als auch bei der Änderung des Tabellenbereichs mit der Anweisung ALTER TABLESPACE festgelegt werden. Der Standardwert für PREFETCHSIZE wird durch den Wert des Konfigurationsparameters DFT_PREFETCH_SZ der Datenbank festgelegt, der je nach Betriebssystem unterschiedlich ist. Sie sollten die Empfehlungen zur Einstellung dieses Parameters im Abschnitt „Standardwert für PREFETCHSIZE (dft_prefetch_sz)" auf Seite 458 nachlesen und Änderungen nach Bedarf vornehmen, um das Versetzen von Daten zu optimieren.

Das folgende Beispiel zeigt die Syntax zur Änderung der Merkmale des Tabellenbereichs RESOURCE:

```
ALTER TABLESPACE RESOURCE
  PREFETCHSIZE 64
  OVERHEAD     19.3
  TRANSFERRATE 0.9
```

Nach Durchführung von Änderungen an den Tabellenbereichen sollten Sie die Anwendungen erneut binden und das Dienstprogramm RUNSTATS zum Sammeln der aktuellsten Statistikdaten zu Indizes ausführen, um sicherzustellen, dass die besten Zugriffspläne verwendet werden.

Auswirkung des Indexierens auf die Abfrageoptimierung

Die Entscheidung, wann ein Index verwendet werden sollte, wird nicht von Ihnen getroffen, sondern das Optimierungsprogramm wählt auf der Grundlage der verfügbaren Informationen über Tabellen und Indizes das am besten geeignete Verfahren aus. Allerdings obliegt Ihnen die wichtige Aufgabe der Erstellung der erforderlichen Indizes, die die Leistung verbessern können. Darüber hinaus haben Sie die Aufgabe, nach der Erstellung der Indizes Statistikdaten zu diesen Indizes (mit dem Dienstprogramm RUNSTATS) zu sammeln oder den Wert für PREFETCHSIZE (wie oben erwähnt) zu ändern sowie die Statistikdaten fortlaufend auf dem neuesten Stand zu halten. Dies setzt voraus, dass Sie mit den verschiedenen Arten von Indizes, die Sie erstellen können, und den Erstellungsmethoden vertraut sind.

Tabellen mit und ohne Index

Wenn für eine der Tabellen, auf die in einer Datenbankabfrage verwiesen wird, kein Index existiert, muss für diese Tabelle eine Tabellensuche durchgeführt werden. Je umfangreicher die Tabelle ist, desto länger dauert die Tabellensuche. Als *Tabellensuche* wird der Vorgang bezeichnet, bei dem der Datenbankmanager sequenziell auf jede Zeile einer Tabelle zugreift. Die Tabellensuche steht einer *Indexsuche* gegenüber, bei der der Datenbankmanager auf die Daten über einen Index zugreift. (Siehe „Konzepte der Indexsuche” auf Seite 188.)

Das Optimierungsprogramm entscheidet sich für die Verwendung eines Index, wenn in der SELECT-Anweisung auf die Indexspalten verwiesen wird und wenn aufgrund der Schätzungen zu erwarten ist, dass eine Indexsuche schneller als eine Tabellensuche durchgeführt werden kann. Indexdateien sind im Allgemeinen kleiner und erfordern weniger Zeit zum Lesen als eine ganze Tabelle, besonders wenn die Tabellen umfangreicher werden. Darüber hinaus braucht eventuell nicht der gesamte Index durchsucht zu werden. Vergleichselemente, die auf einen Index angewandt werden, verringern die Anzahl der Zeilen, die aus den Datenseiten gelesen werden müssen.

Jeder Indexeintrag besteht aus einem Suchschlüsselwert und einem Zeiger auf die Zeile, die den entsprechenden Wert enthält. Die Werte können in umgekehrter Richtung nur dann durchsucht werden, wenn der Parameter ALLOW REVERSE SCANS in der Anweisung CREATE INDEX angegeben wurde. Es ist also möglich, die Suche unter Angabe geeigneter Vergleichselemente zu begrenzen. Ein Index kann auch dazu verwendet werden, Zeilen in einer geordneten Reihenfolge abzurufen und somit zu vermeiden, dass der Datenbankmanager die Zeilen nach dem Lesen aus der Tabelle in einem weiteren Arbeitsgang sortieren muss. Durch Angeben des Parameters ALLOW REVERSE SCANS kann der Index zum direkten Abrufen von Zeilen in Folge vorwärts und rückwärts verwendet werden. Weitere Einzelheiten finden Sie im Handbuch *SQL Reference*.

Ein eindeutiger Index kann außer dem Suchschlüssel und dem Zeilenzeiger auch INCLUDE-Spalten enthalten.

Anmerkung: Sie haben keinen Einfluss darauf, ob sich das Optimierungsprogramm für einen Index entscheidet und ob der Datenbankmanager diesen Index verwendet oder nicht. Beispielsweise kann durch die bloße Existenz eines Index für die abgefragte Datei nicht garantiert werden, dass das Abfrageergebnis in einer geordneten Reihenfolge erstellt wird. Der Datenbankmanager kann sich für diesen Index des Optimierungsprogramms entscheiden und ihn während der Verarbeitung der Abfrage verwenden oder auch nicht. Nur durch die Angabe der Klausel ORDER BY kann die Reihenfolge der Ergebnismenge „garantiert" werden.

Durch Indizes können die Zugriffszeiten erheblich verringert werden. Aber Indizes können auch nachteilige Auswirkungen auf die Leistung haben. Vor der Erstellung von Indizes sind daher die Auswirkungen mehrerer Indizes auf den Plattenspeicherplatz und die Verarbeitungszeit zu bedenken:

- Jeder Index belegt eine bestimmte Speichermenge oder einen bestimmten Plattenspeicherplatz. Die exakte Menge ist von der Größe der Tabelle und der Größe und Anzahl von Spalten, für die der Index erstellt wird, abhängig.
- Jede für eine Tabelle ausgeführte Operation INSERT oder DELETE erfordert eine zusätzliche Aktualisierung aller Indizes für diese Tabelle. Dies gilt auch für jede Operation UPDATE, mit der ein Indexschlüssel geändert wird.
- Das Dienstprogramm LOAD erstellt alle vorhandenen Indizes neu bzw. hängt Daten an sie an.
- Der Parameter `indexfreespace` MODIFIED BY kann für den Befehl LOAD angegeben werden, um den Wert für PCTREE des Index außer Kraft zu setzen, der bei der Erstellung des Index verwendet wurde.

- Jeder Index macht potenziell einen alternativen Zugriffspfad für eine Abfrage verfügbar, den das Optimierungsprogramm berücksichtigt. Dadurch verlängert sich die Kompilierzeit der Abfrage.

Indizes sollten daher mit Umsicht gewählt werden, um den Anforderungen des Anwendungsprogramms gerecht zu werden.

Um festzustellen, ob ein Index in einem bestimmten Paket verwendet wird, können Sie die SQL-EXPLAIN-Einrichtung verwenden, die in „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251, beschrieben wird.

Verwenden von Index Advisor

DB2 Index Advisor ist ein Tool, das Sie bei der Auswahl einer optimalen Gruppe von Indizes für Ihre Tabellendaten unterstützt. Sie haben verschiedene Möglichkeiten, auf dieses Tool zuzugreifen:

- Sie können auf dieses Tool über die Steuerzentrale zugreifen, indem Sie den Ordner **Indizes** auswählen, die Maustaste 2 anklicken und anschließend **Erstellen —> Index mit Assistent** auswählen.
- Sie können auf dieses Tool von der Befehlszeile aus zugreifen, indem Sie `db2advis` eingeben.

Weitere Informationen über DB2 Index Advisor finden Sie in „SQL-Advise-Einrichtung" auf Seite 274.

Verwenden von größeren Indexschlüsseln

Es ist möglich, die Definition von Spalten mit einer Länge von mehr als 255 Bytes als Teil eines Indexschlüssels zu gestatten. Die Registrierungsvariable DB2_INDEX_2BYTEVARLEN gestattet die Verwendung von zwei statt einem Byte zur Speicherung der Länge eines Indexschlüssels.

Änderungen der Registrierungsvariablen wirken sich auf mehrere SQL-Anweisungen aus. Diese sind:

- CREATE TABLE. Primärschlüssel, Fremdschlüssel und eindeutige Schlüssel mit variablen Bestandteilen können größer als 255 Bytes sein.
- CREATE INDEX. Alle Indizes, einschließlich eindeutige Indizes und INCLUDE-Spalten mit variablen Schlüsselbestandteilen können größer als 255 Bytes sein.
- ALTER TABLE. Primärschlüssel, Fremdschlüssel und eindeutige Schlüssel mit variablen Bestandteilen können größer als 255 Bytes sein. Alle Indizes, einschließlich eindeutige Indizes und INCLUDE-Spalten mit variablen Schlüsselbestandteilen können größer als 255 Bytes sein.

  Die für Fremdschlüssel geltende Beschränkung auf 255 Byte wird unabhängig vom Wert der Registrierungsvariablen aufgehoben. Die Wahrheitsbedingung für die Übereinstimmung von Primärschlüssel und Fremdschlüssel setzt jede Beschränkung bzw. Begrenzung durch.

Wenn vorhandene Indizes auf größere Indexschlüssel umgestellt werden sollen, löschen Sie die Indizes, setzen die Registrierungsvariable DB2_INDEX_2BYTEVARLEN auf ON und erstellen die Indizes (unter Verwendung der größeren Spalten) neu.

Weitere Informationen zu SQL-Anweisungen, einschließlich Syntaxbeschreibungen, finden Sie im Handbuch *SQL Reference*.

Richtlinien zur Erstellung von Indizes

Welche Indizes erstellt werden sollten, hängt von den Daten und ihrer beabsichtigten Verwendung ab. Die folgenden Richtlinien enthalten Anhaltspunkte zur Bestimmung, welche Indizes am sinnvollsten wären:

- Definieren Sie überall dort, wo sie anwendbar sind, Primärschlüssel und eindeutige Schlüssel mit der Anweisung CREATE UNIQUE INDEX. (Das Handbuch *SQL Reference* enthält weitere Informationen.) Eindeutige Indizes können dem Optimierungsprogramm helfen, bestimmte Operationen wie Sortierungen zu vermeiden.
- Definieren Sie eindeutige Indizes mit INCLUDE-Spalten, um die Leistung beim Datenabruf zu verbessern. Spalten sind als INCLUDE-Spalten eindeutiger Indizes gut geeignet, wenn sie folgende Kriterien erfüllen:
  - Auf sie wird häufig zugegriffen, so dass die Leistung durch einen reinen Indexzugriff verbessert werden kann.
  - Sie sind zur Begrenzung des Bereichs einer Indexsuche nicht erforderlich.
  - Sie haben keinen Einfluss auf die Reihenfolge oder die Eindeutigkeit des Indexschlüssels.

  Weitere Informationen zu INCLUDE-Spalten finden Sie im Kapitel „Erstellen eines Index oder einer Indexspezifikation" im Band *Systemverwaltung: Konzept*.
- Verwenden Sie Indizes zur Optimierung häufiger Abfragen in Tabellen mit einer größeren Anzahl von Datenseiten, wie in der Spalte NPAGES in der Katalogsicht SYSCAT.TABLES eingetragen. Gehen Sie wie folgt vor:
  - Erstellen Sie einen Index für jede Spalte, die bei der Verknüpfung von Tabellen verwendet werden soll.
  - Erstellen Sie einen Index für jede Spalte, die regelmäßig nach bestimmten Werten durchsucht werden soll.
- Treffen Sie eine Entscheidung zwischen aufsteigender und absteigender Reihenfolge der Schlüssel, je nachdem, welche Reihenfolge primär genutzt bzw. angefordert wird. Die Werte können in umgekehrter Richtung nur dann durchsucht werden, wenn der Parameter ALLOW REVERSE SCANS in der Anweisung CREATE INDEX angegeben wurde. Obwohl Indizes vorwärts und rückwärts durchsucht werden können, zeigt die vorwärts ausgeführte Indexsuche (d. h. in der bei der Erstellung des Index angegebenen Reihenfolge) eine etwas bessere Leistung als die rückwärts ausgeführte Suche. Weitere Einzelheiten finden Sie im Handbuch *SQL Reference*.

- Vermeiden Sie die Erstellung von Indizes, die aus einem Teil der Schlüssel anderer Indexschlüssel für die Spalten bestehen. Wenn es zum Beispiel einen Index für die Spalten a, b und c gibt, ist es im Allgemeinen nicht sinnvoll, einen zweiten Index für die Spalten a und b zu erstellen.
- Verwenden Sie Indizes für Fremdschlüssel, um die Leistung der Operationen DELETE und UPDATE in der übergeordneten Tabelle zu verbessern.
- Verwenden Sie Indizes für Spalten, die häufig zum Sortieren von Daten verwendet werden.
- Bei der Erstellung eines Index für mehrere Spalten wählen Sie, wenn mehrere Spalten für die erste Schlüsselspalte in Frage kommen, die Spalte, die am häufigsten in einem Vergleichselement „=" (equijoin) auftritt, oder die Spalte mit der größten Anzahl unterschiedlicher Werte zuerst aus.
- Die willkürliche Erstellung von Indizes für alle Spalten nimmt nicht nur viel Plattenspeicherplatz in Anspruch, sondern verursacht auch lange Vorbereitungszeiten (PREPARE). Dies gilt besonders für komplexe Abfragen, für die eine Optimierungsklasse mit dynamisch programmierter Verknüpfungsaufzählung (Dynamic Programming Join Enumeration) verwendet wird. (Siehe „Anpassen der Optimierungsklasse" auf Seite 76).
- Die folgende *Faustregel* gibt die typische Anzahl von Indizes für eine Tabelle an. Diese Anzahl richtet sich nach der primären Verwendung der Datenbank:
  - Für Umgebungen zur Online-Transaktionsverarbeitung (OLTP) sollten nur ein oder zwei Indizes erstellt werden.
  - In reinen Abfrageumgebungen (Nur-Lese-Umgebungen) können mehr als fünf Indizes erstellt werden.
  - In gemischten Abfrage-/OLTP-Umgebungen können zwischen zwei und fünf Indizes erstellt werden.
- Ziehen Sie die Definition eines Clusterungsindex in Betracht, um neu eingefügte Zeilen nach diesem Index einzureihen. Ein Clusterungsindex sollte die Notwendigkeit, die Tabelle zu reorganisieren, erheblich herabsetzen.

  Anmerkung: Wenn ein Clusterungsindex definiert wird, sollte die Tabelle mit einem auf jeder Datenseite reservierten freien Speicherbereich geladen werden, um Einfügungen auf diesen Seiten zu ermöglichen. (Freier Speicherbereich wird durch das Schlüsselwort PCTFREE in der Anweisung ALTER TABLE oder durch die Klausel `pagefreespace` MODIFIED BY im Befehl LOAD reserviert.)
- Ziehen Sie bei der Erstellung von Indizes die Verwendung des Schlüsselworts PCTFREE in Betracht. Mit PCTFREE wird Speicherbereich auf Indexseiten für zukünftige Aktualisierungen an diesem Index reserviert. Dadurch kann sich die Häufigkeit von Seitenteilungen verringern und so die Leistung erhöhen.
- Ziehen Sie die Verwendung der Option MINPCTUSED bei der Erstellung von Indizes in Betracht. Mit der Option MINPCTUSED wird die Schwelle für die Mindestgröße des genutzten Speicherbereichs auf einer Indexseite

angegeben und die Online-Indexreorganisation aktiviert. Dadurch kann sich die Notwendigkeit einer Offline-Reorganisation der Daten und des Index verringern.

Anmerkung: Indizes werden nicht für deklarierte temporäre Tabellen unterstützt.

Im folgenden werden typische Fälle beschrieben, in denen die Erstellung eines Index die Leistung verbessern kann:

- Ein Index kann für Spalten erstellt werden, die in Klauseln WHERE der am häufigsten verarbeiteten Abfragen und Transaktionen verwendet werden.

  Betrachten Sie zum Beispiel die folgende Klausel WHERE:

  ```
  WHERE WORKDEPT='A01' OR WORKDEPT='E21'
  ```

  Diese Klausel wird im Allgemeinen von einem Index für die Spalte WORKDEPT profitieren, sofern diese Werte nicht häufig auftreten.

- Ein Index kann auf einer Spalte oder auf Spalten erstellt werden, um die Zeilen in einer Sortierfolge zu ordnen. Das Ordnen ist nicht nur für die Klausel ORDER BY erforderlich, sondern auch für andere Klauseln wie DISTINCT und GROUP BY.

  Im folgenden Beispiel wird die Klausel DISTINCT verwendet:

  ```
  SELECT DISTINCT WORKDEPT
    FROM EMPLOYEE
  ```

  Der Datenbankmanager kann einen Index verwenden, der in aufsteigender oder absteigender Reihenfolge für die Spalte WORKDEPT definiert ist, um doppelte Werte zu eliminieren. Derselbe Index könnte auch verwendet werden, um Werte wie in folgendem Beispiel mit einer Klausel GROUP BY zu gruppieren:

  ```
  SELECT WORKDEPT, AVERAGE(SALARY)
    FROM EMPLOYEE
  GROUP BY WORKDEPT
  ```

- Es kann ein Index für alle Spalten erstellt werden, auf die in einer Anweisung verwiesen wird. Wenn ein Index in dieser Weise angegeben wird, ergibt sich ein reiner Indexzugriff (Index-only), das heißt, Daten können effizienter abgerufen werden, indem ein Zugriff auf die entsprechende Tabelle vermieden wird.

  Betrachten Sie zum Beispiel die folgende SQL-Anweisung:

  ```
  SELECT LASTNAME
    FROM EMPLOYEE
    WHERE WORKDEPT IN ('A00','D11','D21')
  ```

  Wenn ein Index für die Spalten WORKDEPT und LASTNAME der Tabelle EMPLOYEE definiert ist, kann die Anweisung eventuell effizienter verarbeitet werden, indem nur der Index und nicht die gesamte Tabelle durchsucht wird.

Beachten Sie, dass hier die Spalte WORKDEPT die erste Spalte des Index sein sollte, da sich das Vergleichselement auf diese Spalte bezieht.

- INCLUDE-Spalten in einem Index sind eine andere Methode, die Verwendung von Indizes für Tabellen zu verbessern. Für das vorige Beispiel ließe sich ein eindeutiger Index folgendermaßen definieren:

  ```
  CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
  ```

  Die Angabe von lastname als INCLUDE-Spalte und nicht als Teil des Indexschlüssels bedeutet, dass lastname nur auf den äußeren Seiten (Blattseiten) des Index gespeichert wird.

Hinweise zur Leistung für die Verwaltung von Indizes

Die folgenden Erläuterungen geben Hinweise, wie die Leistung durch eine geeignete Verwendung und Verwaltung von Indizes beeinflusst werden kann:

1. **Indexerstellung**

   Bei der Erstellung von Indizes für umfangreiche Tabellen und unter Verwendung einer SMP-Maschine sollten Sie in Betracht ziehen, den Parameter *intra_parallel* auf 1 (YES) bzw. -1 (SYSTEM) zu setzen, um die Leistungsvorteile der Parallelverarbeitung zu nutzen.

   Zum Suchen und Sortieren von Daten können mehrere Prozessoren verwendet werden. Der einzige Fall, in dem die Verwendung mehrerer Prozessoren während der Indexerstellung nicht von Vorteil ist, liegt vor, wenn der Konfigurationsparameter *indexsort* der Datenbank den Wert NO (inaktiv) hat. (Der Standardwert ist YES.) Mit diesem Parameter wird gesteuert, ob die Sortierung von Indexschlüsseln während der Indexerstellung erfolgt oder nicht.

2. **Indextabellenbereich**

   Indizes können in einem anderen Tabellenbereich als dem, der zur Speicherung der anderen Tabellendaten verwendet wird, gespeichert werden. Dadurch kann Plattenspeicher eventuell besser genutzt werden, indem die Bewegungen der Schreib-/Leseköpfe reduziert werden. Sie können Indextabellenbereiche auch so erstellen, dass die Indizes auf schnelleren physischen Einheiten gespeichert werden.

   Einem Indextabellenbereich kann außerdem ein getrennter Pufferpool zugeordnet werden, wodurch verhindert werden kann, dass die Indexseiten durch Einlesen zahlreicher Datenseiten aus dem Puffer verdrängt werden.

   Wenn Indizes nicht in getrennten Tabellenbereichen gespeichert werden, verwenden sowohl die Datenseiten als auch die Indexseiten dieselben Werte für EXTENTSIZE und PREFETCHSIZE. Wenn Sie für Indizes einen anderen Tabellenbereich verwenden, haben Sie die Möglichkeit, verschiedene Werte für alle Merkmale eines Tabellenbereichs zu wählen. Da Indizes in der Regel kleiner als Tabellen sind und sich über weniger Behälter erstrecken, können in der Regel auch kleinere Werte für EXTENTSIZE

(z. B. 8 und 16) verwendet werden. Weitere Informationen finden Sie in „Vorablesezugriff auf Indexseiten" auf Seite 199. Die Verwendung schnellerer Einheiten für Tabellenbereiche wird vom SQL-Optimierungsprogramm berücksichtigt, wie im Abschnitt „Auswirkung des Tabellenbereichs auf die Abfrageoptimierung" auf Seite 107 beschrieben ist. Im Band *Systemverwaltung: Konzept* finden Sie weitere Informationen zu Tabellenbereichen.

3. **Grad der Clusterbildung**

   Wenn eine SQL-Anweisung eine Sortierung erfordert (z. B. durch die Klauseln ORDER BY, GROUP BY, DISTINCT) und ein geeigneter Index zur Erfüllung der Reihenfolge existiert, kann es dennoch Fälle geben, in denen der Datenbankmanager den Index *nicht* verwendet. Solche Fälle liegen zum Beispiel vor:
   - Wenn die Clusterbildung gering ist (siehe die Spalten CLUSTERRATIO und CLUSTERFACTOR im Systemkatalog SYSCAT.INDEXES)
   - Wenn die Tabelle so klein ist, dass es weniger aufwendig ist, die Tabelle zu durchsuchen und die Ergebnismenge im Hauptspeicher zu sortieren
   - Wenn für den Zugriff auf die Tabelle konkurrierende Indizes gibt

   Es wird empfohlen, REORG oder eine Sortierung und LOAD nach der Erstellung eines Clusterungsindex auszuführen. Im Allgemeinen ist die Clusterbildung in einer Tabelle nur anhand eines Index möglich. Ihre Tabellen und Indizes sollten in der Reihenfolge des Clusterungsindex für diese Tabelle generiert werden. Ein Clusterungsindex versucht, eine bestimmte Reihenfolge der Daten zu erhalten, wodurch die vom Dienstprogramm RUNSTATS gesammelten statistischen Werte für CLUSTERRATIO bzw. CLUSTERFACTOR verbessert werden.

   Sie sollten auch die Verwendung des Schlüsselworts PCTFREE beim Ändern einer Tabelle in Betracht ziehen, bevor Sie die Tabelle laden oder reorganisieren. Damit die Clusterung beibehalten werden kann, benötigt jede Tabelle für weitere Einfügungen verfügbaren Speicherbereich auf jeder Datenseite. Wenn der Speicherbereich verfügbar ist, können weitere Einfügungen mit den vorhandenen Daten geclustert werden. Infolgedessen ist es sinnvoll, die Daten erst dann in die Tabelle zu laden, wenn ein Prozentsatz an freiem Speicherbereich auf jeder Seite zur Clusterung weiterer Daten reserviert wurde. Sie können dies erreichen, indem Sie zuerst die Tabelle erstellen und anschließend die Tabelle mit dem Parameter PCTFREE ändern (ALTER TABLE). In ähnlicher Weise sollten Sie auch eine Änderung der Tabelle mit dem Parameter PCTFREE in Betracht ziehen, bevor Sie Ihre Daten reorganisieren. Andernfalls geht durch die Reorganisation jeder reservierte Speicherbereich verloren, wenn PCTFREE nicht definiert wurde.

   Die Clusterung wird zurzeit bei Aktualisierungen nicht beibehalten. Das heißt, wenn jemand einen Datensatz aktualisiert (UPDATE), so dass sich

der Schlüsselwert im Clusterungsindex ändert, wird dieser Datensatz nicht unbedingt auf eine entsprechend andere Seite versetzt, um die Clusterungsreihenfolge zu erhalten. Zum Erhalt der Clusterung können anstelle der Anweisung UPDATE die Anweisung DELETE und anschließend INSERT verwendet werden.

4. **Dienstprogramm RUNSTATS**

   Nach der Erstellung eines neuen Index sollten Sie das Dienstprogramm RUNSTATS ausführen, um Indexstatistikdaten zu sammeln. Anhand dieser Statistikdaten kann das Optimierungsprogramm bestimmen, ob durch die Verwendung des Index die Zugriffsleistung verbessert werden kann. Weitere Informationen finden Sie in „Erfassen statistischer Daten mit dem Dienstprogramm RUNSTATS" auf Seite 131.

5. **Reorganisieren eines Index**

   Um optimale Leistungsvorteile aus den Indizes zu ziehen, sollten Sie eine regelmäßige Reorganisation der Indizes in Betracht ziehen. Durch Aktualisierungen an den Tabellen kann der Vorablesezugriff auf Indexseiten an Effektivität verlieren. Zur Erhaltung der Effektivität des Vorablesezugriffs auf Indexseiten müssen Sie den Index reorganisieren.

   Sie können den Index entweder durch Löschen und Neuerstellen oder mit Hilfe des Dienstprogramms REORG reorganisieren. Weitere Informationen finden Sie in „Reorganisieren von Katalogen und Benutzertabellen" auf Seite 313 .

   Wenn häufige Reorganisationen vermieden werden sollen, können Sie den Parameter PCTFREE beim Erstellen eines Index angeben. Das Angeben des Parameters PCTFREE bei der Indexerstellung bewirkt, dass auf jeder äußeren Seite (Blattseite) des Index bei ihrer Erstellung freier Speicherbereich verfügbar gehalten wird. Dadurch können bei zukünftigen Aktivitäten mit dem Index Datensätze in den Index mit geringerer Wahrscheinlichkeit, Indexseitentrennungen zu verursachen, eingefügt werden. Indexseitentrennungen bewirken, dass Indexseiten nicht mehr direkt aufeinander folgen oder sequenziell sind. Dadurch vermindert sich die Möglichkeit, einen Vorablesezugriff auf Indexseiten durchzuführen. Bei einem geeigneten Wert für PCTFREE für einen Index lassen sich Indexreoganisierungen vermeiden bzw. ihre Häufigkeit herabsetzen.

   Anmerkung: Der Wert für PCTFREE, der bei der Erstellung des Index angegeben wurde, wird während der Reorganisation bei Neuerstellung des Index verwendet.

   Durch Löschen und erneutes Erstellen des Index wird eine neue Menge von Seiten erstellt, die in etwa zusammenhängend und sequenziell sind. Dadurch verbessert sich der Vorablesezugriff auf die Indexseiten, wenn er verwendet wird.

Obwohl das Dienstprogramm REORG bei der Ausführung mehr Aufwand verursacht, stellt es sicher, dass die Datenseiten geclustert werden. Diese Clusterbildung bietet größere Vorteile bei Indexsuchen, durch die auf eine bedeutende Anzahl von Datenseiten zugegriffen wird.

Bei der Ausführung in einer symmetrischen Mehrprozessorumgebung (SMP-Umgebung) nutzt das Dienstprogramm REORG mehrere Prozessoren, wenn der Parameter *intra_parallel* auf YES oder ANY gesetzt ist.

6. **Verwenden von EXPLAIN**

   Führen Sie in regelmäßigen Abständen EXPLAIN für Ihre am häufigsten verwendeten Abfragen aus, und überprüfen Sie, ob jeder Ihrer Indizes wenigstens einmal verwendet wird. Wenn ein Index in keiner Abfrage verwendet wird, empfiehlt es sich, diesen Index zu löschen.

   Verwenden Sie EXPLAIN auch, um festzustellen, ob Tabellensuchen in großen Tabellen als innere Tabellen von Verknüpfungen mit Verschachtelungsschleife verarbeitet werden. Wäre dies der Fall, würde dies bedeuten, dass ein Index für die Spalte im Verknüpfungsvergleichselement entweder fehlt oder bei der Anwendung des Verknüpfungsvergleichselements als nicht effizient eingestuft wird. Eine weitere Möglichkeit wäre, dass vielleicht das Verknüpfungsvergleichselement fehlt.

7. **Flüchtige Tabellen**

   Eine *flüchtige* Tabelle ist dadurch charakterisiert, dass ihr Inhalt zur Ausführungszeit zwischen leer und sehr umfangreich schwanken kann. Die Erstellung eines Zugriffsplans für diese Art von Tabelle kann dazu führen, dass das Optimierungsprogramm (fälschlicherweise) für den Zugriff auf Tabelle, deren Inhalt (Kardinalität) stark schwankt, die Verwendung einer Tabellensuche einer Indexsuche vorzieht.

   Durch die Deklaration einer Tabelle als „flüchtig" in der Anweisung ALTER TABLE...VOLATILE kann das Optimierungsprogramm in die Lage versetzt werden, eine Indexsuche auf der flüchtigen Tabelle auszuführen. In den folgenden Fällen verwendet das Optimierungsprogramm unabhängig von den statistischen Daten eine Indexsuche (und keine Tabellensuche):

   - Wenn alle Spalten, auf die verwiesen wird, Bestandteil des Index sind oder
   - Wenn der Index bei der Indexsuche ein Vergleichselement anwenden kann.

   Wenn es sich bei der Tabelle um eine typisierte Tabelle handelt, wird die Verwendung der Anweisung ALTER TABLE...VOLATILE nur in der Stammtabelle der Hierarchie der typisierten Tabelle unterstützt. Weitere Informationen zu diesem Thema finden Sie im Band *Systemverwaltung: Konzept* oder im Handbuch *SQL Reference*.

Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken

Ein System mit zusammengeschlossenen Datenbanken (Federated System) besteht aus einem DB2-Datenbankverwaltungssystem (DBMS), d. h. der zusammengeschlossenen Datenbank, und einer oder mehreren Datenquellen. Datenquellen werden für die zusammengeschlossene Datenbank mit Hilfe von Anweisungen CREATE SERVER angegeben. Beim Absetzen dieser Anweisungen können außerdem Server-Optionen angegeben werden, die bestimmte Aspekte des Betriebs des Systems mit der zusammengeschlossenen Datenbank bezüglich DB2 und der angegebenen Datenquelle differenzieren und steuern. Server-Optionen können später mit Hilfe von Anweisungen ALTER SERVER geändert werden. Im Handbuch *SQL Reference* finden Sie weitere Informationen zu den Anweisungen CREATE SERVER und ALTER SERVER.

Anmerkung: Sie müssen die Installationsoption *distributed join* installieren und den Parameter FEDERATED des Datenbankmanagers auf den Wert YES setzen, bevor Server erstellt und Serveroptionen angegeben werden können.

Serveroptionen und ihre zugehörigen Werte erleichtern die Pushdown-Analyse von Abfragen, die globale Optimierung und andere Aspekte von Operationen mit zusammengeschlossenen Datenbanken. Beispiel: in der Anweisung CREATE SERVER können Sie bestimmte Leistungsstatistiken als Serveroptionswerte angeben. Das heißt, Sie können die Option *cpu_ratio* auf einen Wert setzen, der die relativen Geschwindigkeiten der CPUs der Datenquelle und des zusammengeschlossenen Servers angibt.

Und Sie können die Option *io_ratio* auf einen Wert setzen, der die relativen Übertragungsgeschwindigkeiten der E/A-Einheiten der Datenquelle und des zusammengeschlossenen Servers angibt. Bei der Ausführung der Anweisung CREATE SERVER werden diese Daten der Katalogsicht SYSCAT.SERVEROPTIONS hinzugefügt, und das Optimierungsprogramm bezieht sie in die Entwicklung eines Zugriffsplans für die Datenquelle mit ein. Falls sich ein Statistikwert ändert (wie es beispielsweise bei einer Aufrüstung der CPU der Datenquelle möglich ist), können Sie die Katalogsicht SYSCAT.SERVEROPTIONS mit Hilfe der Anweisung ALTER SERVER mit dieser Änderung aktualisieren. Das Optimierungsprogramm verwendet dann diese aktualisierten Werte bei der Entwicklung des nächsten Zugriffsplans für die Datenquelle.

Tabelle 8. Server-Optionen und zugehörige Einstellungen

Option	Gültige Einstellungen	Standard-einstellung
collating_sequence	Gibt auf der Basis des codierten Zeichensatzes und der länderspezifischen Informationen an, ob die Datenquelle die gleiche Standardsortierfolge wie die zusammengeschlossene Datenbank verwendet. Wenn eine Datenquelle eine Sortierfolge besitzt, die von der Sortierfolge von DB2 abweicht, können die meisten Operationen, die von der Sortierfolge von DB2 abhängig sind, nicht fern an einer Datenquelle ausgewertet werden. Ein Beispiel hierfür wäre die Ausführung von Spaltenfunktionen MAX an der Zeichenspalte für Kurznamen an einer Datenquelle mit einer anderen Sortierfolge. Da die Ergebnisse verschieden ausfallen könnten, wenn die Funktion MAX an der fernen Datenquelle ausgewertet würde, führt DB2 die Datenverbundoperation und die Funktion MAX lokal aus. Wenn die Abfrage ein Gleichheitszeichen enthält, ist es möglich, diesen Teil der Abfrage zu verschieben (Pushdown), auch wenn die Sortierfolgen unterschiedlich (Wert 'N') sind. Zum Beispiel könnte das Vergleichselement C1 = 'A' an eine Datenquelle verschoben werden. Natürlich können Abfragen nicht verschoben werden, wenn die Sortierfolge an der Datenquelle die Groß-/Kleinschreibung nicht beachtet. Wenn eine Datenquelle die Groß-/Kleinschreibung nicht beachtet, sind die Ergebnisse der Vergleichselemente C1= 'A' und C1 = 'a' identisch, was in einer Umgebung mit Beachtung der Groß-/Kleinschreibung (DB2) nicht akzeptabel ist. Administratoren können zusammengeschlossene Datenbanken mit einer bestimmten Sortierfolge erstellen, die der Sortierfolge der Datenquellen entspricht. Diese Lösung kann die Leistung erhöhen, wenn alle Datenquellen die gleiche Sortierfolge besitzen bzw. wenn die meisten oder alle Spaltenfunktionen auf Datenquellen gerichtet sind, die die gleiche Sortierfolge verwenden. 'Y' Die Sortierfolge der Datenquelle stimmt mit der Sortierfolge der zusammengeschlossenen Datenbank überein. 'N' Die Sortierfolge der Datenquelle stimmt nicht mit der Sortierfolge der zusammengeschlossenen Datenbank überein. 'I' Die Sortierfolge der Datenquelle stimmt nicht mit der Sortierfolge der zusammengeschlossenen Datenbank überein und macht keinen Unterschied zwischen Groß- und Kleinschreibung (z. B. werden 'TOLLESON' und 'TolLESon' als gleichwertig angesehen).	'N'

Tabelle 8. Server-Optionen und zugehörige Einstellungen (Forts.)

Option	Gültige Einstellungen	Standard-einstellung
comm_rate	Gibt die Übertragungsgeschwindigkeit zwischen einem Server einer zusammengeschlossenen Datenbank und den zugeordneten Datenquellen an. Der Wert wird in MB pro Sekunde angegeben. Gültige Werte sind größer als 0 und kleiner als 2147483648. Werte können nur in in ganzen Zahlen angegeben werden, z. B. 12.	'2'
connectstring	Gibt die zum Herstellen einer Verbindung zu einem OLE-Datenbankbetreiber erforderlichen Initialisierungsmerkmale an. Die vollständige Syntax und Semantik der Verbindungszeichenfolge finden Sie im Abschnitt über die API für Kommunikationsverbindung der Kernkomponenten von OLE DB des Handbuchs *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, Microsoft Press, 1998*.	Keine
cpu_ratio	Gibt den Geschwindigkeitsunterschied zwischen der CPU einer Datenquelle und der CPU des Servers für eine zusammengeschlossene Datenbank an. Gültige Werte sind größer als 0 und kleiner als $1x10^{23}$. Werte können in einer beliebigen kombinierten Schreibweise ausgedrückt werden, z. B. 123E10, 123 oder 1,21E4.	'1.0'
dbname	Name der Datenbank einer Datenquelle, auf die der Server einer zusammengeschlossenen Datenbank zugreifen soll. Diese Angabe ist für Datenquellen der DB2-Familie erforderlich, gilt jedoch nicht für Datenquellen von Oracle, da bei Oracle-Exemplaren nur eine Datenbank enthalten ist. Bei DB2 entspricht dieser Wert einer spezifischen Datenbank in einem Exemplar bzw. bei DB2 für OS/390 dem Datenbankwert LOCATION.	Keine.

Tabelle 8. Server-Optionen und zugehörige Einstellungen (Forts.)

Option	Gültige Einstellungen	Standard-einstellung
fold_id (Siehe Anmerkungen 1 und 4 im Anschluss an diese Tabelle.)	Gilt für Benutzer-IDs, die der Server einer zusammengeschlossenen Datenbank an Datenquellen zur Identifikationsüberprüfung sendet. Gültige Werte: 'U' Der Server einer zusammengeschlossenen Datenbank setzt die Benutzer-ID vor dem Senden an die Datenquelle in Großbuchstaben um. Dies ist eine logische Auswahl für Datenquellen der DB2-Familie und von Oracle (siehe Anmerkung 2 im Anschluss an diese Tabelle). 'N' Der Server einer zusammengeschlossenen Datenbank sendet die Benutzer-ID unverändert an die Datenquelle. (Siehe Anmerkung 2 im Anschluss an diese Tabelle.) 'L' Der Server einer zusammengeschlossenen Datenbank setzt die Benutzer-ID vor dem Senden an die Datenquelle in Kleinbuchstaben um. Wenn keine dieser Einstellungen verwendet wird, versucht der Server einer zusammengeschlossenen Datenbank die Benutzer-ID in Großbuchstaben an die Datenquelle zu senden. Wird die Benutzer-ID nicht akzeptiert, versucht der Server, sie in Kleinbuchstaben zu senden.	Keine.
fold_pw (Siehe Anmerkungen 1, 3 und 4 im Anschluss an diese Tabelle.)	Gilt für Kennwörter, die der Server einer zusammengeschlossenen Datenbank an Datenquellen zur Identifikationsüberprüfung sendet. Gültige Werte: 'U' Der Server einer zusammengeschlossenen Datenbank setzt das Kennwort vor dem Senden an die Datenquelle in Großbuchstaben um. Dies ist eine logische Auswahl für Datenquellen der DB2-Familie und von Oracle. 'N' Der Server einer zusammengeschlossenen Datenbank sendet das Kennwort unverändert an die Datenquelle. 'L' Der Server einer zusammengeschlossenen Datenbank setzt das Kennwort vor dem Senden an die Datenquelle in Kleinbuchstaben um. Wenn keine dieser Einstellungen verwendet wird, versucht der Server einer zusammengeschlossenen Datenbank das Kennwort in Großbuchstaben an die Datenquelle zu senden. Wird das Kennwort nicht akzeptiert, versucht der Server, es in Kleinbuchstaben zu senden.	Keine.

Tabelle 8. Server-Optionen und zugehörige Einstellungen (Forts.)

Option	Gültige Einstellungen	Standard-einstellung
io_ratio	Gibt den Geschwindigkeitsunterschied zwischen dem E/A-System einer Datenquelle und dem E/A-System des Servers einer zusammengeschlossenen Datenbank an. Gültige Werte sind größer als 0 und kleiner als $1x10^{23}$. Werte können in einer beliebigen kombinierten Schreibweise ausgedrückt werden, z. B. 123E10, 123 oder 1,21E4.	'1.0'
node	Name, durch den eine Datenquelle als Exemplar des zugehörigen Verwaltungssystems für relationale Datenbanken definiert wird. Diese Angabe ist für alle Datenquellen erforderlich. Für eine Datenquelle der DB2-Familie ist dies der im `DB2-Knotenverzeichnis` der zusammengeschlossenen Datenbank angegebene Knoten. Dieses Verzeichnis kann mit dem Befehl **db2 list node directory** angezeigt werden. Für eine Datenquelle von Oracle ist dieser Name der Server-Name, der in der Datei `tnsnames.ora` von Oracle angegeben ist. Sie können auf der Windows NT-Plattform mit Hilfe der Option zum Anzeigen von Konfigurationsinformationen (**View Configuration Information**) des Oracle-Programms SQL Net Easy Configuration auf diesen Namen zugreifen.	Keine.
password	Gibt an, ob Kennwörter an eine Datenquelle gesendet werden. 'Y' Kennwörter werden immer an die Datenquelle gesendet und ausgewertet. Dies ist der Standardwert. 'N' Kennwörter werden nicht an die Datenquelle gesendet (ohne Rücksicht auf mögliche Benutzerzuordnungen) und werden nicht ausgewertet. 'ENCRYPTION' Kennwörter werden immer in verschlüsselter Form an die Datenquelle gesendet und ausgewertet. Dieser Wert ist nur für Datenquellen der DB2-Familie gültig, die verschlüsselte Kennwörter unterstützen.	'Y'

Tabelle 8. Server-Optionen und zugehörige Einstellungen (Forts.)

Option	Gültige Einstellungen		Standard-einstellung
plan_hints	Gibt an, ob Planhinweise (*plan hints*) aktiviert werden sollen. Planhinweise sind Anweisungsfragmente, die zusätzliche Informationen für Optimierungsprogramme von Datenquellen bereitstellen. Diese Informationen können bei einigen Abfragearten die Abfrageleistung verbessern. Die Planhinweise können das Optimierungsprogramm der Datenquelle bei der Entscheidung unterstützen, ob ein Index, und wenn ja, welcher, zu verwenden ist oder nach welcher Reihenfolge bei der Verknüpfung von Tabellen vorzugehen ist.		'N'
	'Y'	Planhinweise sollen an der Datenquelle aktiviert werden, wenn die Datenquelle Planhinweise unterstützt.	
	'N'	Planhinweise sollen an der Datenquelle nicht aktiviert werden.	
pushdown	'Y'	DB2 zieht in Betracht, die Datenquelle Operationen auswerten zu lassen.	'Y'
	'N'	DB2 ruft lediglich Spalten von der fernen Datenquelle ab und lässt die Datenquelle keine anderen Operationen, wie zum Beispiel Verknüpfungen, auswerten.	
varchar_no_trailing_blanks	Gibt an, ob diese Datenquelle eine VARCHAR-Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen verwendet. Für Zeichenfolgen variabler Länge, die keine folgenden Leerzeichen enthalten, liefert die Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen einiger DBMSs die gleichen Ergebnisse wie die Vergleichssemantik von DB2. Wenn Sie sicher sind, dass alle VARCHAR-Spalten von Tabellen und Sichten an einer Datenquelle keine folgenden Leerzeichen enthalten, können Sie in Betracht ziehen, diese Server-Option auf den Wert 'Y' für eine Datenquelle zu setzen. Diese Option wird häufig für Datenquellen von Oracle verwendet. Stellen Sie sicher, dass Sie alle Objekte, die potenziell Kurznamen haben können (einschließlich Sichten), berücksichtigen.		'N'
	'Y'	Diese Datenquelle verfügt über eine Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen, die der Vergleichssemantik von DB2 ähnlich ist.	
	'N'	Diese Datenquelle verfügt über keine mit DB2 vergleichbare Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen.	

Anmerkungen zu Tabelle 8 auf Seite 122:

1. Dieses Feld wird unabhängig von dem für die Identifikationsüberprüfung (Authentication) angegebenen Wert angewendet.
2. Da DB2 die Benutzer-IDs in Großbuchstaben speichert, sind die Werte 'N' und 'U' logisch äquivalent zueinander.
3. Die Einstellung für 'fold_pw' hat keine Bedeutung, wenn die Einstellung für 'password' den Wert 'N' hat. Da kein Kennwort gesendet wird, spielt die Groß-/Kleinschreibung keine Rolle.
4. Vermeiden Sie Nullwerte für diese Optionen. Ein Nullwert könnte attraktiv erscheinen, weil DB2 mehrere Versuche unternimmt, Benutzer-IDs und Kennwörter aufzulösen. Aber die Leistung könnte dadurch beeinträchtigt werden (es ist möglich, dass DB2 bis zu viermal eine Benutzer-ID und ein Kennwort sendet, bevor die Identifikationsüberprüfung an der Datenquelle erfolgreich durchgeführt wird).

Kapitel 5. Systemkatalogstatistiken

Bei der Optimierung von SQL-Abfragen werden die Entscheidungen, die der SQL-Compiler trifft, wesentlich vom Modell des Optimierungsprogramms über den Inhalt der Datenbank beeinflusst. Dieses Datenmodell dient dem Optimierungsprogramm zur Schätzung des Aufwands alternativer Zugriffspfade, die zur Erfüllung einer bestimmten Abfrage verwendet werden könnten.

Ein Hauptelement in diesem Datenmodell ist die Menge der statistischen Daten, die über die in der Datenbank enthaltenen Daten gesammelt und in den Systemkatalogtabellen gespeichert werden. Dazu gehören Statistiken für Tabellen, Kurznamen, Indizes, Spalten und benutzerdefinierte Funktionen (UDFs). Eine Änderung in den Statistikdaten kann dazu führen, dass ein anderer Zugriffsplan als effizienteste Methode zum Zugriff auf die gewünschten Daten ausgewählt wird.

Die folgenden statistischen Daten sind beispielsweise verfügbar, um das Datenmodell für das Optimierungsprogramm zu definieren:
- Die Anzahl der Seiten in einer Tabelle und die Anzahl der Seiten, die nicht leer sind
- Das Ausmaß, in dem Zeilen von ihrer Originalseite in andere Seiten verschoben wurden (Überlaufseiten)
- Die Anzahl der Zeilen in einer Tabelle
- Die Anzahl der unterschiedlichen Werte einer Spalte
- Der Grad der Clusterbildung eines Index. D. h. das Ausmaß, in dem die physische Reihenfolge der Zeilen einer Tabelle einem Index folgt
- Die Anzahl von Indexstufen und die Anzahl der Blattseiten (Leaf pages) in jedem Index
- Die Anzahl der Vorkommen häufig verwendeter Spaltenwerte (siehe „Erfassen und Verwenden von Verteilungsstatistiken" auf Seite 140)
- Die Verteilung von Spaltenwerten über den Bereich der in der Spalte vorhandenen Werte (siehe „Erfassen und Verwenden von Verteilungsstatistiken" auf Seite 140)
- Aufwandsschätzungen für benutzerdefinierte Funktionen (UDFs)

Statistische Daten für Objekte werden in den Systemkatalogtabellen nur bei expliziter Anforderung aktualisiert. Einige oder alle Statistiken können folgendermaßen aktualisiert werden:
- Verwenden des Dienstprogramms RUNSTATS zur Erfassung statistischer Daten (siehe „Erfassen statistischer Daten mit dem Dienstprogramm RUNSTATS" auf Seite 131)

- Verwenden des Dienstprogramms LOAD unter Angabe der Option zum Sammeln statistischer Daten
- Codieren von SQL-Anweisungen UPDATE, die an einer Gruppe vordefinierter Katalogsichten (siehe „Benutzeraktualisierbare Katalogstatistiken" auf Seite 154) Änderungen vornehmen. Beachten Sie, dass statistische Daten für benutzerdefinierte Funktionen auf diese Weise aktualisiert werden müssen (siehe „Aktualisieren der Statistiken für benutzerdefinierte Funktionen" auf Seite 162). Außer für benutzerdefinierte Funktionen sollten die Kataloge nur dann manuell aktualisiert werden, wenn bestimmte Produktionsbedingungen in der Umgebung eines Testsystems nachgebildet werden sollen oder spezielle „Fallstudien" angestellt werden sollen. Auf geschäftlich genutzten Systemen sollten die Statistiken nicht manuell aktualisiert werden.

In einem System mit zusammengeschlossenen Datenbanken besteht die einzige Möglichkeit, neue Statistikdaten für Kurznamen aus der Datenquelle zu sammeln, darin, den Kurznamen zu löschen (Drop), das Gegenstück zu RUNSTATS an der Datenquelle auszuführen und anschließend den Kurznamen wieder zu erstellen. Bei jeder Erstellung eines Kurznamens werden Statistikdaten über die zugrundeliegende Tabelle aus dem Katalog der Datenquelle abgerufen.

Sie müssen Kurznamen löschen und erneut erstellen, wenn die Datendefinitionsinformationen in der zugrundeliegenden Tabelle geändert werden. Zum Beispiel, wenn eine Spalte einer Tabellendefinition hinzugefügt wurde.

Darüber hinaus sollten Sie die Neuerstellung des Kurznamens in Betracht ziehen, wenn die Abfrageleistung abnimmt. Eine weitere Methode ist die manuelle Aktualisierung von Statistikdaten im Katalog SYSSTAT.TABLES.

Bei der Erstellung eines Kurznamens für eine Sicht ist Vorsicht geboten. Die statistischen Informationen, wie zum Beispiel die Anzahl von Zeilen, die von diesem Kurznamen zurückgegeben werden, entsprechen möglicherweise nicht dem realen Aufwand zur Auswertung dieser Sicht. Wenn die Sicht auf eine einzelne Basistabelle ohne Anwendung von Spaltenfunktionen in der SELECT-Liste definiert ist, sollten die statistischen Informationen, die dem Optimierungsprogramm zur Verfügung stehen, zutreffen. Ist die Sicht hingegen komplex, sollten Sie die Erstellung neuer Sichten über Kurznamen für die Basistabellen der Sicht auf dem DB2-UDB-Server im System mit zusammengeschlossenen Datenbanken in Betracht ziehen, damit das Optimierungsprogramm einen effizienten Plan für den Zugriff auf die Daten generieren kann.

Zusätzliche Informationen:

Die Kataloge SYSCAT und SYSSTAT enthalten Informationen zu den erfassten Statistikdaten. Folgende Informationen finden Sie im Handbuch *SQL Reference*:

- Informationen zu allen Katalogsichten und die in ihnen enthaltenen Spalten
- Informationen zu allen aktualisierbaren Katalogsichten und die in ihnen enthaltenen Spalten. Sie können diesen Abschnitt auch dann lesen, wenn Sie nur an den Statistikspalten der Katalogtabelle interessiert sind.
- Informationen zu Tabellenstatistikdaten
- Informationen zu Spaltenstatistikdaten
- Informationen zu Statistikdaten über die Spaltenverteilung
- Informationen zu Indexstatistikdaten
- Informationen zu Statistikdaten über benutzerdefinierte Funktionen

Erfassen statistischer Daten mit dem Dienstprogramm RUNSTATS

Mit dem Dienstprogramm RUNSTATS werden die Systemkatalogtabellen aktualisiert, um den Prozess zur Optimierung der Abfragen zu unterstützen. Ohne diese Statistiken könnte der Datenbankmanager für die Leistung von SQL-Anweisungen nachteilige Entscheidungen treffen. Das Dienstprogramm RUNSTATS ermöglicht es, statistische Daten über die in den Tabellen und/oder Indizes enthaltenen Daten zu erfassen.

Das Dienstprogramm RUNSTATS dient zur Erfassung statistischer Daten sowohl über die Tabellendaten als auch über die Indexdaten, um genaue Informationen für den Auswahlprozess des Zugriffsplans in folgenden Situationen bereitzustellen:

- Wenn in eine Tabelle Daten geladen wurden und die geeigneten Indizes erstellt sind.
- Wenn eine Tabelle mit dem Dienstprogramm REORG reorganisiert wurde.
- Wenn umfangreiche Aktualisierungen, Löschungen und Einfügungen stattgefunden haben, die eine Tabelle und die zugehörigen Indizes betreffen. („Umfangreich" heißt in diesem Fall, dass 10 bis 20 Prozent der Tabellen- und Indexdaten von den Änderungen betroffen sind.)
- Vor dem Binden von Anwendungsprogrammen, deren Leistung von kritischer Bedeutung ist.
- Wenn ein Vergleich mit vorigen Statistiken erstellt werden soll. Durch Ausführen des Dienstprogramms RUNSTATS in regelmäßigen Abständen können Leistungsprobleme bereits in einer frühen Phase erkannt werden.
- Wenn die Menge der vorab gelesenen Daten (PREFETCHSIZE) geändert wird.
- Wenn das Dienstprogramm REDISTRIBUTE NODEGROUP zur Umverteilung der Daten verwendet wurde.

In einer partitionierten Datenbank erfassen Sie die zu einer Tabelle und den entsprechenden Indizes gehörenden Statistikdaten, indem Sie das Dienst-

programm RUNSTATS auf einem einzigen Knoten ausführen. (Der Knoten, auf dem das Dienstprogramm ausgeführt wird, wird dadurch bestimmt, ob der Knoten, auf dem Sie den Befehl eingeben, Tabellendaten enthält oder nicht. Einzelheiten hierzu finden Sie in „Die Datenbankpartition, auf der RUNSTATS ausgeführt wird".) Da die in den Katalogen gespeicherten Statistikdaten Informationen auf Tabellenebene darstellen sollen, werden die auf Knotenebene vom Datenbankmanager gesammelten Statistikdaten gegebenenfalls mit der Anzahl der Knoten multipliziert, auf die die Tabelle partitioniert ist. Dieses Verfahren liefert einen Näherungswert für die tatsächlichen Statistikdaten, die erfasst würden, wenn RUNSTATS auf jedem Knoten ausgeführt und die Daten zu einem Gesamtergebnis aufsummiert würden.

Anmerkung: Das DB2-Abfrageoptimierungsprogramm nimmt an, dass die Attributwerte (Daten) gleichmäßig über die Datenbankpartitionen des Systems verteilt sind. Wenn die Verteilung nicht gleichmäßig ist, sollten Sie diesen Befehl in einer Datenbankpartition ausführen, die sich nach Ihrer Meinung durch eine repräsentative Tabellenverteilung auszeichnet.

Die Datenbankpartition, auf der RUNSTATS ausgeführt wird

Wenn Sie das Dienstprogramm RUNSTATS für eine Tabelle aufrufen, müssen Sie mit der Datenbank verbunden sein, in der die Tabelle gespeichert ist, jedoch muss die Datenbankpartition, von der aus Sie diesen Befehl ausführen, nicht unbedingt eine Partition für diese Tabelle enthalten:

- Wenn Sie RUNSTATS von einer Datenbankpartition aus ausführen, die eine Partition für die Tabelle enthält, wird das Datenbankpartition in dieser Datenbankpartition ausgeführt.
- Wenn Sie RUNSTATS von einer Datenbankpartition aus ausführen, die keine Partition für die Tabelle enthält, wird eine Anforderung an die erste Datenbankpartition in der Knotengruppe gesendet, die eine Partition für die Tabelle enthält. Das Dienstprogramm wird dann in dieser Datenbankpartition ausgeführt.

Analysieren der Statistikdaten

Die Analyse der Statistiken kann aufzeigen, wann eine Reorganisation der Daten erforderlich ist. Einige Anhaltspunkte dafür sind:

- Clusterbildung in den Indizes

  Bei der Erfassung der Statistik über das Clusterverhältnis (CLUSTERRATIO) wird der entsprechende Wert im Bereich von 0 bis 100 angegeben. Bei der Statistik über den Clusterfaktor (CLUSTERFACTOR) ist dieser Wert eine Zahl aus dem Bereich zwischen 0 und 1. Nur einer dieser Werte zur Erfassung der Clusterbildung wird im Katalog SYSCAT.INDEXES aufgezeichnet. Im allgemeinen kann nur einer der Indizes in einer Tabelle einen hohen Grad der Clusterbildung haben. Durch den Wert -1 wird angegeben, dass keine Statistik verfügbar ist.

Wenn Sie Verhältniswerte miteinander vergleichen möchten, multiplizieren Sie den Clusterfaktor mit 100, um einen Prozentwert für den Grad der Clusterbildung zu erhalten.

Indexsuchen, die **nicht** im reinen Indexzugriff durchgeführt werden, erzielen bei höheren Clusterverhältnissen wahrscheinlich eine bessere Leistung. Ein niedriges Clusterverhältnis führt zu vermehrten Ein-/Ausgabeoperationen für diese Art von Suche, da nach dem ersten Zugriff auf eine Datenseite die Wahrscheinlichkeit geringer ist, dass sich diese Seite immer noch im Pufferpool befindet, wenn der nächste Zugriff auf sie erfolgt. Die Leistung für einen Index ohne Clusterbildung kann durch Erhöhen der Puffergröße verbessert werden. (Lesen Sie im Abschnitt „Vorablesezugriff über Listen" auf Seite 303 die Informationen, wie der Datenbankmanager die Leistung bei der Indexsuche für Indizes mit niedrigen Clusterverhältnissen verbessern kann, und unter „Geclusterte Indizes" auf Seite 197 die Informationen zur Verwendung der Indexstatistik durch das Optimierungsprogramm.)

Wenn die Tabellendaten anfangs bezüglich eines bestimmten Index Clusterbildung aufwiesen und die Clusterinformationen nun anzeigen, dass in Bezug auf denselben Index nur eine geringe Clusterbildung vorhanden ist, kann es sinnvoll sein, die Tabelle neu zu organisieren, um die Daten bezüglich dieses Index wieder in Clustern anzuordnen.

- Überlauf von Zeilen

  Die Überlaufzahl gibt die Anzahl der Zeilen an, die nicht auf ihre ursprünglichen Seiten passen. Dies kann geschehen, wenn Spalten des Typs VARCHAR mit längeren Werten aktualisiert werden. In diesen Fällen wird an der ursprünglichen Stelle der Zeile ein Zeiger gespeichert. Dadurch kann allerdings die Leistung beeinträchtigt werden, da der Datenbankmanager jetzt dem Zeiger folgen muss, um den Inhalt der Zeile zu finden. Dies verursacht eine Verlängerung der Verarbeitungszeit und kann außerdem zu vermehrten Ein-/Ausgabeoperationen führen.

  Mit dem Ansteigen der Anzahl der Überlaufzeilen erhöht sich auch der potenzielle Nutzen einer Reorganisation der Tabellendaten. Durch eine Reorganisieren der Tabellendaten wird der Überlauf von Zeilen beseitigt.

- Vergleich von Dateiseiten

  Die Anzahl der Seiten, die Zeilen enthalten, kann mit der Gesamtanzahl der Seiten, die eine Tabelle enthält, verglichen werden. Leere Seiten werden bei einer Tabellensuche gelesen. Leere Seiten können entstehen, wenn ganze Bereiche von Zeilen gelöscht werden.

  Mit dem Ansteigen der Anzahl leerer Seiten wächst auch die Notwendigkeit einer Reorganisation einer Tabelle. Bei der Reorganisation einer Tabelle kann der Speicherbereich, der von der Tabelle eingenommen wird, komprimiert werden, indem diese leeren Seiten wieder freigegeben werden. Neben einer effizienteren Nutzung des Speicherbereichs kann die Wieder-

nutzbarmachung nicht verwendeter Seiten auch die Leistung bei einer Tabellensuche erhöhen, da weniger Seiten in den Pufferpool gelesen werden.

- Anzahl der Blattseiten (Leaf pages)

  Anhand der Anzahl von Blattseiten kann vorausberechnet werden, wie viele Ein-/Ausgabeoperationen für Indexseiten beim Durchsuchen des gesamten Index erforderlich werden.

  Wahlfreie Aktualisierungen können dazu führen, dass Seiten geteilt werden und dadurch die Größe des Index über den erforderlichen Minimalspeicherbereich hinaus anwächst. Wenn Indizes während der Reorganisation einer Tabelle neu erstellt werden, ist es möglich, jeden Index mit dem minimal erforderlichen Speicherbereich zu erstellen. Weitere Informationen zu den minimalen Speicheranforderungen für einen Index finden Sie in den Abschnitten „Auswirkung des Indexierens auf die Abfrageoptimierung" auf Seite 111 und unter „ Erstellen eines Index oder einer Indexspezifikation" im Handbuch *Systemverwaltung: Konzept*.

  Anmerkung: Bei der Neuerstellung von Indizes werden standardmäßig zehn Prozent freien Speicherbereichs jeder Indexseite nicht belegt. Sie können den Betrag des freien Speicherbereichs mit Hilfe des Parameters PCTFREE bei der Ersterstellung des Index erhöhen. Anschließend wird bei jeder Reorganisation des Index der Wert für PCTFREE verwendet. Ein freier Speicherbereich, der größer als zehn Prozent ist, kann wichtig sein, wenn Sie die Häufigkeit verringern wollen, mit der der Index reorganisiert werden muss. Der freie Speicherbereich dient zur Unterbringung neuer Indexeinfügungen.

Mit dem Dienstprogramm RUNSTATS können Sie auch feststellen, wie sich Änderungen in der Datenbank auf die Leistung auswirken. Die statistischen Daten zeigen die Datenverteilung innerhalb einer Tabelle. Wenn das Dienstprogramm RUNSTATS routinemäßig ausgeführt wird, können Daten zu Tabellen und Indizes über einen Zeitraum hinweg erhoben werden, anhand derer sich im Lauf der Entwicklung des Datenmodells Leistungstrends ablesen lassen.

Im Idealfall sollten Anwendungsprogramme nach der Ausführung von RUNSTATS erneut gebunden werden, damit das Abfrageoptimierungsprogramm einen anderen Zugriffsplan aufgrund der neuen Statistik auswählen kann.

Wenn Sie nicht über genügend Zeit verfügen, alle Statistiken auf einmal zu sammeln, können Sie RUNSTATS auch in regelmäßigen Abständen ausführen und jeweils nur einen Teil der Statistikdaten aktualisieren, die erfasst werden könnten. Wenn aufgrund der Aktivitäten an den Tabellen zwischen den Zeitpunkten, zu denen Sie RUNSTATS zu einer Teilaktualisierung ausführen,

Inkonsistenzen festgestellt werden, wird eine Warnung (SQL0437W, Ursachencode 6) ausgegeben. Sie verwenden zum Beispiel RUNSTATS zunächst, um Statistikdaten zur Tabellenverteilung zu sammeln. Später führen Sie RUNSTATS aus, um Indexstatistikdaten zu erfassen. Wenn Inkonsistenzen wegen der Aktivitäten an der Tabelle festgestellt werden, werden die Statistikdaten zur Tabellenverteilung gelöscht und die Warnung ausgegeben. Wenn dies geschieht, ist es ratsam, RUNSTATS zur Erfassung von Statistikdaten zur Tabellenverteilung ausführen.

RUNSTATS sollte von Zeit zu Zeit zur gleichzeitigen Erfassung von Tabellenstatistikdaten und Indexstatistikdaten ausgeführt werden, um sicherzustellen, dass die Indexstatistikdaten mit den Tabellenstatistikdaten synchronisiert sind. Indexstatistikdaten behalten die Mehrheit der bei der letzten Ausführung von RUNSTATS gesammelten Tabellen- und Spaltenstatistikdaten bei. Wenn seit der letzten Erfassung von Tabellenstatistikdaten umfangreiche Änderungen an der Tabelle vorgenommen wurden, geht durch die Erfassung nur der Indexstatistikdaten für diese Tabelle die Synchronisierung der beiden Arten von Statistikdaten verloren.

In folgenden Situationen kann es sinnvoll sein, statistische Daten nur für Indexdaten zu sammeln:
- Seit der letzten Ausführung des Dienstprogramms RUNSTATS wurde ein neuer Index erstellt, und Sie möchten nicht erneut statistische Daten für die Tabellendaten sammeln.
- Es wurden zahlreiche Änderungen an den Daten vorgenommen, die die erste Spalte eines Index betreffen.

Mit dem Dienstprogramm RUNSTATS können unterschiedliche Ebenen statistischer Daten gesammelt werden. Für Tabellen können statistische Basisdaten oder auch Verteilungsdaten über die Spaltenwerte innerhalb einer Tabelle gesammelt werden (siehe „Erfassen und Verwenden von Verteilungsstatistiken" auf Seite 140). Für Indizes können statistische Basisdaten oder auch Detaildaten gesammelt werden, die das Optimierungsprogramm bei der Ermittlung des Ein-/Ausgabeaufwands für eine Indexsuche unterstützen. (Informationen über diese „Detailstatistik" finden Sie in „Geclusterte Indizes" auf Seite 197).

Anmerkung: Für Spalten mit den Datentypen LONG oder LOB (großes Objekt) bzw. strukturierte Spalten werden keine statistischen Daten gesammelt. Bei Zeilentypen werden die statistischen Daten auf Tabellenebene für NPAGES, FPAGES und OVERFLOW für eine untergeordnete Tabelle nicht erhoben. Für erweiterte Indizes oder deklarierte temporäre Tabellen werden keine statistischen Daten gesammelt.

Die folgenden Tabellen zeigen die Katalogstatistiken, die vom Dienstprogramm RUNSTATS aktualisiert werden:

Tabelle 9. Tabellenstatistiken (SYSCAT.TABLES und SYSSTAT.TABLES)

Statistik	**Beschreibung**	**RUNSTATS-Option**	
		Tabelle	**Indizes**
FPAGES	Anzahl der von einer Tabelle verwendeten Seiten	Ja	Ja
NPAGES	Anzahl der Zeilen enthaltenden Seiten	Ja	Ja
OVERFLOW	Anzahl der Überlaufzeilen	Ja	Nein
CARD	Anzahl der Zeilen in der Tabelle (Kardinalität)	Ja	Ja (Anm. 2)

Anmerkung:
1. In einer partitionierten Datenbank werden Schätzwerte für jede Statistik aus dem Produkt des Werts der Zählung in der Datenbankpartition multipliziert mit der Anzahl der Datenbankpartitionen gebildet.
2. Wenn für die Tabelle keine Indizes erstellt wurden und Sie die Indexstatistik anfordern, wird die CARD-Statistik mit keinem neuen Wert aktualisiert. Die vorige CARD-Statistik wird beibehalten.

Tabelle 10. Spaltenstatistiken (SYSCAT.COLUMNS und SYSSTAT.COLUMNS)

Statistik	**Beschreibung**	**RUNSTATS-Option**	
		Tabelle	**Indizes**
COLCARD	Anzahl der unterschiedlichen Werte der Spalte (Spaltenkardinalität)	Ja (Anm. 1)	Ja (Anm. 2)
AVGCOLLEN	Durchschnittslänge der Spalte	Ja	Ja (Anm. 2)
HIGH2KEY	Zweithöchster Wert der Spalte	Ja	Ja (Anm. 2)
LOW2KEY	Zweitniedrigster Wert der Spalte	Ja	Ja (Anm. 2)
NUMNULLS	Die Anzahl NULLs in einer Spalte	Ja	Ja (Anm. 2)

Anmerkung:
1. COLCARD wird für alle Spalten der Tabelle geschätzt. Wenn in einer partitionierten Datenbank die Spalte ein einspaltiger Partitionierungsschlüssel für die Tabelle ist, wird der Wert als das Produkt aus dem Wert in der Datenbankpartition multipliziert mit der Anzahl der Datenbankpartitionen geschätzt.
2. Spaltenstatistiken werden für die erste Spalte im Indexschlüssel gesammelt.

Tabelle 11. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
NLEAF	Anzahl der Indexblattseiten (leaf pages)	Nein	Ja (Anm. 3)
NLEVELS	Anzahl der Indexstufen	Nein	Ja
CLUSTERRATIO	Grad der Clusterbildung der Tabellendaten	Nein	Ja (Anm. 2)
CLUSTERFACTOR	Feinerer Grad der Clusterbildung	Nein	Detailliert (Anm. 1,2)
DENSITY	Verhältnis (Prozentsatz) von SEQUENTIAL_PAGES zur Anzahl der Seiten im Bereich der vom Index belegten Seiten (Anm. 4)	Nein	Ja
FIRSTKEYCARD	Anzahl der unterschiedlichen Werte in der ersten Spalte des Index	Nein	Ja (Anm. 3)
FIRST2KEYCARD	Anzahl der unterschiedlichen Werte in den ersten beiden Spalten des Index	Nein	Ja (Anm. 3)
FIRST3KEYCARD	Anzahl der unterschiedlichen Werte in den ersten drei Spalten des Index	Nein	Ja (Anm. 3)
FIRST4KEYCARD	Anzahl der unterschiedlichen Werte in den ersten vier Spalten des Index	Nein	Ja (Anm. 3)
FULLKEYCARD	Anzahl der unterschiedlichen Werte in allen Spalten des Index	Nein	Ja (Anm. 3)
PAGE_FETCH_PAIRS	Geschätzte Anzahl der Seitenabrufe für verschiedene Puffergrößen	Nein	Detailliert (Anm. 1,2)

Tabelle 11. Indexstatistiken (SYSCAT.INDEXES und SYSSTAT.INDEXES) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
SEQUENTIAL_PAGES	Anzahl der Blattseiten (äußersten Seiten), die auf der Platte in der durch den Indexschlüssel definierten Reihenfolge mit wenigen oder keinen großen zwischenliegenden Lücken gespeichert sind	Nein	Ja

Anmerkung:

1. Detaillierte Indexstatistikdaten werden gesammelt, indem die Klausel DETAILED im Befehl RUNSTATS oder beim Aufruf der API RUNSTATS für den Parameter `statsopt` die Werte `A`, `Y` oder `X` angegeben werden.
2. CLUSTERFACTOR und PAGE_FETCH_PAIRS werden mit der Klausel DETAILED nur dann gesammelt, wenn die Tabelle eine beträchtliche Größe aufweist. Wenn die Tabelle größer als ca. 25 Seiten ist, werden die Statistikdaten für die Spalten CLUSTERFACTOR und PAGE_FETCH_PAIRS gesammelt. In diesem Fall ist CLUSTERRATIO (Clusterverhältnis) -1 (d. h. wird nicht gesammelt). Wenn die Tabelle relativ klein ist, wird nur die Spalte CLUSTERRATIO vom Dienstprogramm RUNSTATS ausgefüllt, während die Spalten CLUSTERFACTOR und PAGE_FETCH_PAIRS nicht berechnet werden. Wenn die Klausel DETAILED nicht angegeben wird, werden nur die Statistikdaten für CLUSTERRATIO gesammelt.
3. In einer partitionierten Datenbank wird der Wert als Produkt des Werts in der Datenbankpartition multipliziert mit der Anzahl der Datenbankpartitionen geschätzt.
4. Diese Statistik ermittelt den Prozentsatz von Seiten in der Datei, die den Index enthalten, der zu dieser Tabelle gehört. Für eine Tabelle, die nur einen definierten Index hat, sollte DENSITY normalerweise gleich 100 sein. DENSITY wird vom Optimierungsprogramm dazu verwendet, zu schätzen, wie viele irrelevante Seiten von anderen Indizes durchschnittlich vielleicht gelesen werden, wenn die Indexseiten vorabgelesen würden.

Tabelle 12. Spaltenverteilungsstatistiken (SYSCAT.COLDIST und SYSSTAT.COLDIST)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
DISTCOUNT	Wenn TYPE den Wert Q hat, die Anzahl unterschiedlicher Werte, die kleiner oder gleich dem Statistikwert COLVALUE sind.	Verteilung (Anm. 2)	Nein

Tabelle 12. Spaltenverteilungsstatistiken (SYSCAT.COLDIST und SYSSTAT.COLDIST) (Forts.)

Statistik	Beschreibung	RUNSTATS-Option	
		Tabelle	**Indizes**
TYPE	Gibt an, ob die Zeile statistische Daten über die Häufigkeit der Werte oder über Quantilwerte enthält	Verteilung	Nein
SEQNO	Die Stelle in der Häufigkeitsrangfolge einer Folgenummer, die als Hilfe zur eindeutigen Bestimmung der Zeile in der Tabelle verwendet werden kann	Verteilung	Nein
COLVALUE	Datenwert, für den Statistikdaten zur Häufigkeit oder zu Quantilwerten gesammelt werden	Verteilung	Nein
VALCOUNT	Häufigkeit, mit der der Datenwert in der Spalte auftritt, oder bei Quantilwerten die Anzahl der Werte, die kleiner oder gleich dem Datenwert (COLVALUE) sind	Verteilung	Nein

Anmerkung:

1. Verteilungsstatistikdaten über Spalten werden gesammelt, indem die Klausel WITH DISTRIBUTION im Befehl RUNSTATS oder beim Aufruf der API RUNSTATS für den Parameter `statsopt` die Werte `A`, `D` oder `Y` angegeben werden. Beachten Sie, dass Verteilungsstatistikdaten **nicht** gesammelt werden, wenn das Ausmaß der Ungleichmäßigkeit der Werteverteilung in den Spaltenwerten nicht hoch genug ist.
2. DISTCOUNT wird nur für Spalten gesammelt, die die erste Schlüsselspalte in einem Index bilden.
3. In einer partitionierten Datenbank wird der Wert für VALCOUNT als Produkt aus der Anzahl in der Datenbankpartition multipliziert mit der Anzahl der Datenbankpartitionen geschätzt. Hiervon ausgenommen ist der Fall, dass TYPE den Wert 'F' hat und die Spalte den einspaltigen Partitionierungsschlüssel der Tabelle bildet. In diesem Fall ist VALCOUNT einfach der in die Datenbankpartition ermittelte Anzahl.

Weitere Informationen über Verteilungsstatistiken zu Spalten lesen Sie im Abschnitt „Erfassen und Verwenden von Verteilungsstatistiken" auf Seite 140.

Statistische Daten für benutzerdefinierte Funktionen werden vom Dienstprogramm RUNSTATS nicht erfasst. Für diese Funktionen müssen die Statistiken manuell aktualisiert werden. Lesen Sie dazu die Abschnitte „Benutzeraktualisierbare Katalogstatistiken" auf Seite 154 und „Aktualisieren der Statistiken für benutzerdefinierte Funktionen" auf Seite 162.

Erfassen und Verwenden von Verteilungsstatistiken

Der Datenbankmanager kann zwei Arten statistischer Daten, „Häufigkeitswerte" und „Quantile", sammeln, verwalten und verwenden, die in effizienter Weise Schätzwerte über die Verteilung von Datenwerten in einer Spalte bereitstellen. Die Verwendung dieser Statistiken durch das Optimierungsprogramm kann zu bedeutend genaueren Schätzungen über die Anzahl von Zeilen in einer Spalte führen, die ein bestimmtes Gleichheits- oder Bereichsvergleichselement erfüllen. Diese genaueren Schätzwerte wiederum erhöhen die Wahrscheinlichkeit, dass das Optimierungsprogramm den optimalen Zugriffsplan auswählt.

Die statistischen Daten über die Verteilung der Datenwerte können durch die Angabe der Klausel WITH DISTRIBUTION im Befehl RUNSTATS gesammelt werden. Zwar entsteht durch die Erfassung dieser zusätzlichen Statistikdaten ein höherer Systemaufwand bei der Ausführung des Dienstprogramms RUNSTATS, jedoch kann der SQL-Compiler diese Informationen bei der Auswahl des besten Zugriffsplans zur Unterstützung heranziehen.

In einigen Fällen sammelt der Datenbankmanager keine Verteilungsdaten und gibt keinen Fehler aus. Zum Beispiel:

- Die Konfigurationsparameter *num_freqvalues* und *num_quantiles* sind auf Null (0) gesetzt, um anzugeben, dass keine Verteilungsstatistiken erfasst werden sollen. Weitere Informationen zu diesen Parametern finden Sie in folgenden Abschnitten:
  - „Wie viele Statistikdaten sollten erfasst werden?" auf Seite 144
  - „Anzahl der häufigsten Werte (num_freqvalues)" auf Seite 518
  - „Anzahl der Quantile für Spalten (num_quantiles)" auf Seite 519.
- Die Verteilung der Daten ist auch ohne die Verwendung der Verteilungsstatistiken bekannt. Zum Beispiel für eine Spalte, in der kein Datenwert mehr als einmal vorkommt, d. h. wenn alle Datenwerte der Spalte eindeutig sind.
- Es handelt sich um einen Datentyp, für den keine statistischen Daten gesammelt werden. Das heißt, für die Spalte ist ein Datentyp LONG (Langfeld) oder LOB (Large Object) definiert.
- Es gibt nur einen Nichtnullwert in der Spalte und es werden Quantildaten gesammelt.

Die Verteilungsstatistik ist für die erste Spalte von Indizes exakt. Für jede weitere Spalte verwendet der Datenbankmanager Hash-Verfahren und

Stichprobentechniken, um die Verteilungsstatistiken zu schätzen, da zur Berechnung exakter Statistiken zu viel Zeit und Speicher, als zu praktischen Zwecken geeignet sind, erforderlich wären. Diese Techniken sind gängige Methoden der Statistik, die akzeptierte Grade an Genauigkeit liefern.

Verteilungsstatistiken können durch Aktualisieren von SYSSTAT.COLDIST und Festlegen aller COLVALUE- und VALCOUNT-Werte auf 0 oder -1 für die Spalten entfernt werden, für die keine Verteilungsstatistiken mehr benötigt werden.

Die folgenden Themen enthalten Informationen, die Ihnen einen tieferen Einblick in die Verteilungsstatistiken und ihre Verwendung geben sollen:
- Beschreibung der Verteilungsstatistiken.
- Wann sollten Verteilungsstatistiken verwendet werden?
- Wie viele Statistikdaten sollten erfasst werden?
- Wie verwendet das Optimierungsprogramm die Verteilungsstatistiken?
- Modellieren im Einsatz befindlicher Datenbanken.
- Regeln zur Aktualisierung von Verteilungsstatistiken für Spalten.

Beschreibung der Verteilungsstatistiken

Für eine Konstante N>=1 bestehen die *N häufigsten Werte* in einer Spalte aus dem Datenwert, der am häufigsten vorkommt (d. h., der die meisten Duplikate hat), dem Datenwert, der am zweithäufigsten vorkommt, usw. bis zu dem Datenwert, der am N-thäufigsten vorkommt. Die entsprechende *Häufigkeitsstatistik* besteht aus diesen „N" Datenwerten und den zugehörigen Werten für die Anzahl der Vorkommen (Häufigkeit) in der Spalte.

Das *K-Quantil* für eine Spalte ist der kleinste Datenwert `V`, so dass mindestens „K" Zeilen Datenwerte enthalten, die kleiner oder gleich `V` sind. Ein K-Quantil kann berechnet werden, indem die Zeilen in der Spalte nach aufsteigenden Datenwerten sortiert werden. Das K-Quantil ist der Datenwert in der K-ten Zeile der sortierten Spalte.

Betrachten Sie zum Beispiel die folgende Spalte von Daten:

```
C1
--
 B
 E
 Y
 B
 F
 G
 E
 A
 J
 K
 E
 L
```

Diese Spalte kann sortiert werden, so dass sich die folgenden geordneten Werte ergeben:

```
C1'
--
 A
 B
 B
 E
 E
 E
 F
 G
 J
 K
 L
 Y
```

Es gibt neun unterschiedliche Datenwerte in Spalte C1. Für N = 2 ergibt sich folgende Häufigkeitsstatistik:

```
SEQNO    COLVALUE     VALCOUNT
-----    ---------    --------
  1          E            3
  2          B            2
```

Wenn die Anzahl der Quantile, die gesammelt werden, gleich 5 ist (siehe „Anzahl der Quantile für Spalten (num_quantiles)" auf Seite 519), dann ergeben sich folgende K-Quantile für diese Spalten mit K = 1, 3, 6, 9 und 12:

```
SEQNO    COLVALUE     VALCOUNT
-----    ---------    --------
  1          A            1
  2          B            3
  3          E            6
  4          J            9
  5          Y           12
```

In diesem Beispiel ist das 6-Quantil gleich E, da die sechste Zeile in der sortierten Spalte einen Datenwert gleich E aufweist (und 6 Zeilen in der Originalspalte Datenwerte kleiner oder gleich E enthalten).

Derselbe Quantilwert kann mehrmals auftreten, wenn es sich um einen gängigen Wert handelt. Für einen bestimmten Wert werden maximal zwei Quantile gespeichert. Das erste dieser beiden Quantile hat einen VALCOUNT-Wert, der die Anzahl der Zeilen mit einem Wert angibt, der streng kleiner ist als der Wert COLVALUE; das zweite der beiden Quantile gibt die Anzahl der Zeilen an, die einen Wert kleiner oder gleich dem Wert COLVALUE enthalten.

Wann sollten Verteilungsstatistiken verwendet werden?

Bei der Entscheidung, ob Verteilungsstatistikdaten für eine bestimmte Tabelle angelegt werden sollten, spielen zwei Faktoren eine ausschlaggebende Rolle:

1. Die Verwendung statischen oder dynamischen SQLs

   Verteilungsstatistiken sind am besten für dynamisches SQL sowie für SQL, das keine Host-Variablen verwendet, geeignet. Wenn SQL mit Host-Variablen verwendet wird, nutzt das Optimierungsprogramm die Verteilungsstatistikdaten nur in begrenztem Umfang.

2. Ungleichmäßigkeit der Datenverteilungen

   Verteilungsstatistische Daten sind besonders dann hilfreich, wenn mindestens in einer Spalte der Tabelle die Datenwerte höchst *ungleichmäßig* verteilt sind und diese Spalte häufig in Gleichheits- bzw. Bereichsvergleichselementen auftritt, wie zum Beispiel in folgenden Klauseln:

   ```
   WHERE C1 = KEY;
   WHERE C1 IN (KEY1, KEY2, KEY3);
   WHERE (C1 = KEY1) OR (C1 = KEY2) OR (C1 = KEY3);
   WHERE C1 <= KEY;
   WHERE C1 BETWEEN KEY1 AND KEY2;
   ```

   Es gibt zwei Arten der Ungleichmäßigkeit bei einer Datenverteilung, die auch nebeneinander auftreten können:

   - Eine Art der Ungleichmäßigkeit entsteht dann, wenn die Daten nicht gleichmäßig zwischen dem höchsten und dem niedrigsten Datenwert verteilt sind, sondern sich in einem Unterwertebereich Wertecluster bilden, wie in der folgenden Spalte illustriert wird, in der Daten im Bereich (5,10) Clusterbildung zeigen:

     ```
       C1
     -----
      0,0
      5,1
      6,3
      7,1
      8,2
      8,4
      8,5
      9,1
     93,6
     100,0
     ```

     Wenn diese Art von Ungleichmäßigkeit vorliegt, kann es nützlich sein, über Quantilstatistiken zu verfügen.

Das folgende Beispiel zeigt eine Abfrage, die bei der Bestimmung, ob ein hoher Grad an Ungleichmäßigkeit in einer Spalte vorhanden ist, hilfreich sein kann.

```
SELECT C1, COUNT(*) AS OCCURRENCES
  FROM T1
GROUP BY C1
ORDER BY OCCURRENCES DESC;
```

- Eine andere Art der Ungleichmäßigkeit tritt auf, wenn bestimmte Datenwerte eine sehr viel höhere Häufigkeit aufweisen als andere Datenwerte, wie anhand der Spalte mit folgenden Datenwerten und Häufigkeiten zu sehen ist:

  ```
  Datenwert    Häufigkeit
  ----------   ----------
     20            5
     30           10
     40           10
     50           25
     60           25
     70           20
     80            5
  ```

  Wenn diese Art der Ungleichmäßigkeit vorliegt, kann es nützlich sein, sowohl über Quantil- als auch über Häufigkeitsstatistikdaten zu verfügen.

Verteilungsstatistikdaten können mit Hilfe der Klausel WITH DISTRIBUTION im Befehl RUNSTATS oder beim Aufruf der API RUNSTATS durch die Angabe der Werte `D`, `E` oder `A` für den Parameter `statsopt` gesammelt werden. Weitere Informationen zur Anwendungsprogrammierschnittstelle (Application Programming Interface, API) finden Sie im Handbuch *Administrative API Reference*.

Wie viele Statistikdaten sollten erfasst werden?

Wenn eine große Anzahl statistischer Daten über die Werteverteilung in Spalten vorliegt, kann einerseits das Optimierungsprogramm einen besseren Zugriffsplan auswählen, andererseits wächst der Systemaufwand für das Sammeln dieser Statistikdaten und für die Kompilierung der Abfragen entsprechend. Die Größe des Zwischenspeichers für Statistik (siehe „Größe des Statistikzwischenspeichers (stat_heap_sz)" auf Seite 427) kann der Anzahl der statistischen Daten, die berechnet und gespeichert werden können, Grenzen setzen.

Wenn Verteilungsstatistikdaten angefordert werden, speichert der Datenbankmanager standardmäßig die 10 häufigsten Werte für eine Spalte. Eine Anzahl zwischen 10 und 100 häufigsten Werten sollte für die Mehrzahl der praktischen Anwendungen genügen. Im Idealfall sollten ausreichend Häufigkeitsstatistikdaten gespeichert werden, so dass die Häufigkeiten der verbleibenden Werte entweder einander annähernd gleich sind oder im Vergleich zu den Häufigkeiten der häufigsten Werte vernachlässigbar sind.

Zur Festlegung der Anzahl der zu sammelnden Häufigkeitswerte wird der Konfigurationsparameter *num_freqvalues* verwendet, der im Abschnitt „Anzahl der häufigsten Werte (num_freqvalues)" auf Seite 518 beschrieben ist. Der Datenbankmanager sammelt unter Umständen weniger als diese Anzahl von Statistikdaten über die Häufigkeit, da diese Statistikdaten nur für Datenwerte erhoben werden, die mehr als einmal auftreten. Wenn nur Quantilstatistikdaten gesammelt werden, kann dieser Parameter auf den Wert 0 gesetzt werden.

Wenn Verteilungsstatistikdaten angefordert werden, speichert der Datenbankmanager standardmäßig 20 Quantile für eine Spalte. Dieser Wert garantiert einen maximalen Schätzfehler von ungefähr 2,5% für alle einfachen einseitigen Bereichsvergleichselemente (>, >=, < oder <=), und einen maximalen Fehler von 5% für jedes Vergleichselement mit BETWEEN. Als grobe Faustregel zur Bestimmung der Anzahl von Quantilen gilt:

- Bestimmen Sie den maximalen Fehler, der bei der Schätzung der Anzahl von Zeilen jeder Bereichsabfrage tolerierbar ist, als Prozentwert P.
- Die Anzahl der Quantile sollte ca. 100/P sein, wenn es sich um ein BETWEEN-Vergleichselement handelt, und 50/P, wenn das Vergleichselement eine andere Art von Bereichsvergleichselement (<, <=, > oder >=) ist.

Zum Beispiel ergeben 25 Quantile einen maximalen Schätzfehler von 4% bei BETWEEN-Vergleichselementen und 2% bei Vergleichselementen mit ″>″. Im allgemeinen sollten mindestens 10 Quantile gespeichert werden. Mehr als 50 Quantile sind nur bei extrem ungleichmäßig verteilten Daten erforderlich.

Zur Festlegung der Anzahl der Quantile wird der Konfigurationsparameter *num_quantiles* verwendet, der im Abschnitt „Anzahl der Quantile für Spalten (num_quantiles)" auf Seite 519 beschrieben ist. Wenn nur Statistikdaten über die Häufigkeit von Werten gesammelt werden, kann dieser Parameter auf den Wert 0 gesetzt werden. Wird dieser Parameter auf den Wert „1" gesetzt, werden ebenfalls keine Quantilstatistikdaten gesammelt, da der gesamte Bereich von Werten in einem Quantil erfasst würde.

Wie verwendet das Optimierungsprogramm die Verteilungsstatistiken?

Wozu werden statistische Daten zur Werteverteilung gesammelt und gespeichert? Die Antwort liegt in der Tatsache begründet, dass ein Optimierungsprogramm die Anzahl von Zeilen in einer Spalte schätzen muss, die ein Gleichheitsvergleichselement oder ein Bereichsvergleichselement erfüllen, um den Zugriffsplan wählen zu können, der den geringsten Systemaufwand verursacht. Je genauer die Schätzung ist, desto größer ist auch die Wahrscheinlichkeit, dass das Optimierungsprogramm den optimalen Zugriffsplan wählt. Betrachten Sie zum Beispiel die folgende Abfrage:

```
SELECT C1, C2
     FROM TABLE1
    WHERE C1 = 'NEW YORK'
      AND C2 <= 10
```

Nehmen Sie an, dass es einen Index für C1 und einen Index für C2 gibt. Ein möglicher Zugriffsplan besteht darin, über den Index für C1 alle Zeilen mit `C1 = 'NEW YORK'` abzurufen und anschließend jede abgerufene Zeile daraufhin zu überprüfen, ob `C2 <= 10` gilt. Ein alternativer Plan wäre, über den Index für C2 alle Zeilen mit `C2 <= 10` abzurufen und anschließend jede abgerufene Zeile daraufhin zu überprüfen, ob `C1 = 'NEW YORK'` gilt. In der Regel entsteht der Hauptaufwand bei der Ausführung der obigen Abfrage durch das Abrufen der Zeilen, so dass es wünschenswert ist, den Plan auszuwählen, der die geringste Anzahl von Zeilenabrufen erforderlich macht. Zur Auswahl des besten Plans ist es also nötig, die Anzahl der Zeilen, die jedes Vergleichselement erfüllen, im Voraus abzuschätzen.

Wenn Verteilungsstatistikdaten nicht angefordert werden, arbeitet das Optimierungsprogramm nur mit dem zweithöchsten Datenwert (HIGH2KEY), dem zweitniedrigsten Datenwert (LOW2KEY), der Anzahl unterschiedlicher Werte (COLCARD) und der Anzahl der Zeilen für eine Spalte. Die Anzahl der Zeilen, die ein Gleichheitsvergleichselement oder ein Bereichsvergleichselement erfüllen, wird dann unter der Annahme abgeschätzt, dass die Häufigkeiten der Datenwerte in einer Spalte alle gleich und die Datenwerte gleichmäßig über das Intervall (LOW2KEY, HIGH2KEY) verteilt sind. Im Einzelnen wird die Anzahl der Zeilen, die ein Gleichheitsvergleichselement `C1 = KEY` erfüllen, mit dem Wert CARD/COLCARD abgeschätzt. Die Anzahl der Zeilen, die ein Bereichsvergleichselement `C1 BETWEEN KEY1 AND KEY2` erfüllen, wird nach folgender Formel abgeschätzt:

```
      KEY2 - KEY1
------------------  x CARD       (1)
 HIGH2KEY - LOW2KEY
```

Diese Schätzwerte sind nur dann realistisch, wenn die tatsächliche Verteilung der Datenwerte weitgehend gleichmäßig ist. Wenn keine Verteilungsstatistikdaten verfügbar sind und entweder die Häufigkeiten der Datenwerte grob von einander abweichen oder die Datenwerte in einigen wenigen Unterbereichen des Intervalls (LOW_KEY, HIGH_KEY) Cluster bilden, können die Schätzwerte um Größenordnungen von der Realität abweichen, so dass das Optimierungsprogramm möglicherweise einen nicht optimalen Zugriffsplan wählt.

Wenn Verteilungsstatistikdaten verfügbar sind, können die oben beschriebenen Schätzfehler wesentlich verringert werden, indem die statistischen Daten zur Häufigkeit der Werte zur Berechnung der Anzahl von Zeilen, die ein Gleichheitsvergleichselement erfüllen, und sowohl die statistischen Daten zur Häufigkeit der Werte als auch die Quantilstatistik zur Berechnung der Anzahl von Zeilen, die ein Bereichsvergleichselement erfüllen, herangezogen werden.

Beispiel zur Auswirkung auf Gleichheitsvergleichselemente:

Betrachten Sie zunächst ein Vergleichselement der Form `C1 = KEY`. Wenn `KEY` einer der N häufigsten Werte ist, dann verwendet das Optimierungsprogramm einfach die Häufigkeit von `KEY`, die im Katalog gespeichert ist. Wenn `KEY` nicht einer der N häufigsten Werte ist, schätzt das Optimierungsprogramm die Anzahl der Zeilen, die das Vergleichselement erfüllen, unter der Annahme ab, dass die nicht häufigen Werte (COLCARD - N) eine gleichmäßige Verteilung aufweisen. Das heißt, die Anzahl der Zeilen wird folgendermaßen abgeschätzt:

```
CARD - NUM_FREQ_ROWS
--------------------          (2)
   COLCARD - N
```

Hier steht NUM_FREQ_ROWS für die Gesamtzahl der Zeilen mit einem Wert, der gleich einem der N häufigsten Werte ist.

Betrachten Sie zum Beispiel eine Spalte (`C`), in der sich die Häufigkeit der Datenwerte folgendermaßen darstellt:

Datenwert	Häufigkeit
1	2
2	3
3	40
4	4
5	1

Angenommen, es stehen die statistischen Häufigkeitsdaten nur für den häufigsten Wert (d. h. N = 1) zur Verfügung. Für diese Spalte gilt CARD = 50 und COLCARD = 5. Das Vergleichselement `C = 3` wird exakt von 40 Zeilen erfüllt. Unter der Annahme einer gleichmäßigen Datenverteilung wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, als 50/5 = 10 geschätzt, was einen Schätzfehler von -75% mit sich bringt. Bei Verwendung der Häufigkeitsstatistik kann die Anzahl der Zeilen auf 40 geschätzt werden, d. h., es entsteht in diesem Fall kein Fehler.

Analog erfüllen 2 Zeilen das Vergleichselement `C = 1`. Ohne die Häufigkeitsstatistik wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, auf 10 geschätzt; mithin entsteht ein Fehler von 400%. Die folgende Formel kann zur Berechnung des Schätzfehlers (in Prozent) herangezogen werden:

```
geschätzte Zeilen - tatsächliche Zeilen
---------------------------------------  X 100
          tatsächliche Zeilen
```

Bei Verwendung der Häufigkeitsstatistik (N = 1) schätzt das Optimierungsprogramm die Anzahl der Zeilen, die diesen Wert enthalten, anhand der oben gezeigten Formel (2). Zum Beispiel:

```
(50 - 40)
--------- = 3
 (5 - 1)
```

Der Fehler wird dabei um eine Größenordnung verringert, wie im Folgenden gezeigt wird

```
 3 - 2
------- = 50%
   2
```

Die Anzahl der Zeilen, die ein Bereichsvergleichselement erfüllen, kann mit Hilfe von Quantilen, wie in folgenden Beispielen illustriert, abgeschätzt werden. Betrachten Sie eine Spalte (`C`), die folgende Werte enthält:

```
   C
-------
   0,0
   5,1
   6,3
   7,1
   8,2
   8,4
   8,5
   9,1
  93,6
 100,0
```

Angenommen, es stehen K-Quantile zur Verfügung für K = 1, 4, 7 und 10:

```
 K     K-Quantil
---  ----------
 1      0,0
 4      7,1
 7      8,5
10    100,0
```

Betrachten Sie zunächst das Vergleichselement `C <= 8,5`. Für die oben angegebenen Daten erfüllen exakt sieben Zeilen dieses Vergleichselement. Unter der Annahme einer gleichmäßigen Datenverteilung und unter Verwendung der oben angegeben Formel (1), wobei KEY1 durch LOW2KEY ersetzt wird, wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, folgendermaßen abgeschätzt:

```
 8,5 - 5,1
---------- x 10 *= 0
93,6 - 5,1
```

Die Notation `*=` bedeutet „annähernd gleich". Der Fehler bei dieser Schätzung ist annähernd -100%.

Bei Verwendung von Quantilen wird die Anzahl der Zeilen, die dasselbe Vergleichselement (`C <= 8,5`) erfüllen, abgeschätzt, indem 8,5 als einer der K-Quantilwerte aufgesucht und der entsprechende Wert von K, also 7, als Schätzwert verwendet wird. In diesem Fall wird der Fehler auf 0 reduziert.

Betrachten Sie nun das Vergleichselement `C <= 10`. Dieses Vergleichselement wird von exakt acht Zeilen erfüllt. Anders als im vorigen Beispiel ist der Wert 10 keiner der gespeicherten K-Quantile. Unter der Annahme einer gleichmäßigen Datenverteilung und unter Verwendung der Formel (1) wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, auf 1 geschätzt, d. h., es entsteht ein Fehler von -87,5 %.

Bei Verwendung von Quantilen schätzt das Optimierungsprogramm die Anzahl der Zeilen, die das Vergleichselement erfüllen, auf `r_1 + r_2`, wobei `r_1` die Anzahl der Zeilen ist, die das Vergleichselement `C <= 8,5` erfüllen, und `r_2` die Anzahl der Zeilen ist, die das Vergleichselement `C > 8,5 AND C <= 10,0` erfüllen. Wie im vorigen Beispiel gilt `r_1 = 7`.

Zur Abschätzung von r_2 verwendet das Optimierungsprogramm die lineare Interpolation:

```
       10 - 8,5
r_2 *= ---------- x (Anzahl Zeilen mit Wert > 8,5 und <= 100,0)
       100 - 8,5
       10 - 8,5
r_2 *= ---------- x (10 - 7)
       100 - 8,5
       1,5
r_2 *= ---- x (3)
       91,5
r_2 *= 0
```

Die abschließende Schätzung ist r_1 + r_2 *= 7, und der Fehler beträgt nur -12,5 %.

Der Grund dafür, dass die Verwendung von Quantilen die Genauigkeit der Schätzungen in den obigen Beispielen erhöht, liegt darin, dass die realen Datenwerte „Cluster" im Bereich von 5 bis 10 bilden, aber die Standardformeln zur Schätzung von einer gleichmäßigen Verteilung der Werte zwischen 0 und 100 ausgehen. Die Verwendung von Quantilen erhöht auch die Genauigkeit, wenn es wesentliche Unterschiede in den Häufigkeiten verschiedener Datenwerte gibt. Betrachten Sie eine Spalte, die Datenwerte mit den folgenden Häufigkeiten enthält:

Datenwert	Häufigkeit
20	5
30	5
40	15
50	50
60	15
70	5
80	5

Angenommen, es stehen K-Quantile zur Verfügung für K = 5, 25, 75, 95 und 100:

K	K-Quantil
5	20
25	40
75	50
95	70
100	80

Nehmen Sie außerdem an, dass statistische Häufigkeitsdaten für die 3 häufigsten Werte verfügbar sind.

Betrachten Sie das Vergleichselement `C BETWEEN 20 AND 30`. An der Verteilung der Datenwerte können Sie sehen, dass genau 10 Zeilen das Vergleichselement erfüllen. Unter der Annahme einer gleichmäßigen Datenverteilungen und unter Verwendung der Formel (1), wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, wie folgt abgeschätzt:

```
30 - 20
-------  x 100 = 25
70 - 30
```

Diese Abschätzung enthält einen Fehler von 150%.

Unter Verwendung der Häufigkeitsstatistik und der Quantile wird die Anzahl der Zeilen, die das Vergleichselement erfüllen, als `r_1 + r_2` abgeschätzt, wobei `r_1` die Anzahl der Zeilen ist, die das Vergleichselement `(C = 20)` erfüllen, und `r_2` die Anzahl der Zeilen ist, die das Vergleichselement `C > 20 AND C <= 30` erfüllen. Bei Verwendung der Formel (2) wird `r_1` folgendermaßen abgeschätzt:

```
100 - 80
-------- = 5
  7 - 3
```

`r_2` wird mit linearer Interpolation folgendermaßen abgeschätzt:

```
  30 - 20
  ------- x (Anzahl Zeilen mit Wert > 20 und <= 40)
  40 - 20
  30 - 20
= ------- x (25 - 5)
  40 - 20
= 10,
```

Als endgültiger Schätzwert ergibt sich 15, wodurch der Schätzfehler um den Faktor 3 verringert wird.

Erfassen und Verwenden detaillierter Indexstatistikdaten

Sie haben die Möglichkeit, detailliertere Statistikdaten für Indizes zu erfassen, die dem Optimierungsprogramm zu einer besseren Abschätzung des Aufwands für den Zugriff auf eine Tabelle über diesen Index verhelfen. Dazu gibt es zwei Methoden: zum einen können Sie die Klausel DETAILED im Befehl RUNSTATS verwenden, zum anderen können Sie A, Y oder X für den Parameter `statsopt` beim Aufrufen der API RUNSTATS angeben.

Die DETAILED-Statistikdaten für PAGE_FETCH_PAIRS und CLUSTERFACTOR werden nur erfasst, wenn die Tabelle eine ausreichende Größe hat (ca. 25 Seiten). In diesem Fall hat CLUSTERFACTOR einen Wert zwischen 0 und 1, während CLUSTERRATIO den Wert -1 (nicht erfasst) hat. Für Tabellen, die kleiner als 25 Seiten sind, hat CLUSTERFACTOR den Wert -1 (nicht erfasst) und CLUSTERRATIO einen Wert zwischen 0 und 100, auch wenn die Klausel DETAILED für einen Index für diese Tabelle angegeben ist.

Zweck detaillierter Indexstatistikdaten

Mit Hilfe der Klausel DETAILED soll in komprimierter Form die Anzahl der physischen E/A-Operationen erfasst werden, die für den Zugriff auf die Datenseiten einer Tabelle erforderlich werden, wenn eine vollständige Indexsuche bei verschiedenen Puffergrößen ausgeführt wird. Das Dienstprogramm RUNSTATS sucht die Seiten des Index ab und modelliert dabei die verschiedenen Puffergrößen und sammelt Schätzwerte darüber, wie häufig eine Fehlseitenbedingung auftritt. Wenn beispielsweise nur 1 (eine) Pufferseite verfügbar ist, führt jeder neue Verweis des Index auf eine Seite zu einer Fehlseitenbedingung. Schlimmstenfalls kann jede Zeile des Index auf eine andere Seite verweisen, was maximal eine Anzahl von CARD (Kardinalität der Tabelle) an E/A-Operationen verursachen kann. Das andere Extrem wäre der Fall, wenn der Puffer groß genug wäre, um die gesamte Tabelle (abhängig von der maximalen Puffergröße) aufzunehmen. In diesem Fall werden die Seiten der Tabelle (NPAGES) genau einmal physisch gelesen. Die Anzahl der physischen E/A-Operationen muss infolgedessen eine monotone, nicht steigende Funktion der Puffergröße sein.

Das Dienstprogramm RUNSTATS erstellt eine stückweise an diese Schätzwerte angelehnte lineare Kurve, die als Zeichenfolge von 11 Wertepaaren in der PAGE_FETCH_PAIRS-Statistik gespeichert wird. Der erste Wert in jedem Paar stellt eine hypothetische Puffergröße dar, während der zweite Wert jedes Paares die geschätzte Anzahl der physischen E/A-Operationen zum Abrufen der Datenseiten bei einem vollständigen Durchsuchen des Index unter voller Verfügbarkeit der zugehörigen Puffergröße für diese Indexsuche enthält. Das Optimierungsprogramm verwendet die PAGE_FETCH_PAIRS-Statistik, um die Anzahl der physischen E/A-Operationen für Abrufe von Datenseiten in jeder vollständigen oder partiellen Indexsuche in diesem Index abzuschätzen.

Die Form der Kurve, die in PAGE_FETCH_PAIRS für einen Index gespeichert wird, hängt von der Art der Clusterbildung dieses Index ab.

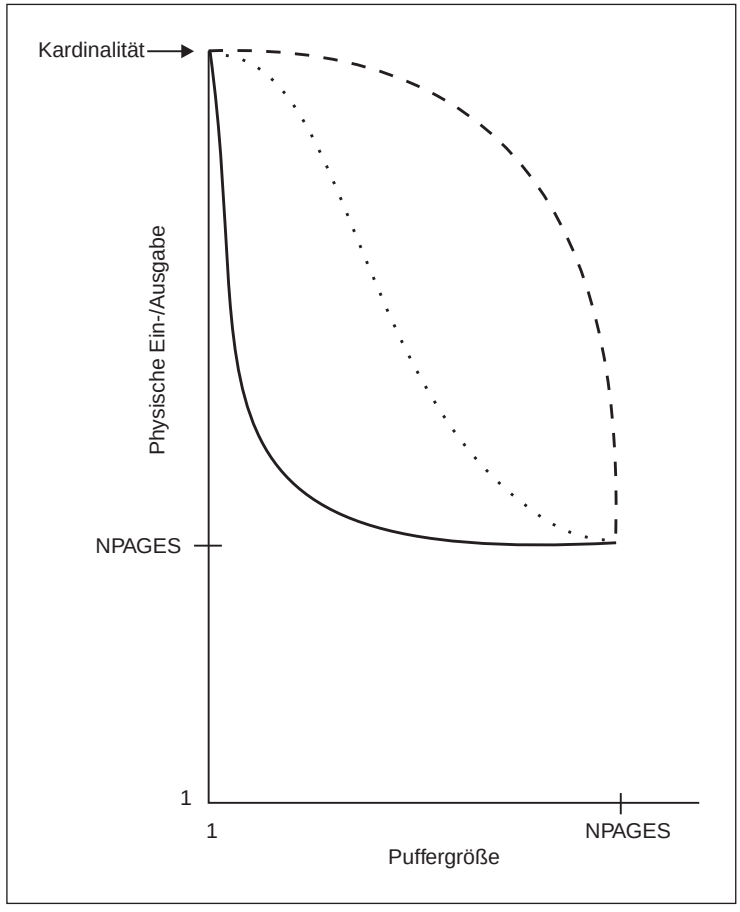


Abbildung 11. Drei Kurven für Indizes mit und ohne Clusterbildung

Es gibt drei Arten möglicher Kurven:

1. Kurve 1 (gestrichelte Linie) zeigt einen Index mit sehr geringer Clusterbildung, der einen Puffer beinahe in der Größe der Tabelle benötigt, bevor Seiten, auf die erneut verwiesen wird, im Puffer gefunden werden. Hierbei handelt es sich um eine Situation, in der Verweise auf dieselbe Seite breit über die Schlüsselwerte des gesamten Index verteilt sind, so dass ein mittelgroßer Puffer nicht ausreicht, um wiederholtes erneutes Verweisen auf dieselbe Seite zu vermeiden. Dies ist der ungünstigste Fall, da hier der größte Pufferspeicher benötigt wird, um eine akzeptable Leistung zu erzielen. Das Optimierungsprogramm verwendet mit hoher Wahrscheinlichkeit die Strategie des Vorablesezugriffs über Listen für solche Indizes, um zu versuchen, die Datenseitenzugriffe für die gewünschten Schlüsselwerte des Index in Clustern zusammenzufassen. Wenn dieser Index häufig verwendet wird, ist für ihn eine Reorganisation sehr vorteilhaft.

2. Kurve 2 (durchgezogene Linie) zeigt einen Index, der lokal weniger Clusterbildung hat. Für sehr kleine Puffer hat er genauso wenig Clusterbildung wie der Index in Kurve 1. Aber sobald einige wenige Pufferseiten verfügbar sind, um die letzten Daten, auf die verwiesen wurde, noch zu enthalten, steigt die Trefferquote für Datenseiten erheblich an. Hierbei handelt es sich um die etwas glückliche Situation, in der, obwohl der Index keine spezielle Clusterbildung aufweist, die Verweise auf dieselben Datenseiten in großer Nähe zueinander in den Schlüsselwerten des Index liegen.
3. Kurve 3 (gepunktete Linie) stellt einen Index dar, der irgendwo zwischen den beiden Extremen liegt und eine gleichmäßige Verbesserung bei steigender Puffergröße zeigt. Dies kommt in der Regel dem Normalfall für Indizes ohne Clusterbildung nahe und stellt die Annahmen des Optimierungsprogramms dar, wenn keine detaillierten Indexdaten (DETAILED) verfügbar sind.

Wann sollten detaillierte Indexstatistiken verwendet werden?

Die detaillierten Indexstatistiken (Klausel DETAILED) sollten verwendet werden, wenn Ihre Abfragen auf Spalten zugreifen, die nicht alle im Index sind. Darüber hinaus sollten detaillierte Indexdaten in folgenden Fällen verwendet werden:

- Wenn es mehrere Indizes ohne Clusterbildung mit verschiedenen Graden von Clusterbildung gibt
- Wenn der Grad der Clusterbildung unter den Schlüsselwerten nicht gleichmäßig ist
- Wenn die Werte im Index ungleichmäßig aktualisiert werden

Unter Umständen ist es nicht einfach, diese Fälle zu identifizieren, ohne im Besitz bestimmter Vorkenntnisse zu sein und ohne zu versuchen, eine Indexsuche bei verschiedenen Puffergrößen zu erzwingen und mit Hilfe des Monitors die sich ergebenden physischen E/A-Operationen zu überwachen. Die Methode des geringsten Aufwands zur Feststellung, ob eine dieser Situationen auftritt, ist, die DETAILED-Statistikdaten für einen Index zu erfassen und sie zu behalten, wenn die resultierenden Wertepaare für PAGE_FETCH_PAIRS einen nichtlinearen Verlauf zeigen.

Benutzeraktualisierbare Katalogstatistiken

Aufgrund der Tatsache, dass ausgewählte Systemkatalogstatistiken aktualisierbar sind, haben Sie folgende Möglichkeiten:

- Sie können die Abfrageleistung auf einem Entwicklungssystem unter Verwendung wirklichkeitsnaher Systemstatistiken eines geschäftlich genutzten Systems nachmodellieren.
- Sie können Fallstudien („Was wäre, wenn?“) für die Abfrageleistung durchführen und die Ergebnisse analysieren.

Auf geschäftlich genutzten Systemen sollten die Statistiken **nicht** aktualisiert werden, da dies dazu führen kann, dass das Optimierungsprogramm nicht den besten Zugriffsplan für eine Abfrage findet.

Zur Aktualisierung der Werte dieser Statistikspalten wird die SQL-Anweisung UPDATE für die Sichten verwendet, die im Schema SYSSTAT definiert sind. Folgende Statistiken können aktualisiert werden:

- Für Tabellen, für die Sie die explizite Berechtigung CONTROL haben. Sie können auch die Statistiken für Spalten und Indizes dieser Tabellen aktualisieren.
- Für Kurznamen, für die Sie die explizite Berechtigung CONTROL in einem System mit zusammengeschlossenen Datenbanken haben. Sie können auch Statistiken für Spalten und Indizes für diese Kurznamen aktualisieren. Beachten Sie, dass die Aktualisierung nur lokale Metadaten betrifft (Tabellenstatistiken der Datenquelle werden nicht geändert). Diese Aktualisierungen betreffen nur die globale Zugriffsstrategie, die vom DB2-Optimierungsprogramm generiert wird.
- Für benutzerdefinierte Funktionen (UDFs), deren Eigner Sie sind (Anleitung siehe „Aktualisieren der Statistiken für benutzerdefinierte Funktionen" auf Seite 162).

Sie können diese Statistiken auch aktualisieren, wenn Ihre Benutzer-ID über eine explizite Berechtigung DBADM für die Datenbank verfügt. Das heißt, wenn Ihre Benutzer-ID in der Tabelle SYSCAT.DBAUTH mit der Berechtigung DBADM eingetragen ist. Durch die Zugehörigkeit zu einer DBADM-Gruppe wird diese Berechtigung nicht explizit erteilt.

Mit Hilfe dieser Sichten kann ein Datenbankadministrator (DBADM) alle Statistikzeilen für alle Benutzer anzeigen. Ein Benutzer ohne die Berechtigung DBADM kann nur die Zeilen anzeigen, die statistische Daten für Objekte enthalten, für die er die Berechtigung CONTROL hat.

Das folgende Beispiel zeigt, wie die Tabellenstatistik für die Tabelle EMPLOYEE aktualisiert werden kann:

```
UPDATE SYSSTAT.TABLES
SET  CARD   = 10000,
     NPAGES = 1000,
     FPAGES = 1000,
     OVERFLOW = 2
WHERE TABSCHEMA = 'userid'
  AND TABNAME   = 'EMPLOYEE'
```

Bei der Aktualisierung der Katalogstatistik ist Vorsicht geboten. Willkürliche Aktualisierungen können ernste Auswirkungen auf die Leistung nachfolgender Abfragen haben. Die folgenden Methoden stehen zur Verfügung, wenn Sie an diesen Tabellen vorgenommene Aktualisierungen ersetzen wollen:

- Rückgängigmachen (ROLLBACK) der Arbeitseinheit, in der Änderungen durchgeführt wurden (unter der Annahme, dass die Arbeitseinheit noch nicht festgeschrieben wurde).
- Ausführen des Dienstprogramms RUNSTATS, um die Katalogstatistiken neu berechnen und aktualisieren zu lassen.
- Aktualisieren der Katalogstatistiken, um anzugeben, dass keine Statistikdaten gesammelt wurden. (Zum Beispiel wird durch den Wert -1 in der Spalte NPAGES angezeigt, dass keine Statistik zur Anzahl von Seiten erfasst wurde.)
- Ersetzen der Katalogstatistiken durch die Daten, die sie vor der Aktualisierung enthielten. Diese Methode ist nur möglich, wenn mit dem Tool *db2look* die Statistiken vor den Änderungen gespeichert wurden, wie es im Abschnitt „Modellieren im Einsatz befindlicher Datenbanken" auf Seite 164 beschrieben wird.

In einigen Fällen kann es vorkommen, dass das Optimierungsprogramm einen bestimmten statistischen Wert oder eine Kombination von Werten als ungültig erkennt, so dass es die entsprechenden Standardwerte verwendet und eine Warnung ausgibt. Situationen dieser Art sind jedoch selten, da der Hauptteil der Gültigkeitsprüfungen bei der Aktualisierung der Statistiken durchgeführt wird.

Zusätzliche Informationen: Informationen zur Aktualisierung von Katalogstatistiken finden Sie in den folgenden Abschnitten:
- „Regeln zur Aktualisierung von Katalogstatistiken"
- „Regeln zur Aktualisierung von Kurznamenstatistiken" auf Seite 157
- „Regeln zur Aktualisierung von Spaltenstatistiken" auf Seite 158
- „Regeln zur Aktualisierung von Verteilungsstatistiken für Spalten" auf Seite 159
- „Regeln zur Aktualisierung von Indexstatistiken" auf Seite 160
- „Aktualisieren der Statistiken für benutzerdefinierte Funktionen" auf Seite 162
- „Modellieren im Einsatz befindlicher Datenbanken" auf Seite 164.

Regeln zur Aktualisierung von Katalogstatistiken

Die wichtigste Regel, die bei einer Aktualisierung der Katalogstatistiken zu beachten ist, ist die Sicherstellung, dass gültige Werte, Wertebereiche und Formate der verschiedenen Statistiken in den Statistiksichten gespeichert werden. Darüber hinaus muss die Konsistenz der Beziehungen zwischen verschiedenen Statistiken gewahrt bleiben.

Zum Beispiel muss der Wert für COLCARD in der Sicht SYSSTAT.COLUMNS kleiner sein als der für CARD in der Sicht SYSSTAT.TABLES (die Anzahl der unterschiedlichen Werte in einer Spalte kann nicht größer sein als die Anzahl der Zeilen). Nehmen Sie an, Sie wollen den Wert für COLCARD von 100 auf 25 und den Wert für CARD von 200 auf 50 verringern. Wenn Sie die Sicht

SYSCAT.TABLES zuerst aktualisieren, sollten Sie eine Fehlernachricht empfangen (da CARD kleiner als COLCARD würde). Die richtige Reihenfolge wäre, zuerst den Wert für COLCARD in der Sicht SYSCAT.COLUMNS und anschließend den Wert für CARD in der Sicht SYSSTAT.TABLES zu aktualisieren. Der Fall stellt sich umgekehrt dar, wenn Sie den Wert von COLCARD von 100 auf 250 und den Wert für CARD von 200 auf 300 erhöhen wollen. In diesem Fall müssten Sie zuerst den Wert CARD und anschließend den Wert COLCARD aktualisieren.

Wenn ein Konflikt zwischen einer aktualisierten Statistik und einer anderen Statistik festgestellt wird, wird eine Fehlernachricht ausgegeben. Jedoch werden vielleicht nicht immer Fehler gemeldet, wenn Konflikte auftreten. In einigen Fällen können die Konflikte nur schwer festgestellt und als Fehler gemeldet werden, besonders wenn die beiden zusammengehörigen Statistiken in verschiedenen Katalogen gespeichert sind. Aus diesem Grund sollten Sie solche Konflikte umsichtig vermeiden.

Die folgenden allgemeinen Prüfungen sollten vor der Aktualisierung einer Katalogstatistik durchgeführt werden:

1. Numerische Statistikdaten müssen -1 bzw. größer oder gleich null (0) sein.
2. Numerische Statistikdaten, die Prozentwerte darstellen (z. B. CLUSTERRATIO in der Katalogsicht SYSSTAT.INDEXES), müssen zwischen 0 und 100 liegen.

Anmerkung: Bei Spaltentypen sind die statistischen Daten auf Tabellenebene für NPAGES, FPAGES und OVERFLOW für eine untergeordnete Tabelle nicht aktualisierbar.

Regeln zur Aktualisierung von Kurznamenstatistiken

Es gibt nur vier statistische Werte, die Sie in der Katalogsicht SYSSTAT.TABLES aktualisieren können: CARD, FPAGES, NPAGES und OVERFLOW. Beachten Sie dabei folgendes:

1. Der Wert für CARD der Tabelle muss größer als alle auf diese Tabelle bezogenen Werte für COLCARD in der Katalogsicht SYSSTAT.COLUMNS sein.
2. Der Wert für CARD muss größer als der Wert für NPAGES sein.
3. Der Wert für FPAGES muss größer als der Wert für NPAGES sein.
4. Der Wert für NPAGES muss kleiner oder gleich jedem Wert für den Abrufteil der Wertepaare in der Spalte PAGE_FETCH_PAIRS für jeden Index sein (unter der Annahme, dass diese Statistik für den Index relevant ist).

5. Der Wert für CARD darf nicht kleiner oder gleich irgendeinem Wert für den Seitenabruf in den Wertepaaren der Spalte PAGE_FETCH_PAIRS für jeden Index sein (unter der Annahme, dass diese Statistik für den Index relevant ist).

Wenn Sie in einem System mit zusammengeschlossenen Datenbanken arbeiten, gehen Sie bei der Erstellung bzw. Aktualisierung von Statistiken für einen Kurznamen über eine ferne Sicht sehr vorsichtig vor. Die statistischen Informationen, wie zum Beispiel die Anzahl von Zeilen, die von diesem Kurznamen zurückgegeben werden, entsprechen möglicherweise nicht dem realen Aufwand zur Auswertung dieser Sicht und können das DB2-Optimierungsprogramm irreführen. Fälle, die von Aktualisierungen der Statistiken profitieren können, sind ferne Sichten, die für eine einzelne ferne Tabelle ohne Anwendung von Spaltenfunktionen in der SELECT-Liste definiert wurden. Komplexe Sichten erfordern möglicherweise einen komplexen Optimierungsprozess, der die Optimierung jeder einzelnen Abfrage erforderlich macht. Ziehen Sie stattdessen die Erstellung lokaler Sichten über Kurznamen in Betracht, damit das DB2-Optimierungsprogramm in der Lage ist, den Aufwand für die Sicht exakter abzuschätzen.

Regeln zur Aktualisierung von Spaltenstatistiken

Bei der Aktualisierung von Statistikdaten in der Katalogsicht SYSSTAT.COLUMNS sind folgende Regeln zu beachten. Einzelheiten zur Aktualisierung von Statistikdaten zur Datenverteilung in den Spalten finden Sie in „Regeln zur Aktualisierung von Verteilungsstatistiken für Spalten” auf Seite 159.

1. Die Werte für HIGH2KEY und LOW2KEY (in der Katalogsicht SYSSTAT.COLUMNS) müssen folgenden Regeln genügen:
   - Der Datentyp jedes Werts für HIGH2KEY, LOW2KEY muss mit dem Datentyp der entsprechenden Spalte des Benutzers übereinstimmen, für die die Statistik gedacht ist. Da die Spalte HIGH2KEY den Datentyp VARCHAR (Zeichenfolge variabler Länge) hat, müssen Sie den Wert in Anführungszeichen setzen. Wenn Sie zum Beispiel die Spalte HIGH2KEY für eine Benutzerspalte mit dem Datentyp INTEGER (ganze Zahl) auf den Wert 25 setzen wollen, muss Ihre Aktualisierungsanweisung den Befehl SET HIGH2KEY = '25' enthalten.
   - Die Länge der Werte für HIGH2KEY, LOW2KEY muss entweder 33 oder die maximale Länge des Datentyps der Zielspalte betragen, je nachdem, welcher der beiden Werte kleiner ist.
   - Der Wert für HIGH2KEY muss größer als der Wert für LOW2KEY sein, wenn es mehr als 3 unterschiedliche Werte in der entsprechenden Spalte gibt. Wenn die Spalte 3 oder weniger unterschiedliche Werte aufweist, kann der Wert für HIGH2KEY gleich dem Wert für LOW2KEY sein.
2. Die Kardinalität einer Spalte (Statistik COLCARD in der Katalogsicht SYSSTAT.COLUMNS) kann nicht größer sein als die Kardinalität der zugehörigen Tabelle (Statistik CARD in der Katalogsicht SYSSTAT.TABLES).

3. Die Anzahl der Nullwerte in einer Spalte (Statistik NUMNULLS in der Katalogsicht SYSSTAT.COLUMNS) kann nicht größer sein als die Kardinalität der zugehörigen Tabelle (Statistik CARD in der Katalogsicht SYSSTAT.TABLES).
4. Für Spalten mit folgenden Datentypen werden keine Statistikdaten unterstützt: LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB.

Regeln zur Aktualisierung von Verteilungsstatistiken für Spalten

Der Abschnitt „Benutzeraktualisierbare Katalogstatistiken" auf Seite 154 enthält allgemeine Informationen zur Aktualisierung von Katalogstatistiken. Sie sollten mit den Informationen des genannten Abschnitts vertraut sein, bevor Sie versuchen, statistische Daten über die Werteverteilung in Spalten zu aktualisieren.

Damit sämtliche Statistiken im Katalog konsistent bleiben, muss bei der Aktualisierung der Verteilungsstatistiken mit großer Vorsicht gearbeitet werden. Im Einzelnen müssen die Katalogeinträge für Häufigkeitsstatistiken und Quantildaten für jede Spalte die folgenden Bedingungen erfüllen:

- Häufigkeitsstatistiken (in der Katalogsicht SYSSTAT.COLDIST). Dazu gehören:
  - Die Werte in der Spalte VALCOUNT müssen bei steigenden Werten für SEQNO unverändert bleiben oder abnehmen.
  - Die Anzahl der Werte in der Spalte COLVALUE muss kleiner oder gleich der Anzahl der unterschiedlichen Werte in der Spalte sein. Diese Anzahl ist in der Spalte COLCARD der Katalogsicht SYSSTAT.COLUMNS gespeichert.
  - Die Summe der Werte in der Spalte VALCOUNT muss kleiner oder gleich der Anzahl der Zeilen in der Spalte sein. Diese Anzahl ist in der Spalte CARD der Katalogsicht SYSSTAT.TABLES gespeichert.
  - In der Regel sollten die Werte in der Spalte COLVALUE zwischen dem zweithöchsten und dem zweitniedrigsten Datenwert für die Spalte liegen. Diese beiden Werte sind in der Spalte HIGH2KEY bzw. LOW2KEY der Katalogsicht SYSSTAT.COLUMNS gespeichert. Es kann einen häufigen Wert geben, der größer als der Wert für HIGH2KEY ist, und einen, der kleiner als der Wert für LOW2KEY ist.
- Quantile (in der Katalogsicht SYSSTAT.COLDIST). Dazu gehören:
  - Die Werte in der Spalte COLVALUE müssen bei steigenden Werten für SEQNO unverändert bleiben oder abnehmen.
  - Die Werte in der Spalte VALCOUNT müssen bei steigenden Werten für SEQNO steigen.
  - Der größte Wert in der Spalte COLVALUE muss einen entsprechenden Eintrag in der Spalte VALCOUNT haben, der gleich der Anzahl von Zeilen in der Spalte ist.

– In der Regel sollten die Werte in der Spalte COLVALUE zwischen dem zweithöchsten und dem zweitniedrigsten Datenwert für die Spalte liegen. Diese beiden Werte sind in der Spalte HIGH2KEY bzw. LOW2KEY der Katalogsicht SYSSTAT.COLUMNS gespeichert.

Nehmen Sie an, es stehen Verteilungsstatistikdaten für eine Spalte C1 mit „Z" Zeilen zur Verfügung, und Sie möchten die Statistiken so modifizieren, dass sie einer Spalte mit identischen relativen Proportionen der Datenwerte, aber mit „(F x Z)" Zeilen entsprechen. Um die Größenordnung der Häufigkeitsstatistiken um einen Faktor F zu erhöhen, muss jeder Eintrag der Spalte VALCOUNT mit F multipliziert werden. Um die Quantilwerte ebenfalls um einen Faktor F zu vergrößern, muss jeder Eintrag der Spalte VALCOUNT mit F multipliziert werden. Werden diese Regeln nicht beachtet, kann das Optimierungsprogramm den falschen Filterfaktor verwenden, was bei der Ausführung der Abfrage zu nicht vorhersehbaren Auswirkungen auf die Leistung führen kann.

Regeln zur Aktualisierung von Indexstatistiken

Bei der Aktualisierung der Statistiken in der Katalogsicht SYSSTAT.INDEXES sind folgende Regeln zu beachten:

1. Die Werte für PAGE_FETCH_PAIRS (in der Katalogsicht SYSSTAT.INDEXES) müssen folgende Regeln erfüllen:
   - Einzelne Werte in der Statistik PAGE_FETCH_PAIRS müssen durch eine Folge von Leerzeichenbegrenzern getrennt werden.
   - Einzelne Werte in der Statistik PAGE_FETCH_PAIRS dürfen eine Länge von 10 Stellen nicht überschreiten und müssen kleiner als der größte ganzzahlige Wert (MAXINT = 2147483647) sein.
   - Es muss immer einen gültigen Wert für PAGE_FETCH_PAIRS geben, wenn für CLUSTERFACTOR ein Wert größer null (0) angegeben ist.
   - Es müssen sich genau 11 Wertepaare in einer einzelnen Statistik PAGE_FETCH_PAIR befinden.
   - Die Einträge für die Puffergrößen in PAGE_FETCH_PAIRS müssen eine aufsteigende Wertefolge bilden.
   - Kein Wert für die Puffergröße in einem Eintrag für PAGE_FETCH_PAIRS darf den Wert MIN(NPAGES, 524287) überschreiten, wobei NPAGES die Anzahl der Seiten in der entsprechenden Tabelle (in der Katalogsicht SYSSTAT.TABLES) ist.
   - Einträge für „Seitenabrufe" in PAGE_FETCH_PAIRS müssen in absteigender Wertefolge angegeben werden, wobei kein einzelner Eintrag für „Seitenabrufe" kleiner als NPAGES sein darf. Einträge für „Seitenabrufe" in PAGE_FETCH_PAIRS können nicht größer als der Wert der Statistik CARD (Kardinalität) der zugehörigen Tabelle sein.

- Wenn der Wert für die Puffergröße in zwei aufeinander folgenden Paaren übereinstimmt, muss der Wert für den Seitenabruf (Page fetch) in beiden Paaren ebenfalls übereinstimmen (in der Katalogsicht SYSSTAT-.TABLES).

Eine gültige Aktualisierung der Wertefolge für PAGE_FETCH_PAIRS ist zum Beispiel:

```
PAGE_FETCH_PAIRS =
 '100 380 120 360 140 340 160 330 180 320 200 310 220 305 240 300
  260 300 280 300 300 300'
```

Dabei gilt folgendes:

```
NPAGES = 300
CARD   = 10000
CLUSTERRATIO  =  -1
CLUSTERFACTOR = 0.9
```

2. Die Werte für CLUSTERRATIO und CLUSTERFACTOR (in der Katalogsicht SYSSTAT.INDEXES) müssen folgende Regeln erfüllen:
   - Gültige Werte für CLUSTERRATIO (Clusterverhältnis) sind -1 bzw. Werte zwischen 0 und 100.
   - Gültige Werte für CLUSTERFACTOR (Clusterfaktor) sind -1 bzw. Werte zwischen 0 und 1.
   - Es muss immer mindestens einer der Werte für CLUSTERRATIO und CLUSTERFACTOR gleich -1 sein.
   - Wenn für CLUSTERFACTOR ein positiver Wert angegeben ist, müssen gültige Statistikdaten in der Spalte PAGE_FETCH_PAIR enthalten sein.
3. Die folgenden Regeln gelten für die Werte der Statistiken FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD und FULLKEYCARD:
   - Der Wert für FIRSTKEYCARD muss gleich dem Wert für FULLKEYCARD für einen einspaltigen Index sein.
   - Der Wert für FIRSTKEYCARD muss gleich dem Wert COLCARD (in SYSSTAT.COLUMNS) für die entsprechende Spalte sein.
   - Wenn einige dieser Indexstatistikwerte nicht relevant sind, sollten Sie sie auf den Wert -1 setzen. Wenn Sie beispielsweise einen Index mit nur drei Spalten haben, setzen Sie FIRST4KEYCARD auf den Wert -1.
   - Für mehrspaltige Indizes gilt die folgende Beziehung, wenn alle Statistikwerte relevant sind:

     ```
     FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
        <= FULLKEYCARD <= CARD
     ```
4. Die folgenden Regeln gelten für SEQUENTIAL_PAGES und DENSITY:
   - Gültige Werte für SEQUENTIAL_PAGES sind -1 bzw. Werte zwischen 0 und NLEAF.
   - Gültige Werte für DENSITY sind -1 bzw. Werte zwischen 0 und 100.

Aktualisieren der Statistiken für benutzerdefinierte Funktionen

Über die Katalogsicht SYSSTAT.FUNCTIONS können Statistiken für benutzerdefinierte Funktionen (UDFs) aktualisiert werden. Wenn diese Statistik verfügbar ist, werden sie vom Optimierungsprogramm zur Abschätzung des Aufwands für verschiedene Zugriffspläne verwendet. Ist diese Statistik nicht verfügbar, enthalten die Spalten der Statistik den Wert -1, und das Optimierungsprogramm verwendet Standardwerte, die von einer einfachen benutzerdefinierten Funktion ausgehen.

Die folgende Tabelle enthält Informationen zu den Spalten mit statistischen Daten, die für benutzerdefinierte Funktionen aktualisiert werden können:

Tabelle 13. Funktionsstatistiken (SYSCAT.FUNCTIONS und SYSSTAT.FUNCTIONS)

Statistik	Beschreibung
IOS_PER_INVOC	Geschätzte Anzahl der Schreib-/Leseanforderungen, die jedes Mal ausgeführt werden, wenn eine Funktion ausgeführt wird
INSTS_PER_INVOC	Geschätzte Anzahl der Maschinenanweisungen, die jedes Mal ausgeführt werden, wenn eine Funktion ausgeführt wird
IOS_PER_ARGBYTE	Geschätzte Anzahl der Schreib-/Leseanforderungen, die für jedes Eingabeargumentbyte ausgeführt werden
INSTS_PER_ARGBYTES	Geschätzte Anzahl der Maschinenanweisungen, die für jedes Eingabeargumentbyte ausgeführt werden
PERCENT_ARGBYTES	Geschätzter Durchschnittsprozentwert der Eingabeargumentbyte, die von der Funktion tatsächlich verarbeitet werden
INITIAL_IOS	Geschätzte Anzahl der Schreib-/Leseanforderungen, die nur beim ersten/letzten Aufruf der Funktion ausgeführt werden
INITIAL_INSTS	Geschätzte Anzahl der Maschinenanweisungen, die nur beim ersten/letzten Aufruf der Funktion ausgeführt werden
CARDINALITY	Geschätzte Anzahl von Zeilen, die von einer Tabellenfunktion generiert werden

Betrachten Sie zum Beispiel eine benutzerdefinierte Funktion (EU_SHOE), die eine amerikanische Schuhgröße in die entsprechende europäische Schuhgröße umwandelt. (Diese beiden Schuhgrößen könnten benutzerdefinierte Datentypen (UDTs) haben.) Für diese UDF sollten Sie die Statistikspalten mit folgenden Werten versehen:

- Für INSTS_PER_INVOC sollte als Wert die geschätzte Anzahl der Maschineninstruktionen festgelegt werden, die zu folgenden Operationen erforderlich sind:
  - Aufrufen der Funktion EU_SHOE
  - Initialisieren der Ausgabezeichenfolge
  - Rückgabe des Ergebnisses
- Für INSTS_PER_ARGBYTE sollte als Wert die geschätzte Anzahl der Maschineninstruktionen festgelegt werden, die zur Umwandlung der Eingabezeichenfolge in die europäische Schuhgröße erforderlich ist.
- Der Wert für PERCENT_ARGBYTES würde auf 100 gesetzt werden, was anzeigt, dass die gesamte Eingabezeichenfolge umzuwandeln ist.
- Die Werte für INITIAL_INSTS, IOS_PER_INVOC, IOS_PER_ARGBYTE und INITIAL_IOS sollten alle auf 0 gesetzt werden, da diese benutzerdefinierte Funktion lediglich Berechnungen ausführt.

PERCENT_ARGBYTES würde für eine Funktion verwendet, die nicht immer die gesamte Eingabezeichenfolge verarbeitet. Ein Beispiel wäre eine benutzerdefinierte Funktion (LOCATE), an die zwei Argumente als Eingabe übergeben werden und die die Anfangsposition des ersten Vorkommens des ersten Arguments innerhalb des zweiten Arguments als Ergebnis zurückliefert. Nehmen Sie an, dass die Länge des ersten Arguments ausreichend klein ist, um im Vergleich zum zweiten Argument kaum eine Rolle zu spielen, und im Durchschnitt 75% des zweiten Arguments zum Auffinden des ersten durchsucht werden. Aufgrund dieser Informationen sollte der Wert für PERCENT_ARGBYTES auf 75 gesetzt werden. Die obige Abschätzung eines Durchschnitts von 75% fußt dabei auf folgenden zusätzlichen Annahmen:

- In der Hälfte der Fälle wird das erste Argument gar nicht gefunden, d. h., das gesamte zweite Argument wird durchsucht.
- Die Wahrscheinlichkeit, dass das erste Argument innerhalb des zweiten Arguments auftritt, ist an allen Stellen gleich groß, d. h., in den Fällen, in denen das erste Argument überhaupt gefunden wird, muss im Durchschnitt die Hälfte des zweiten Arguments durchsucht werden.

Die Werte für INITIAL_INSTS bzw. INITIAL_IOS können zur Aufzeichnung der geschätzten Anzahl von Maschineninstruktionen bzw. Schreib-/Leseanforderungen verwendet werden, die nur beim ersten bzw. letzten Aufruf der Funktion ausgeführt werden. Diese Möglichkeit könnte zum Beispiel zur Aufzeichnung des Aufwands für die Einrichtung eines Arbeitspufferbereichs verwendet werden.

Informationen über Ein-/Ausgaben und Instruktionen, die von einer benutzerdefinierten Funktion verursacht werden, erhalten Sie über die Ausgaben des Compilers für die Programmiersprache bzw. der vom Betriebssystem bereitgestellten Überwachungsprogramme.

Modellieren im Einsatz befindlicher Datenbanken

Manchmal ist es wünschenswert, auf einem Testsystem einen Teil der Daten eines tatsächlich geschäftlich genutzten Systems nachzubilden. Jedoch sind die Zugriffspläne, die auf einem solchen Testsystem ausgewählt werden, nicht unbedingt dieselben wie die, die auf dem tatsächlich genutzten System gewählt würden, sofern nicht die Katalogstatistiken und die Konfigurationsparameter auf dem Testsystem so aktualisiert werden, dass sie mit denen auf dem Produktionssystem übereinstimmen.

Es steht das Tool *db2look* zur Verfügung, das für die Produktionsdatenbank ausgeführt werden kann, um die Aktualisierungsanweisungen (UPDATE-Anweisungen) zu generieren, die erforderlich sind, um die Katalogstatistiken der Testdatenbank in Übereinstimmung mit denen der Produktionsdatenbank zu bringen. Diese Aktualisierungsanweisungen können mit Hilfe des Programms *db2look* mit der Option `-m` (mimic mode) generiert werden. In diesem Fall generiert das Tool *db2look* eine Prozedur für den Befehlsprozessor, die alle Anweisungen enthält, die erforderlich sind, um die Katalogstatistiken der Produktionsdatenbank nachzubilden. Dies kann bei der Analyse von SQL-Anweisungen mit Hilfe von Visual Explain in einer Testumgebung nützlich sein. Sie können Datenbankdatenobjekte einschließlich Tabellen, Sichten, Indizes und andere Objekte in einer Datenbank wieder erstellen, indem Sie die DDL-Anweisungen mit dem Befehl *db2look -e* extrahieren. Sie können die Prozedur für den Befehlsprozessor, die mit diesem Befehl erstellt wurde, in einer anderen Datenbank ausführen, um die Datenbank neu zu erstellen. Sie können die Optionen `-e` und `-m` zusammen verwenden.

Nach der Ausführung der von `db2look` erstellten Aktualisierungsanweisungen im Testsystem kann das Testsystem zur Prüfung der Zugriffspläne verwendet werden, die in der Produktionsdatenbank generiert werden sollen. Da das Optimierungsprogramm die Art und Konfiguration der Tabellenbereiche zur Abschätzung der E/A-Aufwände verwendet, muss das Testsystem über dieselbe Tabellenbereichsanordnung bzw. -konfiguration verfügen. Das heißt, sie muss dieselbe Anzahl von Behältern desselben Typs, SMS oder DMS, haben.

Das Tool *db2look* befindet sich im Unterverzeichnis *bin*.

Weitere Informationen zur Verwendung dieses Tools erhalten Sie, wenn Sie Folgendes in eine Befehlszeile eingeben:

```
db2look -h
```

Weitere Informationen zu diesem Programm finden Sie außerdem im Handbuch *Command Reference*.

Darüber hinaus bietet die Steuerzentrale eine Schnittstelle zum Programm *db2look* namens „DDL generieren für Objektname". Die Steuerzentrale ermöglicht eine Integration der Ergebnisdatei aus dem Dienstprogramm in die Prozedurzentrale. Sie können außerdem den Befehl *db2look* über die Steuerzentrale zeitlich terminieren. Ein Unterschied bei der Verwendung der Steuerzentrale besteht darin, dass eine Analyse nur einer Tabelle durchgeführt werden kann, während in einem Aufruf des Befehls *db2look* bis zu 30 Tabellen analysiert werden können. Beachten Sie darüber hinaus, dass LaTex- und Graphical-Ausgaben über die Steuerzentrale nicht unterstützt werden.

Sie können auch das Dienstprogramm `db2look` für eine OS/390-Datenbank ausführen. Das Dienstprogramm `db2look` extrahiert die Statistikdatenanweisungen DDL und UPDATE für OS/390-Objekte. Diese Funktion ist sehr nützlich, wenn Sie OS/390-Objekte extrahieren und in einer DB2 UDB-Datenbank (UDB - Universal Database) erneut erstellen möchten. Zusätzliche Informationen zum Dienstprogramm `db2look` finden Sie im Handbuch *Command Reference*.

Zwischen den Statistikdaten von DB2 UDB und denen von OS/390 gibt es einige Unterschiede. Das Dienstprogramm `db2look` führt die entsprechenden Umsetzungen von DB2 für OS/390 auf DB2 UDB aus, wenn dies zutreffend ist, und setzt die Statistikdaten von DB2 UDB, für die in DB2 für OS/390 keine Entsprechung vorhanden ist, auf einen Standardwert (-1). Im folgenden wird beschrieben, wie das Dienstprogramm `db2look` die Statistikdaten von DB2 für OS/390 den Statistikdaten von DB2 UDB zuordnet. In den nachfolgenden Erläuterungen steht jeweils „UDB_x" für eine Statistikdatenspalte von DB2 UDB; „S390_x" steht für eine Statistikdatenspalte von DB2 für OS/390.

1. Statistikdaten auf Tabellenebene.

   UDB_CARD = S390_CARDF
   UDB_NPAGES = S390_NPAGES

   Es gibt keinen Parameter S390_FPAGES. DB2 für OS/390 verfügt jedoch über einen anderen Parameter für Statistikdaten mit dem Namen PCTPAGES, der den Prozentsatz von aktiven Tabellenbereichsseiten darstellt, die Zeilen der Tabelle enthalten. Daher kann der Wert des Parameters UDB_FPAGES auf der Basis von S390_NPAGES und S390_PCTPAGES wie folgt berechnet werden:

   ```
   UDB_FPAGES=(S390_NPAGES * 100)/S390_PCTPAGES
   ```

Es gibt keinen Parameter S390_OVERFLOW, der UDB_OVERFLOW zugeordnet werden kann. Daher wird dieser Wert vom Dienstprogramm db2look einfach auf den Standardwert festgelegt:

```
UDB_OVERFLOW=-1
```

2. Statistikdaten auf Spaltenebene.

   UDB_COLCARD = S390_COLCARDF
   UDB_HIGH2KEY = S390_HIGH2KEY
   UDB_LOW2KEY = S390_LOW2KEY

   Es gibt keinen Parameter S390_AVGCOLLEN, der UDB_AVGCOLLEN zugeordnet werden kann. Daher wird dieser Wert vom Dienstprogramm db2look einfach auf den Standardwert festgelegt:

   ```
   UDB_AVGCOLLEN=-1
   ```

3. Statistikdaten auf Indexebene.

   UDB_NLEAF = S390_NLEAF
   UDB_NLEVELS = S390_NLEVELS
   UDB_FIRSTKEYCARD= S390_FIRSTKEYCARD
   UDB_FULLKEYCARD = S390_FULLKEYCARD
   UDB_CLUSTERRATIO= S390_CLUSTERRATIO

   Die anderen Statistikdaten, für die es keine OS/390-Entsprechungen gibt, werden einfach auf den Standardwert festgelegt. Dieser ist wie folgt definiert:

   ```
   UDB_FIRST2KEYCARD = -1
   UDB_FIRST3KEYCARD = -1
   UDB_FIRST4KEYCARD  = -1
   UDB_CLUSTERFACTOR = -1
   UDB_SEQUENTIAL_PAGES = -1
   UDB_DENSITY = -1
   ```

4. Spaltenverteilungsstatistiken.

   In DB2 für OS/390-SYSIBM.SYSCOLUMNS gibt es zwei Arten von Statistikdaten: Die Art „F" für häufige (Frequent) Werte und die Art „C" für Kardinalität (Cardinality). Nur Einträge der Art „F" gelten für DB2 für UDB. Dies sind auch die Einträge, die in Betracht gezogen werden.

   ```
   UDB_COLVALUE = S390_COLVALUE
   UDB_VALCOUNT = S390_FrequencyF * S390_CARD
   ```

   Darüber hinaus gibt es in DB2 für OS/390-SYSIBM.SYSCOLUMNS keine Spalte SEQNO. Diese ist aber für DB2 für UDB erforderlich. Daher generiert das Dienstprogramm db2look eine solche Spalte automatisch.

Unterelementstatistiken

Auch eine Option zum Sammeln und Verwenden von Statistikdaten zu Unterelementen ist vorgesehen. Dabei handelt es sich um Statistikdaten zu Spaltendaten, wenn diese eine Struktur in Form einer Folge von Unterfeldern oder Unterelementen aufweisen, die durch Leerzeichen getrennt sind.

Angenommen, eine Datenbank enthält eine Tabelle namens DOCUMENTS, deren Zeilen jeweils ein Dokument beschreiben, und angenommen, DOCUMENTS enthält eine Spalte namens KEYWORDS, die eine Liste relevanter Schlüsselwörter für das Dokument enthält, die zur Datenabfrage verwendet werden. Die Spalte KEYWORDS könnte z. B. folgende Werte enthalten:

```
'Datenbank Simulation Analytisch Business Intelligence'
'Simulation Modell Fruchtfliege Reproduktion Temperatur'
'Forstwirtschaft Fichte Boden Erosion Regen'
'Wald Temperatur Boden Niederschlag Brand'
```

In diesem Beispiel besteht jede Spalte aus 5 Unterelementen, von denen jedes ein Wort (das Schlüsselwort) ist und durch ein Leerzeichen von den anderen Wörtern getrennt ist.

Bei Abfragen, die für solche Spalten das Vergleichselement LIKE mit dem Platzhalterzeichen % verwenden:

```
SELECT .... FROM DOCUMENTS WHERE KEYWORDS LIKE '%Simulation%'
```

ist es für das Optimierungsprogramm häufig von Nutzen, wenn es einige grundlegende Statistikdaten zur Unterelementstruktur der Spalte kennt, z. B.:

SUB_COUNT
: Die durchschnittliche Anzahl der Unterelemente.

SUB_DELIM_LENGTH
: Die durchschnittliche Länge der einzelnen Begrenzer, die Unterelemente voneinander trennen. Unter Begrenzer wird in diesem Zusammenhang ein Leerzeichen oder mehrere aufeinander folgende Leerzeichen verstanden.

Beim Beispiel der Spalte KEYWORDS hat SUB_COUNT den Wert 5, und SUB-_DELIM_LENGTH den Wert 1, da jeder Begrenzer ein einzelnes Leerzeichen ist.

Der Systemadministrator kann über eine Erweiterung der Registrierungsvariablen DB2_LIKE_VARCHAR auf die Sammlung und Verwendung dieser Statistikdaten Einfluss nehmen. Diese Registrierungsvariable bestimmt, wie das Optimierungsprogramm ein Vergleichselement der folgenden Form behandelt:

```
COLUMN LIKE '%xxxxxx'
```

(wobei *xxxxxx* für eine beliebige Zeichenfolge steht); ein beliebiges LIKE-Vergleichselement also, dessen Suchwert mit einem Prozentzeichen (%) beginnt. (Der Suchwert kann, muss aber nicht auf % enden.) Derartige Vergleichselemente werden in der Folge als "Platzhalter-LIKE-Vergleichselemente" bezeichnet. Das Optimierungsprogramm muss für alle Vergleichselemente schätzen, wie viele Zeilen mit dem Vergleichselement übereinstimmen. Bei LIKE-Vergleichselementen nimmt das Optimierungsprogramm an, dass die verglichene Spalte (COLUMN) eine Struktur in Form einer Folge miteinander verknüpfter Elementen aufweist, die die Spalte bilden, und schätzt die Länge jedes Elements ausgehend von der Länge der Zeichenfolge ohne führende und nachgestellte %-Zeichen. Die neue Syntax lautet:

```
db2set  DB2_LIKE_VARCHAR=[Y|N|S|num1][,Y|N|num2]

Dabei gilt Folgendes:
   - der erste Term (vor dem Komma) bedeutet Folgendes,
      aber nur für Spalten ohne positive
      Statistikdaten zu Unterelementen
      S                 In DB2 Version 2 verwendeten Algorithmus verwenden.
      N                 Unterelement-Algorithmus mit fester Länge verwenden.
      Y (Standardwert)  Unterelement-Algorithmus mit variabler Länge
                        mit Standardwert als Algorithmusparameter verwenden.
      num1              Unterelement-Algorithmus mit variabler Länge verwenden,
                        und num1 als Algorithmusparameter verwenden.

   - der zweite Term (nach dem Komma) bedeutet:
      N (Standardwert)  Keine Statistikdaten zu Unterelementen sammeln oder
      Y                 verwenden. Statistikdaten zu Unterelementen sammeln.
                        Unterelement-Algorithmus mit variabler Länge, der
                        verwendet, zusammen mit einem Standardwert für
                        diese Daten den Algorithmusparameter verwenden, wenn
                        Spalten mit positiven Statistikdaten zu
                        Unterelementen vorliegen.
      num2              Statistikdaten zu Unterelementen sammeln. Unterelement-
                        Algorithmus mit variabler Länge, der diese
                        Daten verwendet, zusammen mit num2 als
                        Algorithmusparameter verwenden, wenn Spalten mit
                        positiven Statistikdaten zu Unterelementen vorliegen.
```

Wenn der Wert von DB2_LIKE_VARCHAR nur den ersten Term enthält, werden keine Statistikdaten zu Unterelementen gesammelt. Bereits gesammelte Daten werden ignoriert. Der angegebene Wert bestimmt auf dieselbe Weise wie zuvor, wie das Optimierungsprogramm die Selektivität von Platzhalter-LIKE-Vergleichselementen berechnet:

- Ist der Wert gleich S, verwendet das Optimierungsprogramm denselben Algorithmus, der in DB2 Version 2 verwendet wurde. Dabei wird nicht vom Unterelementmodell ausgegangen.
- Ist der Wert gleich N, verwendet das Optimierungsprogramm einen Algorithmus, der auf dem Unterelementmodell und auf einer festen Länge der Spalte COLUMN basiert, auch wenn für COLUMN eine variable Länge definiert ist.
- Ist der Wert Y (der Standardwert) oder eine Gleitkommakonstante, verwendet das Optimierungsprogramm einen Algorithmus, der auf dem Unterelementmodell und auf einer variablen Länge, falls definiert, der Spalte COLUMN basiert. Außerdem leitet er Statistikdaten zu Unterelementen aus der Abfrage selbst und nicht aus den Daten ab. Dieser Algorithmus umfasst einen Parameter (den "Algorithmusparameter"), der angibt, um wie viel die Länge des Elements die Länge der in %-Zeichen eingeschlossenen Zeichenfolge übersteigt.
- Ist der Wert gleich Y, verwendet das Optimierungsprogramm den Standardwert 1,9 als Algorithmusparameter.
- Ist der Wert eine Gleitkommakonstante, verwendet das Optimierungsprogramm den angegebenen Wert als Algorithmusparameter. Der Wert dieser Konstante muss im Bereich zwischen 0 und 6,2 liegen.

Wenn der Wert von DB2_LIKE_VARCHAR zwei Terme enthält, von denen der zweite Y oder eine Gleitkommakonstante ist, werden im Rahmen einer RUNSTATS-Operation Unterelementstatistikdaten für Einzelbytezeichensatz-Zeichenfolgespalten des Typs CHAR, VARCHAR, GRAPHIC oder VARGRAPHIC gesammelt und bei der Kompilierung von Abfragen unter Verwendung von Platzhalter-LIKE-Vergleichselementen verwendet. Das Optimierungsprogramm verwendet einen Algorithmus, der auf dem Unterelementmodell basiert, und berechnet die Selektivität des Vergleichselements anhand der Statistikwerte SUB_COUNT und SUB_DELIM_LENGTH sowie eines Algorithmusparameters.

Der Algorithmusparameter wird auf dieselbe Weise angegeben wie der "Ableitungs"-Algorithmus, d. h.:

- Ist der Wert gleich Y, verwendet das Optimierungsprogramm den Standardwert 1,9 als Algorithmusparameter.
- Ist der Wert eine Gleitkommakonstante, verwendet das Optimierungsprogramm den angegebenen Wert als Algorithmusparameter. Der Wert dieser Konstante muss im Bereich zwischen 0 und 6,2 liegen.

Wenn das Optimierungsprogramm während der Kompilierung feststellt, dass für die von der Abfrage betroffene Spalte keine Statistikdaten zu Unterelementen gesammelt wurden, wird der "Ableitungs"-Algorithmus für Unterelemente verwendet. Das ist der Algorithmus, der verwendet wird, wenn nur der erste Term von DB2_LIKE_VARCHAR angegeben wird. Damit das Optimierungsprogramm die Statistikdaten zu Unterelementen berücksichtigt, muss daher der zweite Term von DB2_LIKE_VARCHAR sowohl bei der Ausführung von RUNSTATS als auch bei der Kompilierung angegeben werden.

Die Werte der Unterelementstatistik können durch eine Abfrage in SYSIBM.SYSCOLUMNS abgerufen werden. Zum Beispiel:

```
select substr(NAME,1,16), SUB_COUNT, SUB_DELIM_LENGTH
   from sysibm.syscolumns where tbname = 'DOCUMENTS'
```

Die Spalten SUB_COUNT und SUB_DELIM_LENGTH sind nicht in der Statistiksicht SYSSTAT.COLUMNS enthalten und können daher nicht aktualisiert werden.

Anmerkung: Bei Verwendung dieser Option dauert die Ausführung von RUNSTATS u. U. länger. Bei einer Tabelle mit fünf Zeichenspalten z. B. kann die Ausführung von RUNSTATS zwischen 15 und 40 % länger dauern, wenn die Optionen DETAILED und DISTRIBUTION nicht verwendet werden. Wird die Option DETAILED oder DISTRIBUTION angegeben, ist der Mehraufwand prozentual gesehen u. U. geringer, auch wenn der Mehraufwand absolut gesehen derselbe ist. Wenn Sie die Verwendung dieser Option beabsichtigen, sollten Sie zwischen diesem Mehraufwand und den Verbesserungen der Abfrageleistung abwägen.

Kapitel 6. Der SQL-Compiler

Wenn eine SQL-Abfrage kompiliert wird, wird eine Reihe von Schritten ausgeführt, bevor der „beste" Zugriffsplan entweder ausgeführt oder im Systemkatalog gespeichert wird.

In einer Umgebung mit partitionierten Datenbanken finden alle Operationen, die vom SQL-Compiler an einer SQL-Abfrage ausgeführt werden, in der Datenbankpartition statt, zu der Sie die Verbindung herstellen. Vor der Ausführung wird die kompilierte Abfrage an alle Datenbankpartitionen in der Datenbank gesendet.

Die folgenden Themen enthalten weitere Informationen über die Schritte, die vom SQL-Compiler ausgeführt werden:

- Übersicht über den SQL-Compiler
- Umschreiben der Abfrage durch den SQL-Compiler
- Mischen von Operationen
- Verschieben von Operationen
- Übersetzen von Vergleichselementen
- Datenzugriffskonzepte und Optimierung
- Optimierungsstrategien für partitionsinterne Parallelität
- Automatische Übersichtstabellen
- Abfragecompilerphasen für zusammengeschlossene Datenbanken

Die folgenden Kapitel enthalten darüber hinaus Informationen zu außerhalb des Compilers auftretenden Faktoren, die die Ergebnisse des Compilers beeinflussen können:

- „Kapitel 3. Überlegungen zu Anwendungen" auf Seite 47
- „Kapitel 4. Überlegungen zur Umgebung" auf Seite 103
- „Kapitel 5. Systemkatalogstatistiken" auf Seite 129.

Wie Sie den vom SQL-Compiler ausgewählten Zugriffsplan untersuchen können, wird in „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251, beschrieben.

Übersicht über den SQL-Compiler

Der SQL-Compiler führt mehrere Schritte aus, bevor ein Zugriffsplan erstellt wird, den Sie ausführen können. Eine Darstellung dieser Schritte finden Sie in Abb. 12.

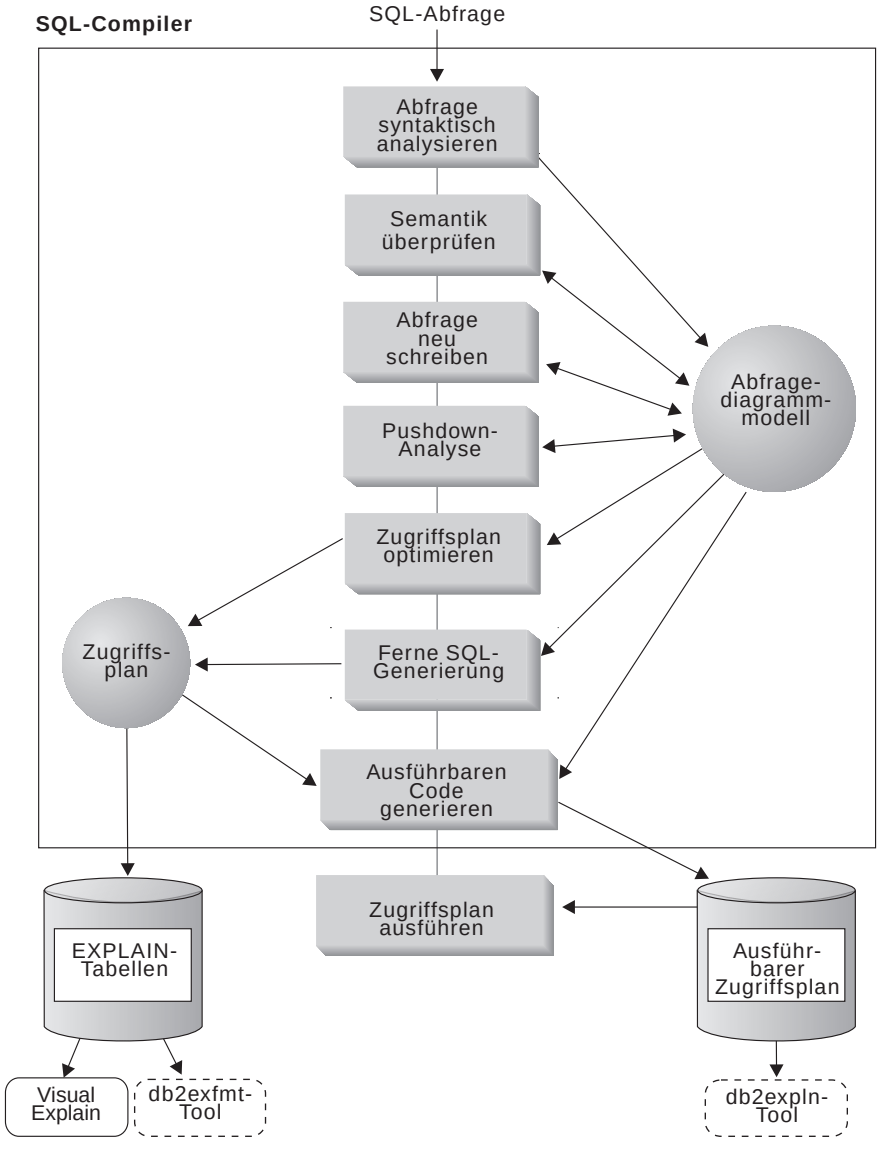


Abbildung 12. Vom SQL-Compiler ausgeführte Schritte

Wie in der Abbildung zu erkennen ist, stellt das **Abfragediagrammmodell** eine Schlüsselkomponente des SQL-Compilers dar. Das *Abfragediagrammmodell* ist

eine interne, im Speicher befindliche Datenbank, die zur Darstellung der Abfrage durch den gesamten Prozess der Abfragekompilierung hindurch verwendet wird, wie im Folgenden näher erläutert wird:

- **Abfrage analysieren**

  Die erste Aufgabe des SQL-Compilers besteht darin, die SQL-Abfrage zu analysieren, um die Syntax auf Gültigkeit zu überprüfen. Wenn Syntaxfehler festgestellt werden, beendet der SQL-Compiler die Verarbeitung und die entsprechende SQL-Fehlernachricht wird an die Anwendung, die versuchte, die SQL-Anweisung zu kompilieren, zurückgemeldet. Wenn die Analyse abgeschlossen ist, wird eine interne Darstellung der Abfrage erstellt.

- **Semantik prüfen**

  Die zweite Aufgabe des Compiler ist es sicherzustellen, dass es keine Inkonsistenzen zwischen Teilen der Anweisung gibt. Eine einfaches Beispiel für diese Semantikprüfung ist die Überprüfung, ob die Spalte für die Skalarfunktion YEAR mit dem Datentyp DATETIME (Datum/Uhrzeit) definiert wurde. Während dieser zweiten Phase fügt der Compiler auch die Bedingungssemantik in das Abfragediagrammodell ein, zu der die Auswirkungen der referenziellen Integritätsbedingungen, der Prüfungen auf Integritätsbedingungen in Tabellen, der Auslöser und der Sichten gehören.

  Das Abfragediagrammodell enthält die gesamte Semantik von Abfragen, einschließlich der Abfrageblöcke, Unterabfragen, Korrelationen, abgeleiteten Tabellen, Ausdrücken, Datentypen, Datentypumsetzungen, Umwandlungen der Codepages und der Partitionierungsschlüssel.

- **Abfrage umschreiben**

  In der dritten Phase verwendet der SQL-Compiler die vom Abfragediagrammodell zur Verfügung gestellte globale Semantik, um die Abfrage in eine Form umzusetzen, die leichter optimiert werden kann. Zum Beispiel kann der Compiler ein Vergleichselement verschieben und somit die Ebene ändern, auf der es angewandt wird, um dadurch potenziell die Abfrageleistung zu erhöhen. Diese Art der Verschiebung einer Operation wird als allgemeine Vergleichselementverschiebung (*General Predicate Pushdown*) bezeichnet. Weitere Informationen finden Sie in „Umschreiben der Abfrage durch den SQL-Compiler" auf Seite 176.

  In einer Umgebung mit partitionierten Datenbanken sind einige Abfrageoperationen für das System aufwendiger, wenn zum Beispiel folgende Funktionen zu verarbeiten sind:

  – Spaltenberechnungen (Aggregation)
  – Umverteilen von Zeilen
  – Korrelierte Unterabfragen

  Eine *korrelierte Unterabfrage ist eine Unterabfrage, die einen Verweis auf eine Spalte einer Tabelle enthält, die sich außerhalb der Unterabfrage befindet.*

In der partitionierten Umgebung kann bei einigen Abfragen eine Dekorrelation im Zusammenhang mit dem Umschreiben der Abfrage erfolgen.

Die übertragene Abfrage wird im 'Abfragediagrammodell' gespeichert.

- **Pushdown-Analyse (zusammengeschlossene Datenbanken)**

  Die Hauptfunktion dieses Schrittes ist, dem DB2-Optimierungsprogramm eine Empfehlung zu liefern, ob eine Operation an eine Datenquelle verschoben und fern ausgewertet werden kann, was als „Pushdown" bezeichnet wird. Diese Art von Pushdown-Aktivität ist für Datenquellenabfragen spezifisch und bildet eine Erweiterung zu den allgemeinen Operationen der Vergleichselementverschiebung.

  Dieser Schritt wird nur ausgeführt, wenn Abfragen für zusammengeschlossene Datenbanken ausgeführt werden. Weitere Informationen finden Sie in „Pushdown-Analyse" auf Seite 234.

- **Zugriffsplan optimieren**

  Das SQL-Optimierungsprogramm des SQL-Compilers verwendet das Abfragediagrammodell als Eingabe und generiert viele alternative Ausführungspläne zur Erfüllung der Benutzeranforderung. Das Programm schätzt den Ausführungsaufwand mit Hilfe der Statistiken für Tabellen, Indizes, Spalten und Funktionen für jeden der verschiedenen Pläne ab, und wählt den Plan mit dem geringsten geschätzten Ausführungsaufwand aus. Das Optimierungsprogramm verwendet das Abfragediagrammodel, um die Abfragesemantik zu analysieren und Informationen zu einer Vielzahl von Faktoren, einschließlich Indizes, Basistabellen, abgeleitete Tabellen, Unterabfragen, Korrelationen und Rekursion, zu erhalten.

  Das Optimierungsprogramm kann außerdem eine andere Art von Verschiebeoperation (Pushdown) in Betracht ziehen, nämlich für *Spaltenberechnungen und Sortierungen*. Die Leistung kann erhöht werden, wenn die Auswertung dieser Operationen an die Komponente der Datenverwaltungsservices (Data Management Services) verschoben werden kann. Weitere Informationen finden Sie in „Pushdown-Operatoren für Spaltenberechnungen und Sortierungen" auf Seite 226.

  Das Optimierungsprogramm berücksichtigt auch, ob es Pufferpools verschiedener Größen gibt, wenn es die Auswahl der Seitengröße festlegt. Der Faktor, dass die Umgebung eine partitionierte Datenbank enthält, wird ebenso berücksichtigt wie die Möglichkeit, den auswählten Plan für eine potenzielle abfrageinterne Parallelität in einer symmetrischen Mehrprozessorumgebung (SMP-Umgebung) auszulegen. Diese Informationen werden vom Optimierungsprogramm zur Auswahl des am besten geeigneten Zugriffsplans für die Abfrage verwendet. Weitere Informationen finden Sie in „Datenzugriffskonzepte und Optimierung" auf Seite 187.

  Die Ausgabe dieses Schritts des SQL-Compilers ist ein „Zugriffsplan". Dieser Zugriffsplan ist die Grundlage für die Informationen, die in den EXPLAIN-Tabellen erfasst werden. Die Informationen, die zur Generierung des

Zugriffsplans dienten, können mit einer Momentaufnahme der EXPLAIN-Einrichtung erfasst werden. (In „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251 finden Sie weitere Informationen zu EXPLAIN.)

- **Generierung von fernem SQL (zusammengeschlossene Datenbanken)**

  Der endgültige Plan, der vom DB2-Optimierungsprogramm gewählt wird, kann aus einer Reihe von Schritten bestehen, die eventuell an einer fernen Datenquelle ausgeführt werden. Für die Operationen, die an der jeweiligen Datenquelle ausgeführt werden, erstellt der Schritt der Generierung von fernem SQL eine effiziente SQL-Anweisung, die auf der SQL-Version der Datenquelle beruht.

  Dieser Schritt wird nur ausgeführt, wenn Abfragen für zusammengeschlossene Datenbanken ausgeführt werden. Weitere Informationen finden Sie unter „Generierung von fernem SQL und globale Optimierung" auf Seite 244.

- **„Ausführbaren" Code generieren**

  Im letzten Schritt des SQL-Compilers werden der *Zugriffsplan* und das Abfragediagrammodell verwendet, um einen ausführbaren Zugriffsplan oder Abschnitt für die Abfrage zu erstellen. Bei dieser Generierung des Codes werden Informationen des Abfragediagrammodells verwendet, um eine Wiederholung der Ausführung von Ausdrücken zu vermeiden, die für eine Abfrage nur einmal berechnet werden müssen. Diese Art der Optimierung ist beispielsweise für Umwandlungen von Codepages und die Verwendung von Host-Variablen möglich.

  Informationen über die Zugriffspläne für statisches SQL werden in den Systemkatalogtabellen gespeichert. Wenn das Paket ausgeführt wird, verwendet der Datenbankmanager die in den Systemkatalogtabellen gespeicherten Informationen, um festzulegen, wie auf die Daten zugegriffen werden soll und die Ergebnisse für eine Abfrage zu erstellen sind. Eben diese Informationen werden vom Programm *db2expln* verwendet. (In „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251 finden Sie weitere Informationen zu EXPLAIN.)

Es ist empfehlenswert, den Befehl RUNSTATS in regelmäßigen Abständen für Tabellen auszuführen, die in Abfragen verwendet werden und für die eine gute Leistung erwünscht ist. Das Optimierungsprogramm ist in diesem Fall besser mit relevanten statistischen Informationen über die Art der Daten ausgerüstet. Als Voraussetzung für die Nutzung der neuen Statistikdaten müssen Sie Ihre Anwendung außerdem erneut binden. Wenn der Befehl RUNSTATS nicht ausgeführt wurde (oder das Optimierungsprogramm annimmt, dass RUNSTATS für leere oder fast leere Tabellen ausgeführt wurde), kann das Optimierungsprogramm entweder Standardwerte verwenden oder versuchen, bestimmte Statistikdaten mit Hilfe der Anzahl der Dateiseiten (FPAGES), die zum Speichern der Tabelle auf Platte verwendet werden, abzuleiten.

Umschreiben der Abfrage durch den SQL-Compiler

Der SQL-Compiler umfasst eine Phase des Umschreibens der Abfrage, die SQL-Anweisungen in eine Form umsetzt, die leichter optimiert werden kann, so dass ein besserer Zugriffsplan ausgewählt werden kann. Das Umschreiben von Abfragen ist besonders wichtig bei Abfragen, die sehr komplex sind, d. h. auch bei solchen Abfragen, die viele Unterabfragen und Verknüpfungen enthalten. Tools zur Generierung von Abfragen erstellen häufig diese sehr komplexen Arten der Abfrage.

Sie können die Anzahl der Regeln für das Umschreiben von Abfragen, die auf eine SQL-Anweisung angewendet werden, beeinflussen, indem Sie die Optimierungsklasse ändern (siehe „Anpassen der Optimierungsklasse" auf Seite 76).

Einige Ergebnisse dieses Umschreibens der Abfrage können mit Hilfe der EXPLAIN-Einrichtung abgefragt oder von Visual Explain angezeigt werden.

Es gibt drei Hauptkategorien des Umschreibens, die der SQL-Compiler durchführen kann:

- Mischen von Operationen
- Verschieben von Operationen
- Übersetzen von Vergleichselementen.

Mischen von Operationen

Der SQL-Compiler schreibt Abfragen um, um Abfrageoperationen so zu mischen, dass nach Möglichkeit die Abfrage die geringste Anzahl von Operationen, insbesondere SELECT-Operationen, enthält. Die folgenden Beispiele zeigen einige der Operationen, die vom SQL-Compiler gemischt werden können:

- Beispiel - Mischen von Sichten

  Die Verwendung von Sichten in einer Anweisung SELECT kann zu Einschränkungen in der Verknüpfungsfolge der Tabelle führen und außerdem überflüssige Verknüpfungen von Tabellen nach sich ziehen. Durch das Mischen der Sichten während des Umschreibens können diese Einschränkungen aufgehoben werden.

- Beispiel - Umsetzungen von Unterabfragen in Verknüpfungen

  Wenn das Optimierungsprogramm eine Unterabfrage in einer SELECT-Anweisung findet, kann seine Auswahl der Verarbeitungsreihenfolge der Tabellen eingeschränkt werden.

- Beispiel - Eliminierung überflüssiger Verknüpfungen

  Während des Umschreibens der Abfrage können überflüssige Verknüpfungen entfernt werden, um die Anweisung SELECT, die optimiert wird, noch weiter zu vereinfachen.

- Beispiel - Gemeinsam benutzte Spaltenberechnung

  Bei Verwendung verschiedener Funktionen kann durch Umschreiben der Abfrage die Zahl der erforderlichen Berechnungen reduziert werden.

Beispiel - Mischen von Sichten

Nehmen Sie zum Beispiel an, Sie haben Zugriff auf die beiden folgenden Sichten der Tabelle EMPLOYEE, von denen die eine die Mitarbeiter mit einer hohen Ausbildungsstufe (EDLEVEL) und die andere die Mitarbeiter mit einem Gehalt (SALARY) über 35.000 Dollar zeigt:

```
CREATE VIEW EMP_EDUCATION (EMPNO, FIRSTNME, LASTNAME, EDLEVEL) AS
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL
     FROM EMPLOYEE
 WHERE EDLEVEL > 17
CREATE VIEW EMP_SALARIES (EMPNO, FIRSTNAME, LASTNAME, SALARY) AS
SELECT EMPNO, FIRSTNME, LASTNAME, SALARY
     FROM EMPLOYEE
 WHERE SALARY > 35000
```

Jetzt wird beispielsweise die folgende Abfrage ausgeführt, um die Mitarbeiter, die eine hohe Ausbildungsstufe haben und deren Gehalt über 35.000 Dollar liegt, aufzulisten:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMP_EDUCATION E1,
       EMP_SALARIES  E2
 WHERE E1.EMPNO = E2.EMPNO
```

Während der Phase des Umschreibens könnten diese beiden Sichten gemischt werden, um folgende Abfrage zu erstellen:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

Durch das Mischen der Anweisungen SELECT der beiden Sichten mit der vom Benutzer geschriebenen Anweisung SELECT kann das Optimierungsprogramm mehr Möglichkeiten bei der Auswahl des Zugriffsplans in Betracht ziehen. Darüber hinaus kann die Abfrage noch weiter umgeschrieben werden, wenn die beiden gemischten Sichten dieselbe Basistabelle verwenden. Eine Beschreibung dieses weiteren Umschreibens finden Sie in „Beispiel - Eliminierung überflüssiger Verknüpfungen” auf Seite 178.

Beispiel - Umsetzungen von Unterabfragen in Verknüpfungen

Nehmen Sie an, eine Abfrage enthält die folgende Unterabfrage:

```
SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO
     FROM EMPLOYEE
WHERE WORKDEPT IN
      (SELECT DEPTNO
         FROM DEPARTMENT
        WHERE DEPTNAME = 'OPERATIONS')
```

Diese Unterabfrage wird vom SQL-Compiler in eine Verknüpfungsabfrage der folgenden Form umgewandelt:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME, PHONENO
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNAME = 'OPERATIONS'
```

Im allgemeinen ist eine Verknüpfung in der Ausführung wesentlich effektiver als eine Unterabfrage.

Beispiel - Eliminierung überflüssiger Verknüpfungen

Manche geschriebenen oder generierten Abfragen enthalten unnötige Verknüpfungen. Abfragen wie die folgende könnten auch während des Umschreibens einer Abfrage, wie in „Beispiel - Mischen von Sichten" auf Seite 177 beschrieben, generiert werden.

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

In dieser Abfrage kann der SQL-Compiler die Verknüpfung eliminieren und die Abfrage auf folgende Form vereinfachen:

```
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL, SALARY
     FROM EMPLOYEE
 WHERE EDLEVEL > 17
   AND SALARY  > 35000
```

Im folgenden Beispiel wird davon ausgegangen, dass für die Kostenstelle (WORKDEPT/DEPNO) der Beispieltabellen EMPLOYEE und DEPARTMENT eine referenzielle Integritätsbedingung vorhanden ist. Zuerst wird eine Sicht erstellt.

```
CREATE VIEW PEPLVIEW
   AS SELECT FIRSTNME, LASTNAME, SALARY, DEPTNO, DEPTNAME, MGRNO
         FROM EMPLOYEE E DEPARTMENT D
         WHERE E.WORKDEPT = D.DEPTNO
```

Eine Abfrage wie die folgende:

```
SELECT LASTNAME, SALARY
   FROM PEPLVIEW
```

wird dann geändert in:

```
SELECT LASTNAME, SALARY
     FROM EMPLOYEE
   WHERE WORKDEPT NOT NULL
```

Beachten Sie bei dieser Situation, dass der Benutzer die Abfrage eventuell nicht umschreiben kann, weil er keinen Zugriff auf die zugrundeliegenden Tabellen hat. Er hat eventuell nur Zugriff auf die Sicht (siehe oben). Daher muss diese Art von Optimierung im Datenbankmanager ausgeführt werden.

Redundanz in Verknüpfungen mit referenzieller Integrität ist in folgenden Fällen wahrscheinlich:

- Sichten sind mit Verknüpfungen definiert.
- Abfragen werden automatisch generiert.

  Es gibt zum Beispiel automatische Tools in Abfrage-Managern, die das Schreiben optimierter Abfragen durch Benutzer verhindern.

Beispiel - Gemeinsam benutzte Spaltenberechnung

Bei Verwendung mehrerer Funktionen in einer Abfrage können zahlreiche Berechnungen entstehen, die zeitintensiv sind. Durch Reduzieren der für die Abfrage erforderlichen Anzahl von Berechnungen kann der Zugriffsplan verbessert werden. Eine Abfrage, die mehrere Funktionen verwendet, wie z. B.:

```
SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  AVG(SALARY+BONUS+COMM) AS OAVG,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE;
```

wird vom SQL-Compiler wie folgt umgewandelt:

```
SELECT OSUM,
  OSUM/OCOUNT
  OCOUNT
FROM (SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE) AS SHARED_AGG;
```

Durch dieses Umschreiben benötigt die Abfrage statt 2 Summen und 2 Zählern nur noch 1 Summe und 1 Zähler.

Verschieben von Operationen

Der SQL-Compiler schreibt Abfragen um, um Abfrageoperationen zu verschieben und die Abfrage dadurch so umzustrukturieren, dass die Anzahl der Operationen und Vergleichselemente minimiert wird. Die folgenden Beispiele zeigen einige der Operationen, die vom SQL-Compiler verschoben werden können:

- Beispiel - Eliminieren von Klauseln DISTINCT

  Während des Umschreibens kann das Optimierungsprogramm die Operation DISTINCT dahin verschieben, wo sie am günstigsten ausgeführt werden kann. In dem gezeigten Beispiel wird die Operation DISTINCT vollständig entfernt.

- Beispiel - Allgemeines Verschieben von Vergleichselementen

  Während des Umschreibens kann die Reihenfolge der Vergleichselemente, die angewendet werden, geändert werden, so dass die Vergleichselemente, die die Auswahl in größerem Maße einschränken, zum frühstmöglichen Zeitpunkt berücksichtigt werden.

- Beispiel - Dekorrelierung

  In einer Umgebung mit partitionierten Datenbanken erfordert das Verschieben von Ergebnismengen zwischen den Datenbankpartitionen hohen Aufwand. Den Umfang des Rundsendebetriebs an andere Datenbankpartitionen und/oder die Anzahl der Rundsendevorgänge zu reduzieren ist eines der Ziele beim Umschreiben von Abfragen.

Beispiel - Eliminieren von Klauseln DISTINCT

Für das folgende Abfragebeispiel wird angenommen, dass die Spalte EMPNO als Primärschlüssel der Tabelle EMPLOYEE definiert wurde:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME
     FROM EMPLOYEE
```

Diese Abfrage würde so umgeschrieben, dass die Klausel DISTINCT entfernt wird:

```
SELECT EMPNO, FIRSTNME, LASTNAME
     FROM EMPLOYEE
```

Da hier der Primärschlüssel ausgewählt wird, weiß der SQL-Compiler, dass jede zurückgelieferte Zeile bereits eindeutig ist. In diesem Fall wird das Schlüsselwort DISTINCT nicht benötigt. Wenn die Abfrage nicht umgeschrieben wird, erstellt das Optimierungsprogramm einen Plan, der die nötigen Verarbeitungsschritte (z. B. einen Sortiervorgang) enthält, um sicherzustellen, dass die Spalten eindeutig sind.

Beispiel - Allgemeines Verschieben von Vergleichselementen

Durch das Ändern der Ebene, auf der Vergleichselemente normalerweise angewendet werden, lässt sich eventuell eine Leistungsverbesserung erreichen. Zum Beispiel sei die folgende Sicht gegeben, die eine Liste aller Mitarbeiter der Abteilung „D11" zusammenstellt:

```
CREATE VIEW D11_EMPLOYEE
 (EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM)
AS SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM
     FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
```

Es wird nun die folgende Abfrage aufgesetzt:

```
SELECT FIRSTNME, PHONENO
  FROM D11_EMPLOYEE
 WHERE LASTNAME = 'BROWN'
```

In der Phase des Umschreibens verschiebt der Compiler das Vergleichselement LASTNAME = 'BROWN' nach unten in die Sicht D11_EMPLOYEE. Dadurch kann das Vergleichselement früher und möglicherweise effektiver angewendet werden. Die tatsächliche Abfrage, die in diesem Beispiel zur Ausführung kommen könnte, lautet folgendermaßen:

```
SELECT FIRSTNME, PHONENO
     FROM EMPLOYEE
 WHERE LASTNAME = 'BROWN'
   AND WORKDEPT = 'D11'
```

Das Verschieben von Vergleichselementen ist nicht auf Sichten beschränkt. Beispiele anderer Situationen, in denen Vergleichselemente verschoben werden können, sind Klauseln UNION, GROUP BY und abgeleitete Tabellen (verschachtelte Tabellenausdrücke oder allgemeine Tabellenausdrücke).

Beispiel - Dekorrelierung

In einer Umgebung mit partitionierten Datenbanken kann der SQL-Compiler die folgende Abfrage umschreiben:

Lokalisieren aller Mitarbeiter, die in der Programmierung arbeiten und unterbezahlt sind.

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
  FROM EMPLOYEE E, PROJECT P
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND E.SALARY+E.BONUS+E.COMM <
     (SELECT AVG(E1.SALARY+E1.BONUS+E1.COMM)
        FROM EMPLOYEE E1, PROJECT P1
        WHERE P1.PROJNAME LIKE '%PROGRAMMING%'
          AND P1.PROJNO = A.PROJNO
          AND E1.EMPNO = P1.EMPNO)
```

Da die Abfrage korreliert ist und wahrscheinlich weder PROJECT noch EMPLOYEE über die Spalte PROJNO partitioniert sind, wird möglicherweise jedes Projekt im Rundsendebetrieb an jede Datenbankpartition übermittelt. Außerdem müsste die Unterabfrage viele Male ausgewertet werden.

Der SQL-Compiler kann die Abfrage wie folgt umschreiben:

- Die explizite Liste der Mitarbeiter (Employees) ermitteln, die an Programmierungsprojekten arbeiten, und diese DIST_PROJS nennen. Diese Liste muss explizit sein, damit sichergestellt wird, dass die Spaltenberechnung nur einmal pro Projekt erfolgt:

```
WITH DIST_PROJS(PROJNO, EMPNO) AS
(SELECT DISTINCT PROJNO, EMPNO
 FROM PROJECT P1
 WHERE P1.PROJNAME LIKE '%PROGRAMMING%')
```

- Die explizite Liste der Mitarbeiter, die an Programmierungsprojekten arbeiten mit der Mitarbeitertabelle verknüpfen, um die durchschnittliche Entlohnung je Projekt (AVG_PER_PROJ) zu ermitteln:

```
AVG_PER_PROJ(PROJNO, AVG_COMP) AS
(SELECT P2.PROJNO, AVG(E1.SALARY+E1.BONUS+E1.COMM)
 FROM EMPLOYEE E1, DIST_PROJS P2
 WHERE E1.EMPNO = P2.EMPNO
 GROUP BY P2.PROJNO)
```

- Die neue Abfrage würde wie folgt lauten:

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
 FROM PROJECT P, EMPLOYEE E, AVG_PER_PROG A
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND P.PROJNO = A.PROJNO
  AND E.SALARY+E.BONUS+E.COMM < A.AVG_COMP
```

Die umgeschriebene SQL-Abfrage berechnet AVG_COMP je Projekt (AVG_PRE-_PROJ) und kann das Ergebnis an alle Datenbankpartitionen übermitteln, in denen die Tabelle EMPLOYEE enthalten ist.

Übersetzen von Vergleichselementen

Der SQL-Compiler schreibt Abfragen um, um vorhandene Vergleichselemente für eine bestimmte Abfrage in eine optimierte Form zu bringen. Die folgenden Beispiele zeigen einige der Vergleichselemente, die vom SQL-Compiler übersetzt werden können:

- Beispiel - Hinzufügen implizierter Vergleichselemente

  Während des Umschreibens können Vergleichselemente der Abfrage hinzugefügt werden, um dem Optimierungsprogramm die Möglichkeit zu geben, weitere Tabellenverknüpfungen bei der Auswahl des günstigsten Zugriffsplans für die Abfrage in Betracht zu ziehen.

- Beispiel - Transformationen von OR zu IN

  Während des Umschreibens kann ein Vergleichselement OR in ein Vergleichselement IN übersetzt werden, um die Auswahl eines effektiveren Zugriffsplans zu ermöglichen. Der SQL-Compiler kann auch ein Vergleichselement IN in ein Vergleichselement OR übersetzen, wenn diese Transformation zur Auswahl eines günstigeren Zugriffsplans führen würde.

Beispiel - Hinzufügen implizierter Vergleichselemente

Die folgende Abfrage erstellt eine Liste der Manager, deren Abteilung an Abteilung „E01" berichten, und der Projekte, für die die Manager verantwortlich sind:

```
SELECT DEPT.DEPTNAME DEPT.MGRNO, EMP.LASTNAME, PROJ.PROJNAME
  FROM DEPARTMENT DEPT,
       EMPLOYEE   EMP,
       PROJECT    PROJ
 WHERE DEPT.ADMRDEPT = 'E01'
   AND DEPT.MGRNO = EMP.EMPNO
   AND EMP.EMPNO  = PROJ.RESPEMP
```

Beim Umschreiben der Abfrage wird das folgende implizierte Vergleichselement hinzugefügt:

```
DEPT.MGRNO = PROJ.RESPEMP
```

Als Ergebnis dieses Umschreibens kann das Optimierungsprogramm weitere Verknüpfungen in die Berechnung mit einbeziehen, wenn es versucht, den günstigsten Zugriffsplan für die Abfrage auszuwählen.

Neben der oben gezeigten Ausnutzung der Transitivität werden durch das Umschreiben auch zusätzliche lokale Vergleichselemente aufgrund der durch Gleichheitsvergleichselemente implizierten Transitivität abgeleitet. Zum Beispiel stellt die folgende Abfrage eine Liste der Namen der Abteilungen (deren Abteilungsnummer größer als „E00" ist) und der Mitarbeiter, die in diesen Abteilungen arbeiten, zusammen.

```
SELECT EMPNO, LASTNAME, FIRSTNAME, DEPTNO, DEPTNAME
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNO > 'E00'
```

Dieser Abfrage wird in der Phase des Umschreibens das folgende implizierte Vergleichselement hinzugefügt:

```
EMP.WORKDEPT > 'E00'
```

Das Ergebnis dieses Umschreibens besteht darin, dass das Optimierungsprogramm die Anzahl der Zeilen, die verknüpft werden müssen, verringern kann.

Beispiel - Transformationen von OR zu IN

Nehmen Sie an, eine Klausel OR verknüpft zwei oder mehr einfache Gleichheitsvergleichselemente für dieselbe Spalte, wie im folgenden Beispiel:

```
SELECT *
   FROM EMPLOYEE
WHERE DEPTNO = 'D11'
   OR DEPTNO = 'D21'
   OR DEPTNO = 'E21'
```

Wenn es für die Spalte DEPTNO keinen Index gibt, erlaubt die Umwandlung der Klausel OR in das folgende Vergleichselement IN eine effektivere Verarbeitung der Abfrage:

```
SELECT *
   FROM EMPLOYEE
WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

Anmerkung: In einigen Fällen wandelt der Datenbankmanager ein Prädikat IN in eine Gruppe von Klauseln OR um, so dass OR-Verknüpfungen für Indizes durchgeführt werden können. Weitere Informationen zu OR-Verknüpfungen von Indizes finden Sie in „Zugriff über mehrere Indizes” auf Seite 196.

Behandeln von Spaltenkorrelation

Sie haben vielleicht Anwendungen, die Abfragen enthalten, die mit Verknüpfungen konstruiert sind, die mehr als ein Verknüpfungsvergleichselement zum Verknüpfen zweier Tabellen haben. Das mag zwar kompliziert klingen, aber eine solche Situation ist nicht ungewöhnlich, wenn Sie versuchen, die Abhängigkeiten zwischen ähnlichen, in Beziehung stehenden Spalten zwischen Tabellen festzustellen.

Zum Beispiel produziert ein Hersteller Produkte aus Rohmaterial verschiedener Farben, Elastizitäten und Qualitäten. Das fertige Produkt hat dieselbe Farbe und Elastizität wie das Rohmaterial, aus dem es hergestellt ist. Der Hersteller führt die folgende Abfrage aus:

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY FROM PRODUCT, RAWMATERIAL
   WHERE PRODUCT.COLOR      =  RAWMATERIAL.COLOR
     AND PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Diese Abfrage liefert die Namen und die Qualität des Rohmaterials aller Produkte. Zwei Vergleichselemente werden für die Verknüpfung verwendet:

```
PRODUCT.COLOR       =  RAWMATERIAL.COLOR
PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

Beim Auswählen eines Plans zur Ausführung dieser Abfrage berechnet das DB2-UDB-Optimierungsprogramm die Selektivität der beiden Vergleichselemente und nimmt an, dass die Vergleichselemente unabhängig sind, d. h., dass alle Variationen von Elastizität für jede Farbe vorkommen und umgekehrt für jede Elastizitätsstufe Rohmaterial in jeder Farbe verfügbar ist. Anschließend verwendet es Statistiken über die Anzahl der in jeder Tabelle vorhandenen Elastizitätsstufen und verschiedenen Farben, um die Gesamtselektivität des Vergleichselementpaares zu berechnen. Aufgrund dieser Berechnungen kann es beispielsweise eine Verknüpfung über Verschachtelungsschleife (Nested Loop Join) einer Mischverknüpfung (Merge Join) vorziehen oder umgekehrt.

Es kann jedoch vorkommen, dass diese beiden Vergleichselemente nicht unabhängig sind. Zum Beispiel könnte es der Fall sein, dass die hochelastischen Materialien nur in wenigen Farben verfügbar sind und die sehr wenig elastischen Materialien nur in einigen anderen Farben (die sich von den elastischen unterscheiden) verfügbar sind. In diesem Fall ist die kombinierte Selektivität der Vergleichselemente geringer (eliminiert weniger Zeilen), so dass die Abfrage mehr Zeilen liefert. Betrachten Sie zum besseren Verständnis dieses Sachverhalts einmal den extremen Fall, dass es nur eine Stufe von Elastizität für jede Farbe gibt und umgekehrt. Hier könnte jedes der beiden Vergleichselemente logisch ganz weggelassen werden, da es vom jeweils anderen impliziert wird. Die Planauswahl des Optimierungsprogramms ist vielleicht nicht mehr optimal. Es ist beispielsweise möglich, dass ein Plan mit Verknüpfung über Verschachtelungsschleife ausgewählt wird, aber die Mischverknüpfung schneller wäre.

Bei anderen Datenbankprodukten versuchten Datenbankadministratoren dieses Leistungsproblem dadurch in den Griff zu bekommen, dass sie Statistiken im Katalog aktualisierten, um eines der Vergleichselemente weniger selektiv erscheinen zu lassen, jedoch kann dieses Vorgehen zu unerwünschten Nebeneffekten bei anderen Abfragen führen.

Das Optimierungsprogramm von DB2 UDB versucht, eine Korrelation von Verknüpfungsvergleichselementen zu entdecken und zu kompensieren, wenn Sie folgende Maßnahmen ergreifen:

1. Sie definieren eindeutige Indizes für die korrelierten Spalten, das heißt, für die Spalten einer Tabelle, die in korrelierten Vergleichselementen vorkommen.
2. Setzen Sie die Registrierungsvariable DB2_CORRELATED_PREDICATES nicht auf „NO" (Nein).

Im obigen Beispiel sollten Sie einen eindeutigen Index definieren, der folgende Spaltenpaare abdeckt. Entweder:

```
PRODUCT.COLOR, PRODUCT.ELASTICITY
```

oder

```
RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

oder beide.

Damit eine Korrelation erkannt werden kann, müssen die Nicht-INCLUDE-Spalten dieses Index ausschließlich korrelierte und keine anderen Spalten sein. Der Index kann wahlfrei INCLUDE-Spalten enthalten.

Im Allgemeinen kann es mehr als zwei korrelierte Spalten in Vergleichselementen zur Verknüpfung geben. Daher sollten Sie sicherstellen, dass der definierte eindeutige Index alle diese Spalten abdeckt.

In vielen Fällen bilden korrelierte Spalten innerhalb einer Tabelle den Primärschlüssel der Tabelle. Ein Primärschlüssel ist immer eindeutig, so dass bei Vorhandensein eines Primärschlüssels über die korrelierten Spalten kein weiterer eindeutiger Index mehr definiert werden muss.

Stellen Sie anschließend sicher, dass die Tabellenstatistiken aktuell sind und nicht aus irgendeinem Grund (z. B., um das Optimierungsprogramm zu beeinflussen) von den wahren Werten abweichend geändert wurden.

Das Optimierungsprogramm verwendet die FIRSTnKEYCARD- und FULLKEYCARD-Informationen der Statistikdaten für eindeutige Indizes, um Fälle von Korrelation zu erkennen und die errechneten kombinierten Selektivitäten der korrelierten Vergleichselemente dynamisch anzupassen, um so eine realistischere Schätzung der Größe und des Aufwands der Verknüpfung zu erhalten.

Zusätzlich zur JOIN-Vergleichselementkorrelation ist das Optimierungsprogramm auch für die Korrelation mit einfachen gleichen Vergleichselementen der Art COL = „constant" (konstant) verantwortlich. Stellen Sie sich beispielsweise eine Tabelle mit verschiedenen Arten von Fahrzeugen vor, die alle über die Spalten HERSTELLER, MODELL, AUSFÜHRUNG (d. h. Limousine, Kombi, Sport- oder Nutzfahrzeug), JAHR und FARBE verfügen. Vergleichselemente in der Spalte FARBE sind wahrscheinlich von denen in den Spalten HERSTELLER, MODELL, AUSFÜHRUNG oder JAHR unabhängig, da die meisten Hersteller dieselben Standardfarben für alle ihre Modelle und Ausführungen in jedem Jahr verfügbar machen.

Die Vergleichselemente HERSTELLER und MODELL sind jedoch in keinem Fall unabhängig voneinander, da nur jeweils ein Fahrzeughersteller ein Modell mit einem bestimmten Namen produziert. Identische Modellnamen, die von zwei oder mehr Fahrzeugherstellern verwendet werden, sind sehr unwahrscheinlich und mit Sicherheit von den Fahrzeugherstellern nicht gewollt. Wenn für die beiden Spalten HERSTELLER und MODELL ein Index vorhanden ist, verwendet das Optimierungsprogramm die Statistikdaten aus diesem Index, um die kombinierte Zahl an unterschiedlichen Werten festzustellen und die Selektivitäts- oder Kardinalitätsschätzung für die Korrelation zwischen diesen beiden Spalten anzupassen. Für Vergleichselemente, die keine Verknüpfungsvergleichselemente sind, muss kein eindeutigen Index vorhanden sein, damit das Optimierungsprogramm Anpassung vornehmen kann.

Datenzugriffskonzepte und Optimierung

Bei der Kompilierung einer SQL-Anweisung, schätzt das Optimierungsprogramm den Ausführungsaufwand der verschiedenen Methoden ab, die die Anforderung erfüllen würden. Auf der Grundlage dieser Abschätzung wählt das Optimierungsprogramm den Zugriffsplan aus, den es für optimal hält. Ein *Zugriffsplan* gibt die Reihenfolge von Operationen an, die erforderlich sind, um eine SQL-Anweisung auszuführen. Wenn ein Anwendungsprogramm gebunden wird, wird ein *Paket* erstellt. Dieses Paket enthält Zugriffspläne für alle statischen SQL-Anweisungen in dem entsprechenden Anwendungsprogramm. Die Zugriffspläne für dynamische SQL-Anweisungen werden zum Zeitpunkt der Ausführung der Anwendung erstellt.

Es gibt zwei Möglichkeiten, auf Daten in einer Tabelle zuzugreifen: durch direktes Lesen der Tabelle (Tabellensuche) oder über den Zugriff auf einen Index für die Tabelle (Indexsuche).

Eine *Tabellensuche* ist der Prozess, bei dem der Datenbankmanager sequenziell auf jede Zeile einer Tabelle zugreift. Lesen Sie im Abschnitt „Konzepte der Indexsuche" auf Seite 188 die Informationen zur Funktionsweise von Indexsuchen und in „Tabellensuche und Indexsuche" auf Seite 200 die Informationen, unter welchen Bedingungen jeweils die eine oder andere Art der Suche verwendet wird.

Die folgenden Themen behandeln andere Methoden, die ebenfalls in einem Zugriffsplan zum Zugriff auf Daten in einer Tabelle und zur Lieferung von Ergebnissen für die Abfrage verwendet werden können:

- „Vergleichselementterminologie" auf Seite 200
- „Verknüpfungskonzepte" auf Seite 203
- „Verknüpfungsstrategien in einer partitionierten Datenbank" auf Seite 217
- „Einfluss des Sortierens auf das Optimierungsprogramm" auf Seite 225.

Weitere zugehörige Abschnitte:

- Der Abschnitt „Anpassen der Optimierungsklasse" auf Seite 76 enthält Informationen zur Steuerung der Anzahl alternativer Zugriffspläne, die vom SQL-Compiler ausgewertet werden.
- „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251, enthält Informationen, wie Sie Informationen über den vom SQL-Compiler ausgewählten Zugriffplan erhalten können.

Konzepte der Indexsuche

Als *Indexsuche* wird der Vorgang bezeichnet, bei dem der Datenbankmanager zu folgenden Zwecken auf einen Index zugreift:

- Eingrenzen der Menge der den Kriterien entsprechenden Zeilen (durch Durchsuchen eines bestimmten Bereichs des Index) vor dem Zugriff auf die Basistabelle. Der *Suchbereich* des Index (der Start- und der Stoppunkt der Suche) wird durch die Werte in der Abfrage festgelegt, mit denen Indexspalten verglichen werden.
- Sortieren der Ausgabe
- Vollständiges Abrufen der angeforderten Daten. Wenn sich alle angeforderten Daten im Index befinden, findet kein Zugriff auf die Basistabelle statt. Dies wird als *reiner Indexzugriff* bezeichnet.

Suchoperationen können in Indizes auch in der entgegengesetzten Richtung zu der Richtung durchgeführt werden, mit der sie definiert wurden. Weitere Informationen finden Sie in der Beschreibung der Option ALLOW REVERSE SCANS in der Anweisung CREATE INDEX im Handbuch *SQL Reference*.

Die folgenden weiteren Themen werden behandelt:

- Indexstruktur
- Indexsuchen zur Begrenzung von Bereichen
- Indexsuchen zur Sortierung von Daten
- Reiner Indexzugriff
- Zugriff über mehrere Indizes
- Geclusterte Indizes
- Vorablesezugriff auf Indexseiten.

Indexstruktur

Der Datenbankmanager verwendet eine B+-Baumstruktur zur Speicherung der Indizes. Eine B+-Baumstruktur hat eine oder mehrere Ebenen, wie die folgende Abbildung zeigt (`rid` steht für Satz/Zeilen-ID):

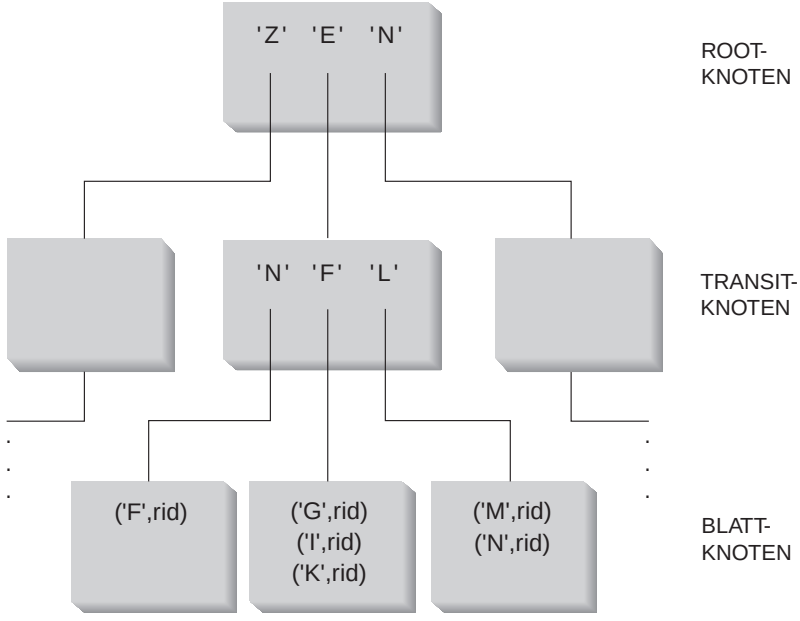


Abbildung 13. B+-Baumstruktur

Die oberste Ebene wird als *Wurzelknoten* bezeichnet. Die unterste Ebene besteht aus *Blattknoten*, in denen die tatsächlichen Werte des Indexschlüssels und ein Zeiger auf die tatsächliche Zeile in der Tabelle gespeichert sind. Die Ebenen zwischen dem Wurzelknoten und dem Blattknoten werden als *innere Knoten* bezeichnet.

Bei der Suche nach einem bestimmten Indexschlüsselwert durchsucht der Indexmanager den Indexbaum ausgehend vom Wurzelknoten. Der Wurzelknoten enthält einen Schlüssel für jeden Knoten auf der folgenden Ebene. Der Wert jedes dieser Schlüssel ist jeweils der größte vorhandene Schlüsselwert für den entsprechenden Knoten auf der nächsten Ebene.

Wenn zum Beispiel ein Index drei Ebenen hat, wie in Abb. 13 auf Seite 189 gezeigt, dann durchsucht der Indexmanager zum Auffinden eines Indexschlüsselwerts den Wurzelknoten nach dem ersten Schlüsselwert, der größer oder gleich dem gesuchten Schlüsselwert ist. Dieser Wurzelknotenschlüssel enthält dann einen Zeiger auf einen bestimmten inneren Knoten. Dieselbe Prozedur wird dann für diesen inneren Knoten durchgeführt, um festzustellen, in welchem Blattknoten die Suche fortzusetzen ist. Der endgültige Indexschlüssel wird dann im Blattknoten gefunden. In Abb. 13 auf Seite 189 wird zum Beispiel nach „I" gesucht. Der erste Schlüssel im Wurzelknoten, der größer oder gleich „I" ist, ist „N". Dieser Wert zeigt auf den mittleren Knoten auf der nächsten Ebene. Der erste Schlüssel in diesem inneren Knoten, der größer oder gleich „I" ist, ist „L". Dieser Wert zeigt auf einen bestimmten Blattknoten, in dem der Indexschlüssel für „I" zusammen mit den entsprechenden Zeilen-IDs gefunden wird (die Zeilen-ID der entsprechenden Zeilen in der Basistabelle).

Anmerkung: Auf der Blattknotenebene kann es frühere Blattzeiger geben. Dies kann sehr vorteilhaft sein, da der Indexmanager, wenn er einmal einen bestimmten Schlüsselwert im Index beim Durchlaufen der Baumstruktur gefunden hat, die Blattknoten in beide Richtungen durchsuchen kann, um einen Bereich von Werten abzurufen. Diese Möglichkeit, den Index in beide Richtungen zu durchsuchen, ist nur gegeben, wenn der Index mit dem Parameter ALLOW REVERSE SCANS erstellt wurde.

Weitere Informationen finden Sie in der Beschreibung der Optionen für die Anweisung CREATE INDEX im Handbuch *SQL Reference*.

Indexsuchen zur Begrenzung von Bereichen

Bei der Bestimmung, ob ein Index für eine bestimmte Abfrage verwendet werden kann, wertet das Optimierungsprogramm jede Spalte des Index beginnend bei der ersten Spalte aus, um zu überprüfen, ob sie zur Erfüllung der folgenden Vergleichselemente verwendet werden kann:

- Eines der Gleichheitsvergleichselemente in der Klausel WHERE der Anweisung
- Eines der anderen Vergleichselemente in der Klausel WHERE

Ein *Vergleichselement* ist ein Element einer Suchbedingung in einer Klausel WHERE, das eine Vergleichsoperation definiert oder impliziert. Vergleichselemente, die zur Eingrenzung des Bereichs einer Indexsuche verwendet werden können, sind solche, die eine Indexspalte betreffen, für die eine der folgenden Bedingungen gilt:

- Die Indexspalte wird auf Gleichheit mit einer Konstanten, einer Host-Variablen, einem Ausdruck, aus dem sich ein konstanter Wert errechnet, oder einem Schlüsselwort getestet.
- Der Test der Indexspalte ergibt den Wert „IS NULL" (gleich null) oder „IS NOT NULL" (nicht gleich null).
- Der Test prüft auf Gleichheit mit einer einfachen Unterabfrage (d. h. einer, die kein ANY, ALL oder SOME enthält), und die Unterabfrage hat keinen Verweis über eine korrelierte Spalte auf den unmittelbar übergeordneten Abfrageblock (d. h. auf die Anweisung SELECT, für die diese Unterabfrage eine Unterauswahl ist).
- Der Test besteht aus einem Ungleichheitsvergleichselement, das die unten beschriebenen Bedingungen erfüllt.

Zum Beispiel sei ein Index mit der folgenden Definition gegeben:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

Die folgenden Vergleichselemente könnten verwendet werden, um den Bereich der Indexsuche in Index IX1 einzugrenzen:

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
```

oder

```
WHERE  MGR   = :hv1
 AND  NAME   = :hv2
 AND  DEPT   = :hv3
```

Beachten Sie, dass im zweiten Beispiel die Vergleichselemente nach WHERE nicht in derselben Reihenfolge angegeben werden müssen, in der die Schlüsselspalten im Index erscheinen. In den Beispielen werden Host-Variablen (hv = Host-Variable) verwendet, jedoch hätten Parametermarken, Ausdrücke und Konstanten dieselbe Wirkung.

Ein einzelner Index, der mit dem Parameter ALLOW REVERSE SCANS in der Anweisung CREATE INDEX erstellt wurde, kann vorwärts und rückwärts durchsucht werden. Das heißt, solche Indizes unterstützen Suchoperationen in der Richtung, die bei der Erstellung des Index definiert wurde, und Suchoperationen in entgegengesetzter oder umgekehrter Richtung. Die Anweisung könnte ungefähr wie folgt aussehen:

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

In diesem Fall wird der Index (`iname`) nach den absteigenden (DESCending) Werten der Spalte `cname` gebildet. Durch Zulassen von umgekehrten Suchoperationen kann eine Suche in aufsteigender Folge durchgeführt werden, obwohl der Index für die Spalte für Suchoperationen in absteigender Folge definiert ist. Die tatsächliche Verwendung des Index in beiden Richtungen wird nicht von Ihnen, sondern vom Optimierungsprogramm bei der Erstellung und Auswahl von Zugriffsplänen gesteuert.

In der folgenden Klausel WHERE werden nur die Vergleichselemente für NAME und DEPT verwendet, um den Bereich der Indexsuche einzugrenzen, jedoch nicht die Vergleichselemente für SALARY und YEARS:

```
WHERE  NAME   = :hv1
 AND   DEPT   = :hv2
 AND   SALARY = :hv4
 AND   YEARS  = :hv5
```

Der Grund dafür ist der, dass es eine Schlüsselspalte (MGR) gibt, die diese Spalten von den ersten beiden Indexschlüsselspalten trennt, so dass die Reihenfolge nicht gewährleistet wäre. Wenn aber der Bereich einmal durch die Vergleichselemente NAME = :hv1 und DEPT = :hv2 festgelegt ist, können die verbleibenden Vergleichselemente an den verbleibenden Indexschlüsselspalten ausgewertet werden.

Neben den beschriebenen Gleichheitsvergleichselementen können auch einige Ungleichheitsvergleichselemente zur Eingrenzung des Bereichs einer Indexsuche verwendet werden. Im folgenden werden zwei Arten von Ungleichheitsvergleichselemente behandelt: strenge Ungleichheit und einschließende Ungleichheit.

Vergleichselement strenger Ungleichheit: Die Operatoren der strengen (ausschließenden) Ungleichheit, die für bereichsbegrenzende Vergleichselemente verwendet werden können, sind > und <.

Zur Eingrenzung eines Bereichs für eine Indexsuche wird nur eine Spalte mit Vergleichselementen strenger Ungleichheit berücksichtigt. Im folgenden Beispiel können die Vergleichselemente für die Spalten NAME und DEPT verwendet werden, um den Bereich einzugrenzen, aber das Vergleichselement für die Spalte MGR nicht.

```
WHERE  NAME   = :hv1
  AND  DEPT   > :hv2
  AND  DEPT   < :hv3
  AND  MGR    < :hv4
```

Vergleichselemente einschließender Ungleichheit: Die folgenden Operatoren einschließender Ungleichheit können in Vergleichselementen zur Eingrenzung von Bereichen verwendet werden:

- >= und <=
- BETWEEN
- LIKE

Zur Eingrenzung eines Bereichs für eine Indexsuche werden mehrere Spalten mit Vergleichselementen einschließender Ungleichheit berücksichtigt. Im folgenden Beispiel können alle Vergleichselemente zur Eingrenzung des Bereichs der Indexsuche verwendet werden:

```
WHERE  NAME  =  :hv1
  AND  DEPT  >= :hv2
  AND  DEPT  <= :hv3
  AND  MGR   <= :hv4
```

Zur weiteren Verdeutlichung dieses Beispiels seien die folgenden Werte angenommen: `:hv2 = 404`, `:hv3 = 406` und `:hv4 = 12345`. Der Datenbankmanager durchsucht den Index für die Abteilungen 404 und 405 ganz, bricht aber die Suche in Abteilung 406 ab, wenn er auf den ersten Manager trifft, der eine Personalnummer (Spalte MGR) über dem Wert 12345 hat.

Weitere Informationen finden Sie in „Bereichsbegrenzende und bei Indexsuchen als Suchargument verwendbare Vergleichselemente” auf Seite 201.

Indexsuchen zur Sortierung von Daten

Wenn die Abfrage eine Sortierung erfordert, kann ein Index zum Sortieren der Daten verwendet werden, wenn die ordnenden Spalten nacheinander, angefangen bei der ersten Indexschlüsselspalte, im Index auftreten. (Ordnen oder Sortieren kann das Ergebnis solcher Operationen wie ORDER BY, DISTINCT, GROUP BY, Unterabfrage mit „= ANY", Unterabfrage mit „> ALL", Unterabfrage mit „< ALL", INTERSECT bzw. EXCEPT sowie UNION sein.) Eine Ausnahme ist der Fall, wenn die Indexschlüsselspalten auf Gleichheit mit „konstanten Werten" überprüft werden (d. h., mit einem Ausdruck, aus dem sich eine Konstante errechnet). In diesem Fall kann die ordnende Spalte eine andere als die ersten Indexschlüsselspalten sein. Betrachten Sie zum Beispiel folgende Abfrage:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

Der Index könnte verwendet werden, um die Zeilen zu sortieren, da die Spalten NAME und DEPT immer dieselben Werte haben und so geordnet werden. Anders ausgedrückt, die gezeigten Klauseln WHERE und ORDER BY sind äquivalent mit folgenden Klauseln:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY NAME, DEPT, MGR
```

Ein eindeutiger Index kann auch zum Verkürzen einer Sortieranforderung verwendet werden. Betrachten Sie zum Beispiel die folgende Indexdefinition und Klausel ORDER BY:

```
UNIQUE INDEX IX0:  PROJNO  ASC
SELECT PROJNO, PROJNAME, DEPTNO
   FROM PROJECT
ORDER BY PROJNO, PROJNAME
```

Eine zusätzliche Sortierung anhand der Spalte PROJNAME ist nicht erforderlich, da der Index IX0 bereits sicherstellt, dass die Werte der Spalte PROJNO eindeutig sind. Diese Eindeutigkeit sorgt dafür, dass es nur einen Wert für PROJNAME für jeden Wert von PROJNO gibt.

Reiner Indexzugriff

In einigen Fällen können alle angeforderten Daten aus dem Index abgerufen werden, ohne auf die Tabelle zuzugreifen. Dies wird als *reiner Indexzugriff* bezeichnet.

Betrachten Sie die folgende Indexdefinition zur Illustration des reinen Indexzugriffs:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

In diesem Fall kann die folgende Abfrage nur durch den Zugriff auf den Index ohne Lesen der Basistabelle erfüllt werden:

```
SELECT NAME, DEPT, MGR, SALARY
     FROM   EMPLOYEE
  WHERE NAME = 'SMITH'
```

In anderen Fällen kann es Spalten geben, die nicht im Index vertreten sind. Um Daten für diese Spalten abzurufen, müssen Zeilen der Basistabelle gelesen werden. Wenn zum Beispiel der Index IX1 vorhanden ist, muss für die folgende Abfrage auf die Basistabelle zugegriffen werden, um Daten der Spalten PHONENO und HIREDATE abzurufen:

```
SELECT NAME, DEPT, MGR, SALARY, PHONENO, HIREDATE
   FROM EMPLOYEE
  WHERE NAME = 'SMITH'
```

Durch Erstellen eines eindeutigen Index mit INCLUDE-Spalten können Sie die Leistung bei Datenabfragen durch Erhöhen der Anzahl von Zugriffsversuchen, die nur auf Indizes abzielen, verbessern.

Betrachten Sie zur Verdeutlichung der Verwendung von INCLUDE-Spalten die folgende Indexdefinition:

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
    INCLUDE (DEPT, MGR, SALARY, YEARS)
```

Diese Anweisungen erstellen einen eindeutigen Index, der die Eindeutigkeit der Spalte NAME gewährleistet und außerdem die Daten für die Spalten DEPT, MGR, SALARY und YEARS speichert und pflegt.

Die folgende Abfrage kann nur durch den Zugriff auf den Index ohne Lesen der Basistabelle erfüllt werden:

```
SELECT NAME, DEPT, MGR, SALARY
     FROM   EMPLOYEE
   WHERE NAME='SMITH'
```

Zugriff über mehrere Indizes

In allen oben aufgeführten Beispielen wurde eine einzelne Indexsuche zur Erzielung der Ergebnisse durchgeführt. Zur Erfüllung der Vergleichselemente einer Klausel WHERE kann das Optimierungsprogramm auch mehrere Indizes durchsuchen. Betrachten Sie zum Beispiel die beiden folgenden Indexdefinitionen:

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

Die folgenden Vergleichselemente könnten mit Hilfe dieser beiden Indizes aufgelöst werden:

```
WHERE DEPT = :hv1
  OR (JOB   = :hv2
  AND YEARS >= :hv3)
```

In diesem Beispiel liefert das Durchsuchen des Index IX2 eine Liste von Zeilen-IDs (RIDs), die das Vergleichselement `DEPT = :hv1` erfüllen. Das Durchsuchen des Index IX3 liefert eine Liste der RIDs, die das Vergleichselement `JOB = :hv2 AND YEARS >= :hv3` erfüllen. Diese beiden Listen von RIDs können kombiniert und doppelte Werte entfernt werden, bevor auf die Tabelle zugegriffen wird. Diese Methode wird als *OR-Verknüpfung von Indizes* (Oder-Verknüpfung) bezeichnet.

Die OR-Verknüpfung von Indizes kann auch für Vergleichselemente mit dem Ausdruck IN wie in folgendem Beispiel verwendet werden:

```
WHERE DEPT IN (:hv1, :hv2, :hv3)
```

Das Ziel der OR-Verknüpfung von Indizes ist es, doppelte RIDs vollständig zu entfernen; das Ziel der *AND-Verknüpfung von Indizes* (Index ANDing) ist es dagegen, gemeinsame RIDs zu finden. Die AND-Verknüpfung von Indizes kann bei Anwendungen auftreten, wenn es mehrere Indizes für entsprechende Spalten innerhalb derselben Tabelle gibt und eine Abfrage mit mehreren AND-Vergleichselementen für die Tabelle ausgeführt wird. Mehrere Indexsuchen für alle mit Indizes versehenen Spalten in einer solchen Abfrage ergeben Werte, mit denen in einem Hash-Verfahren Bitzuordnungen erstellt werden. Die zweite Bitzuordnung wird zur Prüfung der ersten Bitzuordnung verwendet, um die gesuchten Zeilen zu generieren, die zur Erstellung der endgültigen zurückzugebenden Datenmenge abgerufen werden.

Betrachten Sie zum Beispiel die beiden folgenden Indexdefinitionen:

```
INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

Die folgenden Vergleichselemente könnten mit Hilfe dieser beiden Indizes aufgelöst werden:

```
WHERE SALARY BETWEEN 20000 AND 30000
   AND COMM BETWEEN 1000 AND 3000
```

In diesem Beispiel generiert das Durchsuchen des Index IX4 eine Bitzuordnung, die das Vergleichselement SALARY BETWEEN 20000 AND 30000 erfüllt. Das Durchsuchen von IX5 und Prüfen gegen die Bitzuordnung für IX4 liefert eine Liste von RIDs, die beide Vergleichselemente erfüllen. Dies wird als „dynamische AND-Verknüpfung über Bitzuordnungen" bezeichnet. Diese Methode wird nur angewandt, sofern die Tabelle ausreichend Kardinalität hat und die Spalten genügend Werte in dem gesuchten Bereich oder genügend Duplizität haben, wenn Gleichheitsvergleichselemente verwendet werden.

Zur Umsetzung der Leistungsvorteile dynamischer Bitzuordnungen beim Durchsuchen mehrerer Indizes kann es notwendig sein, den Wert des Konfigurationsparameters Größe des Sortierspeichers (*sortheap*) der Datenbank und des Konfigurationsparameters Schwellenwert für Sortierspeicher (*sheapthres*) des Datenbankmanagers zu ändern.

Bei Verwendung dynamischer Bitzuordnungen in Zugriffsplänen wird zusätzlicher Sortierspeicher benötigt. Wenn der Wert von *sheapthres* relativ nahe am Wert von *sortheap* liegt (d. h. weniger als ein Faktor von 2- oder 3-mal pro gleichzeitiger Abfrage), steht dynamischen Bitzuordnungen mit Zugriff über mehrere Indizes wesentlich weniger Speicher zur Verfügung als das Optimierungsprogramm veranschlagt hat.

Die Lösung besteht darin, den Wert von *sheapthres* relativ zu *sortheap* zu erhöhen.

Anmerkung: Beim Zugriff auf eine einzige Tabelle werden von DB2 die AND-Verknüpfung und die OR-Verknüpfung von Indizes nicht kombiniert.

Geclusterte Indizes

Bei der Auswahl des Zugriffsplans bezieht das Optimierungsprogramm den Ein-/Ausgabeaufwand für das Laden von Seiten von der Platte in den Pufferpool in die Kalkulation mit ein. In den Berechnungen schätzt das Optimierungsprogramm die Anzahl der E/A-Operationen ab, die zur Erfüllung einer Abfrage erforderlich sind. Diese Schätzung schließt eine Voraussage über den Bedarf an Pufferpool mit ein, da zum Lesen von Zeilen einer Seite, die sich bereits im Pufferpool befindet, keine weiteren E/A-Operationen anfallen.

Für Indexsuchen verwendet das Optimierungsprogramm Informationen aus den Systemkatalogtabellen (SYSCAT.INDEXES), um die Abschätzung des Ein-/Ausgabeaufwands zum Lesen von Datenseiten in den Pufferpool zu unterstützen. Die folgenden Spalten aus der Tabelle SYSCAT.INDEXES werden verwendet:

- Die Spalte CLUSTERRATIO, die den Grad angibt, zu dem die Tabellendaten in Relation zu diesem Index in Gruppen zusammengefasst (″geclustert″) sind. Ein höherer Wert bedeutet, dass die Zeilen auf den Datenseiten in der Reihenfolge des Indexschlüssels geordnet sind. Daher können alle Zeilen auf einer Datenseite gelesen werden, während sich die Seite im Puffer befindet. Wenn der Wert dieser Spalte -1 ist, versucht das Optimierungsprogramm die Spalten PAGE_FETCH_PAIRS und CLUSTERFACTOR zu verwenden.

  oder

- Die Spalte PAGE_FETCH_PAIRS, die verschiedene Paare von Ziffern enthält, die jeweils die Anzahl der E/A-Operationen zum Lesen der Datenseiten in Pufferpools verschiedener Größen angeben, zusammen mit CLUSTERFACTOR. Bei der Sammlung der Statistiken für einen Index werden diese Informationen als detaillierte statistische Daten angesehen.

Wenn keine Statistiken verfügbar sind, verwendet das Optimierungsprogramm Standardwerte für die statistischen Daten, die von einem geringen Grad der Clusterbildung der Daten bezüglich des Index ausgehen. Lesen Sie dazu auch die Informationen in „Kapitel 5. Systemkatalogstatistiken” auf Seite 129 und im Abschnitt „Erfassen statistischer Daten mit dem Dienstprogramm RUNSTATS” auf Seite 131.

Sie können einen Clusterungsindex angeben, der zur Clusterung der Zeilen während einer Tabellenreorganisation und zur Erhaltung dieses Merkmals während der Verarbeitung von INSERT-Operationen dient. (Lesen Sie im Abschnitt „Reorganisieren von Katalogen und Benutzertabellen” auf Seite 313 die Informationen zur Reorganisation von Tabellen.) Nachfolgende Aktualisierungen und Einfügungen können den Index weniger gut geclustert zurücklassen (wie mit den von RUNSTATS gesammelten Statistiken zu messen ist), so dass von Zeit zu Zeit eine Reorganisation der Tabelle erforderlich wird. Um die Häufigkeit der Reorganisation einer Tabelle zu verringern, die aufgrund von INSERT-, UPDATE- und DELETE-Anweisungen häufig geändert wird, können Sie beim Ändern einer Tabelle den Parameter PCTFREE verwenden. Dieser Parameter ermöglicht es, weitere Einfügungen so in die vorhandenen Daten einzugliedern, dass die Clusterung erhalten bleibt.

Der Grad, zu dem die Daten in Bezug auf einen Index in Gruppen zusammengefasst (d. h. geclustert) sind, kann bedeutende Auswirkungen auf die Leistung haben, so dass einer der Indizes für eine Tabelle auf einem Grad nahe an 100% Clusterbildung gehalten werden sollte. Im allgemeinen kann nur ein Index eine 100-prozentige Clusterbildung aufweisen. Eine Ausnahme bilden nur solche Fälle, in denen die Schlüssel eines anderen Index eine Obermenge der Schlüssel des Clusterungsindex sind oder in denen es eine De-Facto-Korrelation zwischen den Schlüsselspalten der beiden Indizes gibt.

Weitere Informationen zu leistungsrelevanten Gründen für die Verwendung von Clusterungsindizes finden Sie in „Hinweise zur Leistung für die Verwaltung von Indizes" auf Seite 117. Weitere Informationen zur Erstellung eines Clusterungsindex finden Sie unter CREATE INDEX im Handbuch *SQL Reference*.

Clusterung von Seitenleseoperationen mit Hilfe des Vorablesezugriffs über Listen: Wenn das Optimierungsprogramm einen Index zum Zugriff auf Zeilen verwendet, kann es das Lesen der Datenseiten verzögern, bis alle Zeilen-IDs (RIDs) aus dem Index empfangen wurden. Betrachten Sie zum Beispiel den bereits früher definierten Index IX1:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

Betrachten Sie folgende Suchkriterien:

```
WHERE NAME BETWEEN 'A' and 'I'
```

In diesem Fall könnte das Optimierungsprogramm eine Indexsuche in IX1 durchführen, um die Zeilen (und Datenseiten), die abzurufen sind, zu ermitteln. Wenn die Daten nach diesem Index nicht geclustert sind, enthält der Vorablesezugriff über Listen einen Schritt zum Sortieren der durch die Indexsuche ermittelten Liste von Zeilen-IDs (RIDs). Weitere Informationen finden Sie in „Vorablesezugriff über Listen" auf Seite 303.

Vorablesezugriff auf Indexseiten

Der Datenbankmanager versucht nach Möglichkeit das Auftreten eines sequenziellen Zugriffs auf Indexseiten festzustellen und entsprechende Anforderungen zum Vorablesen zu generieren. Auf diese Weise wird die benötigte Zeit für nichtselektive Indexsuchen sowie die benötigte Zeit für Indexsuchen, die auf einen wesentlichen Teil des Index zugreifen, bedeutend verringert.

Das Optimierungsprogramm stützt sich bei der Abschätzung des zu erwartenden Volumens des Vorablesezugriffs auf Indexseiten auf Indexstatistikdaten wie DENSITY und SEQUENTIAL_PAGES, die Merkmale der Tabellenbereiche, in denen sich der Index befindet, und die Informationen über die Auswirkung etwaiger bereichsbegrenzender Vergleichselemente. Von diesen Schätzwerten ausgehend wird der Gesamtaufwand für die Verwendung eines bestimmten Index abgeschätzt.

Weitere Informationen finden Sie in „Sequenzieller Vorablesezugriff" auf Seite 301.

Tabellensuche und Indexsuche

Das Optimierungsprogramm wählt eine Tabellensuche, wenn für die Abfrage kein Index verwendet werden kann oder wenn das Optimierungsprogramm feststellt, dass eine Indexsuche aufwendiger ist. Eine Indexsuche ist in folgenden Fällen auswendiger:

- Wenn die Tabelle klein ist.
- Wenn der Grad der Index-Clusterung gering ist.
- Wenn auf den größten Teil der Tabelle zugegriffen wird.

Mit den SQL-EXPLAIN-Einrichtungen können Sie feststellen, ob Ihr Zugriffsplan eine Tabellensuche oder eine Indexsuche verwendet. Weitere Informationen finden Sie in „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251.

Vergleichselementterminologie

Eine Benutzeranwendung fordert eine Menge von Zeilen aus der Datenbank mit Hilfe einer SQL-Anweisung an, in der die gewünschten Zeilen mit Vergleichselementen näher eingegrenzt werden. Wenn das Optimierungsprogramm eine SQL-Anweisung auswertet, wird jedes Vergleichselement einer von vier Kategorien zugeordnet. Die Kategorie wird dadurch bestimmt, wie und wann das jeweilige Vergleichselement im Auswertungsprozess verwendet wird. Diese Kategorien werden im Folgenden in der Reihenfolge von der höchsten bis zur niedrigsten Leistung geordnet aufgelistet:

1. Bereichsbegrenzende Vergleichselemente
2. Bei Indexsuchen als Suchargument verwendbare Vergleichselemente (Index SARGable)
3. Bei Datensuchen als Suchargument verwendbare Vergleichselemente (Data SARGable)
4. Restvergleichselemente

(Die Bezeichnung *SARGable* ist aus dem Begriff *s*earch *arg*ument abgeleitet.)

Der Abschnitt „Zusammenfassung der Verwendung von Vergleichselementen" auf Seite 202 enthält einen Vergleich der Merkmale, die sich auf die Leistung der verschiedenen Vergleichselementkategorien auswirken.

Bereichsbegrenzende und bei Indexsuchen als Suchargument verwendbare Vergleichselemente

Bereichsbegrenzende Vergleichselemente werden zur Eingrenzung einer Indexsuche verwendet. Sie definieren Start- und/oder Stoppschlüsselwerte für die Indexsuche. Bei Indexsuchen als Suchargument verwendbare Vergleichselemente dienen nicht zur Eingrenzung einer Suche, aber sie können mit Hilfe des Index ausgewertet werden, da die im Vergleichselement verwendeten Spalten Teil des Indexschlüssels sind. Betrachten Sie zum Beispiel den zuvor definierten Index IX1 (siehe „Konzepte der Indexsuche" auf Seite 188) und die folgende Klausel WHERE:

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

Die ersten beiden Vergleichselemente (NAME = :hv1, DEPT = :hv2) wären bereichsbegrenzende Vergleichselemente, während YEARS > :hv5 ein bei Indexsuchen als Suchargument verwendbares Vergleichselement wäre.

Der Datenbankmanager verwendet die Indexdaten bei der Auswertung dieser Vergleichselemente, anstatt die Basistabelle zu lesen. Diese bei Indexsuchen als Suchargumente verwendbaren Vergleichselemente (*Index SARGable*) verringern die Anzahl der Datenseiten, auf die zugegriffen wird, indem die Menge der Zeilen, die aus der Tabelle zu lesen sind, verkleinert wird. Diese Vergleichselementkategorien wirken sich nicht auf die Anzahl der Indexseiten aus, auf die zugegriffen wird.

Bei Datensuchen als Suchargument verwendbare Vergleichselemente

Vergleichselemente, die nicht vom Indexmanager ausgewertet werden können, sondern nur von den Datenverwaltungsservices, werden als bei Datensuchen verwendbare (*Data SARGable*) Vergleichselemente bezeichnet. In der Regel machen diese Vergleichselemente den Zugriff auf einzelne Zeilen aus einer Basistabelle erforderlich. Bei Bedarf rufen die Datenverwaltungsservices die zur Auswertung des Vergleichselements benötigten Spalten und andere Spalten ab, um die Spalten für die SELECT-Liste, die nicht aus dem Index abgerufen werden konnten, zur Verfügung zu stellen.

Betrachten Sie zum Beispiel einen einzelnen Index, der für die Tabelle PROJECT definiert ist:

```
INDEX IX0:  PROJNO ASC
```

Bei der Ausführung der folgenden Abfrage würde das Vergleichselement DEPTNO = 'D11' als bei Datensuchen als Suchargument verwendbares Vergleichselement eingestuft.

```
SELECT PROJNO, PROJNAME, RESPEMP
  FROM PROJECT
 WHERE DEPTNO = 'D11'
 ORDER BY PROJNO
```

Restvergleichselemente

Restvergleichselemente sind typischerweise solche Vergleichselemente, die E/A-Operationen über den einfachen Zugriff auf eine Basistabelle hinaus erforderlich machen. Beispiele von Restvergleichselementen sind solche mit korrelierten Unterabfragen, quantifizierten Unterabfragen (d. h. Abfragen mit ANY, ALL, SOME oder IN) oder Vergleichselemente, für die Daten der Typen LONG VARCHAR oder LOB (die von der Tabelle getrennt gespeichert werden) gelesen werden. Diese Vergleichselemente werden von den Services für relationale Daten (Relational Data Services) ausgewertet.

Es kommt vor, dass Vergleichselemente, die nur auf einen Index angewandt wurden, noch einmal angewandt werden müssen, wenn auf die Datenseite zugegriffen wird. Zum Beispiel wenden Zugriffspläne mit OR-Verknüpfung oder AND-Verknüpfung von Indizes (siehe „Zugriff über mehrere Indizes" auf Seite 196) die Vergleichselemente immer ein weiteres Mal als Restvergleichselemente an, wenn auf die Datenseite zugegriffen wird.

Zusammenfassung der Verwendung von Vergleichselementen

Die Verwendung von Vergleichselementen in einer Abfrage kann die Verringerung der zur Erfüllung der Abfrage zu lesenden Daten unterstützen. Verschiedene Kategorien von Vergleichselementen haben verschiedene Auswirkungen auf die Leistung einer Abfrage. Die Auswirkungen werden vom Optimierungsprogramm in der Aufwandskalkulation berücksichtigt. Die folgende Tabelle zeigt die Rangordnung der verschiedenen Vergleichselementkategorien und die Art der Auswirkung, die jede Kategorie auf die Leistung haben kann.

Tabelle 14. Zusammenfassung der Merkmale der Vergleichselementkategorien

Merkmal	**Vergleichselementkategorie**			
	Bereichs-begrenzend	**Index-suchargument**	**Daten-suchargument**	**Restvergleichs-element**
Verringern der Index-Ein-/Ausgabe	Ja	Nein	Nein	Nein
Verringern der Datenseiteneinein-/ausgabe	Ja	Ja	Nein	Nein
Verringern der Anzahl intern übergebener Zeilen	Ja	Ja	Ja	Nein
Verringern der Anzahl der den Kriterien entsprechenden Zeilen	Ja	Ja	Ja	Ja

Verknüpfungskonzepte

Als *Verknüpfung* wird die Verkettung von Zeilen einer Tabelle mit Zeilen einer oder mehrerer anderer Tabellen bezeichnet. Betrachten Sie zum Beispiel die folgenden beiden Tabellen:

```
      TABELLE1                TABELLE2
-----------------       -----------------
 PROJ     PROJ_ID        PROJ_ID     NAME
------    -------       -------    ------
  A          1             1        Sam
  B          2             3        Joe
  C          3             4        Mary
  D          4             1        Sue
                           2        Mike
```

Die Verknüpfung von Tabelle1 und Tabelle2 an den Stellen, an denen die ID-Spalten gleiche Werte enthalten, wird durch die folgende SQL-Anweisung ausgedrückt:

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

Diese Anweisung ergäbe die folgende Menge von Ergebniszeilen:

```
 PROJ     PROJ_ID     NAME
------    -------    ------
  A          1        Sam
  A          1        Sue
  B          2        Mike
  C          3        Joe
  D          4        Mary
```

Bei der Verknüpfung zweier Tabellen wird die eine Tabelle als äußere Tabelle und die andere als innere Tabelle ausgewählt. Auf die äußere Tabelle wird zuerst zugegriffen, und sie wird nur einmal durchsucht. Ob die innere Tabelle mehrere Male durchsucht wird, hängt von der Art der Verknüpfung ab und davon, welche Indizes vorhanden sind. Unabhängig davon, ob durch eine Abfrage zwei oder mehr Tabellen miteinander verknüpft werden, verknüpft das Optimierungsprogramm jeweils nur zwei Tabellen gleichzeitig. Bei Bedarf werden temporäre Tabellen mit Zwischenergebnissen erstellt.

Das Optimierungsprogramm wählt eine von zwei Verknüpfungsmethoden (Verknüpfung über Verschachtelungsschleife oder Mischverknüpfung) in Abhängigkeit davon, ob ein Verknüpfungsvergleichselement (Definition siehe „Mischverknüpfung” auf Seite 206) vorhanden ist und welche Verarbeitungsaufwände anhand der Tabellen- und Indexstatistiken ermittelt werden.

Verknüpfung über Verschachtelungsschleife

Eine Verknüpfung über Verschachtelungsschleife (Nested Loop Join) wird auf eine von zwei Arten ausgeführt:

1. Durch Durchsuchen der inneren Tabelle für jede Zeile der äußeren Tabelle, auf die zugegriffen wird

   Betrachten Sie zum Beispiel die Spalte A in den Tabellen T1 und T2 mit folgenden Werten:

```
Äußere Tabelle T1: Spalte A   Innere Tabelle T2: Spalte A
-----------------------       -----------------------
          2                             3
          3                             2
          3                             2
                                        3
                                        1
```

Folgende Schritte werden bei der Verknüpfung über Verschachtelungsschleife ausgeführt:

- Lesen der ersten Zeile aus T1. Der Wert für A ist „2".
- Durchsuchen von T2, bis ein übereinstimmender Wert („2") gefunden wird, und anschließendes Verknüpfen der beiden Zeilen
- Durchsuchen von T2, bis der nächste übereinstimmende Wert („2") gefunden wird, und anschließendes Verknüpfen der beiden Zeilen
- Durchsuchen von T2 bis zum Ende der Tabelle
- Zurückkehren zu T1 und Lesen der nächsten Zeile („3")
- Durchsuchen von T2 von der ersten Zeile an, bis ein übereinstimmender Wert („3") gefunden wird, und anschließendes Verknüpfen der beiden Zeilen
- Durchsuchen von T2, bis der nächste übereinstimmende Wert („3") gefunden wird, und anschließendes Verknüpfen der beiden Zeilen
- Durchsuchen von T2 bis zum Ende der Tabelle
- Zurückkehren zu T1 und Lesen der nächsten Zeile („3")
- Durchsuchen von T2 wie vorher und Verknüpfen aller übereinstimmenden Zeilen („3")

2. Durch eine Indexsuche für die innere Tabelle für jede Zeile der äußeren Tabelle, auf die zugegriffen wird

   Diese Methode kann für angegebene Vergleichselemente verwendet werden, wenn es ein Vergleichselement der folgenden Form gibt:

   ```
   ausdr(äußere_tabelle.spalte)  relop  innere_tabelle.spalte
   ```

   Dabei gilt, dass `relop` ein relativer Operator (z. B. =, >, >=, < oder <=) und `ausdr` ein gültiger Ausdruck für die äußere Tabelle ist. Beispiele:

   ```
   ÄUSSERE.C1 + ÄUSSERE.C2 <= INNERE.C1
   ```

   und

   ```
   ÄUSSERE.C4 < INNERE.C3
   ```

   Diese Methode ist eine Möglichkeit, die Anzahl der Zeilen, auf die in der inneren Tabelle für jeden Zugriff auf die äußere Tabelle zugegriffen wird, wesentlich zu verringern (obwohl dies von einer Reihe von Faktoren abhängig ist, zu denen auch die Selektivität des Verknüpfungsvergleichselements zählt).

Bei der Auswertung einer Verknüpfung über eine Verschachtelungsschleife bestimmt das Optimierungsprogramm auch, ob die äußere Tabelle sortiert wird oder nicht, bevor die Verknüpfung durchgeführt wird. Durch Sortieren der äußeren Tabelle auf der Grundlage der Verknüpfungsspalten kann die Anzahl der Leseoperationen zum Zugriff auf Seiten auf der Platte für die innere Tabelle verringert werden, da es wahrscheinlicher wird, dass sich die Seiten bereits im Pufferpool befinden. Wenn die Verknüpfung einen Index mit einem hohen Grad der Clusterbildung verwendet, um auf die innere Tabelle zuzugreifen, kann die Anzahl der Indexseiten, auf die zugegriffen wird, minimiert werden, wenn die äußere Tabelle sortiert wurde.

Darüber hinaus kann das Optimierungsprogramm eine Sortierung vor der Verknüpfung durchführen, wenn zu erwarten ist, dass durch die Verknüpfung eine spätere Sortierung aufwendiger wird. Eine spätere Sortierung könnte erforderlich sein, um die Klauseln GROUP BY, DISTINCT, ORDER BY oder eine Mischverknüpfung zu unterstützen.

Mischverknüpfung

Eine Mischverknüpfung (Merge Join) erfordert ein Vergleichselement der Form tabelle1.spalte = tabelle2.spalte. Ein solches Vergleichselement wird als *Gleichheitsverknüpfungsprädikat* bezeichnet. Eine Mischverknüpfung erfordert entweder über einen Indexzugriff oder durch eine Sortierung geordnete Eingaben für die Verknüpfungsspalten. Um eine Mischverknüpfung verwenden zu können, darf die Verknüpfungsspalte keine Spalte mit Langfelddaten (LONG) oder mit LOB-Daten sein.

Die verknüpften Tabellen werden gleichzeitig durchsucht. Die äußere Tabelle wird bei der Mischverknüpfung nur einmal durchsucht. Die innere Tabelle wird auch nur einmal durchsucht, sofern es keine sich wiederholenden Werte in der äußeren Tabelle gibt. Wenn es doppelte Werte in der äußeren Tabelle gibt, kann eine Gruppe von Zeilen der inneren Tabelle noch einmal durchsucht werden. Betrachten Sie zum Beispiel die Spalte A in den Tabellen T1 und T2 mit folgenden Werten:

```
Äußere Tabelle T1: Spalte A   Innere Tabelle T2: Spalte A
-----------------------       -----------------------
          2                             1
          3                             2
          3                             2
                                        3
                                        3
```

Bei der Mischverknüpfung werden folgende Schritte ausgeführt:

- Lesen der ersten Zeile aus T1. Der Wert für A ist „2".
- Durchsuchen von T2, bis ein übereinstimmender Wert gefunden wird, und anschließendes Verknüpfen der beiden Zeilen
- Fortsetzen des Durchsuchens von T2, solange die Spalten übereinstimmen, und dabei Verknüpfen der Zeilen
- Wenn der Wert „3" in T2 gelesen wird, Zurückkehren zu T1 und Lesen der nächsten Zeile
- Der nächste Wert in T1 ist „3", der mit T2 übereinstimmt, also Verknüpfen der Spalten
- Fortsetzen des Durchsuchens von T2, solange die Spalten übereinstimmen, und dabei Verknüpfen der Zeilen
- Erreichen des Endes von T2
- Zurückkehren zu T1, um die nächste Zeile zu lesen. Beachten Sie, dass der nächste Wert in T1 derselbe ist wie der vorige Wert aus T1, so dass T2 noch einmal, angefangen vom ersten Wert „3" in T2, durchsucht wird (der Datenbankmanager speichert diese Position).

Hash-Verknüpfung

Eine Hash-Verknüpfung erfordert ein oder mehrere Vergleichselemente der Form tabelle1.spalteX = tabelle2.spalteY, wobei es sich um die **gleichen** Spaltentypen handeln muss. Spalten des Typs CHAR müssen die gleiche Länge aufweisen. Bei Spalten des Typs DECIMAL muss die Genauigkeit und die Anzahl der Kommastellen übereinstimmen. Der Spaltentyp kann keine LONG-Feldspalte oder LOB-Spalte sein.

Zunächst wird eine Tabelle (als innere Tabelle (INNER) bezeichnet) durchsucht, und die Zeilen werden in Speicherpuffer kopiert, die aus dem Sortierspeicher zugeordnet werden (siehe Konfigurationsparameter „Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422 der Datenbank). Die Speicherpuffer werden nach einem „Hash-Code", der anhand der Spalten des Verknüpfungsvergleichselements (bzw. der Vergleichselemente) berechnet wird, in Partitionen unterteilt. Wenn die Größe der ersten Tabelle die Größe des Sortierspeichers überschreitet, werden Puffer aus ausgewählten Partitionen in temporäre Tabellen geschrieben. Nach Abschluss der Verarbeitungsschritte für die innere Tabelle wird die zweite Tabelle (als äußere Tabelle (OUTER) bezeichnet) durchsucht. Zeilen der äußeren Tabelle werden mit Zeilen aus der inneren Tabelle abgeglichen, indem zuerst ein aus den Spalten des Verknüpfungsvergleichselements (bzw. der Vergleichselemente) generierter „Hash-Code" verglichen wird. Wenn der „Hash-Code" der äußeren Zeilen mit dem „Hash-Code" der inneren Zeile übereinstimmt, werden die tatsächlichen Spalten des Verknüpfungsvergleichselements (bzw. der Vergleichselemente) miteinander verglichen.

Zeilen der äußeren Tabelle, die Partitionen entsprechen, die nicht in eine temporäre Tabelle geschrieben wurden, werden sofort mit den Zeilen der inneren Tabelle im Speicher abgeglichen. Andernfalls werden die äußeren Zeilen, wenn die entsprechende Partition der inneren Tabelle in eine temporäre Tabelle geschrieben wurde, ebenfalls in eine temporäre Tabelle geschrieben. Schließlich werden übereinstimmende Paare von Partitionen aus den temporären Tabellen gelesen, die „Hash-Codes" ihrer Zeilen abgeglichen und die Verknüpfungsvergleichselemente geprüft.

Zur Umsetzung der Leistungsvorteile der Hash-Verknüpfung kann es notwendig sein, den Wert des Konfigurationsparameters *sortheap* der Datenbank und des Konfigurationsparameters *sheapthres* des Datenbankmanagers zu ändern.

Bei Entscheidungshilfeabfragen benötigen Zugriffspläne mit Hash-Verknüpfungen mehr Sortierspeicher als Zugriffspläne mit anderen Verknüpfungsarten. Wenn der Wert von *sheapthres* relativ nahe am Wert von *sortheap* liegt (d. h. weniger als ein Faktor 2 oder 3 pro gleichzeitiger Abfrage), arbeitet eine Hash-Verknüpfung mit wesentlich weniger Speicher als das Optimierungsprogramm veranschlagt hat. Bei der Ausführung mit begrenztem Hauptspeicher können Hash-Verknüpfungen sehr langsam sein. Das Problem tritt in Abfragen mit mehreren Sortierungen und Hash-Verknüpfungen auf, in denen sich die Sortierungen bzw. Hash-Verknüpfungen den größten Teil des verfügbaren Speichers reservieren.

Die Lösung besteht darin, den Parameter *sheapthres* ausreichend groß (im Vergleich zu *sortheap*) zu konfigurieren.

Festlegen von äußerer und innerer Tabelle

Wie wird bei der Verknüpfung festgelegt, welche Tabelle die äußere und welche die innere ist? Im folgenden wird allgemein erläutert, wie das Optimierungsprogramm festlegt, welche Tabelle die innere und welche die äußere Tabelle sein soll.

Im Fall einer **Hash-Verknüpfung** wird die innere Tabelle in Speicherpuffern behalten. Wenn nicht genügend Speicherpuffer zur Verfügung stehen, ist die Hash-Verknüpfung gezwungen, einen Überlauf zu verursachen. Das Optimierungsprogramm versucht, dies zu vermeiden, und wählt daher die kleinere der beiden Tabellen als innere Tabelle und die größere als äußere Tabelle.

Die Reihenfolge, in der auf die Tabellen zugegriffen wird, spielt besonders bei einer **Verknüpfung über Verschachtelungsschleife** eine wichtige Rolle, da auf die äußere Tabelle nur einmal zugegriffen wird, während auf die innere Tabelle einmal für jede Zeile der äußeren Tabelle zugegriffen wird. Das

Optimierungsprogramm wählt die äußere und innere Tabelle auf der Grundlage von Aufwandsschätzungen aus. Diese Aufwandsschätzungen werden von folgenden Faktoren beeinflusst:

- Größe

  Die kleinere Tabelle wird häufig als äußere Tabelle festgelegt, um die Anzahl der erneuten Zugriffe auf die innere Tabelle zu verringern. Allerdings kann ein Vorablesezugriff gerade das Gegenteil bewirken. Der Vorablesezugriff kann den Aufwand für den Zugriff auf eine umfangreiche Tabelle erheblich reduzieren. Jedoch ist der Vorablesezugriff nur für die äußere Tabelle einer Verknüpfung effizient. Daher kann auch auf die größere Tabelle zuerst zugegriffen werden. Weitere Informationen finden Sie in „Vorablesen von Daten in den Pufferpool" auf Seite 301.

- Vergleichselemente

  Eine Tabelle wird mit größerer Wahrscheinlichkeit als äußere Tabelle festgelegt, wenn selektive Vergleichselemente auf sie angewandt werden können, da auf die innere Tabelle nur für Zeilen zugegriffen wird, die die auf die äußere Tabelle angewandten Vergleichselemente erfüllen.

- Puffern

  Wenn die gesamte innere Tabelle für jede Zeile der äußeren Tabelle durchsucht werden muss (d. h., es kann keine Indexsuche für die innere Tabelle durchgeführt werden), kann die kleinere der beiden Tabellen als innere Tabelle festgelegt werden, um die Vorteile der Pufferung nutzen zu können. Diese Festlegung wird von der Größe der Tabelle und des Pufferpools beeinflusst. Beachten Sie, dass die Entscheidungen über die Verknüpfung von der Größe des Pufferpools beeinflusst werden und daher der Zugriffsplan für Anwendungen geändert werden kann, wenn nach einer Änderung der Pufferpoolgröße die Anwendungen erneut an die Datenbank gebunden werden.

  Die Möglichkeit, mehr als einen Pufferpool zu erstellen und die Größe dieses Pufferpools zu ändern sowie die Tabellenbereiche, die diesen Pufferpool verwenden, zu steuern, kann sich darauf auswirken, wann das Puffern innerhalb innerer und äußerer Tabellen verwendet wird.

- Indizes

  Wenn für eine der Tabellen eine Indexsuche durchgeführt werden kann, dann bietet sich diese Tabelle zur Verwendung als innere Tabelle besonders an. Auf sie könnte dann über eine Suche nach einem Indexschlüssel mit Hilfe des Verknüpfungsschlüsselvergleichselements der äußeren Tabelle als eines der Schlüsselwerte zugegriffen werden. Wenn eine Tabelle keinen Index hat, ist sie zur Verwendung als innere Tabelle nicht besonders geeignet, da in diesem Fall die gesamte innere Tabelle für jede Zeile der äußeren Tabelle durchsucht werden müsste.

- Anforderungen an die Reihenfolge

  Auf eine Tabelle, für die eine Reihenfolge der Zeilen angefordert wurde, wird eventuell zuerst zugegriffen. Wenn zum Beispiel die Ausgabe der Verknüpfung zwischen t1 und t2 nach t1.c sortiert werden soll, kann der Zugriff auf t1 als äußere Tabelle mit einem Index für t1.c eine gute Methode sein. Die Ausgabe der Verknüpfung wäre geordnet, so dass keine weitere Sortierung erforderlich würde.

  ```
  SELECT * FROM t1, t2
    WHERE t1.a = t2.b
    ORDER BY t1.c
  ```

Die Reihenfolge, in der auf die Tabellen zugegriffen wird, spielt bei einer **Mischverknüpfung** eine nicht so wichtige Rolle, weil sowohl die innere als auch die äußere Tabelle nur einmal gelesen werden. Allerdings werden Teile der inneren Tabelle, die mehrfach auftretenden Werten in der Verknüpfungsspalte der äußeren Tabelle entsprechen, in einem speicherinternen Puffer behalten. Der Puffer wird erneut gelesen, wenn die nächste äußere Zeile mit der vorigen äußeren Zeile übereinstimmt. Ansonsten wird der Puffer neu belegt. Wenn die Anzahl der Vorkommen eines mehrfach auftretenden Verknüpfungswerts die Kapazität des speicherinternen Puffers überschreitet, werden nicht alle auftretenden Werte im Puffer behalten. Dies geschieht nur, wenn die Duplizität eines Werts groß ist und der Wert einen übereinstimmenden Wert in der äußeren Tabelle hat.

Bei allen Überlegungen zu diesen mehrfach auftretenden Werten wird doch in den meisten Fällen die Tabelle als äußere Tabelle in einer Verknüpfung ausgewählt, die weniger mehrfach auftretende Werte enthält. Letzten Endes wählt das Optimierungsprogramm die äußeren und inneren Tabellen jedoch auf der Grundlage detaillierter Aufwandsschätzungen aus.

Suchstrategien zur Auswahl der optimalen Verknüpfungsmethode

Das Optimierungsprogramm kann die optimalen Methoden zur Verknüpfung mit Hilfe verschiedener Suchstrategien bestimmen. Die verwendete Suchstrategie wird durch der Optimierungsklasse festgelegt (siehe „Anpassen der Optimierungsklasse" auf Seite 76). Die Suchstrategien und ihre Merkmale sind folgende:

- Schnelle Verknüpfungsaufzählung (Greedy Join Enumeration)
  - Effizient im Hinblick auf Speicherbedarf und Zeit
  - Aufzählung in einer Richtung, d. h., wenn eine Verknüpfungsmethode für zwei Tabellen ausgewählt ist, wird sie im Laufe der weiteren Optimierung nicht mehr geändert
  - Eventuell wird nicht der optimale Zugriffsplan gewählt, wenn viele Tabellen verknüpft werden. Wenn eine Abfrage nur zwei oder drei Tabellen verknüpft, ist der Zugriffsplan, der von der schnellen Verknüpfungsaufzählung ausgewählt wird, derselbe wie der Zugriffsplan, der von der

dynamisch programmierten Verknüpfungsaufzählung (Dynamic Programming Join Enumeration) ausgewählt wird. Dies gilt besonders dann, wenn die Abfrage viele Verknüpfungsvergleichselemente (entweder explizit angegebene oder implizit durch die Anwendung der Transitivität generierte) für dieselbe Spalte hat.

- Dynamisch programmierte Verknüpfungsaufzählung (Dynamic Programming Join Enumeration)
  - Die Anforderungen an Speicher und Zeit wachsen exponenziell mit der Anzahl der Tabellen, die verknüpft werden.
  - Effiziente und erschöpfende Suche nach dem besten Zugriffsplan
  - Ähnlich der Strategie, die von DB2 für OS/390 verwendet wird

Der Algorithmus für die Verknüpfungsaufzählung spielt die entscheidende Rolle bei der Bestimmung der Anzahl von Zugriffsplankombinationen, die vom Optimierungsprogramm geprüft werden.

Suchstrategien für eine Sternverknüpfung (Star Join)

Im allgemeinen sollten die Tabellen, auf die in einer Abfrage zugegriffen wird, durch Verknüpfungsvergleichselemente miteinander verbunden sein. Wenn zwei Tabellen ohne Verknüpfungsvergleichselement verknüpft werden, wird das kartesische Produkt der beiden Tabellen gebildet. Das heißt, jede ausgewählte Zeile der ersten Tabelle wird mit jeder ausgewählten Zeile der zweiten Tabelle verknüpft. Das Ergebnis ist eine Tabelle, die aus dem Kreuzprodukt in der Größe der beiden Tabellen besteht und in der Regel sehr groß ist. Da ein solcher Plan wahrscheinlich keine gute Leistung zulässt, vermeidet das Optimierungsprogramm sogar die Aufwandsabschätzung für einen Plan dieser Art. Die einzige Ausnahme dieser Vorgehensweise tritt ein, wenn die Optimierungsklasse 9 definiert wurde oder der folgende Spezialfall eines „Sternschemas" (Star Scheme) vorliegt. Weitere Informationen finden Sie in „Anpassen der Optimierungsklasse" auf Seite 76.

Die Fälle, in denen Zugriffspläne mit kartesischen Produkten eine gute Leistung zeigen, liegen in der Regel dann vor, wenn es sich um umfangreiche Entscheidungshilfedatenbanken handelt, die in der Technik eines Sternschemas (Star Schema) aufgebaut sind. Mit Sternschema wird der Aufbau einer Datenbank bezeichnet, bei dem das Gros der Rohdaten in einer einzigen umfangreichen Tabelle mit zahlreichen Spalten gespeichert wird, die gemeinhin als „Fakttabelle" bezeichnet wird. Viele der Spalten enthalten verschlüsselte Werte, die die Dimensionen eines bestimmten, in der Fakttabelle gespeicherten Faktums charakterisieren. Zur einfachen Ermöglichung der Analyse einer bestimmten Untermenge der Fakten werden die Dimensionstabellen verwendet, um die verschlüsselten Werte zu decodieren. Eine typische Abfrage bestünde aus mehreren lokalen Vergleichselementen, die auf decodierte Werte in den Dimensionstabellen verweisen, und enthielte Verknüpfungsvergleichselemente, die die Dimensionstabellen mit der Fakttabelle verbinden. Für diese

Arten von Abfragen kann es vorteilhaft sein, das kartesische Produkt mehrerer kleiner Dimensionstabellen zu bilden und erst anschließend auf die umfangreiche Fakttabelle zuzugreifen. Diese Technik ist dann von Vorteil, wenn mehrere Verknüpfungsvergleichselemente einem mehrspaltigen Index entsprechen.

DB2 kann Abfragen erkennen, die für Datenbanken durchgeführt werden, die mit Sternschemen aufgebaut sind und mindestens zwei Dimensionstabellen haben, und kann den Suchbereich vergrößern, um potenzielle Zugriffspläne mit kartesischen Produkten von Dimensionstabellen zu berücksichtigen. Wenn der Plan mit den kartesischen Produkten den niedrigsten geschätzten Aufwand verursacht, wird er vom Optimierungsprogramm ausgewählt.

Bei der bisher behandelten Sternschementechnik wurde angenommen, dass Primärschlüsselindizes in der Verknüpfung verwendet werden. Eine andere Situation liegt vor, wenn Fremdschlüsselindizes verwendet werden. Ausgehend von der Annahme, dass die Fremdschlüsselspalten in der Fakttabelle einspaltige Indizes sind und dass es eine relativ hohe Selektivität über alle Dimensionstabellen hinweg gibt, könnte die folgende Methode der Sternverknüpfung zur Anwendung kommen:

1. Jede Dimensionstabelle wird wie folgt verarbeitet:
   - Durchführen einer einfachen Gleichheitsverknüpfung zwischen der Dimensionstabelle und dem Fremdschlüsselindex für die Fakttabelle.
   - Dynamisches Erstellen einer Bitzuordnung durch ein Hash-Verfahren für die Werte der Zeilen-IDs (RID).
2. Jede Bitzuordnung wird mit AND-Vergleichselementen auf die vorige Bitzuordnung angewandt (siehe „Zugriff über mehrere Indizes" auf Seite 196).
3. Feststellen der verbliebenen RIDs, nachdem die letzte Bitzuordnung verarbeitet wurde.
4. Wahlfreies Sortieren dieser RIDs.
5. Abrufen einer Zeile aus der Basistabelle.
6. Wiederverknüpfen der Fakttabelle mit jeder der zugehörigen Dimensionstabellen, dabei Zugreifen auf die Spalten der Dimensionstabellen, die für die SELECT-Klausel benötigt werden.
7. Erneutes Anwenden der Vergleichselemente (Restvergleichselemente).

Für diese Methode sind keine mehrspaltigen Indizes erforderlich. Explizite referenzielle Integritätsbedingungen zwischen Fakttabelle und Dimensionstabellen müssen für diese Methode nicht ausgewählt werden, obwohl die Beziehung zwischen Fakttabelle und Dimensionstabellen über dieses Merkmal verfügen sollte.

Die im Rahmen der Sternverknüpfungsmethode erstellten und verwendeten Bitzuordnungen verwenden Sortierspeicher. Weitere Informationen über den Konfigurationsparameter Größe des Sortierspeichers (*sortheap*) der Datenbank finden Sie in Kapitel 13, ″Konfigurieren von DB2″ im Handbuch *Systemverwaltung: Optimierung*.

Zusammengesetzte Tabellen

Ein anderer wichtiger Parameter bestimmt die Reihenfolge der Verknüpfungen in einer Abfrage. Das Ergebnis der Verknüpfung zweier Tabellen wird als zusammengesetzte Tabelle bezeichnet. In der Regel wird diese zusammengesetzte Ergebnistabelle als äußere Tabelle in einer Verknüpfung mit einer weiteren inneren Tabelle verwendet. In diesem Fall handelt es sich also um eine „zusammengesetzte äußere Tabelle". In einigen Fällen, besonders bei Verwendung der schnellen Verknüpfungsaufzählung (Greedy Join Enumeration), ist es sinnvoll, das Ergebnis der Verknüpfung zweier Tabellen zur inneren Tabelle einer späteren Verknüpfung zu machen. Wenn die innere Tabelle einer Verknüpfung selbst aus dem Ergebnis einer Verknüpfung zweier oder mehrerer Tabellen besteht, spricht man davon, dass der Plan eine „zusammengesetzte innere Tabelle" enthält. Betrachten Sie zum Beispiel die folgende Abfrage:

```
SELECT COUNT(*)
FROM T1, T2, T3, T4
WHERE T1.A = T2.A AND
      T3.A = T4.A AND
      T2.Z = T3.Z
```

Hier könnte es von Vorteil sein, die Tabellen T1 und T2 zu verknüpfen (T1xT2), anschließend die Tabellen T3 und T4 zu verknüpfen (T3xT4) und schließlich das Ergebnis der ersten Verknüpfung als äußere Tabelle und das Ergebnis der zweiten Verknüpfung als innere Tabelle auszuwählen. Im daraus resultierenden Plan ((T1xT2) x (T3xT4)) ist das Ergebnis der Verknüpfung (T3xT4) eine zusammengesetzte innere Tabelle. Abhängig von der Abfrageoptimierungsklasse belegt das Optimierungsprogramm die maximale Anzahl von Tabellen, die als innere Tabelle einer Verknüpfung verwendet werden können, mit unterschiedlichen Einschränkungen. Zusammengesetzte innere Tabellen sind bei den Optimierungsklassen 5, 7 und 9 zulässig.

Replizierte Übersichtstabellen

Durch die Verwendung replizierter Übersichtstabellen in einer Umgebung mit partitionierten Datenbanken können Sie die Leistung verbessern, indem Sie die Datenbank vorberechnete Werte der Basistabellendaten verwalten lassen. Zum Beispiel würde die im Folgenden gezeigte Abfrage von der Erstellung der nachfolgend definierten replizierten Übersichtstabelle profitieren. Dazu gelten folgende Annahmen:

- Die Tabelle SALES (Verkauf) befindet sich im Tabellenbereich REGIONTABLESPACE, der in mehrere Partitionen unterteilt ist, und ist über die Spalte REGION partitioniert.

- Die Tabellen EMPLOYEE (Mitarbeiter) und DEPARTMENT (Abteilung) sind in einer Knotengruppe mit Einzelpartition.

Anschließend erstellen Sie eine replizierte Übersichtstabelle auf der Basis der Informationen in der Tabelle EMPLOYEE.

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM   EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

Wenn die replizierte Übersichtstabelle erstellt ist, wird ihr Inhalt durch Ausführung der folgenden Anweisung aktualisiert:

```
REFRESH TABLE R_EMPLOYEE;
```

Im folgenden Beispiel werden der Verkauf nach Mitarbeiter, die Summe für die Abteilung und die Gesamtsumme berechnet:

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

Anstatt die Tabelle EMPLOYEE, die nur in einer Datenbankpartition ist, zu verwenden, verwendet der Datenbankmanager die Tabelle R_EMPLOYEE , die in jeder der Datenbankpartitionen repliziert wird, in der sich die Tabelle SALES befindet. Der Leistungsvorteil ergibt sich daraus, dass die Mitarbeiterinformationen zur Berechnung der Verknüpfung nicht zu jeder Datenbankpartition über das Netzwerk gesendet werden muss.

Replizierte Übersichtstabellen können als Hilfe bei der Zusammenfassung von Verknüpfungen durch Kollokation verwendet werden. Wenn beispielsweise ein Sternschema mit einer großen Fakttabelle vorliegt, die sich über zwanzig Knoten erstreckt, sind die Verknüpfungen zwischen der Fakttabelle und den Dimensionstabellen dann am effizientesten, wenn diese Tabellen durch Kollokation zusammenfasst werden.

Wenn alle Tabellen in derselben Knotengruppe platziert würden, würde zumindest eine Dimensionstabelle korrekt für eine zusammengefasste Verknüpfung partitioniert. Alle anderen Dimensionstabellen könnten nicht in einer zusammengefassten Verknüpfung verwendet werden, da die Verknüpfungsspalte(n) in der Fakttabelle nicht mit dem Partitionierungsschlüssel der Fakttabelle übereinstimmen würde(n).

Beispielsweise könnte eine Tabelle namens FACT (C1, C2, C3, ...) in C1 partitioniert sein, eine Tabelle namens DIM1 (C1, dim1a, dim1b, ...) in C1, eine Tabelle namens DIM2 (C2, dim2a, dim2b, ...) in C2 usw.

In diesem Beispiel können Sie erkennen, dass die Verknüpfung zwischen FACT und DIM1 korrekt ist, da das Vergleichselement DIM1.C1 = FACT.C1 zusammengefasst würde. Die beiden Tabellen werden in der Spalte C1 partitioniert.

Die Verknüpfung mit DIM2 mit dem Vergleichselement WHERE DIM2.C2 = FACT.C2 kann nicht zusammengefasst werden, da FACT in der Spalte C1 partitioniert wird, nicht in der Spalte C2.

In diesem Fall sollte DIM2 in der Knotengruppe der Fakttabelle repliziert werden. Dadurch kann die Verknüpfung lokal in jeder Partition erstellt werden.

Anmerkung: Die vorliegende Erörterung von replizierten Übersichtstabellen hängt mit der datenbankübergreifenden Replikation zusammen. Datenbankübergreifende Replikation umfasst Subskriptionen, Steuertabellen und Daten in unterschiedlichen Datenbanken und unter unterschiedlichen Betriebssystemen. Wenn Sie sich für datenbankübergreifende Replikation interessieren, finden Sie weitere Informationen in *Replikation Benutzer- und Referenzhandbuch*.

Die Quellentabelle beim Erstellen einer replizierten Übersichtstabelle kann aus einer Knotengruppe mit einem oder mit mehreren Knoten stammen. Meistens handelt es sich um eine kleine Tabelle, die in einer Knotengruppe mit einem Knoten platziert werden kann. Sie können die Datenmenge für die Replikation einschränken, indem Sie nur einen Teil der Spalten der Tabelle angeben, die Zeilenzahl mit Hilfe der verwendeten Vergleichselemente begrenzen oder beide Methoden anwenden, wenn Sie die replizierte Übersichtstabelle erstellen.

Anmerkung: Für das Funktionieren replizierter Übersichtstabellen ist die Option zur Datenerfassung nicht erforderlich.

Die replizierte Übersichtstabelle könnte auch in einer Knotengruppe mit mehreren Knoten erstellt werden. Die Knotengruppe stimmt mit der überein, in der Sie Ihre großen Tabellen platziert haben. In diesem Fall werden in allen Partitionen der Knotengruppe Kopien der Quellentabelle erstellt. Es ist besser, Verknüpfungen zwischen einer großen Fakttabelle und den Dimensionstabellen in dieser Umgebung lokal zu erstellen, als die Quellentabelle an alle Partitionen verteilen zu müssen.

Indizes für replizierte Tabellen werden nicht automatisch erstellt. Indizes werden erstellt und können sich von den in der Quellentabelle erstellten Indizes unterscheiden.

Anmerkung: Sie können keine eindeutigen Indizes für replizierte Tabellen erstellen (und keine Integritätsbedingungen für sie festlegen). Dadurch werden ungültige Integritätsbedingungen vermieden, die in den Quellentabellen nicht vorhanden sind. Diese Integritätsbedingungen sind unzulässig, selbst wenn die Quellentabelle eine identische Integritätsbedingung enthält.

Nach der Verwendung der Anweisung REFRESH sollte RUNSTATS in der replizierten Tabelle wie in anderen Tabellen ausgeführt werden können.

Sie können auf die replizierten Tabellen direkt in einer Abfrage verweisen. Sie können jedoch bei einer replizierten Tabelle nicht das Vergleichselement NODENUMBER() verwenden, um die Tabellendaten in einer bestimmten Partition anzuzeigen.

Anhand der Funktion EXPLAIN können Sie feststellen, ob eine erstellte replizierte Übersichtstabelle benutzt wurde (vorausgesetzt, eine Abfrage hat auf die Quellentabelle verwiesen). Stellen Sie zunächst fest, ob die EXPLAIN-Tabellen vorhanden sind. Erstellen Sie anschließend einen EXPLAIN-Plan für die Anweisung SELECT, für die Sie sich interessieren. Formatieren Sie abschließend die EXPLAIN-Ausgabe mit dem Dienstprogramm db2exfmt.

Ob der vom Optimierungsprogramm ausgewählte Zugriffsplan die replizierte Übersichtstabelle tatsächlich verwendet, hängt von den zu verknüpfenden Informationen ab. Die replizierte Übersichtstabelle könnte u. U. dann nicht verwendet werden, wenn das Optimierungsprogramm feststellt, dass es weniger aufwendig wäre, die ursprüngliche Quellentabelle an die anderen Partitionen der Knotengruppe zu verteilen.

Verknüpfungsstrategien in einer partitionierten Datenbank

In den folgenden Abschnitten werden die Verknüpfungsstrategien beschrieben, die in einer Umgebung mit partitionierten Datenbanken möglich sind. Das DB2-Optimierungsprogramm wählt abhängig von den Anforderungen der jeweiligen Anwendung automatisch die beste Verknüpfungsstrategie aus. Die Verknüpfungsstrategien werden hier vorgestellt, um Ihnen einen besseren Einblick in die verschiedenen Verfahren zu geben. Eine „Tabellenwarteschlange" ist ein Mechanismus zur Übertragung von Zeilen zwischen Datenbankpartitionen oder zwischen Prozessoren in einer Datenbank mit einer Einzelpartition.

In den folgenden Ausführungen bezieht sich der Ausdruck *übertragene* Tabellenwarteschlange auf eine Tabellenwarteschlange, deren Zeilen über ein Hash-Verfahren einer der empfangenden Datenbankpartitionen zugeführt wurden. Eine *rundgesendete* Tabellenwarteschlange (Broadcast) ist eine Tabellenwarteschlange, deren Zeilen an alle empfangenden Datenbankpartitionen gesendet werden (d. h. ohne Hash-Verfahren). In den Abbildungen dieses Abschnitts beziehen sich q1, q2 und q3 (für „Queue") auf die Tabellenwarteschlangen in den Beispielen. Außerdem sind die Tabellen, auf die zugegriffen wird, für den Zweck dieser Beispielszenarios auf zwei Datenbankpartitionen verteilt. Die Pfeile zeigen die Richtung an, in der die Tabellenwarteschlangen übertragen bzw. gesendet werden. Der Koordinatorknoten ist Partition 0.

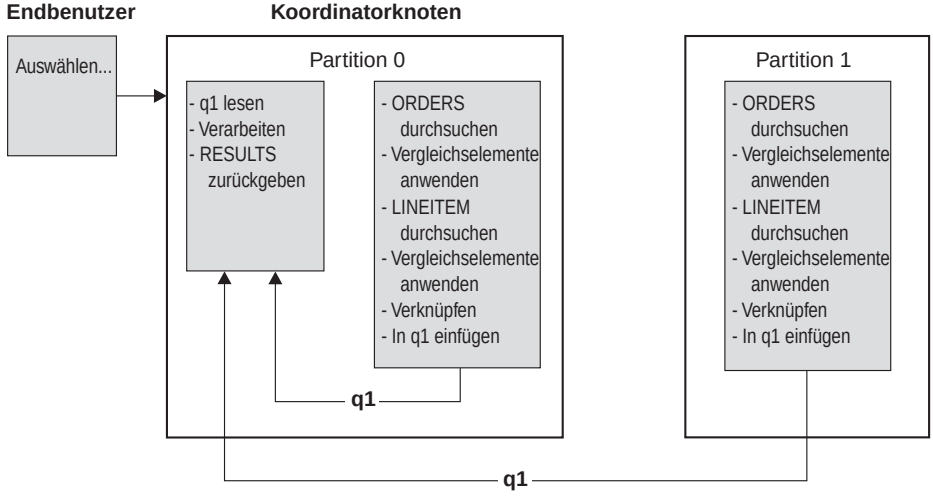
Ein Gesichtspunkt bei Tabellen, die häufig an Verknüpfungen in einer partitionierten Datenbank beteiligt sind, ist die Tabellenkollokation, d. h. die physische Zusammenfassung von Tabellen. Die Tabellenkollokation stellt eine Methode in einer partitionierten Datenbank dar, Daten aus einer Tabelle mit Hilfe eines gleichen Partitionierungsschlüssels mit den Daten aus einer anderen Tabelle in derselben Partition zu suchen. Wenn die Daten einmal in dieser Weise zusammengefasst sind, können zu verknüpfende Daten an einer Abfrage beteiligt sein, ohne als Teil der Aktivität der Abfrage von einer Datenbankpartition in eine andere übertragen werden zu müssen. Nur die Ergebnismenge einer Verknüpfung wird an den Koordinatorknoten übergeben. Weitere Informationen zu diesem Thema finden Sie in „Tabellenkollokation" im Band *Systemverwaltung: Konzept*.

Informationen zu Verknüpfungsabhängigkeiten finden Sie im Handbuch *SQL Reference*.

Zusammengefasste Verknüpfungen

Damit das Optimierungsprogramm eine zusammengefasste Verknüpfung in Erwägung ziehen kann, müssen die zu verknüpfenden Tabellen in einer Partition zusammengefasst sein (Kollokation), und alle Paare mit dem entsprechenden Partitionierungsschlüssel müssen in den Gleichheitsverknüpfungsprädikaten vertreten sein. Abb. 14 zeigt ein Beispiel.

Anmerkung: Replizierte Übersichtstabellen erhöhen die Wahrscheinlichkeit zusammengefasster Verknüpfungen. Weitere Informationen finden Sie in „Replizierte Übersichtstabellen" auf Seite 213.

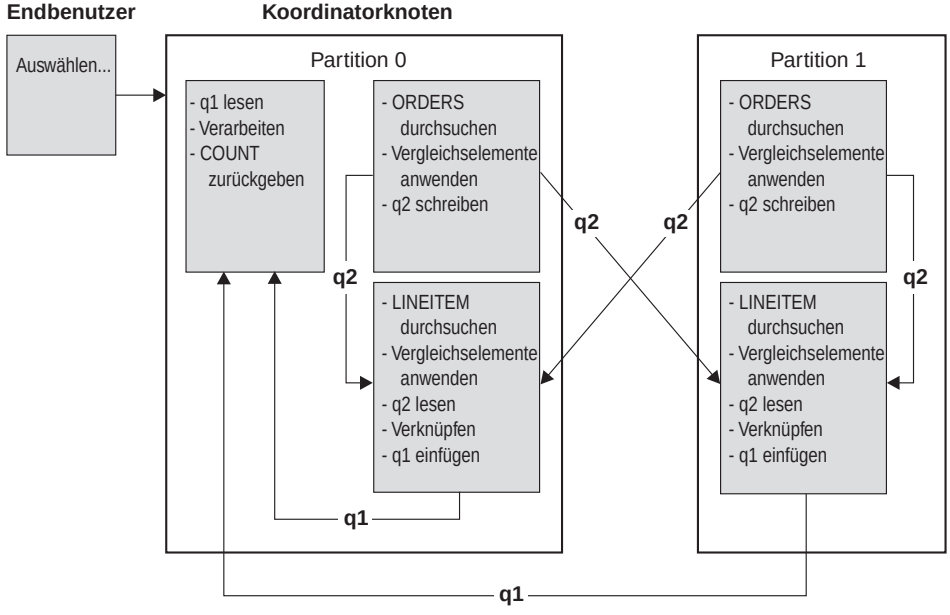


Die beiden Tabellen LINEITEM und ORDERS werden über die Spalte ORDERKEY partitioniert. Die Verknüpfung erfolgt lokal in jeder Datenbankpartition.
In diesem Beispiel wird folgendes Vergleichselement für die Verknüpfung angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 14. Beispiel für eine zusammengefasste Verknüpfung

Verknüpfungen mit rundgesendeter äußerer Tabelle

Diese parallele Verknüpfungsstrategie kann angewandt werden, wenn es zwischen den zu verknüpfenden Tabellen keine Gleichheitsverknüpfungsprädikate gibt. Sie kann außerdem in solchen Fällen verwendet werden, in denen sie die Methode mit dem geringsten Aufwand darstellt. Solche Fälle liegen typischerweise dann vor, wenn eine sehr umfangreiche und eine sehr kleine Tabelle an der Verknüpfung beteiligt sind, von denen keine über die Spalten, auf die die Vergleichselemente angewandt werden, partitioniert ist. Anstatt beide Tabellen auf Partitionen zu verteilen, kann es „billiger" sein, die kleine Tabelle an alle Partitionen mit der größeren Tabelle rundzusenden. Abb. 15 zeigt ein Beispiel.

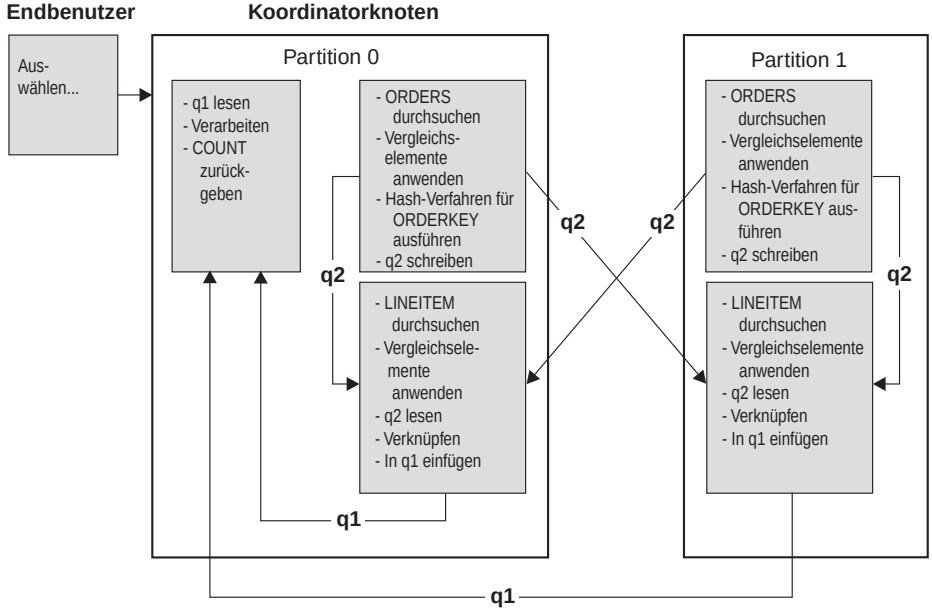


Die Tabelle ORDERS wird an alle Datenbankpartitionen mit der Tabelle LINEITEM gesendet.
Tabellenwarteschlange q2 wird im Rundsendebetrieb an alle Datenbankpartitionen der inneren Tabelle gesendet.

Abbildung 15. Beispiel für eine Verknüpfung mit rundgesendeter äußerer Tabelle

Verknüpfungen mit übertragener äußerer Tabelle

Bei dieser Verknüpfungsstrategie wird jede Zeile der äußeren Tabelle an eine Datenbankpartition der inneren Tabelle (entsprechend den Partitionierungsattributen der inneren Tabelle) übertragen. Die Verknüpfung erfolgt in dieser Datenbankpartition. Abb. 16 zeigt ein Beispiel.



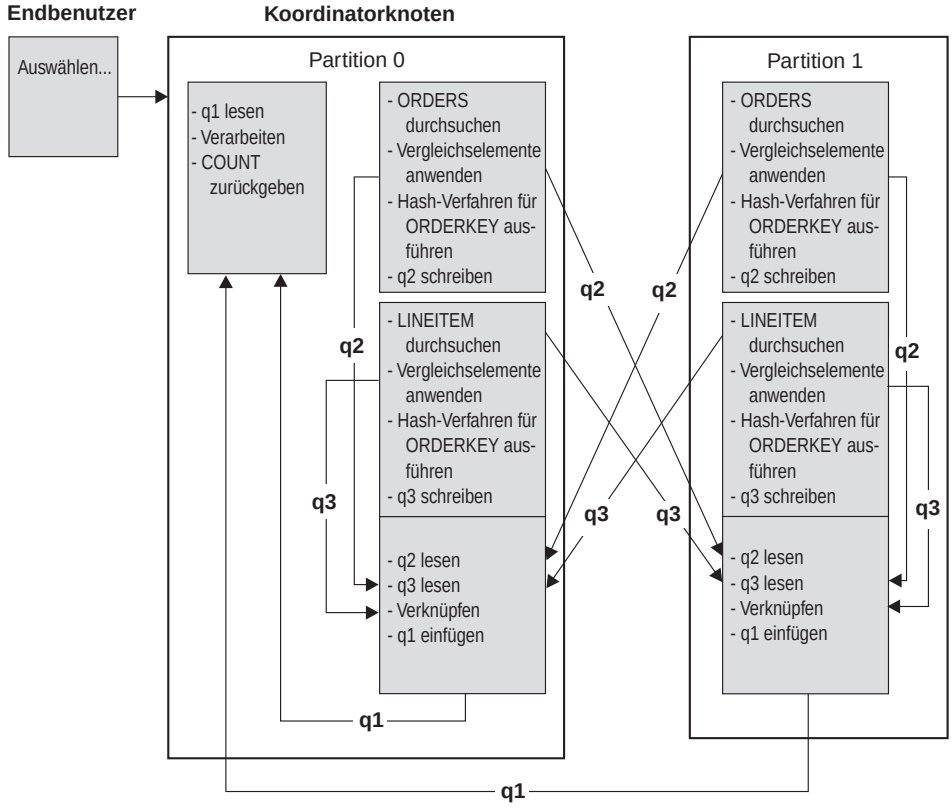
Die Tabelle LINEITEM wird über die Spalte ORDERKEY partitioniert.
Die Tabelle ORDERS wird über eine andere Spalte partitioniert.
Für die Tabelle ORDERS wird das Hash-Verfahren ausgeführt, und sie wird an die richtige Datenbankpartition der Tabelle LINEITEM gesendet.

In diesem Beispiel wird folgendes Vergleichselement für die Verknüpfung angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 16. Beispiel für eine Verknüpfung mit übertragener äußerer Tabelle

Verknüpfungen mit übertragener innerer und äußerer Tabelle

Bei dieser Strategie werden Zeilen der äußeren und inneren Tabellen entsprechend den Werten der Verknüpfungsspalten an eine Gruppe von Datenbankpartitionen übertragen. Die Verknüpfung erfolgt in diesen Datenbankpartitionen. Abb. 17 zeigt ein Beispiel.

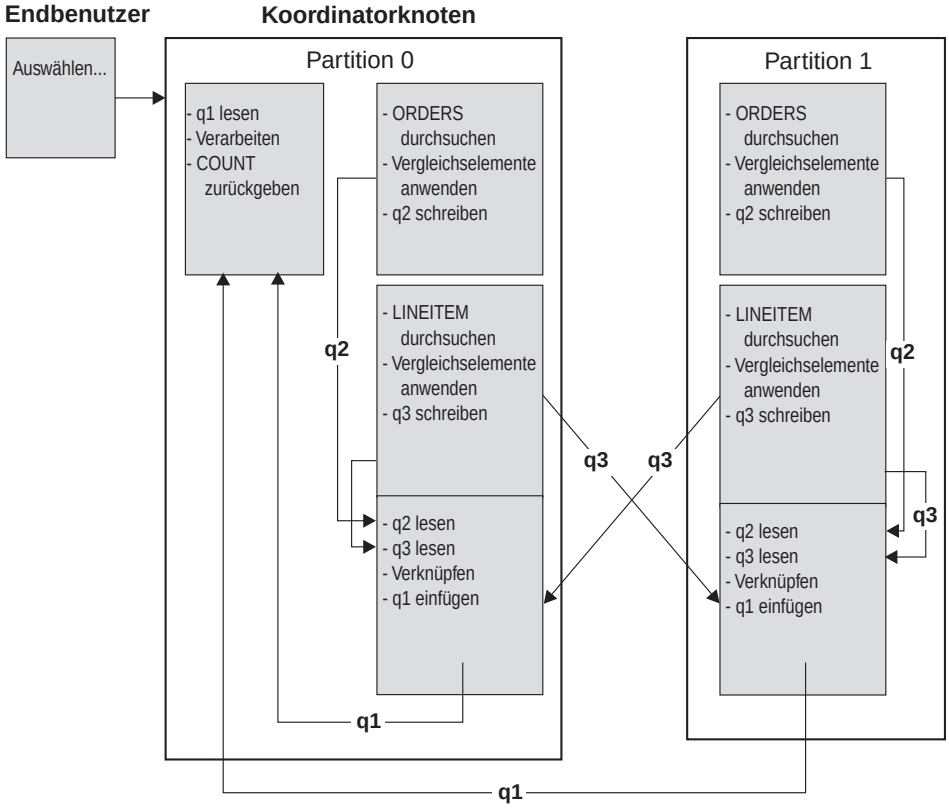


Es wird keine Tabelle über die Spalte ORDERKEY partitioniert.
Für beide Tabellen wird das Hash-Verfahren ausgeführt, und sie werden an neue Datenbankpartitionen gesendet und dort verknüpft.
Beide Tabellenwarteschlangen q2 und q3 werden übertragen.
In diesem Beispiel wird folgendes Vergleichselement für die Verknüpfung angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY

Abbildung 17. Beispiel für eine Verknüpfung mit übertragener innerer und äußerer Tabelle

Verknüpfungen mit rundgesendeter innerer Tabelle

Bei dieser Strategie wird die innere Tabelle an alle Datenbankpartitionen der äußeren Tabelle rundgesendet. Abb. 18 zeigt ein Beispiel.

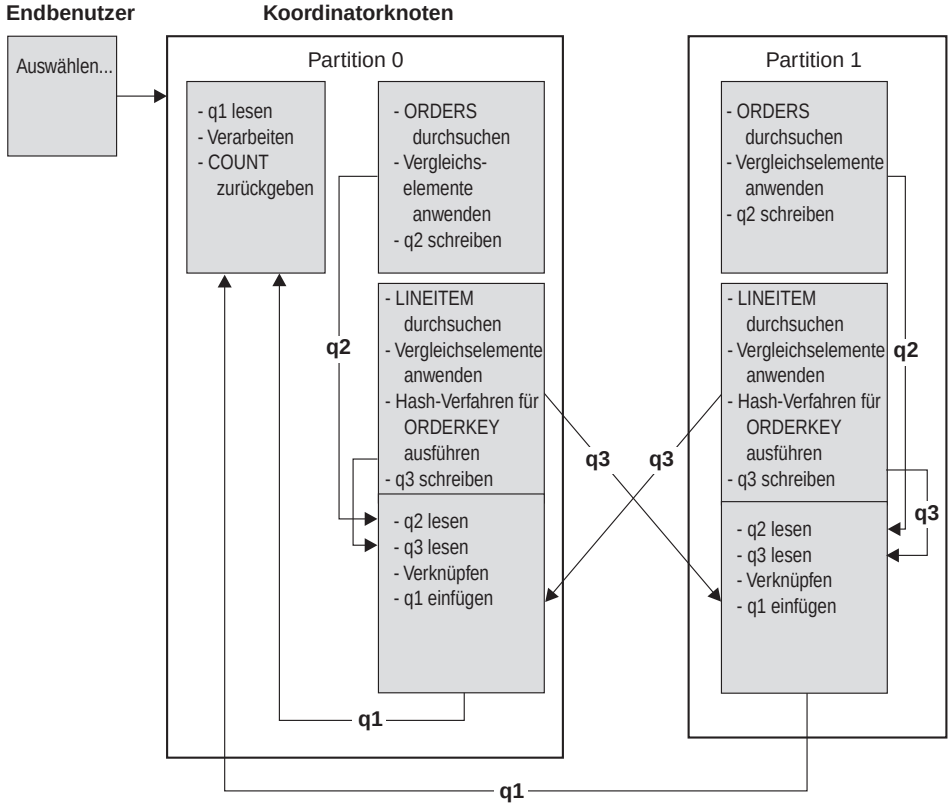


Die Tabelle LINEITEM wird an alle Datenbankpartitionen mit der Tabelle ORDERS gesendet.
Tabellenwarteschlange q3 wird im Rundsendebetrieb an alle Datenbankpartitionen der äußeren Tabelle gesendet.

Abbildung 18. Beispiel für eine Verknüpfung mit rundgesendeter innerer Tabelle

Verknüpfungen mit übertragener innerer Tabelle

Bei dieser Verknüpfungsstrategie wird jede Zeile der inneren Tabelle an eine Datenbankpartition der äußeren Tabelle (entsprechend den Partitionierungsattributen der äußeren Tabelle) übertragen. Die Verknüpfung erfolgt in dieser Datenbankpartition. Abb. 19 zeigt ein Beispiel.



Die Tabelle ORDERS wird über die Spalte ORDERKEY partitioniert.
Die Tabelle LINEITEM wird über eine andere Spalte partitioniert.
Für die Tabelle LINEITEM wird das Hash-Verfahren ausgeführt, und sie wird an die richtige Datenbankpartition der Tabelle ORDERS gesendet.
In diesem Beispiel wird folgendes Vergleichselement für die Verknüpfung angenommen:
ORDERS.ORDERKEY = LINEITEM.ORDERKEY.

Abbildung 19. Beispiel für eine Verknüpfung mit übertragener innerer Tabelle

Tabellenwarteschlangen

Eine Tabellenwarteschlange hat folgende Funktionen:

- Übertragen von Tabellendaten von einer Datenbankpartition zu einer anderen bei Verwendung partitionsübergreifender Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung partitionsinterner Parallelität
- Übertragen von Tabellendaten innerhalb einer Datenbankpartition bei Verwendung einer einzigen Datenbankpartition

Eine Tabellenwarteschlange dient zur Übergabe der Daten in eine einzige Richtung.

Der Compiler entscheidet, wo Tabellenwarteschlangen erforderlich sind, und nimmt sie in den Plan auf. Bei der Ausführung des Plans werden die Tabellenwarteschlangen durch die Verbindungen zwischen den Datenbankpartitionen initiiert. Die Tabellenwarteschlangen werden am Ende der Prozesse wieder geschlossen.

Es gibt verschiedene Arten von Tabellenwarteschlangen:

- *Asynchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als asynchron bezeichnet, weil sie Zeilen vor einer FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, vorablesen. Wenn eine FETCH-Anweisung abgesetzt wird, wird die Zeile aus der Tabellenwarteschlange abgerufen.

  Asynchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung verwenden. Wenn Sie Zeilen nur mit FETCH abrufen, ist eine asynchrone Tabellenwarteschlange schneller.
- *Synchrone Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden als synchron bezeichnet, weil sie für jede FETCH-Anweisung, die durch eine Anwendung abgesetzt wird, eine Zeile lesen. In jeder Datenbankpartition wird der Cursor in die nächste Zeile gesetzt, die aus der jeweiligen Datenbankpartition zu lesen ist.

  Synchrone Tabellenwarteschlangen werden verwendet, wenn Sie die Klausel FOR FETCH ONLY in der SELECT-Anweisung nicht angeben. In einer Umgebung mit partitionierten Datenbanken verwendet der Datenbankmanager synchrone Tabellenwarteschlangen, wenn Zeilen aktualisiert werden.
- *Tabellenwarteschlangen für Mischverknüpfung.* In dieser Art von Tabellenwarteschlangen wird die Reihenfolge gewahrt.
- *Reguläre Tabellenwarteschlangen.* Diese Tabellenwarteschlangen sind nicht für Mischverknüpfungen geeignet. Bei ihnen bleibt die Reihenfolge nicht erhalten.

- *Empfangende Tabellenwarteschlangen.* Diese Tabellenwarteschlangen werden mit korrelierten Unterabfragen verwendet. Die Korrelationswerte werden an die Unterabfrage übergeben, und die Ergebnisse mit Hilfe dieser Art von Tabellenwarteschlange an den übergeordneten Abfrageblock zurückgeliefert.

Einfluss des Sortierens auf das Optimierungsprogramm

Wenn das Optimierungsprogramm einen Zugriffsplan auswählt, kalkuliert es die Auswirkungen einer Sortierung von Daten auf die Leistung mit ein. Sortieroperationen werden durchgeführt, wenn es keinen Index gibt, über den die angeforderte Reihenfolge der abgerufenen Daten hergestellt werden kann. Eine Sortieroperation kann außerdem erfolgen, wenn sie vom Optimierungsprogramm als weniger aufwendig als eine Indexsuche eingestuft wird. Das Optimierungsprogramm kann beim Sortieren von Daten eine der folgenden Aktionen ausführen:

- Über-Pipe-Leiten der Sortierergebnisse, wenn die Abfrage ausgeführt wird. Siehe Abschnitte „Über Pipe geleitete und nicht über Pipe geleitete Sortierungen" und „Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung" auf Seite 103.
- Interne Behandlung der Sortierung innerhalb des Datenbankmanagers. Siehe Abschnitt „Pushdown-Operatoren für Spaltenberechnungen und Sortierungen" auf Seite 226.

Über Pipe geleitete und nicht über Pipe geleitete Sortierungen

Wenn nach Beendigung der Sortierung die endgültige sortierte Liste von Daten in einem einzigen sequenziellen Vorgang gelesen werden kann, können die Ergebnisse über eine *Pipe* geleitet werden. Dieses *Piping* ist schneller als die Verwendung anderer (nicht über Pipe geleiteter) Mechanismen zur Übertragung der Sortierergebnisse. Das Optimierungsprogramm wählt, wenn möglich, die Pipe zur Übergabe der Sortierergebnisse.

Unabhängig davon, ob eine Sortierung über eine Pipe geleitet wird, ist die Dauer der Sortierung von einer Reihe von Faktoren abhängig, wie z. B. der Anzahl der zu sortierenden Zeilen, der Sortierschlüsselgröße und der Zeilenlänge. Wenn die zu sortierenden Zeilen mehr als den im Zwischenspeicher für Sortierlisten verfügbaren Speicherbereich in Anspruch nehmen, werden mehrere Sortierarbeitsgänge durchgeführt, wobei in jedem Arbeitsgang eine Untermenge der Gesamtmenge von Zeilen sortiert wird. Jeder Arbeitsgang des Sortiervorgangs wird in einer temporären Tabelle im Pufferpool gespeichert. (Im Rahmen der Pufferpoolverwaltung ist es möglich, dass Seiten aus dieser temporären Tabelle auf Platte geschrieben werden.) Wenn alle Arbeitsgänge des Sortiervorgangs abgeschlossen sind, müssen die sortierten Untermengen zu einer einzigen sortierten Menge von Zeilen zusammengefügt werden.

Wenn die Sortierung über eine Pipe geleitet wird, werden die Zeilen nach dem Zusammenfügen direkt an die Services für relationale Daten (Relational Data Services) weitergegeben.

Weitere Informationen finden Sie in „Anzeichen für Probleme bei der Sortierleistung" auf Seite 310 oder in der Diskussion des Konfigurationsparameters *sortheap* im Abschnitt „Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung" auf Seite 103.

Pushdown-Operatoren für Spaltenberechnungen und Sortierungen

In einigen Fällen kann sich das Optimierungsprogramm entscheiden, einen Sortiervorgang oder eine Spaltenberechnung (Aggregation) von der Komponente der Services für relationale Daten an die Komponente der Datenverwaltungsservices zu „verschieben" („Pushdown"). Die Verschiebung dieser Operationen verbessert die Leistung, da nun die Datenverwaltungsservices Daten direkt an eine Sortier- oder Spaltenberechnungsroutine übergeben können. Ohne diese Verschiebung würden die Datenverwaltungsservices diese Daten zunächst an die Services für relationale Daten übergeben, die anschließend wiederum mit den Sortier- bzw. Spaltenberechnungsroutinen kommunizieren würden. Die folgende Abfrage beispielsweise kann von dieser Art der Optimierung profitieren:

```
SELECT WORKDEPT, AVG(SALARY) AS AVG_DEPT_SALARY
    FROM   EMPLOYEE
  GROUP BY WORKDEPT
```

Spaltenberechnung beim Sortieren

Wenn das Sortieren dazu dient, die erforderliche Reihenfolge für eine Operation GROUP BY herzustellen, hat das Optimierungsprogramm die Möglichkeit, einige oder alle Spaltenberechnungen (Aggregation) für GROUP BY während des Sortierens durchzuführen. Dies ist vorteilhaft, wenn die Anzahl der Zeilen in jeder Gruppe sehr groß ist. Es wird sogar noch vorteilhafter, wenn die Durchführung eines Teils der Gruppierung während des Sortierens die Notwendigkeit, dass die Sortierung einen Überlauf auf die Festplatte verursacht, verringert oder ausschließt.

Wenn Spaltenberechnung in einer Sortierung verwendet wird, sind bis zu drei Phasen der Spaltenberechnung erforderlich, um sicherzustellen, dass die richtigen Ergebnisse berechnet werden. Die erste Phase der Spaltenberechnung, die „partielle Spaltenberechnung" errechnet die Ergebniswerte, bis der Sortierspeicher voll ist. Die partielle Spaltenberechnung ist der Prozess, durch den nicht berechnete Daten entgegengenommen und partielle Ergebniswerte erstellt werden. Wenn der Sortierspeicher voll ist, läuft der Rest der Daten auf die Festplatte über und enthält alle partiellen Spaltenberechnungsergebnisse, die während des aktuellen Füllens des Sortierspeichers berechnet wurden. Nach dem Zurücksetzen des Sortierspeichers werden neue Spaltenberechnungen gestartet.

Die zweite Phase der Spaltenberechnung, die „Zwischenberechnung" nimmt alle übergelaufenen Sortierdurchläufe und setzt die Spaltenberechnung für die Gruppierschlüssel fort. Die Spaltenberechnung kann nicht zu Ende geführt werden, weil die Gruppierschlüsselspalten eine Untergruppe der Partitionierungsschlüsselspalten sind. Die Zwischenberechnung nimmt vorhandene partielle Spaltenberechnungsergebnisse und produziert neue partielle Berechnungsergebnisse. Diese Phase ist wahlfrei und wird sowohl für partitionsinterne Parallelität als auch für partitionsübergreifende Parallelität verwendet. Im letzteren Fall ist die Gruppierung beendet, wenn ein globaler Gruppierschlüssel verfügbar ist. Bei partitionsübergreifender Parallelität würde dies eintreten, wenn der Gruppierschlüssel eine Untergruppe des Partitionierungsschlüssels wäre, der Gruppen über Partitionen hinweg teilt und so eine erneute Partitionierung zur Beendigung der Spaltenfunktion erforderlich machte. Ein ähnlicher Fall liegt vor, wenn bei partitionsinterner Parallelität jeder Agent seine übergelaufenen Sortierdurchgänge zusammengefügt hat, bevor auf einen einzigen Agenten reduziert wird, um die Spaltenberechnung zu beenden.

Die letzte Phase der Spaltenberechnung, die „Endberechnung", greift alle partiellen Berechnungsergebnisse auf und führt die Spaltenberechnung zu Ende. Die Endberechnung greift partielle Berechnungsergebnisse auf und produziert Endergebnisse. Dieser Schritt findet immer in einem Operator GROUP BY statt. Das Sortieren kann die Spaltenberechnung nicht bis zu Ende durchführen, weil es keine Garantie gibt, dass die Sortierung nicht geteilt wird. Die Komplettberechnung nimmt nicht berechnete Daten auf und produziert Endergebnisse. Diese Methode der Spaltenberechnung wird in der Regel dann verwendet, wenn die Gruppierdaten bereits in der richtigen Reihenfolge vorliegen und keine Partitionierung die Verwendung dieser Methode verhindert.

Optimierungsstrategien für partitionsinterne Parallelität

Das Optimierungsprogramm kann einen Zugriffsplan so wählen, dass eine Abfrage parallel innerhalb einer Datenbankpartition ausgeführt wird, wenn ein Grad von Parallelität bei der Kompilierung der SQL-Anweisung angegeben wird.

Während der Ausführung werden mehrere Datenbankagenten, so genannte „Subagenten", zur Ausführung der Abfrage erstellt. Die Anzahl von Subagenten ist kleiner oder gleich dem Grad von Parallelität, die bei der Kompilierung der SQL-Anweisung festgelegt wurde.

Weitere Informationen zur Einstellung des Grads der Parallelität für SQL-Anweisungen finden Sie in „Parallele Verarbeitung von Anwendungen” auf Seite 100. Weitere Informationen zu Agenten und Subagenten finden Sie in „Datenbankagenten” auf Seite 320.

In einer partitionierten Datenbank gilt der Grad der Parallelität für jede Partition. Beispielsweise wird der Teil einer Abfrage, der in einer bestimmten Datenbankpartition ausgeführt wird, entsprechend dem Grad der Parallelität, der in dieser Datenbankpartition für die vorliegende SQL-Anweisung definiert wurde, noch weiter in parallele Ausführungseinheiten unterteilt.

Der Zugriffsplan wird parallelisiert, indem er in einen Teil, der von jedem Subagenten und einen Teil, der vom koordinierenden Agenten ausgeführt wird, aufgeteilt wird. Die Subagenten übergeben Daten über Tabellenwarteschlangen an den koordinierenden Agenten oder andere Subagenten. In einer partitionierten Datenbank können Subagenten Daten an Subagenten in anderen Datenbankpartitionen senden oder von ihnen empfangen.

In diesem Abschnitt werden die Parallelisierungsstrategien innerhalb einer einzelnen Datenbankpartition beschrieben.

Strategien für paralleles Suchen

Tabellensuchen und Indexsuchen können parallel in derselben Tabelle oder demselben Index ausgeführt werden. Für parallele Tabellensuchen wird die Tabelle in Seiten- oder Zeilenbereiche unterteilt. Je ein Bereich von Seiten oder Zeilen wird einem Subagenten zugewiesen. Ein Subagent durchsucht den ihm zugewiesenen Bereich und erhält einen anderen Bereich zugewiesen, wenn er mit dem Durchsuchen des aktuellen Bereichs fertig ist.

Für parallele Indexsuchen wird der Index in Bereiche von Datensätzen entsprechend den Indexschlüsselwerten und der Anzahl von Indexeinträgen für einen Schlüsselwert unterteilt. Die parallele Indexsuche wird wie die parallele Tabellensuche mit Subagenten durchgeführt, denen jeweils ein Bereich von Datensätzen zugewiesen wird. Einem Subagenten wird ein neuer Bereich zugewiesen, wenn er die Suche im aktuellen Bereich beendet hat.

Die Sucheinheit (Seite oder Zeile) sowie die Suchunterteilung werden vom Optimierungsprogramm festgelegt.

Beim parallelen Suchen wird die Arbeit gleichmäßig unter den Subagenten verteilt. Der Zweck der parallelen Suche besteht darin, die Belastung unter den Subagenten ausgewogen zu verteilen und sie gleichmäßig auszulasten. Wenn die Anzahl aktiver Subagenten gleich der Anzahl verfügbarer Prozessoren ist und die Platten nicht mit E/A-Anforderungen überlastet sind, werden die Ressourcen der Maschine effektiv genutzt.

Andere Zugriffsplanoperationen können eine unausgewogene Datenverteilung bei der Ausführung der Abfrage verursachen. Das Optimierungsprogramm wählt die Parallelstrategien so aus, dass die Datenausgewogenheit erhalten bleibt.

Strategien für paralleles Sortieren

Das Optimierungsprogramm kann eine der folgenden parallelen Sortierstrategien auswählen:

Reihumverteiltes Sortieren

Dies kann auch als „Umverteilungssortieren" bezeichnet werden. Dabei handelt es sich um eine effiziente Sortiermethode im gemeinsamen Speicher, bei der versucht wird, die Daten so gleichmäßig wie möglich an alle Subagenten zu verteilen. Zur Realisierung der gleichmäßigen Verteilung wird eine Art Reihumverteilungsalgorithmus verwendet. Zunächst wird ein Sortiervorgang für jeden Subagenten erstellt. Während der Einfügephase fügen Subagenten Zeilen reihum in jeden der einzelnen Sortiervorgänge ein. Dadurch wird eine gleichmäßigere Verteilung von Daten erreicht.

Partitioniertes Sortieren

Dies ist dem reihumverteilten Sortieren insofern ähnlich, als dass für jeden Subagenten ein Sortiervorgang erstellt wird. Die Subagenten wenden eine Hash-Funktion auf die Sortierspalten an, um festzulegen, in welchen Sortiervorgang eine Zeile eingefügt werden sollte. Wenn beispielsweise die innere und die äußere Tabelle einer Mischverknüpfung in einem partitionierten Sortiervorgang sind, kann ein Subagent mit Hilfe einer Mischverknüpfung die entsprechenden Partitionen verknüpfen. Dadurch kann die Mischverknüpfung parallel erfolgen.

Repliziertes Sortieren

Diese Art der Sortierung wird in Fällen verwendet, in denen alle Subagenten die gesamte Sortierausgabe benötigen. Es wird nur ein Sortiervorgang erstellt. Beim Einfügen von Zeilen in den Sortiervorgang sind die Subagenten synchronisiert. Nach Beendigung der Sortierung liest jeder Subagent das gesamte Sortierergebnis. Diese Art der Sortierung kann verwendet werden, um den Datenstrom wieder auszugleichen, wenn die Anzahl von Zeilen gering ist.

Gemeinsames Sortieren

Diese Art der Sortierung entspricht der replizierten Sortierung, abgesehen davon, dass die Subagenten eine parallele Suche über das Sortierergebnis öffnen. Dadurch werden die Daten unter den Subagenten ähnlich wie bei der reihumverteilten Sortierung verteilt.

Parallele temporäre Tabellen

Subagenten können kooperieren, um eine temporäre Tabelle durch Einfügen von Zeilen in dieselbe Tabelle zu erstellen. Eine solche Tabelle ist eine `gemeinsame temporäre Tabelle`. Die Subagenten können private oder parallele Suchoperationen über die gemeinsame temporäre Tabelle öffnen, je nachdem, ob der Datenstrom zu replizieren oder zu partitionieren ist.

Strategien für parallele Spaltenberechnung

Spaltenberechnungen (Aggregation) können von Subagenten parallel durchgeführt werden. Eine Spaltenberechnung setzt voraus, dass die Daten nach den Gruppierungsspalten geordnet sind. Wenn ein Subagent sicher sein kann, dass er alle Zeilen für eine Reihe von Gruppierungsspaltenwerten erhält, kann er eine vollständige Spaltenberechnung (Aggregation) durchführen. Dies ist möglich, wenn der Strom bereits aufgrund einer früheren partitionierten Sortierung auf den Gruppierungsspalten partitioniert ist.

Andernfalls kann der Subagent eine partielle Spaltenberechnung durchführen und eine andere Strategie zur Vervollständigung der Spaltenberechnung anwenden. Einige dieser Strategien sind:

- Senden der teilweise berechneten Daten an den Koordinationsagenten über eine Tabellenwarteschlange für Mischverknüpfungen. Der Koordinationsagent vervollständigt die Spaltenberechnung.
- Einfügen der teilweise berechneten Daten in eine partitionierte Sortierung. Die Sortierung ist auf den Gruppierungsspalten partitioniert. Dadurch ist sichergestellt, dass alle Zeilen für eine Reihe von Gruppierungsspalten in einer Sortierpartition enthalten sind.
- Wenn der Strom zu Ausgleichszwecken repliziert werden muss, können die teilweise berechneten Daten in eine replizierte Sortierung eingefügt werden. Die einzelnen Subagenten vervollständigen die Spaltenberechnung über die replizierte Sortierung und erhalten eine identische Kopie des Ergebnisses der Spaltenberechnung.

Parallele Verknüpfungsstrategien

Verknüpfungsoperationen können von Subagenten parallel durchgeführt werden. Parallele Verknüpfungsstrategien werden durch die Merkmale des Datenstroms festgelegt.

Eine Verknüpfung kann parallelisiert werden, indem der Datenstrom nach der inneren und äußeren Tabelle der Verknüpfung partitioniert und/oder repliziert wird. Zum Beispiel kann eine Verknüpfung über Verschachtelungsschleife parallelisiert werden, wenn der äußere Datenstrom aufgrund einer Parallelsuche partitioniert ist und der innere Datenstrom von jedem Subagenten unabhängig neu ausgewertet wird. Eine Mischverknüpfung kann parallelisiert werden, wenn der innere und der äußere Datenstrom aufgrund partitionierter Sortierungen nach ihren Werten partitioniert sind.

Automatische Übersichtstabellen

Übersichtstabellen sind eine leistungsfähige Methode zum Verbessern der Antwortzeit von Abfragen. In vielen Umgebungen, bei denen einige der grundlegenden Strukturen von Abfragen vorhergesehen werden können, können Übersichtstabellen für die folgenden Zwecke verwendet werden:

- Daten aus einer oder mehreren Dimensionen ansammeln
- Daten aus einer Gruppe von Tabellen verknüpfen und ansammeln
- Eine Untergruppe von Daten (d. h. eine sofort für Verarbeitungsoperationen bereite horizontale oder vertikale Partition), auf die allgemein zugegriffen wird, identifizieren
- Eine Tabelle oder einen Teil einer Tabelle in einer Umgebung mit partitionierten Datenbanken erneut partitionieren

In den SQL-Compiler sind Kenntnisse von Übersichtstabellen integriert. Im SQL-Compiler sind das Umschreiben von Abfragen (siehe „Umschreiben der Abfrage durch den SQL-Compiler" auf Seite 176) und das Optimierungsprogramm (siehe „Datenzugriffskonzepte und Optimierung" auf Seite 187) daran beteiligt, Abfragen mit Übersichtstabellen abzugleichen und zu bestimmen, ob eine Übersichtstabelle durch eine Abfrage ersetzt werden soll, die auf die Basistabellen zugreift. Wenn Übersichtstabellen zur Beantwortung von Abfragen verwendet werden, können die EXPLAIN-Einrichtungen (siehe „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251) dafür benutzt werden zu ermitteln, welche Übersichtstabelle ausgewählt wurde. Da Übersichtstabellen sich in vielerlei Hinsicht wie reguläre Tabellen verhalten, treffen die Überlegungen zum Optimieren des Datenzugriffs unter Verwendung von Tabellenbereichsdefinitionen, durch Erstellen von Indizes und Ausgeben von RUNSTATS auch auf Übersichtstabellen zu.

Um Ihnen die Leistungsfähigkeit von Übersichtstabellen zu verdeutlichen, wird im folgenden Beispiel eine mehrdimensionale Analyseabfrage dargestellt und aufgezeigt, wie dabei Übersichtstabellen genutzt werden.

In diesem Beispiel wird von einem Szenario ausgegangen, in dem ein Geschäft eine Reihe von Kunden und eine Reihe von Kreditkartenkonten umfasst. Das Geschäft zeichnet die Gruppe der Transaktionen auf, die mit den Kreditkarten durchgeführt wurden. Alle Transaktionen enthalten eine Reihe von Artikeln, die gemeinsam gekauft wurden. Diese Umgebung wird als Mehrfachstern (Multi-Star) kategorisiert, da zwei große Tabellen vorhanden sind, von denen die eine Transaktionsartikel enthält und die andere die Kauftransaktionen identifiziert, die beide zusammen die Nabe des Sterns bilden.

Es gibt drei hierarchische Dimensionen, die eine Transaktion beschreiben: Produkt, Standort und Zeit. Die Produkthierarchie wird in zwei normalisierten Tabellen aufgezeichnet, die die Produktgruppe und die Produktlinie darstellen. Die Standorthierarchie enthält Informationen zu Ort, Bundesland und Staat und wird in einer einzigen denormalisierten Tabelle dargestellt. Die Zeithierarchie enthält Informationen zu Tag, Monat und Jahr und ist in einem einzigen Datumsfeld codiert. Die Datumsdimensionen werden aus dem Datumsfeld der Transaktion unter Verwendung integrierter Funktionen extrahiert. Es gibt auch andere Tabellen in diesem Szenario, die Kontoinformationen für Kunden und Kundeninformationen darstellen.

Eine Übersichtstabelle wird mit der Summe und der Anzahl der Verkäufe für jede Stufe der folgenden Hierarchien erstellt:
- Produkthierarchie
- Standorthierarchie
- Zeithierarchie bestehend aus Jahr, Monat, Tag

Eine breite Palette von Abfragen kann ihre Antworten aus diesen gespeicherten Ergebnisdaten extrahieren. Im folgenden Beispiel werden die Summe und die Anzahl der Verkäufe für die Dimensionen 'Produktgruppe' (Product Group) und 'Produktlinie (Product Line) berechnet; danach wird dies für die Dimensionen 'Ort' (City), 'Bundesland' (State) und 'Staat' (Country) und zuletzt für die Dimension 'Zeit' (Time) durchgeführt. Das Beispiel enthält auch einige weitere Spalten in der Klausel GROUP BY.

```
CREATE TABLE dba.PG_SALESSUM
  AS (
      SELECT l.id AS prodline, pg.id AS pgroup,
             loc.country, loc.state, loc.city,
             l.name AS linename, pg.name AS pgname,
             YEAR(pdate) AS year, MONTH(pdate) AS month,
             t.status,
             SUM(ti.amount) AS amount,
             COUNT(*) AS count
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
         AND YEAR(pdate) > 1990
      GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
               year(pdate), month(pdate), t.status, l.name, pg.name
     )
DATA INITIALLY DEFERRED REFRESH DEFERRED;

REFRESH TABLE dba.SALESCUBE;
```

Die Übersichtstabelle ist normalerweise viel kleiner als die Basisfakttabellen. Sie können durch Angeben der Option DEFERRED steuern, wann die Übersichtstabelle aktualisiert wird (wie im aufgeführten Beispiel gezeigt).

Abfragen, die solche vorberechneten Summen nutzen können, sind unter anderem:
- Verkaufsdaten nach Monat und Produktgruppe
- Gesamtvertrieb für Jahre nach 1990
- Verkauf für 1995 oder 1996
- Summe des Verkaufs für eine Produktgruppe oder Produktlinie
- Summe des Verkaufs für eine bestimmte Produktgruppe oder Produktlinie UND für 1995, 1996
- Summe des Verkaufs für ein bestimmtes Land

Obwohl die präzise Anwort für keine dieser Abfragen in der Übersichtstabelle enthalten ist, könnte der Aufwand zur Berechnung der Antwort mit Hilfe der Übersichtstabelle erheblich geringer ausfallen als bei Verwendung der umfangreichen Basistabelle, da ein Teil der für die Antwort benötigten Berechnungen bereits erfolgt ist. Aufwendige Verknüpfungen, Sortierungen und Spaltenberechnungen von Basisdaten werden mit Hilfe von Übersichtstabellen reduziert oder sogar vermieden.

Es folgen Beispielabfragen, die erhebliche Leistungsverbesserungen erfahren würden, weil sie die in der Übersichtstabelle bereits berechneten Ergebnisse verwenden könnten. Das erste Beispiel liefert die Gesamtverkäufe für 1995 und 1996:

```
SET CURRENT REFRESH AGE=ANY

SELECT YEAR(pdate) AS year, SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY year(pdate);
```

Das zweite Beispiel liefert die Gesamtverkäufe nach Produktgruppe für 1995 und 1996:

```
SET CURRENT REFRESH AGE=ANY

SELECT pg.id AS "PRODUCT GROUP",
       SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY pg.id;
```

Deutlichere Verbesserungen in der Antwortzeit für solche Abfragen können mit größeren Datenbanken erreicht werden. Dies liegt daran, dass die Übersichtstabelle langsamer anwächst als die Basistabelle. Ein Vorteil von Übersichtstabellen besteht darin, dass DB2 Universal Database sie dazu verwendet, überlappende Arbeit zwischen Abfragen effektiv zu beseitigen, indem die Berechnung nur einmal bei der Erstellung der Übersichtstabellen durchgeführt und ihr Inhalt für eine sehr große Zahl von Abfragen erneut verwendet wird.

Abfragecompilerphasen für zusammengeschlossene Datenbanken

In diesem Abschnitt werden die zusätzlichen Phasen der Abfrageverarbeitung in einem System zusammengeschlossener Datenbanken beschrieben. Es enthält darüber hinaus Empfehlungen für die Verbesserung der Leistung von Abfragen für zusammengeschlossene Datenbanken. Zu den Hauptthemen gehören:

- „Pushdown-Analyse"
- „Generierung von fernem SQL und globale Optimierung" auf Seite 244.

Pushdown-Analyse

Die Pushdown-Analyse gibt dem DB2-Optimierungsprogramm darüber Auskunft, ob eine Operation an einer fernen Datenquelle durchgeführt werden kann. Bei dieser Operation kann es sich um eine Funktion, wie zum Beispiel einen relationalen Operator, eine System- oder Benutzerfunktion oder einen SQL-Operator (GROUP BY, ORDER BY usw.) handeln.

Funktionen, die nicht zur Ausführung an die Datenquelle verschoben (Pushdown) werden können, können die Abfrageleistung erheblich beeinträchtigen. Betrachten Sie die Konsequenzen für den Fall, dass ein selektives Vergleichselement lokal anstatt an der Datenquelle ausgewertet werden muss. Dieses Verfahren würde DB2 zwingen, die gesamte Tabelle von der fernen Datenquelle abzurufen und anschließend lokal über das Vergleichselement zu filtern. Wenn die Netzwerkkapazitäten begrenzt sind und die Tabelle groß ist, könnte die Leistung beeinträchtigt werden.

Operatoren, die nicht an die Datenquelle verschoben werden, können die Abfrageleistung ebenfalls erheblich beeinträchtigen. Wenn zum Beispiel ein Operator GROUP BY für ferne Daten lokal ausgeführt wird, muss DB2 auch in diesem Fall die gesamte Tabelle von der fernen Datenquelle abrufen.

Nehmen Sie zum Beispiel an, dass der Kurzname N1 auf die Tabelle EMPLOYEE in einer Datenquelle unter DB2 für OS/390 verweist. Nehmen Sie weiter an, dass die Tabelle 10.000 Zeilen besitzt, wobei eine der Spalten die Familiennamen (Lastname) und eine andere die Gehälter (Salary) enthält. Es sei folgende Anweisung gegeben:

```
SELECT LASTNAME, COUNT(*)  FROM N1
   WHERE LASTNAME  > 'B' AND SALARY > 50000
   GROUP BY LASTNAME;
```

Für diese Abfrage kommen mehrere Möglichkeiten in Betracht:

- Wenn die Sortierfolgen in DB2 und in DB2 für OS/390 übereinstimmen, ist es wahrscheinlich, dass das Abfragevergleichselement an DB2 für OS/390 verschoben wird (Pushdown). In der Regel ist es effizienter, die Ergebnisse an der Datenquelle zu filtern und zu gruppieren, als die gesamte Tabelle nach DB2 zu kopieren und die Operationen lokal auszuführen. Die Pushdown-Analyse in Systemen zusammengeschlossener Datenbanken stellt fest, ob Operationen an der Datenquelle ausgeführt werden können. In diesem Fall kann die Verarbeitung des Vergleichselements und der Operation GROUP BY an der Datenquelle stattfinden.
- Wenn die Sortierfolgen nicht übereinstimmen, stellt die Pushdown-Analyse fest, dass das gesamte Vergleichselement nicht an der Datenquelle ausgewertet werden kann. Allerdings kann das Optimierungsprogramm entscheiden, den Teil mit `SALARY > 50000` des Vergleichselements an die Datenquelle zu verschieben. Der Vergleich des Bereichs muss immer noch in DB2 ausgeführt werden.

- Wenn die Sortierfolgen übereinstimmen und dem Optimierungsprogramm bekannt ist, dass der lokale DB2-Server sehr schnell ist, ist es möglich, dass das Optimierungsprogramm entscheidet, dass die lokale Ausführung der Operation GROUP BY in DB2 das günstigste Verfahren (d. h. mit dem geringsten Aufwand) ist. Das Vergleichselement wird an der Datenquelle ausgeführt. Dies wäre ein Beispiel für die Pushdown-Analyse in Kombination mit globaler Optimierung. DB2 betrachtet die verfügbaren Pfade und wählt dann einen Plan, der der effizienteste ist.

Im allgemeinen besteht das Ziel darin, sicherzustellen, dass Funktionen und Operatoren für die Auswertung auf Datenquellen vom Optimierungsprogramm in Betracht gezogen werden können. Zahlreiche Faktoren können die Entscheidung beeinflussen, ob eine Funktion oder ein SQL-Operator an der fernen Datenquelle ausgewertet wird. Die Hauptfaktoren werden in drei Gruppen behandelt: Server-Merkmale, Kurznamenmerkmale und Abfragemerkmale.

Server-Merkmale mit Auswirkung auf die Pushdown-Möglichkeiten

In den folgenden Abschnitten werden datenquellenspezifische Faktoren behandelt, die Auswirkung auf die Pushdown-Möglichkeiten haben. Diese Faktoren gibt es im Allgemeinen deswegen, weil DB2 die Verwendung einer variantenreichen SQL-Version zur Übergabe von Abfragen ermöglicht. Diese SQL-Version bietet u. U. mehr Funktionalität als die SQL-Version, die von einem Server unterstützt wird, auf den während einer DB2-Abfrage zugegriffen wird. DB2 kann diesen Funktionsmangel auf dem Daten-Server kompensieren, allerdings kann dies dazu führen, dass die Operation in DB2 lokal ausgeführt werden muss.

SQL-Leistungsspektrum: Jede Datenquelle unterstützt eine Variante der SQL-Version und verschiedene Funktionalitätsebenen. Betrachten Sie zum Beispiel die GROUP BY-Liste. Die meisten Datenquellen unterstützen den Operator GROUP BY. Aber einige haben Einschränkungen hinsichtlich der Anzahl von Elementen in der GROUP BY-Liste oder Einschränkungen hinsichtlich der Zulässigkeit bestimmter Ausdrücke in der GROUP BY-Liste. Wenn es eine Einschränkung an der fernen Datenquelle gibt, muss DB2 die Operation GROUP BY eventuell lokal ausführen.

SQL-Einschränkungen: Jede Datenquelle kann unterschiedliche SQL-Einschränkungen haben. Zum Beispiel fordern einige Datenquellen, dass Parametermarken Werte für ferne SQL-Anweisungen einbinden. Daher müssen die Einschränkungen für Parametermarken überprüft werden, um sicherzustellen, dass jede Datenquelle solche Bindeverfahren unterstützen können. Wenn DB2 keine gute Methode zum Einbinden eines Werts für eine Funktion finden kann, muss diese Funktion lokal ausgewertet werden.

SQL-Grenzwerte: DB2 lässt vielleicht die Verwendung größerer ganzer Zahlen (Integer) als die fernen Datenquellen zu. Allerdings können Werte, die das Maximum überschreiten, nicht in Anweisungen eingebettet werden, die an Datenquellen gesendet werden. Daher muss eine Funktion oder ein Operator, die bzw. der mit einer solchen Konstanten operiert, lokal ausgewertet werden.

Server-spezifische Faktoren: In diese Kategorie fallen verschiedene Faktoren. Ein Beispiel ist die Sortierung von NULL-Werten (höchste oder niedrigste Position bzw. abhängig von der angeforderten Reihenfolge). Wenn zum Beispiel der NULL-Wert an der Datenquelle anders sortiert wird als von DB2, können Operationen ORDER BY für einen Ausdruck mit wahlfreier Dateneingabe nicht fern ausgewertet werden.

Sortierfolge: Die Konfiguration einer zusammengeschlossenen Datenbank zur Verwendung derselben Sortierfolge, die von einer Datenquelle verwendet wird, und das Setzen der Serveroption *collating_sequence* auf den Wert 'Y' ermöglicht dem Optimierungsprogramm, ein Verschieben (Pushdown) von Vergleichselementen zum Vergleich von Zeichenbereichen an die Datenquelle in Betracht zu ziehen.

Wenn eine Abfrage aus einem Server für zusammengeschlossene Datenbanken eine Sortierung erfordert, ist die Auswahl der Stelle, an der die Sortierung verarbeitet wird, von verschiedenen Faktoren abhängig. Wenn die Sortierfolge der zusammengeschlossenen Datenbank mit der der Datenquelle übereinstimmt, auf der die abgefragten Daten gespeichert sind, kann die Sortierung an der Datenquelle erfolgen. Bei gleichen Sortierfolgen kann das Optimierungsprogramm entscheiden, ob eine lokale Sortierung oder eine Sortierung an der Datenquelle der effizienteste Weg zur Durchführung der Abfrage ist. In ähnlicher Weise kann ein Vergleich von Zeichendaten, wenn die Abfrage einen solchen erforderlich macht, ebenfalls an der Datenquelle durchgeführt werden.

Numerische Vergleiche können im Allgemeinen an beiden Stellen stattfinden, selbst wenn die Sortierfolgen unterschiedlich sind. Unerwartete Ergebnisse kommen eventuell dann zustande, wenn die Wertigkeit von Nullzeichen zwischen der zusammengeschlossenen Datenbank und der Datenquelle unterschiedlich ist. Analog gilt auch für Vergleichsanweisungen, dass Vorsicht geboten ist, wenn Anweisungen an eine Datenquelle übergeben werden, an der die Groß- und Kleinschreibung nicht unterschieden wird. Die Wertigkeiten der Zeichen ″I″ und ″i″ sind bei einer Datenquelle ohne Unterscheidung der Groß-/Kleinschreibung identisch. DB2 beachtet standardmäßig die Groß-/Kleinschreibung und weist den Zeichen unterschiedliche Wertigkeiten zu.

Wenn die Sortierfolgen der zusammengeschlossenen Datenbank und der Datenquelle voneinander abweichen, ruft DB2 die Daten auf die zusammengeschlossene Datenbank ab, so dass die Sortierung und der Vergleich lokal durchgeführt werden können. Dies geschieht, weil Benutzer die Abfrageergebnisse nach der für den Server der zusammengeschlossenen Datenbank definierten Sortierfolge geordnet erwarten. Durch das lokale Durchführen der Datensortierung, stellt der Server der zusammengeschlossenen Datenbank sicher, dass diese Erwartung erfüllt wird.

Das Abrufen von Daten für lokale Sortierungen und Vergleiche setzt in der Regel den Durchsatz herab. Daher sollte in Erwägung gezogen werden, die zusammengeschlossene Datenbank so zu konfigurieren, dass sie dieselbe Sortierfolge wie die Datenquellen verwendet. So lässt sich der Durchsatz vielleicht erhöhen, weil der Server der zusammengeschlossenen Datenbank die Durchführung von Sortierungen und Vergleichsoperationen an den Datenquellen zulassen kann. Zum Beispiel werden in DB2 UDB für OS/390 Sortierungen, die durch Klauseln ORDER BY definiert werden, mit Hilfe einer Sortierfolge implementiert, die auf einer EBCDIC-Codepage (Extended Binary-Coded Decimal Interchange Code) basiert. Wenn Sie den Server der zusammengeschlossenen Datenbank zum Abrufen von Daten einer Datenquelle unter DB2 für OS/390 verwenden wollen und die Daten mit Hilfe von Klauseln ORDER BY sortiert werden sollen, ist es empfehlenswert, die zusammengeschlossene Datenbank so zu konfigurieren, dass sie eine vordefinierte, auf der EBCDIC-Codepage basierende Sortierfolge verwendet.

Wenn die Sortierfolgen der zusammengeschlossenen Datenbank und der Datenquelle verschieden sind und Sie die Daten nach der Reihenfolge der Sortierfolge der Datenquelle geordnet abrufen müssen, können Sie die Abfrage im Durchgangsmodus übergeben oder die Abfrage in einer Datenquellensicht definieren.

Im Handbuch *Systemverwaltung: Konzept* finden Sie weitere Informationen zu Sortierfolgen und deren Konfiguration. Tabelle 8 auf Seite 122 enthält weitere Informationen zur Serveroption *collating_sequence*.

Serveroptionen: Verschiedene Serveroptionen können die Pushdown-Möglichkeiten beeinflussen. Prüfen Sie insbesondere Ihre Einstellungen für die Serveroptionen *collating_sequence*, *varchar_no_trailing_blanks* und *pushdown*. Im Abschnitt „Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken" auf Seite 121 finden Sie Informationen zur Einstellung dieser Optionen.

Faktoren der DB2-Typzuordnung und DB2-Funktionszuordnung: Die Standardzuordnungen von lokalen Datentypen, die von DB2 bereitgestellt werden (Informationen zu Datentyptabellen finden Sie im Handbuch *Application Development Guide*) sind so ausgelegt, dass ausreichend Pufferspeicher für

jeden Datentyp der Datenquelle reserviert wird (um Datenverlust zu vermeiden). Ein Benutzer hat die Möglichkeit, die Typzuordnung für eine bestimmte Datenquelle an die Anforderungen bestimmter Anwendungen anzupassen. Wenn Sie zum Beispiel auf eine Spalte des Datentyps DATE einer Oracle-Datenquelle (die standardmäßig dem DB2-Datentyp TIMESTAMP zugeordnet wird) zugreifen, können Sie den lokalen Datentyp in den DB2-Datentyp DATE ändern.

DB2 kann Funktionen, die von einer Datenquelle nicht unterstützt werden, kompensieren. Diese Funktionskompensation tritt in drei Fällen auf:

- Die Funktion ist an der fernen Datenquelle einfach nicht vorhanden.
- Die Funktion ist vorhanden, aber die Merkmale des Operanden verletzen die Einschränkungen der Funktion. Ein Beispiel für diesen Fall ist der relationale Operator IS NULL. Die meisten Datenquellen unterstützen ihn, aber einige haben möglicherweise Einschränkungen, wie zum Beispiel, dass nur ein Spaltenname auf der linken Seite des Operators IS NULL zulässig ist.
- Die Funktion kann bei ferner Auswertung ein anderes Ergebnis liefern. Ein Beispiel für diesen Fall ist der Operator '>' (größer als). Für Datenquellen mit abweichenden Sortierfolgen kann der Operator 'größer als' andere Ergebnisse liefern als bei einer lokalen Auswertung durch DB2.

Kurznamenmerkmale mit Auswirkung auf die Pushdown-Möglichkeiten

In den folgenden Abschnitten werden kurznamenspezifische Faktoren behandelt, die Auswirkung auf die Pushdown-Möglichkeiten haben.

Lokaler Datentyp einer Kurznamenspalte: Stellen Sie sicher, dass der lokale Datentyp einer Spalte nicht verhindert, dass ein Vergleichselement an der Datenquelle ausgewertet werden kann. Wie bereits früher erwähnt, stehen die Standarddatentypzuordnungen bereit, um jeden möglichen Überlauf zu vermeiden. Jedoch wird ein verknüpfendes Vergleichselement zwischen zwei Spalten unterschiedlicher Längen eventuell nicht an der Datenquelle ausgewertet, deren Verknüpfungsspalte kürzer ist, je nachdem, wie DB2 die längere Spalte einbindet. Dieser Umstand kann sich auf die Anzahl der Möglichkeiten in einer Verknüpfungssequenz auswirken, die vom DB2-Optimierungsprogramm ausgewertet werden. Zum Beispiel erhalten Spalten einer Oracle-Datenquelle, die mit dem Datentyp INTEGER bzw. INT erstellt wurden, den Typ NUMBER(38). Eine Kurznamenspalte für diesen Oracle-Datentyp erhält den lokalen Datentyp FLOAT, weil die Werte einer ganzen Zahl in DB2 den Bereich von 2**31 bis (-2**31)-1 umfassen, was grob dem Typ NUMBER(9) entspricht. In diesem Fall können Verknüpfungen zwischen einer DB2-Spalte des Typs Integer und einer Oracle-Spalte des Typs Integer nicht an der DB2-Datenquelle stattfinden (kürzere Verknüpfungsspalte). Wenn jedoch der Wertebereich dieser Oracle-Spalte des Typs Integer in dem DB2-Datentyp INTEGER

untergebracht werden kann, ändern Sie den lokalen Datentyp der Spalte mit der Anweisung ALTER NICKNAME, so dass die Verknüpfung an der DB2-Datenquelle stattfinden kann.

Spaltenoptionen: Die SQL-Anweisung ALTER NICKNAME kann zum Hinzufügen oder Ändern von Spaltenoptionen für Kurznamen verwendet werden.

Eine dieser Optionen ist *varchar_no_trailing_blanks*. Mit ihrer Hilfe kann eine Spalte angegeben werden, die keine folgenden Leerzeichen enthält. Der Push-down-Analyseschritt des Compilers berücksichtigt diese Information bei der Prüfung aller Operationen für Spalten, die so markiert sind. Aufgrund dieser Angabe kann DB2 eine andere, aber äquivalente Form eines Vergleichselements generieren, das in einer fernen, an eine Datenquelle gesendete SQL-Anweisung verwendet werden soll. Ein Benutzer bemerkt vielleicht, dass an der Datenquelle ein anderes Vergleichselement ausgewertet wird, das Nettoergebnis sollte jedoch äquivalent sein.

Eine weitere Spaltenoption ist *numeric_string*. Diese Option dient zur Angabe, ob die Werte in der betreffenden Spalte immer Ziffern ohne folgende Leerzeichen sind.

Tabelle 15 enthält die möglichen Werte und die Standardwerte für Spaltenoptionen.

Tabelle 15. Spaltenoptionen und zugehörige Einstellungen

Option	Gültige Einstellungen		Standard-einstellung
numeric_string	'Y'	Der Wert 'Y' (Yes) gibt an, dass diese Spalte nur Zeichenfolgen mit numerischen Daten enthält. WICHTIG: Wenn diese Spalte nur numerische Zeichenfolgen mit folgenden Leerzeichen enthält, ist die Angabe von 'Y' nicht zu empfehlen.	'N'
	'N'	Der Wert 'N' (No) gibt an, dass diese Spalte nicht auf Zeichenfolgen mit numerischen Daten beschränkt ist.	
	Durch Einstellen der Spaltenoption numeric_string auf den Wert 'Y' teilen Sie dem Optimierungsprogramm mit, dass diese Spalte keine Leerzeichen enthält, die einen störenden Einfluss auf das Sortieren der Daten dieser Spalte haben könnten. Diese Option ist in solchen Fällen nützlich, in denen die Sortierfolge einer Datenquelle von der Sortierfolge in DB2 abweicht. Mit dieser Option markierte Spalten werden nicht von der lokalen Bewertung (an der Datenquelle) aufgrund unterschiedlicher Sortierfolgen ausgeschlossen.		

Tabelle 15. Spaltenoptionen und zugehörige Einstellungen (Forts.)

Option	Gültige Einstellungen	Standard-einstellung
varchar_no_trailing_blanks	Gibt an, ob diese Datenquelle eine VARCHAR-Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen verwendet. Für Zeichenfolgen variabler Länge, die keine folgenden Leerzeichen enthalten, liefert die Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen einiger DBMSs die gleichen Ergebnisse wie die Vergleichssemantik von DB2. Wenn Sie sicher sind, dass alle VARCHAR-Spalten von Tabellen und Sichten an einer Datenquelle keine folgenden Leerzeichen enthalten, können Sie in Betracht ziehen, diese Server-Option auf den Wert 'Y' für eine Datenquelle zu setzen. Diese Option wird häufig für Datenquellen von Oracle verwendet. Stellen Sie sicher, dass Sie alle Objekte, die potenziell Kurznamen haben können (einschließlich Sichten), berücksichtigen. 'Y' Diese Datenquelle verfügt über eine Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen, die der Vergleichssemantik von DB2 ähnlich ist. 'N' Diese Datenquelle verfügt über keine mit DB2 vergleichbare Vergleichssemantik für nicht mit Leerzeichen aufgefüllte Zeichenfolgen.	'N'

Abfragemerkmale mit Auswirkung auf Pushdown-Möglichkeiten

Eine Abfrage kann auf einen SQL-Operator verweisen, der Kurznamen aus verschiedenen Datenquellen mit einbezieht. Wenn DB2 die Ergebnisse aus zwei angegebenen Datenquellen mit Hilfe eines Operators, wie zum Beispiel eines Gruppenoperators (z. B. UNION) kombinieren muss, muss die Operation in DB2 stattfinden. Der Operator kann nicht direkt an der fernen Datenquelle ausgewertet werden.

Analysieren und Verstehen der Pushdown-Analyseergebnisse

Das Umschreiben von SQL-Anweisungen kann zusätzliche Pushdown-Möglichkeiten für die DB2-Abfrageverarbeitung bereitstellen. Dieser Abschnitt enthält eine Einführung in Programme, mit denen festgestellt werden kann, wo eine Abfrage ausgewertet wird, führt allgemeine Fragen (und zur Untersuchung empfohlene Bereiche) auf, die sich auf die Abfrageanalyse beziehen, und schließt mit einem kurzen Abschnitt zum Thema Erweitern von Datenquellen.

Analysieren, wo eine Abfrage ausgewertet wird: DB2 wird mit zwei Dienstprogrammen geliefert, die zeigen, wo Abfragen ausgewertet werden:

- Visual Explain. Starten Sie das Programm über den Befehl **db2cc** oder **db2vexp**. Dieses Programm dient zum Anzeigen des Zugriffsplan-

diagramms. Die Ausführungsposition für jeden Operator ist der detaillierten Anzeige für einen Operator zu entnehmen.

Wenn eine Abfrage vollständig an eine Datenquelle verschoben wird (Pushdown), sollten Sie über einem Operator RQUERY einen Operator RETURN sehen. Der Operator RETURN ist ein Standardoperator von DB2. Der Operator RQUERY ist hingegen für Operationen für zusammengeschlossene Datenbanken spezifisch. RQUERY sendet eine SQL-Anweisung SELECT an eine Datenquelle, um das Abfrageergebnis abzurufen. Die SELECT-Anweisung wird mit Hilfe der von der Datenquelle unterstützten SQL-Version generiert. Sie kann jede für diese Datenquelle gültige Abfrage enthalten.

- SQL-EXPLAIN. Starten Sie dieses Programm mit dem Befehl **db2expln** oder **dynexpln**. Dieses Programm dient zur Erstellung einer Textausgabe des Zugriffsplans.

Verstehen, warum eine Abfrage an einer Datenquelle oder in DB2 ausgewertet wird: Dieser Abschnitt führt typische Fragen zur Plananalyse und Untersuchungsbereiche auf, die die Pushdown-Möglichkeiten erhöhen. Zu den wichtigsten Fragen gehören die folgenden:

- Warum wird dieses Vergleichselement nicht fern ausgewertet?

  Diese Frage erhebt sich, wenn ein Vergleichselement sehr selektiv ist und daher zum Filtern von Zeilen herangezogen werden und den Netzwerkverkehr verringern könnte. Die ferne Auswertung von Vergleichselementen wirkt sich auch darauf aus, ob eine Verknüpfung zwischen zwei Tabellen derselben Datenquelle fern ausgewertet werden kann.

  Zu untersuchende Bereiche sind:

  – Vergleichselemente mit Unterabfragen. Enthält dieses Vergleichselement eine Unterabfrage, die sich auf eine andere Datenquelle bezieht? Enthält dieses Vergleichselement eine Unterabfrage mit einem SQL-Operator, der von dieser Datenquelle nicht unterstützt wird? Nicht alle Datenquellen unterstützen Gruppenoperatoren in einem Vergleichselement.

  – Funktionen in Vergleichselementen. Enthält dieses Vergleichselement eine Funktion, die von dieser fernen Datenquelle nicht ausgewertet werden kann? Relationale Operatoren werden als Funktionen klassifiziert.

  – Bindeanforderungen von Vergleichselementen. Erfordert dieses Vergleichselement, wenn es fern ausgewertet werden soll, das Einbinden eines bestimmten Werts? Ist dies der Fall, würde der Wert die SQL-Einschränkungen an dieser Datenquelle verletzen?

  – Globale Optimierung. Das Optimierungsprogramm kann entschieden haben, dass der Aufwand bei der lokalen Verarbeitung geringer ist. Weitere Informationen finden Sie unter „Generierung von fernem SQL und globale Optimierung” auf Seite 244.

- Warum wird der Operator GROUP BY nicht fern ausgewertet?

  Es gibt verschiedene Bereiche, die überprüft werden können:
  - Wird die Eingabe für den Operator GROUP BY fern ausgewertet? Ist dies zu verneinen, untersuchen Sie die Eingabe.
  - Besitzt die Datenquelle Einschränkungen für diesen Operator? Beispiele hierfür sind:
    - Begrenzte Anzahl von GROUP BY-Elementen
    - Begrenzte Byteanzahl kombinierter GROUP BY-Elemente
    - Spaltenspezifikation nur für die GROUP BY-Liste
  - Unterstützt die Datenquelle diesen SQL-Operator?
  - Globale Optimierung. Das Optimierungsprogramm kann entschieden haben, dass der Aufwand bei der lokalen Verarbeitung geringer ist. Weitere Informationen finden Sie unter „Generierung von fernem SQL und globale Optimierung" auf Seite 244.
- Warum wird der Gruppenoperator nicht fern ausgewertet?

  Es gibt verschiedene Bereiche, die überprüft werden können:
  - Werden beide Operanden vollständig an derselben Datenquelle ausgewertet? Ist dies nicht der Fall, obwohl es der Fall sein sollte, untersuchen Sie jeden Operanden.
  - Besitzt die Datenquelle Einschränkungen für diesen Gruppenoperator? Sind zum Beispiel große Objekte oder Langfelder gültige Eingaben für diesen speziellen Gruppenoperator?
- Warum wird die Operation ORDER BY nicht fern ausgeführt?

  Betrachten Sie folgendes:
  - Wird die Eingabe für die Operation ORDER BY fern ausgewertet? Ist dies zu verneinen, untersuchen Sie die Eingabe.
  - Enthält die Klausel ORDER BY einen Zeichenausdruck? Wenn ja, besitzt die ferne Datenquelle nicht die gleiche Sortierfolge wie DB2?
  - Besitzt die Datenquelle Einschränkungen für diesen Operator? Gibt es beispielsweise eine begrenzte Anzahl von ORDER BY-Elementen? Schränkt die Datenquelle die Spaltenangaben für die ORDER BY-Liste ein?

Erweitern und Anpassen von Datenquellen: Obwohl der SQL-Compiler von DB2 über zahlreiche Informationen zur SQL-Unterstützung der Datenquelle verfügt, müssen diese Daten eventuell mit der Zeit angepasst werden, weil Datenquellen erweitert und/oder angepasst werden können. In solchen Fällen müssen Erweiterungen DB2 durch Ändern der lokalen Kataloginformationen bekannt gemacht werden. Verwenden Sie DDL-Anweisungen von DB2 (z. B. CREATE FUNCTION MAPPING und ALTER SERVER), um den Katalog zu aktualisieren. Weitere Informationen finden Sie im Handbuch *SQL Reference*.

Generierung von fernem SQL und globale Optimierung

Diese Phase hilft bei der Herstellung einer global optimalen Zugriffsstrategie zur Auswertung einer Abfrage. Für eine Abfrage einer zusammengeschlossenen Datenbank kann die Zugriffsstrategie vorsehen, die Originalabfrage in eine Gruppe ferner Abfrageeinheiten zu zerlegen und anschließend die Ergebnisse zu kombinieren.

Das Optimierungsprogramm verwendet die Ausgabe der Pushdown-Analyse als Empfehlung und entscheidet, ob eine Operation lokal in DB2 oder fern an einer Datenquelle ausgewertet wird. Die Entscheidung stützt sich auf die Ausgabe des Aufwandsmodells, das nicht nur den Aufwand für die Auswertung der Operation, sondern auch den Aufwand für die Übertragung der Daten oder Nachrichten zwischen DB2 und den Datenquellen berücksichtigt.

Das Ziel ist die Erstellung einer optimierten Abfrage. Die Ausgabe aus der globalen Optimierung und somit auch die Leistung der Abfrage wird jedoch von zahlreichen Faktoren beeinflusst. Die Schlüsselfaktoren werden im Folgenden in zwei Gruppen behandelt: Server-Merkmale und Kurznamenmerkmale.

Server-Merkmale/-Optionen mit Auswirkung auf die globale Optimierung

Zu den Server-Faktoren einer Datenquelle, die sich auf die globale Optimierung auswirken können, gehören folgende:

- Relatives Verhältnis der CPU-Geschwindigkeiten

  Geben Sie mit Hilfe der Serveroption *cpu_ratio* an, um wie viel schneller oder langsamer die CPU-Geschwindigkeit der Datenquelle im Verhältnis zur CPU von DB2 ist. Ein niedriger Wert bedeutet, dass die CPU der Workstation der Datenquelle schneller als die CPU der DB2-Workstation ist. Bei niedrigen Verhältniswerten zieht das DB2-Optimierungsprogramm mit größerer Wahrscheinlichkeit in Betracht, CPU-intensive Operationen an die Datenquelle zu verschieben (Pushdown). Weitere Informationen zu dieser Geschwindigkeitsangabe finden Sie in „Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken” auf Seite 121.

- Relatives Verhältnis der E/A-Geschwindigkeiten

  Geben Sie mit Hilfe der Serveroption *io_ratio* an, um wie viel schneller oder langsamer die E/A-Geschwindigkeit des Datenquellensystems im Verhältnis zum DB2-System ist. Ein niedriger Wert bedeutet, dass die E/A-Geschwindigkeit der Workstation der Datenquelle schneller als die E/A-Geschwindigkeit der DB2-Workstation ist. Bei niedrigen Verhältniswerten zieht das DB2-Optimierungsprogramm mit größerer Wahrscheinlichkeit in Betracht, E/A-intensive Operationen an die Datenquelle zu verschieben (Pushdown). Weitere Informationen zu dieser Geschwindigkeitsangabe finden Sie in „Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken” auf Seite 121.

- Übertragungsgeschwindigkeit zwischen DB2 und der Datenquelle

  Geben Sie mit Hilfe der Serveroption *comm_rate* die Netzwerkkapazität an. Langsame Geschwindigkeiten (die eine langsame Kommunikation zwischen DB2 und der Datenquelle bedeuten) veranlassen das DB2-Optimierungsprogramm, die Anzahl der Nachrichten zu reduzieren, die zwischen DB2 und dieser Datenquelle gesendet werden. Wenn die Geschwindigkeit auf den Wert 0 gesetzt wird, erstellt das Optimierungsprogramm eine Abfrage mit minimalen Anforderungen an das Netzwerk. Weitere Informationen zu dieser Geschwindigkeitsangabe finden Sie in „Server-Optionen mit Auswirkung auf Abfragen für zusammengeschlossene Datenbanken" auf Seite 121.
- Sortierfolge der Datenquelle

  Geben Sie mit Hilfe der Serveroption *collating_sequence* an, ob die Sortierfolge einer Datenquelle mit der lokalen DB2-Sortierfolge übereinstimmt. Wenn die Option nicht auf den Wert 'Y' gesetzt ist, betrachtet das Optimierungsprogramm die von dieser Datenquelle abgerufenen Daten als unsortiert. Im Abschnitt „Sortierfolge" auf Seite 237 finden Sie weitere Informationen zur Leistungsrelevanz der Sortierfolge.
- Ferne Planhinweise

  Geben Sie mit Hilfe der Serveroption *plan_hints* an, ob an einer Datenquelle Planhinweise unterstützt werden. Planhinweise sind Anweisungsfragmente, die zusätzliche Informationen für Optimierungsprogramme von Datenquellen bereitstellen. Diese Informationen können bei einigen Abfragearten die Abfrageleistung verbessern. Die Planhinweise können das Optimierungsprogramm der Datenquelle bei der Entscheidung unterstützen, ob ein Index, und wenn ja, welcher, zu verwenden ist oder nach welcher Reihenfolge bei der Verknüpfung von Tabellen vorzugehen ist.

  Wenn Planhinweise aktiviert sind, enthält die Abfrage, die an die Datenquelle gesendet wird, zusätzliche Informationen. Zum Beispiel könnte eine Anweisung, die an ein Oracle-Optimierungsprogramm mit Planhinweisen gesendet wird, folgendermaßen aussehen:

  ```
  SELECT /*+ INDEX (table1, t1index)*/
     col1
     FROM table1
  ```

  Der Planhinweis ist hier die Zeichenfolge `/*+ INDEX (table1, t1index)*/`.
- Informationen in der Wissensbasis des DB2-Optimierungsprogramms

  DB2 besitzt eine Wissensbasis für das Optimierungsprogramm, die Daten über spezifische Datenquellen enthält. Das DB2-Optimierungsprogramm generiert keine fernen Zugriffspläne, die nicht von spezifischen Datenbankverwaltungssystemen (DBMSs) generiert werden können. In anderen Worten, DB2 vermeidet die Generierung von Plänen, die von Optimierungsprogrammen an fernen Datenquellen nicht verstanden bzw. akzeptiert werden.

Kurznamenmerkmale mit Auswirkung auf die globale Optimierung

In den folgenden Abschnitten werden kurznamenspezifische Faktoren behandelt, die Auswirkung auf die globale Optimierung haben.

Überlegungen zu Indizes: DB2 kann Informationen über Indizes an Datenquellen zur Optimierung von Abfragen heranziehen. Daher ist es wichtig, dass die für DB2 verfügbaren Indexinformationen aktuell sind. Die Indexinformationen für Kurznamen werden zuerst bei der Erstellung des Kurznamens empfangen. Indexinformationen werden für Sichtkurznamen nicht gesammelt.

Erstellen von Indexspezifikationen für Kurznamen: Es kann eine Indexspezifikation für einen Kurznamen erstellt werden. Indexspezifikationen bilden eine Indexdefinition (keinen tatsächlichen Index) im Katalog, die vom DB2-Optimierungsprogramm verwendet werden kann. Indexspezifikationen werden mit der Anweisung CREATE INDEX SPECIFICATION ONLY erstellt. Die Syntax für die Erstellung einer Indexspezifikation für einen Kurznamen ist der Syntax zur Erstellung eines Index für eine lokale Tabelle ähnlich. Weitere Informationen finden Sie im Handbuch *Systemverwaltung: Konzept*.

Die Erstellung von Indexspezifikationen kommt unter folgenden Bedingungen in Betracht:

- DB2 kann während der Kurznamenerstellung keine Indexinformationen von einer Datenquelle abrufen.
- Sie wünschen einen Index für einen Sichtkurznamen.
- Sie wollen das DB2-Optimierungsprogramm veranlassen, einen speziellen Kurznamen als innere Tabelle einer Verknüpfung mit Verschachtelungsschleife zu verwenden. Der Benutzer kann einen Index für die Verknüpfungsspalte erstellen, falls keiner vorhanden ist.

Kalkulieren Sie Ihre Anforderungen, bevor Sie Anweisungen CREATE INDEX für einen Kurznamen einer Sicht ausführen. In einem Fall, wenn die Sicht eine einfache SELECT-Abfrage auf eine Tabelle mit einem Index ist, kann die Erstellung von Indizes für den Kurznamen (lokal), die mit den Indizes für die Tabelle an der Datenquelle übereinstimmen, die Abfrageleistung erheblich erhöhen. Wenn aber Indizes lokal über Sichten erstellt werden, die keine einfachen SELECT-Anweisungen sind (z. B. eine Sicht, die durch Verknüpfen zweier Tabellen erstellt wird), kann die Abfrageleistung sinken. Wenn zum Beispiel ein Index über eine Sicht erstellt wird, die auf der Verknüpfung zweier Tabellen basiert, kann das Optimierungsprogramm diese Sicht möglicherweise als inneres Element in einer Verknüpfung über Verschachtelungsschleife (Nested Loop Join) festlegen. Die Abfrage wird eine geringe Leistung erzielen, weil in diesem Fall die Verknüpfung mehrere Male ausgewertet wird. Eine Alternative wäre, Kurznamen für jede der Tabellen, auf die in der Sicht der Datenquelle verwiesen wird, zu erstellen und eine lokale Sicht unter DB2 zu definieren, die auf beide Kurznamen verweist.

Überlegungen zu Systemkatalogstatistiken: Katalogstatistiken beschreiben die allgemeine Größe von Kurznamen und den Wertebereich in den zugehörigen Spalten. Sie werden vom Optimierungsprogramm bei der Berechnung des Pfads mit dem günstigsten Aufwand zur Verarbeitung von Abfragen verwendet, die Kurznamen enthalten. Die statistischen Daten über Kurznamen werden in den gleichen Katalogsichten wie Tabellenstatistiken gespeichert. In „Kapitel 5. Systemkatalogstatistiken" auf Seite 129, und im Abschnitt „Regeln zur Aktualisierung von Kurznamenstatistiken" auf Seite 157 finden Sie weitere Informationen zu Statistikarten und dazu, wie sie lokal aktualisiert werden.

Obwohl DB2 die an einer Datenquelle befindlichen statistischen Daten abrufen kann, kann DB2 Aktualisierungen an vorhandenen Statistikdaten an Datenquellen nicht automatisch erkennen. Zudem verfügt DB2 über keine Methode zur Behandlung von Definitions- oder Strukturänderungen (z. B. Hinzufügen einer Spalte) an Objekten der Datenquelle. Wenn die Statistik- bzw. Strukturdaten für ein Objekt geändert wurden, haben Sie zwei Möglichkeiten:

- Führen Sie an der Datenquelle das Programm aus, das dem Dienstprogramm RUNSTATS entspricht. Löschen Sie anschließend den aktuellen Kurznamen. Erstellen Sie den Kurznamen erneut. Diese Methode ist zu verwenden, wenn sich Strukturinformationen geändert haben.
- Aktualisieren Sie die Statistiken in der Sicht SYSSTAT.TABLES manuell. Diese Methode hat zwar weniger Schritte, funktioniert jedoch nicht, falls sich Strukturinformationen geändert haben.

Analysieren und Verstehen der Entscheidungen der globalen Optimierung

Dieser Abschnitt enthält eine Einführung in Programme zur Analyse der Abfrageoptimierung und stellt allgemeine Fragen (und zur Untersuchung empfohlene Bereiche) im Zusammenhang mit der Abfrageoptimierung vor.

Analysieren der Abfrageoptimierung: DB2 stellt zwei Dienstprogramme zur Verfügung, die globale Zugriffspläne zeigen:

- Visual Explain. Starten Sie das Programm über den Befehl **db2cc** oder **db2vexp**. Dieses Programm dient zum Anzeigen des Zugriffsplandiagramms. Die Ausführungsposition für jeden Operator ist der detaillierten Anzeige für einen Operator zu entnehmen. Sie können darüber hinaus die ferne SQL-Anweisung, die für die jeweilige Datenquelle generiert wurde, dem Operator RQUERY (SELECT-Operation) entnehmen. Durch eine Untersuchung der Einzelangaben für jeden Operator können Sie die Anzahl der Zeilen feststellen, die vom DB2-Optimierungsprogramm als Eingabe für den jeweiligen Operator und als Ausgabe aus diesem Operator geschätzt wird. Weiterhin können Sie den für die Ausführung des jeweiligen Operators geschätzten Aufwand, einschließlich des Übertragungsaufwands, ablesen. Weitere Informationen finden Sie in „Anhang C. EXPLAIN-Programme (SQL)" auf Seite 645.

- SQL-EXPLAIN. Starten Sie dieses Programm mit dem Befehl **db2expln** oder **dynexpln**. Dieses Programm dient zur Erstellung einer Textausgabe des Zugriffsplans. SQL-EXPLAIN bietet keine Informationen über den Aufwand. Aber Sie können der Ausgabe den Zugriffsplan entnehmen, der von dem fernen Optimierungsprogramm für die Datenquellen generiert wurde, die von der fernen EXPLAIN-Funktion unterstützt werden. Weitere Informationen finden Sie in „Anhang C. EXPLAIN-Programme (SQL)" auf Seite 645.

Verstehen der Entscheidungen der DB2-Optimierung: Dieser Abschnitt enthält eine Liste von Fragen zur Optimierung und Schlüsselbereiche, die zur Verbesserung der Leistung untersucht werden können. Zu den wichtigsten Fragen gehören die folgenden:

- Warum wird eine Verknüpfung zwischen zwei Kurznamen derselben Datenquelle nicht fern an der Datenquelle ausgewertet?

  Zu untersuchende Bereiche sind:

  – Verknüpfungsoperationen. Werden sie von der Datenquelle unterstützt?

  – Verknüpfungsvergleichselemente. Können die Verknüpfungsvergleichselemente an der fernen Datenquelle ausgewertet werden? Wenn dies zu verneinen ist, untersuchen Sie das Verknüpfungsvergleichselement. Weitere Informationen finden Sie in „Verstehen, warum eine Abfrage an einer Datenquelle oder in DB2 ausgewertet wird" auf Seite 242.

  – Anzahl der Zeilen im Verknüpfungsergebnis (mit Visual Explain). Ergibt sich aus der Verknüpfung eine wesentlich größere Gruppe von Zeilen als durch die Kombination der beiden Kurznamen? Ist die jeweilige Anzahl sinnvoll? Wenn dies zu verneinen ist, ziehen Sie eine manuelle Aktualisierung der Kurznamenstatistik (SYSSTAT.TABLES) in Betracht.

- Warum wird der Operator GROUP BY nicht fern ausgewertet?

  Zu untersuchende Bereiche sind:

  – Operatorsyntax. Stellen Sie sicher, dass der Operator an der fernen Datenquelle ausgewertet werden kann. Weitere Informationen finden Sie in „Verstehen, warum eine Abfrage an einer Datenquelle oder in DB2 ausgewertet wird" auf Seite 242.

  – Anzahl von Zeilen. Prüfen Sie die geschätzte Anzahl von Zeilen in der Eingabe und Ausgabe des Operators GROUP BY mit Hilfe von Visual Explain. Liegt die eine Anzahl sehr nahe an der anderen? Ist dies zu bejahen, betrachtet das DB2-Optimierungsprogramm es als effizienter, diesen Operator GROUP BY lokal auszuwerten. Ist die jeweilige Anzahl zudem sinnvoll? Wenn dies zu verneinen ist, ziehen Sie eine manuelle Aktualisierung der Kurznamenstatistik (SYSSTAT.TABLES) in Betracht.

- Warum wird die Anweisung nicht vollständig durch die ferne Datenquelle ausgewertet?

  Das DB2-Optimierungsprogramm führt eine aufwandsorientierte Optimierung durch. Auch wenn die Pushdown-Analyse ergibt, dass jeder Operator an der fernen Datenquelle ausgewertet werden kann, stützt sich das Optimierungsprogramm bei der Generierung eines global optimalen Plans auf die Aufwandsschätzung. Es gibt eine Vielzahl von Faktoren, die in diesen Plan einfließen können. Es ist beispielsweise möglich, dass eine ferne Datenquelle zwar jede einzelne Operation in der ursprünglichen Abfrage ausführen kann, aber die CPU-Geschwindigkeit der Datenquelle wesentlich langsamer ist als die des DB2-Systems, so dass es vorteilhafter ist, die Operationen in DB2 lokal auszuführen. Wenn die Ergebnisse nicht zufrieden stellend sind, überprüfen Sie die Server-Statistikdaten in SYSCAT.SERVEROPTIONS.
- Warum zeigt ein vom Optimierungsprogramm generierter Plan, der vollständig an einer fernen Datenquelle ausgewertet wird, eine wesentlich schlechtere Leistung als die Originalabfrage bei direkter Ausführung an der fernen Datenquelle?

  Zu untersuchende Bereiche sind:

  – Die vom DB2-Optimierungsprogramm generierte ferne SQL-Anweisung. Überprüfen Sie, ob sie mit der Originalabfrage identisch ist. Suchen Sie nach Änderungen in der Reihenfolge der Vergleichselemente. Ein gutes Abfrageoptimierungsprogramm sollte nicht von der Reihenfolge der Vergleichselemente einer Abfrage abhängig sein. Leider sind nicht alle DBMS-Optimierungsprogramme identisch, so dass das Optimierungsprogramm der fernen Datenquelle vielleicht aufgrund der Reihenfolge der Vergleichselemente in der Eingabe einen anderen Plan generiert. Wenn dies der Fall ist, liegt das Problem bei dem fernen Optimierungsprogramm. Sie können in diesem Fall entweder eine Änderung der Reihenfolge der Vergleichselemente in der Eingabe für DB2 in Betracht ziehen oder die Serviceorganisation der fernen Datenquelle um Hilfe bitten.

    Prüfen Sie darüber hinaus auf Ersetzungen von Vergleichselementen. Ein gutes Abfrageoptimierungsprogramm sollte nicht von der Ersetzung äquivalenter Vergleichselemente abhängig sein. Leider sind nicht alle DBMS-Optimierungsprogramme identisch, so dass das Optimierungsprogramm der fernen Datenquelle vielleicht aufgrund des Vergleichselements in der Eingabe einen anderen Plan generiert. Zum Beispiel können einige Optimierungsprogramme keine Anweisungen für Vergleichselemente unter Verwendung der Transitivität generieren.

- Die Anzahl der zurückgegebenen Zeilen. Diese Anzahl kann mit Hilfe von Visual Explain festgestellt werden. Wenn die Abfrage eine große Anzahl von Zeilen zurückgibt, kann der Netzwerkverkehr zum potenziellen Engpass werden.
- Zusätzliche Funktionen. Enthält die ferne SQL-Anweisung im Vergleich zur Originalabfrage zusätzliche Funktionen? Es können einige zusätzliche Funktionen zur Umsetzung von Datentypen generiert werden. Stellen Sie sicher, dass sie erforderlich sind.

Kapitel 7. Die SQL-EXPLAIN-Einrichtung

Die SQL-EXPLAIN-Einrichtung ist Bestandteil des SQL-Compilers und dient zur Erfassung von Informationen über die Umgebung, in der die statischen oder dynamischen SQL-Anweisungen kompiliert werden. Die erfassten Informationen geben Aufschluss über die Struktur und die potenzielle Ausführungsleistung von SQL-Anweisungen. Folgende Informationen werden bereitgestellt:

- Informationen über die Reihenfolge der Operationen zur Verarbeitung der Abfrage
- Informationen über den Aufwand
- Schätzwerte zu Vergleichselementen und Selektivität
- Statistiken für alle Objekte, auf die in der SQL-Anweisung zur Zeit der Ausführung von EXPLAIN verwiesen wird

Diese Informationen können Sie zu folgenden Zwecken verwenden:

- Das Verständnis des für eine Abfrage ausgewählten Ausführungsplans
- Die Hilfe bei der Analyse und beim Entwurf von Anwendungsprogrammen
- Die Ermittlung, wann eine Anwendung erneut gebunden werden sollte
- Den Datenbankentwurf

Die folgenden Themen werden behandelt:

- „Auswählen eines EXPLAIN-Programms" auf Seite 252
- „Verwenden der SQL-EXPLAIN-Einrichtung" auf Seite 254
- „Grundkonzepte für EXPLAIN" auf Seite 257
- „Organisation von EXPLAIN-Informationen" auf Seite 260
- „Abrufen von EXPLAIN-Daten" auf Seite 267
- „Richtlinien zur Verwendung der EXPLAIN-Ausgabe" auf Seite 271
- „Visual Explain" auf Seite 273
- „SQL-Advise-Einrichtung" auf Seite 274.

Die EXPLAIN-Ausgabe wird in relationalen Tabellen und optionsweise in einem Format gespeichert, das mit Hilfe des Programms Visual Explain grafisch angezeigt werden kann. Sie sollten auf die EXPLAIN-Tabellen zurückgreifen, wenn Sie bestimmte Abfragen finden möchten, die für die EXPLAIN-Tabellen ausgeführt werden. Weitere Informationen zu den Tabellen, die von der EXPLAIN-Einrichtung verwendet werden, und zu ihrer Erstellung finden Sie in „Anhang B. EXPLAIN-Tabellen und Definitionen" auf Seite 609.

Auswählen eines EXPLAIN-Programms

DB2 verfügt unter vergleichbaren Systemen über die umfangreichste EXPLAIN-Einrichtung, die detaillierte Informationen des Optimierungsprogramms über den für eine mit EXPLAIN bearbeitete SQL-Anweisung gewählten Zugriffsplan zur Verfügung stellt. Sie haben die Wahl zwischen mehreren Methoden, die Ihnen die nötige Flexibilität bei der Erfassung und Auswertung von EXPLAIN-Informationen geben.

Detaillierte Informationen des Optimierungsprogramms, die eine eingehende Analyse eines Zugriffsplans ermöglichen, werden getrennt vom eigentlichen Zugriffsplan in EXPLAIN-Tabellen gespeichert. Es gibt drei Möglichkeiten, auf Informationen der EXPLAIN-Tabellen zuzugreifen:

1. Schreiben eigener Abfragen (mit Hilfe der Beschreibungen der EXPLAIN-Tabellen in „Anhang B. EXPLAIN-Tabellen und Definitionen" auf Seite 609)
2. Das Programm *db2exfmt*
3. Visual Explain (zur Anzeige von EXPLAIN-Momentaufnahmen)

Der Zugriff auf die EXPLAIN-Tabellen, die Informationen zu statischen und dynamischen SQL-Anweisungen enthalten, ist auf allen unterstützten Plattformen möglich. Sie können auf die EXPLAIN-Tabellen mit SQL-Anweisungen zugreifen, die eine einfache Bearbeitung der Ausgabe sowie Vergleiche zwischen verschiedenen Abfragen bzw. Vergleiche derselben Abfrage über einen bestimmten Zeitraum hinweg ermöglichen. Wenn Sie die Informationen der EXPLAIN-Tabellen in einem vordefinierten Format erhalten möchten, können Sie das Programm *db2exfmt* verwenden. Weitere Informationen zu diesem Programm finden Sie in „Anhang D. db2exfmt - EXPLAIN-Tool für Tabellenformat" auf Seite 695. Alternativ können Sie eigene Anweisungen zum Zugriff auf die Tabellen erstellen.

Anmerkung: Dieses Programm (wie auch *db2batch*, *dynexpln*, *db2vexp* und *db2_all*) befindet sich im Unterverzeichnis *misc* des Verzeichnisses *sqllib*. Wenn dieses Tool aus diesem Pfad versetzt wurde, funktioniert der oben genannte Befehl vielleicht nicht.

Visual Explain ermöglicht die Analyse des Zugriffplans und der Informationen des Optimierungsprogramms in den EXPLAIN-Tabellen mit Hilfe einer grafischen Schnittstelle. Mit diesem Tool können sowohl statische als auch dynamische SQL-Anweisungen analysiert werden. Visual Explain wird in der Regel über die Steuerzentrale aufgerufen. Die Steuerzentrale kann ihrerseits über die Befehlszeile durch Eingabe des Befehls *db2cc* aufgerufen werden. Visual Explain kann jedoch auch für eine einzelne SQL-Anweisung direkt über die Befehlszeile mit Hilfe des Befehls *db2vexp* aufgerufen werden. Auf einigen Plattformen kann Visual Explain mit Hilfe eines Ordners im Ordner von DB2 Universal Database aufgerufen werden. Visual Explain steht nicht auf allen unterstützten Plattformen zur Verfügung.

Das Handbuch *Einstieg* für Ihre Plattform enthält Informationen darüber, ob Visual Explain unterstützt wird. Visual Explain ermöglicht Ihnen, auf einer anderen Plattform erfasste oder erstellte Momentaufnahmen anzuzeigen. Zum Beispiel kann ein Windows NT-Client Momentaufnahmen darstellen, die auf einem Server unter DB2 für HP-UX generiert wurden. Dazu müssen beide Plattformen jedoch die Version 5 oder höher haben. Die Ausgabe von Visual Explain kann zur weiteren Analyse nicht einfach bearbeitet werden, und auch andere Anwendungen können auf diese Informationen nicht zugreifen. Weitere Informationen zum Befehl *db2vexp* können Sie mit dem Befehl *db2vexp -h* über die Befehlszeile anzeigen oder im Handbuch *Command Reference* nachlesen. Weitere Informationen zu Visual Explain enthält die Online-Hilfefunktion in der Steuerzentrale, auf die Sie mit dem Befehl *db2cc* zugreifen können.

Informationen zu Zugriffsplänen für statische SQL-Anweisungen werden generiert und im Systemkatalog als Teil eines Pakets gespeichert. Zugriffsplaninformationen, die für ein oder mehrere Pakete verfügbar sind, können mit dem Programm *db2expln* über die Befehlszeile angezeigt werden. Das Programm *db2expln* zeigt die tatsächliche Implementierung des gewählten Zugriffsplans. Informationen des Optimierungsprogramms werden nicht angezeigt.

Das Programm *dynexpln*, das intern auf *db2expln* zurückgreift, stellt eine schnelle Methode zur Analyse dynamischer SQL-Anweisungen dar, die keine Parametermarken enthalten. Die Verwendung von *db2expln* innerhalb des Programms *dynexpln* wird durch die Transformation der Eingabeanweisung in eine statische SQL-Anweisung innerhalb eines Pseudopakets ermöglicht. Wenn dieses Verfahren angewandt wird, sind die Informationen nicht immer ganz exakt. Wenn es auf Genauigkeit ankommt, sollten Sie die EXPLAIN-Einrichtung verwenden, die im Abschnitt „Verwenden der SQL-EXPLAIN-Einrichtung” auf Seite 254 beschrieben wird.

Das Programm *db2expln* liefert einen relativ kompakten und englisch aufbereiteten Überblick über die Operationen, die bei der Ausführung stattfinden, in dem der tatsächlich generierte Zugriffsplan analysiert wird. (Auf Seite 175 finden Sie weitere Informationen über die Generierung des Codes). Weitere Einzelheiten zur Verwendung von *db2expln* und zur Interpretation der Ausgabe finden Sie in „Anhang C. EXPLAIN-Programme (SQL)” auf Seite 645.

Tabelle 16 auf Seite 254 enthält eine Übersicht über die verschiedenen, in der EXPLAIN-Einrichtung von DB2 verfügbaren Tools und ihre Merkmale. Verwenden Sie die Tabelle zur Auswahl des Tools, das für Ihre Anforderungen und Ihre Umgebung am besten geeignet ist.

Tabelle 16. Tools der EXPLAIN-Einrichtung

Merkmale	Visual Explain	db2vexp	EXPLAIN-Tabellen	db2exfmt	db2expln	dynexpln
Grafische Benutzerschnittstelle	Ja	Ja				
Textausgabe				Ja	Ja	Ja
Kurze und schnelle Analyse statischen SQLs					Ja	
Unterstützung für statisches SQL	Ja		Ja	Ja	Ja	
Unterstützung für dynamisches SQL	Ja	Ja	Ja	Ja		Ja*
Unterstützung für CLI-Anwendungen	Ja		Ja	Ja		
Verfügbar für DRDA-Anwendungs-Requester			Ja			
Detaillierte Informationen des Optimierungsprogramms	Ja	Ja	Ja	Ja		
Geeignet zur Analyse mehrerer Anweisungen			Ja	Ja	Ja	Ja
Zugriff auf die Informationen aus einer Anwendung heraus			Ja			

Anmerkung:

* Verwendet indirekt db2expln; es gelten einige Einschränkungen.

Verwenden der SQL-EXPLAIN-Einrichtung

EXPLAIN-Informationen können mit Hilfe verschiedener Mittel erfasst werden:

1. BIND/PREP-Optionen EXPLAIN und EXPLSNAP
2. Sonderregister CURRENT EXPLAIN MODE und CURRENT EXPLAIN SNAPSHOT
3. SQL-Anweisung EXPLAIN
4. Programm *db2vexp* (ruft zum Anzeigen der Informationen Visual Explain ebenfalls direkt auf)

Es gibt drei Gründe, aus denen das Sammeln und Verwenden von EXPLAIN-Daten wünschenswert sein kann:

1. Um die verschiedenen Schritte, d. h. den Zugriffsplan, zu verstehen, die der Datenbankmanager zur Erfüllung Ihrer Abfrage ausführen muss. Im Abschnitt „Datenzugriffskonzepte und Optimierung" auf Seite 187 finden Sie Informationen, die Sie vielleicht zum Verständnis der EXPLAIN-Ausgabe benötigen.
2. Um die Ergebnisse von Maßnahmen zur Leistungsverbesserung bewerten zu können. Es gibt eine Anzahl von Maßnahmen, die Ihnen helfen, die Leistung Ihrer Abfragen zu verbessern. Viele dieser möglichen Maßnahmen werden in Abschnitten der folgenden Kapitel beschrieben:
   - „Kapitel 3. Überlegungen zu Anwendungen" auf Seite 47
   - „Kapitel 4. Überlegungen zur Umgebung" auf Seite 103
   - „Kapitel 5. Systemkatalogstatistiken" auf Seite 129.

   Nach Änderungen in einem dieser Bereiche können Sie mit der SQL-EXPLAIN-Einrichtung die Auswirkungen ermitteln, die die Änderungen auf den ausgewählten Zugriffsplan haben. Wenn Sie zum Beispiel entsprechend den Empfehlungen im Abschnitt „Auswirkung des Indexierens auf die Abfrageoptimierung" auf Seite 111 einen Index hinzufügen, können Sie anhand der EXPLAIN-Daten herausfinden, ob der Index tatsächlich in erwarteter Weise verwendet wird.

   Zwar erlaubt die EXPLAIN-Ausgabe das Feststellen des ausgewählten Zugriffsplans und des für ihn erforderlichen relativen Aufwands, aber die einzige Möglichkeit, die Verbesserung der Leistung genau zu messen, sind Vergleichstests (eine Beschreibung finden Sie in „Kapitel 12. Durchführen von Vergleichstests" auf Seite 371).
3. Zum Verständnis der Ursachen für Änderungen in der Abfrageleistung benötigen Sie die EXPLAIN-Informationen vor und nach den Änderungen, um die Auswirkungen analysieren zu können. Daher sollten Sie beim Kompilieren einer SQL-Anweisung für die Datenbank Folgendes durchführen:
   - Erfassen der Zugriffsplaninformationen mit Hilfe der EXPLAIN-Einrichtung vor Ihren Änderungen und Speichern der resultierenden EXPLAIN-Tabelle; oder Speichern der Ausgabe des EXPLAIN-Programms `db2exfmt`.

- Speichern und/oder Drucken der aktuellen Katalogstatistiken, wenn Sie zum Anzeigen der entsprechenden Informationen nicht auf Visual Explain zugreifen wollen oder können. (Das im Abschnitt „Modellieren im Einsatz befindlicher Datenbanken" auf Seite 164 beschriebene Produktivitäts-Tool `db2look` kann zum Ausführen dieser Aufgabe verwendet werden.)
- Speichern und/oder Drucken der Anweisungen der Datendefinitionssprache (DDL), einschließlich der Anweisungen für CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE TABLESPACE

Die auf diese Weise ermittelten Informationen geben Ihnen einen *Vorher*-Status, den Sie für die zukünftige Analyse als Referenzpunkt verwenden können. Bei dynamischen SQL-Anweisungen können Sie diese Informationen auch bei der ersten Ausführung Ihrer Anwendung erfassen. Bei statischen SQL-Anweisungen können Sie diese Informationen auch beim Binden erfassen.

Wenn Sie die Ursache einer Änderung in der Leistung analysieren möchten, können Sie den ermittelten *Vorher*-Status mit den Informationen vergleichen, die Sie über die Abfrage und die Umgebung sammeln, wenn Sie Ihre Analyse starten (d. h. den *Nachher*-Daten).

Ein einfaches Beispiel hierfür wäre z. B., wenn Ihre Analyse ergäbe, dass ein Index nicht mehr als Bestandteil des Zugriffspfads verwendet wird. Mit Hilfe der in Visual Explain angezeigten Informationen der Katalogstatistiken könnten Sie feststellen, dass die Anzahl von Indexstufen (Spalte NLEVELS) nun wesentlich höher ist als zu dem Zeitpunkt, als die Abfrage zum ersten Mal an die Datenbank gebunden wurde. Sie hätten in diesem Fall folgende Möglichkeiten:

- Reorganisieren des Index
- Sammeln neuer Statistikdaten für die Tabelle und Indizes
- Sammeln von EXPLAIN-Informationen beim erneuten Binden der Abfrage

Anschließend stellen Sie möglicherweise fest, dass der Index im Zugriffsplan wieder verwendet wird und dass die Leistung der Abfrage kein Problem mehr darstellt.

Grundkonzepte für EXPLAIN

Anhand der EXPLAIN-Informationen können Sie den Zugriffsplan analysieren, den das Optimierungsprogramm aufgrund der im Abschnitt „Datenzugriffskonzepte und Optimierung" auf Seite 187 beschriebenen Angaben ausgewählt hat. Zum Beispiel kann aus den EXPLAIN-Informationen hervorgehen, dass eine Indexsuche (siehe „Konzepte der Indexsuche" auf Seite 188) vom Optimierungsprogramm ausgewählt wurde. Außerdem können Sie anhand dieser Informationen auch Folgendes ermitteln:

- Wie viele Indexspalten als Suchkriterium verwendet werden. Hinweise hierzu finden Sie in „Bereichsbegrenzende und bei Indexsuchen als Suchargument verwendbare Vergleichselemente" auf Seite 201.
- Ob ein reiner Indexzugriff verwendet wird. Hinweise hierzu finden Sie in „Reiner Indexzugriff" auf Seite 194.
- Ob ein Vorablesezugriff über Listen zum Lesen der Seiten verwendet wird. Hinweise hierzu finden Sie in „Vorablesezugriff über Listen" auf Seite 303.

Weiterhin können die EXPLAIN-Informationen Ihnen auch einen Einblick geben, wie zwei Tabellen miteinander verknüpft werden:

- Die Verknüpfungsmethode
- Die Reihenfolge, in der die Tabellen verknüpft werden
- Auftreten und Art von Sortierungen

Obwohl EXPLAIN für SQL-Anweisungen SELECT, SELECT INTO, UPDATE, INSERT, VALUES, VALUES INTO und DELETE verwendet werden kann, dient es primär dazu, die Zugriffspfade für die SELECT-Teile Ihrer Anweisungen zu beobachten.

Zum Ausführen einer SQL-Abfrage unternimmt der Datenbankmanager in der Regel folgende Schritte:

- Er greift auf ein oder mehrere Datenobjekte (eine Tabelle, einen Index oder beides) zu.
- Er führt eine oder mehrere Operationen (z. B. Tabellensuche, Indexsuche oder Verknüpfung) durch.
- Er übergibt die Ergebnismenge an die aufrufende Anwendung.

Für eine einfache SQL-Abfrage wie

```
SELECT DEPTNO, DEPTNAME
  FROM DEPARTMENT
```

könnte die folgende grafische Darstellung der ausgeführten Schritte von Visual Explain angezeigt werden:

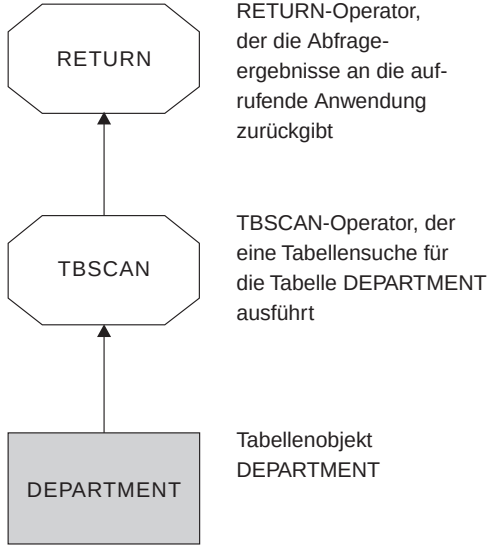


Abbildung 20. Grafische Anzeige der EXPLAIN-Ausgabe

In den folgenden Abschnitten werden die Angaben beschrieben, die über die Objekte und Operatoren angezeigt werden können:

- „EXPLAIN-Informationen für Datenobjekte"
- „EXPLAIN-Informationen für Datenoperatoren" auf Seite 259

EXPLAIN-Informationen für Datenobjekte

Ein einzelner Zugriffsplan kann zur Ausführung der SQL-Anweisung ein oder mehrere Datenobjekte verwenden.

Objektstatistiken: Die EXPLAIN-Einrichtung zeichnet Informationen über das Objekt auf, wie z. B. folgende Angaben:

- Die Erstellungszeit
- Den Zeitpunkt, an dem zum letzten Mal Statistiken für das Objekt gesammelt wurden (siehe „Kapitel 5. Systemkatalogstatistiken" auf Seite 129)
- Eine Angabe, ob die Daten im Objekt sortiert sind (nur Tabellen- oder Indexobjekte)
- Die Anzahl von Spalten im Objekt (nur Tabellen- oder Indexobjekte)
- Die geschätzte Anzahl von Zeilen im Objekt (nur Tabellen- oder Indexobjekte)
- Die Anzahl von Seiten, die das Objekt im Pufferpool einnimmt
- Den geschätzten Gesamtaufwand in Millisekunden für eine einzelne wahlfreie Ein-/Ausgabeoperation für den angegebenen Tabellenbereich, in dem dieses Objekt gespeichert ist

- Die geschätzte Übertragungsrate in Millisekunden zum Lesen einer 4-KB-Seite aus dem angegebenen Tabellenbereich
- Die Größen für PREFETCHSIZE und EXTENTSIZE (in 4-KB-Seiten)
- Den Grad der Datenclusterbildung mit dem Index
- Die Anzahl von Blattseiten (Leaf Pages), die vom Index dieses Objekts verwendet werden, und die Anzahl der Indexstufen in der Indexbaumstruktur
- Die Anzahl unterschiedlicher vollständiger Schlüsselwerte im Index dieses Objekts
- Die Gesamtzahl der Überlaufsätze in der Tabelle

EXPLAIN-Informationen für Datenoperatoren

Ein einziger Zugriffsplan kann mehrere Operationen mit den Daten ausführen, um die SQL-Anweisung auszuführen und die Ergebnisse an Sie zurückzugeben. Der SQL-Compiler ermittelt, welche Operationen erforderlich sind, z. B. eine Tabellensuche, eine Indexsuche, eine Verknüpfung über Verschachtelungsschleife oder ein GROUP BY-Operator. Einzelheiten zu vielen dieser Operatoren finden Sie in „Datenzugriffskonzepte und Optimierung" auf Seite 187.

Über die Anzeige der verschiedenen im Zugriffsplan verwendeten Operatoren hinaus bietet EXPLAIN auch Informationen über jeden Operator sowie über die kumulativen Auswirkungen des Zugriffsplans.

Informationen über den geschätzten Aufwand: Informationen über die folgenden geschätzten kumulativen Aufwände können für die Operatoren angezeigt werden. Dieser Aufwand bezieht sich auf den ausgewählten Zugriffsplan bis zu dem Operator (einschließlich), für den die Informationen erfasst wurden.

- Der Gesamtaufwand (in Timerons)
- Die Anzahl der Seiten-E/As
- Die Anzahl von CPU-Instruktionen
- Der Aufwand (in Timerons) für das Abrufen der ersten Zeile einschließlich eines evtl. zusätzlich erforderlichen einleitenden Systemaufwands
- Der Übertragungsaufwand (in Rahmen (Frames))

Timerons ist die Bezeichnung für eine künstliche relative Maßeinheit. Timerons werden vom Optimierungsprogramm auf der Basis von Intervallwerten bestimmt, z. B. Statistikdaten, die sich aufgrund der Datenbanknutzung ändern. Daher kann nicht sichergestellt werden, dass das Timerons-Maß für eine SQL-Anweisung bei jeder Ermittlung der geschätzten Kosten in Timerons identisch ist.

Operatoreigenschaften: Die folgenden Informationen werden von der EXPLAIN-Einrichtung aufgezeichnet, um die Eigenschaften jedes Operators zu beschreiben:

- Die Menge der Tabellen, auf die zugegriffen wurde
- Die Menge der Spalten, auf die zugegriffen wurde
- Die Spalten, nach denen die Daten sortiert werden, wenn das Optimierungsprogramm ermittelte, dass diese Sortierung von nachfolgenden Operatoren verwendet werden kann
- Die Menge der Vergleichselemente, die angewandt wurden
- Die geschätzte Anzahl von Zeilen, die übergeben werden sollen (Kardinalität)

Organisation von EXPLAIN-Informationen

Sämtliche EXPLAIN-Informationen sind nach dem Konzept eines EXPLAIN-Exemplars organisiert. Ein EXPLAIN-Exemplar stellt einen Aufruf der EXPLAIN-Einrichtung für eine oder mehrere SQL-Anweisungen dar. Ein EXPLAIN-Exemplar enthält die EXPLAIN-Informationen für:

- Alle auswählbaren SQL-Anweisungen eines Pakets für **statische** SQL-Anweisungen
- Eine bestimmte SQL-Anweisung für SQL-Anweisungen zum inkrementellen Binden (Incremental Bind)
- Eine bestimmte SQL-Anweisung für **dynamische** SQL-Anweisungen
- Jede SQL-Anweisung EXPLAIN (dynamisch und statisch)

Die EXPLAIN-Informationen, die innerhalb eines EXPLAIN-Exemplars erfasst werden, schließen die Umgebung der SQL-Kompilierung und den zur Ausführung der SQL-Anweisung, die kompiliert wird, ausgewählten Zugriffsplan mit ein. EXPLAIN-Informationen werden für drei Untergruppen getrennt organisiert:

EXPLAIN-Exemplare	Informationen zur Kompilierungsumgebung, die für jedes EXPLAIN-Exemplar erfasst werden
EXPLAIN-Momentaufnahmen	Informationen, die von Visual Explain verwendet werden
EXPLAIN-Tabellen	Informationen, die gesammelt werden, wenn die EXPLAIN-Tabelleninformationen angefordert werden

Informationen über EXPLAIN-Exemplare

Informationen über EXPLAIN-Exemplare werden in der Tabelle EXPLAIN_INSTANCE gespeichert. Zusätzliche spezifische Informationen zu jeder SQL-Anweisung, die innerhalb eines EXPLAIN-Exemplars bearbeitet wurde, werden in der Tabelle EXPLAIN_STATEMENT gespeichert.

Kennzeichnung von EXPLAIN-Exemplaren: Anhand der folgenden Informationen können Sie jedes EXPLAIN-Exemplar identifizieren und die Informationen für SQL-Anweisungen einem bestimmten Aufruf der Einrichtung zuordnen:

- Der Benutzer, der die EXPLAIN-Informationen anforderte
- Der Zeitpunkt, zu dem die EXPLAIN-Anforderung begann
- Der Name des Pakets, aus dem die mit EXPLAIN bearbeitete SQL-Anweisung kam
- Das Schema des Pakets, aus dem die mit EXPLAIN bearbeitete SQL-Anweisung kam
- Eine Angabe, ob eine Momentaufnahme Teil der EXPLAIN-Anforderung war

Einstellungen der Umgebung: Es werden Umgebungsinformationen darüber erfasst, wie der SQL-Compiler die Abfragen optimierte. Zu den Umgebungsinformationen gehören die folgenden:

- Die Versions- und Release-Nummer für die verwendete Stufe von DB2
- Der Grad der zum Kompilieren der Abfrage verwendeten Parallelität. Zur Festlegung des Grads der Parallelität beim Kompilieren einer bestimmten Abfrage kann das Sonderregister CURRENT DEGREE, die Bindeoption DEGREE, die API SET RUNTIME DEGREE und der Konfigurationsparameter *dft_degree* verwendet werden.
- Die Angabe, ob es sich um eine dynamische oder statische SQL-Anweisung handelt
- Die zum Kompilieren der Abfrage verwendete Optimierungsklasse. Weitere Informationen finden Sie in „Anpassen der Optimierungsklasse" auf Seite 76.
- Der Typ der Cursorblockung, der beim Kompilieren der Abfrage angegeben wurde. Weitere Informationen zu Cursorn finden Sie im Handbuch *SQL Reference*. Weitere Informationen zur Cursorblockung finden Sie in „Zeilenblockung" auf Seite 90.
- Die beim Kompilieren der Abfrage verwendete Isolationsstufe. Weitere Informationen finden Sie in „Gemeinsamer Zugriff" auf Seite 47.
- Die Werte verschiedener Konfigurationsparameter zum Zeitpunkt der Kompilierung der Abfrage. Im Abschnitt „Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung" auf Seite 103 finden Sie weitere Informationen über die Konfigurationsparameter, die Auswirkungen auf die

Abfrageoptimierung haben können, einschließlich der folgenden Parameter, die aufgezeichnet werden, wenn eine EXPLAIN-Momentaufnahme erstellt wird:

- „Größe des Pufferpools (buffpage)" auf Seite 404
- „Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422
- „Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)" auf Seite 463
- „Zwischenspeicher für Datenbank (dbheap)" auf Seite 408
- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
- „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450
- „CPU-Geschwindigkeit (cpuspeed)" auf Seite 549
- „Übertragungsbandbreite (comm_bandwidth)" auf Seite 549.

Kennzeichnung von SQL-Anweisungen: Für jedes EXPLAIN-Exemplar können mehrere SQL-Anweisungen mit EXPLAIN bearbeitet worden sein. Anhand der folgenden Informationen zusammen mit Informationen, die das EXPLAIN-Exemplar eindeutig kennzeichnen, kann jede einzelne SQL-Anweisung identifiziert werden.

- Die Anweisungsart: SELECT, DELETE, INSERT, UPDATE, positioniertes DELETE, positioniertes UPDATE.
- Die Anweisungs- und Abschnittsnummer des Pakets, das die SQL-Anweisung absetzt, wie in der Katalogsicht SYSCAT.STATEMENTS gespeichert.

In der Tabelle EXPLAIN_STATEMENT enthalten die Felder QUERYTAG und QUERYNO Kennungen, die im Rahmen des EXPLAIN-Prozesses für Sie mit Werten gefüllt werden.

Für dynamische SQL-EXPLAIN-Anweisungen, die in einer Sitzung des Befehlszeilenprozessors (CLP) oder einer Sitzung von Call Level Interface (CLI) übergeben wurden, wird als Wert für QUERYTAG „CLP" bzw. „CLI" gespeichert, wenn EXPLAIN MODE oder EXPLAIN SNAPSHOT aktiv ist. Wenn dies der Fall ist, wird für QUERYNO standardmäßig eine Nummer angegeben, die um eins oder mehr für jede Anweisung erhöht wird.

Für alle anderen dynamischen SQL-EXPLAIN-Anweisungen (d. h. nicht über CLP, CLI bzw. mit Hilfe der SQL-Anweisung EXPLAIN gefordert) wird das Feld QUERYTAG mit Leerzeichen und das Feld QUERYNO immer mit dem Wert 1 gefüllt.

Schätzung des Aufwands: Für jede mit EXPLAIN bearbeitete Anweisung wird ein Schätzwert des relativen Aufwands zur Ausführung des ausgewählten Zugriffsplans aufgezeichnet. Dieser Schätzwert wird in einer künstlichen

relativen Maßeinheit, den *Timerons*, angegeben. Schätzwerte für die benötigten Ausführungszeiten werden aus folgenden Gründen **nicht** zur Verfügung gestellt:

- Das SQL-Optimierungsprogramm schätzt nicht die benötigte Zeit, sondern den Ressourcenbedarf ab.
- Das Optimierungsprogramm modelliert nicht alle Faktoren nach, die die benötigte Zeit beeinflussen können. Es werden die Faktoren ignoriert, die keine Auswirkung auf die Effizienz des Zugriffsplans haben. Die benötigte Zeit **wird** jedoch von einer Reihe Laufzeitfaktoren beeinflusst, zu denen die folgenden gehören: die Systemauslastung, die Ressourcenverfügbarkeit, der Umfang der Parallelverarbeitung und der Ein-/Ausgabeoperationen, der Aufwand für die Rückgabe von Zeilen an den Benutzer sowie die Übertragungszeit zwischen Client und Server.

Anweisungstext: Für jede mit EXPLAIN bearbeitete Anweisung werden zwei Versionen des Texts der SQL-Anweisung aufgezeichnet. Eine Version ist der Text, wie er vom SQL-Compiler empfangen wurde. Die andere Version des Anweisungstexts ist eine Rückübersetzung aus der internen Compilerdarstellung der Abfrage. Diese Rückübersetzung sieht zwar anderen SQL-Anweisungen sehr ähnlich, folgt aber **nicht** unbedingt der richtigen SQL-Syntax und spiegelt nicht in jedem Fall den tatsächlichen Inhalt der internen Darstellung als Ganzes wider. Sie wird nur für das Verständnis des SQL-Kontexts, in dem das Optimierungsprogramm den Zugriffsplan auswählte, zur Verfügung gestellt. Der Vergleich des Textes der benutzererstellten Anweisung mit der internen Darstellung der SQL-Anweisung kann Ihnen dabei helfen, zu verstehen, wie der SQL-Compiler Ihre Abfrage zur Optimierung umgeschrieben hat. (Siehe „Umschreiben der Abfrage durch den SQL-Compiler" auf Seite 176.) Der Anweisungstext zeigt Ihnen auch, welche anderen Elemente Ihrer Umgebung, z. B. Auslöser (Trigger) und Integritätsbedingungen, Auswirkungen auf Ihre Anweisung haben. Einige Schlüsselwörter, die in diesem „optimierten" Text verwendet werden, sind:

$Cn	Der Name der abgeleiteten Spalte, wobei n ein ganzzahliger Wert ist.
$CONSTRAINT$	Dieses Kennzeichen markiert den Namen einer Integritätsbedingung, die der ursprünglichen SQL-Anweisung bei der Kompilierung hinzugefügt wurde. Es ist in Kombination mit dem Präfix $WITH_CONTEXT$ zu sehen.
$DERIVED.Tn	Der Name einer abgeleiteten Tabelle, wobei n ein ganzzahliger Wert ist.
$INTERNAL_FUNC$	Dieses Kennzeichen markiert das Vorhandensein einer Funktion, die vom SQL-Compiler für die mit EXPLAIN bearbeitete Abfrage

verwendet wurde, jedoch nicht zur allgemeinen Verwendung verfügbar ist.

$INTERNAL_PRED$ Dieses Kennzeichen markiert das Vorhandensein eines Vergleichselements, das vom SQL-Compiler bei der Kompilierung der mit EXPLAIN bearbeiteten Abfrage hinzugefügt wurde. Auch ein solches Vergleichselement ist nicht zur allgemeinen Verwendung verfügbar. Vom Compiler wird ein internes Vergleichselement verwendet, um den als Ergebnis von Auslösern und Integritätsbedingungen der ursprünglichen SQL-Anweisung hinzugefügten Bedingungskontext zu erfüllen.

RID Dieses Kennzeichen dient zur Identifizierung der Spalte der Satz-ID (RID) für eine bestimmte Zeile.

$TRIGGER$ Dieses Kennzeichen markiert den Namen eines Auslösers, der der ursprünglichen SQL-Anweisung bei der Kompilierung hinzugefügt wurde. Es ist in Kombination mit dem Präfix $WITH_CONTEXT$ zu sehen.

$WITH_CONTEXT$(...) Dieses Präfix tritt am Anfang des Texts auf, wenn zusätzliche Auslöser oder Integritätsbedingungen in die ursprüngliche SQL-Anweisung eingefügt wurden. Diesem Präfix folgt eine Liste der Namen aller Auslöser oder Integritätsbedingungen, die sich auf die Kompilierung und Auflösung der SQL-Anweisung auswirken.

Informationen der EXPLAIN-Momentaufnahmen

Wenn eine Momentaufnahme (Snapshot) angefordert wird, werden zusätzliche EXPLAIN-Informationen aufgezeichnet, die den Zugriffsplan beschreiben, der vom SQL-Optimierungsprogramm ausgewählt wurde. Diese Informationen werden in der Spalte SHAPSHOT der Tabelle EXPLAIN_STATEMENT im für Visual Explain erforderlichen Format gespeichert. Dieses Format kann von anderen Anwendungen nicht verwendet werden.

Weitere Informationen zum Inhalt der Informationen der EXPLAIN-Momentaufnahmen erhalten Sie direkt mit Hilfe von Visual Explain sowie in folgenden Abschnitten:

- „EXPLAIN-Informationen für Datenobjekte" auf Seite 258
- „EXPLAIN-Informationen für Datenoperatoren" auf Seite 259.

Informationen der EXPLAIN-Tabellen

Wenn EXPLAIN-Tabelleninformationen angefordert werden, werden zusätzliche Informationen aufgezeichnet, die den vom SQL-Optimierungsprogramm ausgewählten Zugriffsplan beschreiben. Diese Informationen werden in den folgenden EXPLAIN-Tabellen gespeichert:

- EXPLAIN_ARGUMENT. Diese Tabelle enthält die spezifischen Merkmale für jeden einzelnen Operator (sofern vorhanden).
- EXPLAIN_INSTANCE. Diese Tabelle ist die Hauptsteuertabelle für alle EXPLAIN-Informationen. Jede Datenzeile in den EXPLAIN-Tabellen ist explizit mit einer eindeutigen Zeile in dieser Tabelle verbunden. In dieser Tabelle werden grundlegende Informationen über die Quelle der SQL-Anweisungen, die mit EXPLAIN bearbeitet werden, und Umgebungsinformationen gespeichert.
- EXPLAIN_OBJECT. Diese Tabelle identifiziert die Datenobjekte, die für den Zugriffsplan, der zur Erfüllung der SQL-Anweisung generiert wurde, erforderlich sind.
- EXPLAIN_OPERATOR. Diese Tabelle enthält alle Operatoren, die zur Erfüllung der SQL-Anweisung durch den SQL-Compiler benötigt werden.
- EXPLAIN_PREDICATE. Diese Tabelle enthält Informationen darüber, welche Vergleichselemente von einem bestimmten Operator angewandt werden.
- EXPLAIN_STATEMENT. Diese Tabelle enthält den Text der SQL-Anweisung, wie sie für die verschiedenen Stufen der EXPLAIN-Informationen existiert. Die ursprüngliche SQL-Anweisung, wie sie vom Benutzer eingegeben wurde, wird in dieser Tabelle zusammen mit der vom Optimierungsprogramm zum Auswählen eines Zugriffsplans für die Ausführung der SQL-Anweisung verwendeten Version gespeichert.
- EXPLAIN_STREAM. Diese Tabelle stellt die Eingabe- und Ausgabedatenströme zwischen einzelnen Operatoren und Datenobjekten dar. Die Datenobjekte selbst werden in der Tabelle EXPLAIN_OBJECT dargestellt. Die an einem Datenstrom beteiligten Operatoren werden in der Tabelle EXPLAIN_OPERATOR dargestellt.
- ADVISE_WORKLOAD. Diese Tabelle ermöglicht Benutzern, die durch Ihre Arbeit entstehende Auslastung für die Datenbank zu beschreiben. Jede Zeile in dieser Auslastungstabelle stellt eine SQL-Anweisung dar und wird durch eine zugeordnete Häufigkeit beschrieben. Diese Tabelle wird vom Programm `db2advis` und vom *Assistent: Index* verwendet, um Auslastungsinformationen aufzunehmen und zu speichern.
- ADVISE_INDEX. In dieser Tabelle werden Informationen über empfohlene Indizes gespeichert. Die Tabelle wird durch den SQL-Compiler, das Dienstprogramm `db2advis`, den *Assistent: Index* oder einen Benutzer mit Werten gefüllt. Diese Tabelle wird zu zwei Zwecken verwendet:
  - Zum Abrufen empfohlener Indizes.

– Zum Auswerten von Indizes auf der Grundlage von Eingaben zu vorgeschlagenen Indizes.

Die oben genannten Tabellen werden nicht standardmäßig erstellt. Sie können durch Ausführen der Prozedur EXPLAIN.DDL erstellt werden, die sich im Unterverzeichnis `misc` des Unterverzeichnisses `sqllib` befindet. Stellen Sie die Verbindung (Connect) zu der Datenbank her, in der die EXPLAIN- und ADVISE-Tabellen benötigt werden. Geben Sie dann den Befehl `db2 -tf EXPLAIN.DDL` ein, um die Tabellen zu erstellen. Die Tabellen könnten ebenso bei Bedarf automatisch durch den *Assistent: Index* erstellt werden.

Jeder von Visual Explain rechteckig dargestellte *Objektknoten* entspricht einer Zeile in der Tabelle EXPLAIN_OBJECT. Jeder von Visual Explain achteckig dargestellte „Operatorknoten" entspricht einer Zeile in der Tabelle EXPLAIN_OPERATOR. Jede Verbindung zwischen Operatoren bzw. Operatorobjekten entspricht einer Zeile in der Tabelle EXPLAIN_STREAM.

Die EXPLAIN-Tabelleninformationen ähneln inhaltlich den Informationen, die für eine EXPLAIN-Momentaufnahme aufgezeichnet werden, jedoch werden diese Informationen in regulären relationalen Tabellen gespeichert, was den Zugriff mit Hilfe von SQL-Standardanweisungen ermöglicht.

Wie auch das Zugriffsplandiagramm von Visual Explain sind EXPLAIN-Tabellen dazu gedacht, die Beziehungen zwischen Operatoren und Datenobjekten innerhalb des Zugriffsplans darzustellen. Die folgende Abbildung zeigt die Beziehungen zwischen diesen Tabellen.

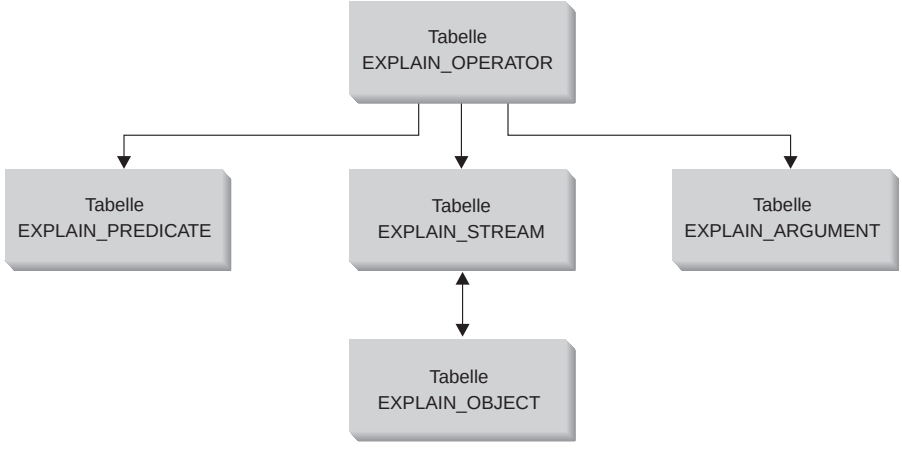


Abbildung 21. Beziehungen zwischen den EXPLAIN-Tabellen (nicht alle Tabellen gezeigt)

Es ist möglich, EXPLAIN-Tabellen zu haben, die zu mehr als einem Benutzer gehören. Die EXPLAIN-Tabellen können für einen Benutzer definiert wer-

den. In diesem Fall können Aliasnamen mit Hilfe desselben Namens für jeden weiteren Benutzer definiert werden, die auf die definierten Tabellen verweisen. Jeder Benutzer, der auf die gemeinsamen EXPLAIN-Tabellen zugreift, muss das Zugriffsrecht INSERT zum Einfügen für diese Tabellen haben.

In „Anhang C. EXPLAIN-Programme (SQL)" auf Seite 645 finden Sie weitere Informationen zu den EXPLAIN-Tabellen und zu ihrer Erstellung. Weitere Informationen zum Inhalt der EXPLAIN-Tabelleninformationen finden Sie in folgenden Abschnitten:

- „EXPLAIN-Informationen für Datenobjekte" auf Seite 258
- „EXPLAIN-Informationen für Datenoperatoren" auf Seite 259.

Das Tool `db2exfmt`, das im Unterverzeichnis `misc` des Verzeichnisses `sqllib` zur Verfügung steht, kann zur Formatierung des Inhalts der EXPLAIN-Tabellen in eine lesbare, strukturierte Ausgabe verwendet werden.

Abrufen von EXPLAIN-Daten

Bevor Sie EXPLAIN-Daten für eine SQL-Anweisung abrufen können, müssen Sie über eine Reihe von EXPLAIN-Tabellen verfügen, die mit demselben Schema wie die zum Aufrufen der EXPLAIN-Einrichtung verwendete Berechtigungs-ID definiert sind. Informationen über das Erstellen der Tabellen finden Sie in „Tabellendefinitionen für EXPLAIN-Tabellen" auf Seite 634.

Erfassen von EXPLAIN-Tabelleninformationen

Wenn diese Tabellen definiert sind, werden EXPLAIN-Daten erfasst, wenn eine SQL-Anweisung kompiliert wird und EXPLAIN-Daten angefordert wurden:

- Bei **statischen** SQL-Anweisungen oder SQL-Anweisungen für **inkrementelles Binden** (Incremental Bind) werden EXPLAIN-Tabelleninformationen erfasst, wenn eine der beiden Optionen EXPLAIN ALL oder EXPLAIN YES in den Befehlen BIND oder PREP angegeben wird oder wenn eine statische SQL-Anweisung EXPLAIN im Quellenprogramm verwendet wird.

  Anmerkung: Wenn SQL-Anweisungen zum inkrementellen Binden (Incremental Bind) zur Laufzeit kompiliert werden, werden Sie zur Laufzeit, nicht während der Bindeoperation, in den EXPLAIN-Tabellen platziert. Das Qualifikationsmerkmal und die Berechtigungs-ID der EXPLAIN-Tabelle, die für den Einfügevorgang in die EXPLAIN-Tabellen verwendet werden, sind die des Paketeigners und nicht die des Benutzers, der das Paket ausführt.
- Bei **dynamischen** SQL-Anweisungen werden unter einer der folgenden Bedingungen EXPLAIN-Tabelleninformationen erfasst:

– Eine SQL-Anweisung EXPLAIN wird verwendet. Alle EXPLAIN-Informationen werden erfasst und in den EXPLAIN-Tabellen gespeichert, sofern nicht die Klausel FOR SNAPSHOT angegeben wird.

  Beispiel einer SQL-Anweisung EXPLAIN:

  ```
  EXPLAIN PLAN FOR <eine gültige SQL-Anweisung DELETE, INSERT, SELECT, SELECT INTO,
     UPDATE, VALUES oder VALUES INTO>
  ```

– Das Sonderregister CURRENT EXPLAIN MODE ist auf YES gesetzt. Diese Einstellung bewirkt, dass der SQL-Compiler EXPLAIN-Daten erfasst, wobei die SQL-Anweisung ausgeführt wird und die Ergebnisse der Abfrage zurückgibt.
– Das Sonderregister CURRENT EXPLAIN MODE ist auf EXPLAIN gesetzt. Diese Einstellung bewirkt, dass der SQL-Compiler EXPLAIN-Daten erfasst, jedoch wird die SQL-Anweisung nicht ausgeführt.
– Das Sonderregister CURRENT EXPLAIN MODE ist auf RECOMMEND INDEXES gesetzt. Diese Einstellung weist den SQL-Compiler an, EXPLAIN-Daten zu erfassen, und sorgt dafür, dass empfohlene Indizes in die Tabelle ADVISE_INDEX eingefügt werden. Die SQL-Anweisung wird jedoch nicht ausgeführt.
– Das Sonderregister CURRENT EXPLAIN MODE ist auf EVALUATE INDEXES gesetzt. Diese Einstellung bewirkt, dass der SQL-Compiler die vom Benutzer in die Tabelle ADVISE_INDEX eingefügten Indizes verwendet. Der Benutzer fügt eine neue Zeile für jeden Index ein, der ausgewertet werden soll. Die für jeden Index erforderlichen Informationen sind: Indexname, Tabellenname und die Spaltennamen, aus denen der auszuwertende Index besteht. Wenn das Sonderregister CURRENT EXPLAIN MODE eingegeben wurde, sollte es auf EVALUATE INDEXES gesetzt werden. Der SQL-Compiler durchsucht dann die Tabelle ADVISE_INDEX, in der das Feld USE_INDEX auf „Y" festgelegt ist (diese Indizes werden als virtuelle Indizes bezeichnet). Alle im Modus EVALUATE INDEXES ausgeführten dynamischen Anweisungen werden von EXPLAIN so bearbeitet, als wären diese virtuellen Indizes verfügbar. Der SQL-Compiler wählt in diesem Fall die virtuellen Indizes, wenn sie die Leistung der Anweisungen verbessern. Ansonsten werden die Indizes ignoriert. Durch Prüfen der EXPLAIN-Ergebnisse können Sie feststellen, ob die vom Benutzer vorgeschlagenen Indizes vom SQL-Compiler verwendet wurden. Die verwendeten Indizes sollten zur Verbesserung des Zugriffs implementiert werden.
– Die Option EXPLAIN ALL wurde im Befehl BIND oder PREP angegeben. Diese Einstellung bewirkt, dass der SQL-Compiler EXPLAIN-Daten für dynamisches SQL zur Laufzeit erfasst, auch wenn die Einstellung des Sonderregisters CURRENT EXPLAIN MODE den Wert NO hat. Die SQL-Anweisung wird ausgeführt, und die Abfrageergebnisse werden zurückgegeben.

Anmerkung: Die EXPLAIN-Informationen werden nur erfasst, wenn die SQL-Anweisung kompiliert wird. Nach der einleitenden Kompilierung werden dynamische SQL-Anweisungen nur erneut kompiliert, wenn eine Änderung an der Umgebung das erneute Kompilieren einer Anweisung erforderlich macht. Wenn dieselbe Anweisung PREPARE mehrmals für die gleiche SQL-Anweisung ausgeführt wird, erfolgt das Kompilieren der SQL-Anweisung und daher auch das Erfassen der EXPLAIN-Daten nur beim ersten Ausführen der Anweisung PREPARE, da angenommen wird, dass sich die Umgebung nicht ändert.

Weitere Informationen zur Verwendung der SQL-Anweisung EXPLAIN bzw. zur Verwendung des Registers CURRENT EXPLAIN MODE finden Sie im Handbuch *SQL Reference*. Weitere Informationen zu den Befehlen BIND und PREP finden Sie im Handbuch *Command Reference*.

Erfassen von EXPLAIN-Momentaufnahmedaten

EXPLAIN-Momentaufnahmedaten werden erfasst, wenn eine SQL-Anweisung kompiliert wird und EXPLAIN-Daten angefordert wurden:

- Bei **statischen** SQL-Anweisungen oder SQL-Anweisungen für **inkrementelles Binden** (Incremental Bind) wird eine EXPLAIN-Momentaufnahme erfasst, wenn eine der beiden Klauseln EXPLSNAP ALL oder EXPLSNAP YES in den Befehlen BIND oder PREP angegeben wird oder wenn eine statische SQL-Anweisung EXPLAIN mit der Klausel FOR SNAPSHOT oder WITH SNAPSHOT im Quellenprogramm verwendet wird.

  Anmerkung: Wenn SQL-Anweisungen zum inkrementellen Binden (Incremental Bind) zur Laufzeit kompiliert werden, werden Sie zur Laufzeit, nicht während der Bindeoperation, in den EXPLAIN-Tabellen platziert. Das Qualifikationsmerkmal und die Berechtigungs-ID der EXPLAIN-Tabelle, die für den Einfügevorgang in die EXPLAIN-Tabellen verwendet werden, sind die des Paketeigners und nicht die des Benutzers, der das Paket ausführt.

- Für **dynamische** SQL-Anweisungen wird unter einer der folgenden Bedingungen eine EXPLAIN-Momentaufnahme erfasst:
  - Eine SQL-Anweisung EXPLAIN mit der Klausel FOR SNAPSHOT oder WITH SNAPSHOT wird verwendet. Bei Angabe der Klausel FOR SNAPSHOT werden außer den Daten für die EXPLAIN-Momentaufnahme keine EXPLAIN-Tabelleninformationen erfasst. Mit der Klausel WITH SNAPSHOT werden neben den Daten für die EXPLAIN-Momentaufnahme auch alle EXPLAIN-Tabelleninformationen erfasst.

Das folgende Beispiel zeigt eine SQL-Anweisung EXPLAIN mit der Angabe zur Erstellung einer EXPLAIN-Momentaufnahme:

```
EXPLAIN PLAN FOR SNAPSHOT FOR <eine gültige SQL-Anweisung DELETE,
INSERT, SELECT, SELECT INTO, UPDATE, VALUES oder VALUES INTO SQL>
```

Es wird lediglich eine EXPLAIN-Momentaufnahme erstellt, und die erfassten Informationen werden in den Tabellen EXPLAIN_INSTANCE und EXPLAIN_STATEMENT gespeichert.

- Das Sonderregister CURRENT EXPLAIN SNAPSHOT ist auf YES gesetzt. Diese Einstellung bewirkt, dass der SQL-Compiler eine Momentaufnahme der EXPLAIN-Daten erstellt, wobei die SQL-Anweisung ausgeführt wird und die Abfrageergebnisse übergeben werden.
- Das Sonderregister CURRENT EXPLAIN SNAPSHOT ist auf EXPLAIN gesetzt. Diese Einstellung bewirkt, dass der SQL-Compiler eine Momentaufnahme der EXPLAIN-Daten erstellt, wobei die SQL-Anweisung jedoch nicht ausgeführt wird.
- Die Option EXPLSNAP ALL wurde im Befehl BIND oder PREP angegeben. Diese Einstellung bewirkt, dass der SQL-Compiler eine Momentaufnahme der EXPLAIN-Daten zur Laufzeit erstellt, auch wenn die Einstellung des Sonderregisters CURRENT EXPLAIN SNAPSHOT auf NO gesetzt ist. Die SQL-Anweisung wird ausgeführt, und die Abfrageergebnisse werden zurückgegeben.

Anmerkung: Die EXPLAIN-Informationen werden nur erfasst, wenn die SQL-Anweisung kompiliert wird. Nach der einleitenden Kompilierung werden dynamische SQL-Anweisungen nur erneut kompiliert, wenn eine Änderung an der Umgebung das erneute Kompilieren einer Anweisung erforderlich macht. Wenn dieselbe Anweisung PREPARE mehrmals für die gleiche SQL-Anweisung ausgeführt wird, erfolgt das Kompilieren der SQL-Anweisung und daher auch das Erfassen der EXPLAIN-Daten nur beim ersten Ausführen der Anweisung PREPARE, da angenommen wird, dass sich die Umgebung nicht ändert.

Weitere Informationen zur Verwendung der SQL-Anweisung EXPLAIN und der Klauseln FOR SNAPSHOT bzw. WITH SNAPSHOT sowie zur Verwendung des Sonderregisters CURRENT EXPLAIN SNAPSHOT finden Sie im Handbuch *SQL Reference*. Weitere Informationen zu den Befehlen BIND und PREP finden Sie im Handbuch *Command Reference*.

Richtlinien zur Verwendung der EXPLAIN-Ausgabe

Die Analyse der EXPLAIN-Daten kann Ihnen auf verschiedene Arten dabei helfen, Ihre Abfragen und Ihre Umgebung zu optimieren. Beispiele:

- **Werden Indizes verwendet?**

  Wie im Abschnitt „Auswirkung des Indexierens auf die Abfrageoptimierung" auf Seite 111 erläutert, kann durch die Verwendung der geeigneten Indizes eine erhebliche Leistungssteigerung erzielt werden. Anhand der EXPLAIN-Ausgabe können Sie ermitteln, ob die von Ihnen für eine bestimmte Gruppe von Abfragen erstellten Indizes tatsächlich verwendet werden. In der EXPLAIN-Ausgabe sollten Sie folgende Bereiche auf die Verwendung von Indizes hin überprüfen:
  - Verknüpfungsvergleichselemente
  - Lokale Vergleichselemente
  - Die Klausel GROUP BY
  - Die Klausel ORDER BY
  - Die SELECT-Liste

  Sie können die EXPLAIN-Einrichtung auch verwenden, um zu ermitteln, ob ein anderer Index anstelle des vorhandenen Index oder kein Index verwendet werden kann. Nach Erstellen eines neuen Index sollten Sie Statistikdaten für den betreffenden Index sammeln (mit dem Befehl RUNSTATS) und Ihre Abfrage erneut kompilieren. Mit der Zeit werden Sie möglicherweise durch die EXPLAIN-Daten feststellen, dass anstelle einer Indexsuche eine Tabellensuche verwendet wird. Dies kann sich aus einer Änderung in der Clusterbildung der Tabellendaten ergeben. Wenn der zuvor verwendete Index nun ein niedriges Clusterverhältnis aufweist, sollten Sie folgende Maßnahmen in Erwägung ziehen:
  - Reorganisieren der Tabelle, so dass die Clusterbildung für die Daten gemäß dem entsprechenden Index gestaltet wird
  - Aktualisieren der Katalogstatistik für die Tabelle und den Index mit dem Befehl RUNSTATS
  - Erneutes Kompilieren der Abfrage
  - Prüfen der Ausgabe von EXPLAIN, um herauszufinden, ob das Reorganisieren Ihrer Tabelle zu einer Verbesserung des Zugriffsplans geführt hat

- **Ist die Zugriffsart für Ihre Anwendung geeignet?**

  Sie können die EXPLAIN-Ausgabe analysieren und nach Arten des Zugriffs auf die Daten durchsuchen, die normalerweise für den verwendeten Anwendungstyp nicht optimal sind. Beispiele:

  - **Abfragen der Online-Transaktionsverarbeitung (OLTP-Abfragen)**

    In OLTP-Anwendungen werden häufig Indexsuchen mit Vergleichselementen durchgeführt, die eine Bereichsbegrenzung vornehmen, weil diese Anwendungen in der Regel nur wenige Zeilen zurückgeben, die den mit einem Gleichheitsvergleichselement für eine Schlüsselspalte angegebenen Kriterien entsprechen. Wenn Ihre OLTP-Abfragen eine

Tabellensuche verwenden, können Sie die EXPLAIN-Daten analysieren, um herauszufinden, warum keine Indexsuche verwendet wurde.

– **Abfragen, bei denen nur Daten angezeigt werden**

  Die Suchkriterien für eine Abfrage, bei der Daten nur angezeigt werden, können sehr vage sein, was bewirkt, dass eine große Menge von Zeilen den Kriterien entspricht. Wenn der Benutzer sich normalerweise nur einige Seiten der Ausgabedaten anzeigen lässt, können Sie dafür sorgen, dass nicht die gesamte Antwortmenge errechnet werden muss, bevor einige Ergebnisse übergeben werden. In diesem Fall unterscheiden sich die Ziele des Benutzers vom grundlegenden Verarbeitungsprinzip des Optimierungsprogramms, das versucht, den Ressourcenbedarf für die gesamte Abfrage und nicht nur für die ersten wenigen Anzeigen mit Daten zu minimieren. Wenn z. B. die EXPLAIN-Ausgabe zeigt, dass Operatoren sowohl für Mischverknüpfungen als auch für Sortierungen im Zugriffsplan verwendet wurden, wird die gesamte Antwortmenge in einer temporären Tabelle gespeichert, bevor Zeilen an die Anwendung zurückgegeben werden. In diesem Fall können Sie versuchen, den Zugriffsplan durch die Verwendung der Klausel OPTIMIZE FOR in der SELECT-Anweisung zu ändern. (Weitere Informationen zur Klausel OPTIMIZE FOR finden Sie in „Klausel OPTIMIZE FOR n ROWS" auf Seite 87.) Auf diese Weise kann das Optimierungsprogramm versuchen, einen Zugriffsplan auszuwählen, der nicht die gesamte Antwortmenge in einer temporären Tabelle erstellt, bevor die ersten Zeilen an die Anwendung zurückgegeben werden.

- **Welche Verknüpfungsart wird verwendet?**

  Wenn bei einer Abfrage zwei Tabellen verknüpft werden, können Sie die Art der verwendeten Verknüpfungsverarbeitung überprüfen. Verknüpfungen mit relativ vielen Zeilen, wie sie z. B. bei Abfragen von Entscheidungshilfedaten auftreten, werden in der Regel schneller als Mischverknüpfungen (Merge Joins) ausgeführt. Verknüpfungen, die nur einige wenige Zeilen betreffen, wie z. B. für OLTP-Abfragen, sind hingegen als Verknüpfungen über Verschachtelungsschleife (Nested Loop Join) effizienter. In beiden Fällen kann es jedoch auch Umstände geben, wie beispielsweise die Verwendung lokaler Vergleichselemente oder Indizes, die die normale Verarbeitung dieser Verknüpfungen ändern. (Informationen über die Funktionsweise dieser beiden Verknüpfungsmethoden finden Sie in den Abschnitten „Verknüpfung über Verschachtelungsschleife" auf Seite 204 und „Mischverknüpfung" auf Seite 206).

Visual Explain

Visual Explain kann zur genaueren Untersuchung von Abfragen verwendet werden, wenn es zum Vergleich mit anderen Methoden, besonders solcher mit komplexen Folgen von Operationen, herangezogen wird. Visual Explain steht nicht auf allen unterstützten Plattformen zur Verfügung. Das Handbuch *Einstieg* für Ihre Plattform enthält Informationen darüber, ob Visual Explain unterstützt wird.

Visual Explain zeigt den Zugriffsplan für mit EXPLAIN bearbeitete SQL-Anweisungen in einem Diagramm an. Die dem Diagramm zu entnehmenden Informationen können bei der Optimierung Ihrer SQL-Abfragen für bessere Leistung dienlich sein. Außerdem kann mit Visual Explain eine SQL-Anweisung dynamisch mit EXPLAIN bearbeitet und der sich ergebende Zugriffsplan im Diagramm angezeigt werden.

Das Optimierungsprogramm wählt einen Zugriffsplan, und Visual Explain zeigt die Informationen in Form eines Zugriffsplandiagramms an, in dem Tabellen und Indizes sowie jede an ihnen ausgeführte Operation als Knoten und der Datenfluss als Verbindungslinien zwischen den Knoten dargestellt werden.

Um ein Zugriffsplandiagramm anzeigen zu können, müssen Sie eine EXPLAIN-Momentaufnahme erstellt haben. Einem Zugriffsplandiagramm können Sie Einzelheiten zu folgendem entnehmen:

- Tabellen und Indizes (und den zugehörigen Spalten)
- Operatoren (z. B. Tabellensuchen, Sortierungen und Verknüpfungen)
- Tabellenbereiche und Funktionen

Darüber hinaus kann Visual Explain zu folgendem verwendet werden:

- Anzeigen der Statistiken, die zur Zeit der Optimierung verwendet wurden. Sie können als eine Maßnahme zur Bestimmung, ob ein erneutes Binden des Pakets die Leistung verbessern könnte, diese Statistiken mit den aktuellen Katalogstatistiken vergleichen.
- Feststellen, ob ein Index zum Zugriff auf eine Tabelle verwendet wurde oder nicht. Wenn kein Index verwendet wurde, kann Visual Explain Ihnen hilfreiche Hinweise geben, für welche Spalten ein Index möglicherweise von Vorteil wäre.
- Ermitteln der Auswirkungen verschiedener Optimierungstechniken, indem Sie die "Vorher-Nachher-Versionen" des Zugriffsplandiagramms für eine Abfrage miteinander vergleichen.
- Anzeigen von Informationen zu jeder Operation im Zugriffsplan, einschließlich der Informationen zum Gesamtaufwand und der Anzahl von Zeilen, die abgerufen werden (Kardinalität).

Weitere Einzelheiten zu Visual Explain entnehmen Sie bitte den Online-Informationen, die über die Steuerzentrale zur Verfügung gestellt werden. Die Steuerzentrale wird durch Eingabe des Befehls `db2cc` über die Befehlszeile aufgerufen.

SQL-Advise-Einrichtung

Index Advisor ist ein Verwaltungsprogramm, das Sie teilweise von der Notwendigkeit befreit, selbst geeignete Indizes für die Daten entwerfen und definieren zu müssen.

Index Advisor dient folgenden Zwecken:

- Ermitteln der besten Indizes für eine problematische Abfrage.
- Ermitteln der besten Indizes für eine Gruppe von Abfragen (eine Auslastung) unter Ressourcengrenzen, die wahlfrei angewandt werden.
- Testen eines Index für eine Auslastung, ohne den Index erstellen zu müssen.

Es gibt einige Begriffe, die mit der SQL-Advise-Einrichtung verbunden sind. Da ist zunächst der Begriff der *Auslastung* (Workload). Eine Auslastung ist eine Gruppe von SQL-Anweisungen, die der Datenbankmanager über einen bestimmten Zeitraum hinweg verarbeiten muss. Die SQL-Anweisungen können folgende sein: SELECT, INSERT, UPDATE und DELETE. Zum Beispiel könnte der Datenbankmanager über einen Monat hinweg 1 000 INSERT-, 10 000 UPDATE-, 10 000 SELECT- und 1 000 DELETE-Operationen verarbeiten müssen. Die Informationen in der Auslastung betreffen die Art und Häufigkeit der SQL-Anweisungen über einen bestimmten Zeitraum hinweg. Die Advisor-Steuerkomponente verwendet diese Informationen zur Auslastung in Verbindung mit den Datenbankinformationen, um Indizes zu empfehlen. Der Zweck der Advisor-Steuerkomponente besteht darin, den Gesamtaufwand für die Auslastung zu minimieren.

Sodann gibt es den Begriff eines *virtuellen Index*. Virtuelle Indizes sind Indizes, die im aktuellen Datenbankschema nicht vorhanden sind. Diese Indizes können entweder von der Advise-Einrichtung empfohlene Indizes sein oder Indizes, die Sie von der Advise-Einrichtung ausgewertet haben wollen. Diese Indizes können außerdem solche sein, die von der Advise-Einrichtung als Teil des Prozesses in Betracht gezogen werden, aber anschließend wieder verworfen werden, weil sie nicht zu empfehlen sind. Virtuelle Indizes werden von Ihnen an die Advise-Einrichtung und von der Advise-Einrichtung an Sie mit Hilfe der Tabelle ADVISE_INDEX übergeben.

Die Advise-Einrichtung verwendet eine Auslastung und Statistikdaten aus der Datenbank, um empfohlene Indizes zu generieren.

Die Advise-Einrichtung verwendet zwei EXPLAIN-Tabellen:

- ADVISE_WORKLOAD

  In dieser Tabelle wird die zu berücksichtigende Auslastung beschrieben. Jede Zeile in der Tabelle stellt eine SQL-Anweisung dar und wird durch eine zugeordnete Häufigkeit beschrieben. Es gibt eine Kennung für jede Auslastung, die in einem Feld namens WORKLOAD_NAME der Tabelle gespeichert wird. Alle SQL-Anweisungen, die zur selben Auslastung gehören, müssen denselben Namen für WORKLOAD_NAME haben.

  Der *Assistent: Index* und das Programm `db2advis` verwenden die Tabelle zum Abrufen und Speichern von Informationen zu Auslastungen.

- ADVISE_INDEX

  In dieser Tabelle werden Informationen über empfohlene Indizes gespeichert. Informationen werden in diese Tabelle vom SQL-Compiler, vom *Assistent: Index*, vom Programm `db2advis` oder von Ihnen eingefügt.

  Diese Tabelle wird zu zwei Zwecken verwendet:

  – Zum Abrufen von Indexempfehlungen aus der Advise-Einrichtung
  – Zur Auswertung von Indizes

Anmerkung: Zur Erstellung dieser Tabellen führen Sie die Prozedur EXPLAIN.DDL aus, die sich im Unterverzeichnis `misc` des Unterverzeichnisses `sqllib` befindet. Wenn sie nicht bereits erstellt wurden, kann der *Assistent: Index* diese Tabellen ebenfalls erstellen.

Das Verfahren zur Verwendung von Index Advisor umfasst Eingaben, das Aufrufen von Advisor, Ausgaben und einige Sonderfälle, die betrachtet werden sollten.

Die Eingabe für Index Advisor kann auf drei Arten erstellt werden:

- Erfassen einer Auslastung

  Verwenden Sie eine der folgenden Methoden, um das auszuwertende SQL zu erstellen:

  – Verwenden des Überwachungsprogramms, um dynamisches SQL zu erhalten.
  – Verwenden der Katalogsicht SYSSTMT, um statisches SQL zu erhalten.
  – Hinzufügen von Anweisungen und Häufigkeiten (Frequency) durch Ausschneiden und Einfügen der Werte in die Tabelle ADVISE_INDEX.

- Ändern der Häufigkeiten für Auslastungen, um die Wichtigkeit von Abfragen zu erhöhen oder zu senken.
- Festlegen der Integritätsbedingungen, falls überhaupt, für die Daten.

Es gibt vier Methoden, Index Advisor aufzurufen:

- Über die Steuerzentrale

  Dies ist die empfohlene Methode, Index Advisor zu verwenden. Erweitern Sie von der Steuerzentrale aus die Objektbaumstruktur, bis Sie den Ordner **Indizes** finden. Klicken Sie mit der Maustaste 2 den Ordner **Indizes** an, und wählen Sie **Erstellen–>Index mit Assistent** im Kontextmenü aus. Daraufhin wird der *Assistent: Index* geöffnet. Zum *Assistent: Index* ist eine umfassende Hilfe verfügbar, und er ist einfach zu verwenden. Der Assistent enthält außerdem Einrichtungen zur Bildung einer Auslastung, indem er nach kürzlich ausgeführtem SQL sucht bzw. kürzlich verwendete Pakete durchsucht oder indem ihm manuell SQL-Anweisungen hinzugefügt werden.
- Über den Befehlszeilenprozessor

  Geben Sie in die Befehlszeile den Befehl `db2advis` ein. Der Befehl `db2advis` beginnt, indem er eine Auslastung aus einer von drei Positionen liest:
  - Von der Befehlszeile
  - Aus den Anweisungen in einer Textdatei
  - Aus der Tabelle ADVISE_WORKLOAD, nachdem Zeilen mit der angenommenen Auslastung (SQL und Häufigkeit) eingefügt sind.

  Das Programm verwendet dann das Register CURRENT EXPLAIN MODE, um die empfohlenen Indizes zusammen mit einem internen Optimierungsalgorithmus zur Auswahl der besten Indizes zu erhalten. Die Ausgabe erfolgt auf dem Terminalbildschirm, in die Tabelle ADVISE_INDEX und in eine Ausgabedatei, falls erwünscht.

  Im folgenden Beispiel soll das Programm Indizes für eine einfache Abfrage „select count(*) from sales where region = 'Quebec'" empfehlen:

```
$ db2advis -d sample \
 -s "select count(*) from sales where region = 'Quebec'" \
 -t 1
performing auto-bind

Bind is successful. Used bindfile: /home3/valentin/sqllib/bnd/db2advis.bnd

Calculating initial cost (without recommended indexes) [31.198040] timerons
Initial set of proposed indexes is ready.
Found maximum set of [1] recommended indexes
Cost of workload with all indexes included [2,177133] timerons
cost without index [0] is [31,198040] timerons.  Derived benefit is
[29,020907]
total disk space needed for initial set [1] MB
total disk space constrained to         [-1] MB
```

```
  1  indexes in current solution
 [31,198040] timerons  (without indexes)
 [2,177133] timerons  (with current solution)
 [%93,02] improvement

Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ689 ON VALENTIN.SALES (REGION DESC)
===========================
Index Advisor tool is finished.
```

Das Programm db2advis darüber hinaus zur Empfehlung von Indizes für eine Auslastung verwendet werden. Sie können eine Eingabedatei mit dem Namen „sample.sql" erstellen:

```
--#SET FREQUENCY 100
select count(*) from sales where region = ?;
--#SET FREQUENCY 3
select projno, sum(comm) tot_comm  from employee, emp_act
where employee.empno = emp_act.empno and
     employee.job='DESIGNER'
group by projno
order by tot_comm desc;
--#SET FREQUENCY 50
select * from sales where sales_date = ?;
```

Führen Sie anschließend den folgenden Befehl aus:

```
$ db2advis -d sample -i sample.sql -t 0
  found [3] SQL statements from the input file

Calculating initial cost (without recommended indexes) [62,331280] timerons
Initial set of proposed indexes is ready.
Found maximum set of [2] recommended indexes
Cost of workload with all indexes included [29,795755] timerons
cost without index [0] is [58,816662] timerons.  Derived benefit is
[29,020907]
cost without index [1] is [33,310373] timerons.  Derived benefit is
[3,514618]
total disk space needed for initial set [2] MB
total disk space constrained to         [-1] MB
  2  indexes in current solution
 [62,331280] timerons  (without indexes)
 [29,795755] timerons  (with current solution)
 [%52,20] improvement

Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ119 ON VALENTIN.SALES (SALES_DATE DESC,
SALES_PERSON DESC)
```

```
index[2], 1MB CREATE INDEX WIZ63 ON VALENTIN.SALES (REGION DESC)
==========================
Index Advisor tool is finished.
```

- Über selbstanweisende Methoden unter Verwendung von EXPLAIN-Modi und PREP-Befehlsoptionen

  Zum Beispiel wird das Sonderregister CURRENT EXPLAIN MODE auf RECOMMEND INDEXES gesetzt. Diese Einstellung weist den SQL-Compiler an, EXPLAIN-Daten zu erfassen, und sorgt dafür, dass empfohlene Indizes in die Tabelle ADVISE_INDEX eingefügt werden. Die SQL-Anweisung wird jedoch nicht ausgeführt.

  Oder das Sonderregister CURRENT EXPLAIN MODE wird auf EVALUATE INDEXES gesetzt. Diese Einstellung bewirkt, dass der SQL-Compiler die vom Benutzer in die Tabelle ADVISE_INDEX eingefügten Indizes verwendet. Der Benutzer fügt eine neue Zeile für jeden Index ein, der ausgewertet werden soll. Die für jeden Index erforderlichen Informationen sind: Indexname, Tabellenname und die Spaltennamen, aus denen der auszuwertende Index besteht. Wenn das Sonderregister CURRENT EXPLAIN MODE eingegeben wurde, sollte es auf EVALUATE INDEXES gesetzt werden. Der SQL-Compiler durchsucht dann die Tabelle ADVISE_INDEX, nach Indizes, bei denen das Feld USE_INDEX=„Y" ist (diese Indizes werden als virtuelle Indizes bezeichnet). Alle im Modus EVALUATE INDEXES ausgeführten dynamischen Anweisungen werden von EXPLAIN so bearbeitet, als wären diese virtuellen Indizes verfügbar. Der SQL-Compiler wählt in diesem Fall die virtuellen Indizes, wenn sie die Leistung der Anweisungen verbessern. Ansonsten werden die Indizes ignoriert. Durch Prüfen der EXPLAIN-Ergebnisse können Sie feststellen, ob die vom Benutzer vorgeschlagenen Indizes vom SQL-Compiler verwendet wurden. Die verwendeten Indizes sollten zur Verbesserung des Zugriffs implementiert werden.
- Über Call Level Interface (CLI).

  Wenn Sie diese Schnittstelle zum Schreiben von Anwendungen verwenden, können Sie Index Advisor ebenfalls verwenden.

Die Ergebnisse aus Index Advisor bieten drei Verwendungsmöglichkeiten:

- Interpretieren der Ausgabe aus Index Advisor

  Mit Hilfe der folgenden Abfrage können Sie ermitteln, welche Indizes von der Advise-Einrichtung empfohlen wurden:

  ```
  SELECT CAST(CREATION_TEXT as CHAR(200))
     FROM ADVISE_INDEX
  ```

- Anwenden der Empfehlungen von Index Advisor
- Feststellen, wann ein Index zu löschen ist

Um bessere Empfehlungen für eine bestimmte Abfrage zu erhalten, ist es zu empfehlen, die Advise-Einrichtung nur für diese Abfrage zu verwenden. Sie können sich mit Hilfe des *Assistent: Index* Indizes für eine einzelne Abfrage empfehlen lassen, indem Sie eine Auslastung erstellen, die nur diese Abfrage enthält.

Eine Beispielauslastung kann der Ausgabe von Event Monitor entnommen werden. Mit Event Monitor können Ausführungen von dynamischen SQL erfasst werden. Diese Anweisungen können anschließend als Eingabe für die Advise-Einrichtung verwendet werden.

Der *Assistent: Index* ist eine einfache, benutzerfreundliche visuelle Schnittstelle, die einen hervorragenden Zugriff auf die Advise-Einrichtung bietet.

Teil 3. Optimieren und Konfigurieren des Systems

Kapitel 8. Leistung bei der Ausführung

Unter folgenden Themen finden Sie Informationen dazu, wie Sie die Leistung einer SQL-Abfrage während der Laufzeit beeinflussen können:

- Verwendung des Speichers durch DB2
- Verwalten des Datenbankpufferpools
- Verwalten mehrerer Datenbankpufferpools
- Vorablesen von Daten in den Pufferpool
- Konfigurieren von E/A-Servern für Vorablesezugriff und parallele E/A
- Sortieren
- Reorganisieren von Katalogen und Benutzertabellen
- Überlegungen zur Leistung bei DMS-Einheiten
- Verwalten des Initialisierungsaufwands
- Datenbankagenten
- Verwenden des Datenbanksystemmonitors
- Erweitern von Speicher.

Weitere Informationen zur Beeinflussung der Leistung finden Sie in folgenden Kapiteln:

- „Kapitel 3. Überlegungen zu Anwendungen" auf Seite 47
- „Kapitel 4. Überlegungen zur Umgebung" auf Seite 103
- „Kapitel 5. Systemkatalogstatistiken" auf Seite 129.

Darüber hinaus können Sie auch die Informationen zum physischen Datenbankdesign im Handbuch *Systemverwaltung: Konzept* zu Rate ziehen.

Verwendung des Speichers durch DB2

Viele der für DB2 verfügbaren Konfigurationsparameter beeinflussen die Verwendung des Hauptspeichers auf dem System. Einige betreffen den Speicher auf dem Server, andere den Speicher auf dem Client und wieder andere sowohl den Speicher auf dem Server als auch den Speicher auf dem Client. Darüber hinaus wird Speicher zu verschiedenen Zeiten und aus verschiedenen Bereichen des Systems zugeordnet und wieder freigegeben.

Ein Systemadministrator sollte auf eine ausgewogene Gesamtverwendung von Speicher auf dem System Wert legen. Verschiedene Anwendungen, die unter dem Betriebssystem ausgeführt werden, können den Speicher auf unterschiedliche Arten verwenden. Zum Beispiel können einige Anwendungen den Cache

des Betriebssystems verwenden, während der Datenbankmanager über einen eigenen Pufferpool zum Zwischenspeichern von Daten verfügt und die Einrichtung des Betriebssystems nicht verwendet. Weitere Hinweise finden Sie in „Einstellen von Parametern mit Auswirkung auf die Speicherbelegung" auf Seite 290.

Abb. 22 zeigt, dass der Datenbankmanager verschiedene Arten von Speicher verwendet. Dabei ist anzunehmen, dass bei dieser Abbildung, die die Verwendung des Speichers darstellt, nicht die Enterprise – Extended Edition oder eine Umgebung mit mehreren logischen Knoten gezeigt wird. In der Enterprise – Extended Edition oder einer Umgebung mit mehreren logischen Knoten gibt es mehrere gemeinsam benutzte Speicher des Datenbankmanagers (einen pro Knoten).

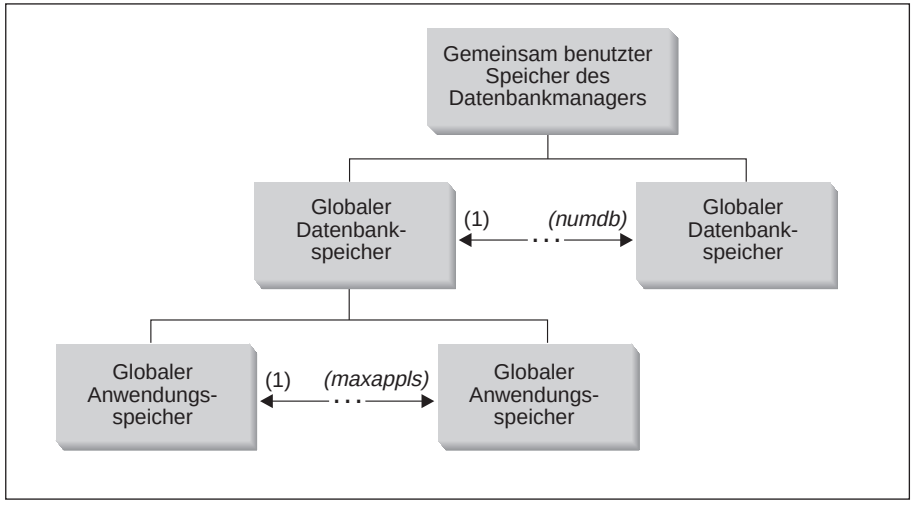


Abbildung 22. Arten von Speicher, die vom Datenbankmanager verwendet werden

Der Speicher wird für jedes Exemplar des Datenbankmanagers zu folgenden Zeitpunkten zugeordnet:

- Wenn der Datenbankmanager gestartet (`db2start`) wird, wird der Bereich „Gemeinsamer Speicher des Datenbankmanagers" zugeordnet, der zugeordnet bleibt, bis der Datenbankmanager wieder gestoppt (`db2stop`) wird. Dieser Bereich enthält Informationen, die vom Datenbankmanager benötigt werden, um die Aktivitäten für alle Datenbankverbindungen zu verwalten. Wenn die erste Anwendung die Verbindung zu einer Datenbank herstellt, werden sowohl globale als auch private Speicherbereiche zugeordnet.
- Beim ersten Aktivieren oder Herstellen einer Verbindung zu einer Datenbank wird der „globale Datenbankspeicher" zugeordnet. Der globale Datenbankspeicher wird für alle Anwendungen verwendet, die die Verbindung zur Datenbank herstellen, und enthält solche Speicherbereiche wie die

Pufferpools, die Sperrenliste, den Datenbankzwischenspeicher und den Zwischenspeicher für Dienstprogramme.

- Wenn eine Anwendung die Verbindung zu einer Datenbank herstellt, wird der „globale Anwendungsspeicher" zugeordnet (dies geschieht nur in Umgebungen mit partitionierten Datenbanken oder wenn der Konfigurationsparameter *intra_parallel* aktiviert ist). Dieser Speicher wird von Agenten verwendet, die im Auftrag der Anwendung für die gemeinsame Datennutzung sorgen und untereinander Aktivitäten koordinieren.
- (Nicht im obigen Diagramm dargestellt:) Wenn ein Agent zur Unterstützung einer bestimmten Anwendung zugeordnet wird (als Ergebnis einer CONNECT-Anforderung oder - in einer parallelen Umgebung - einer neuen SQL-Anforderung), wird „privater Agentenspeicher" für diesen Agenten zugeordnet. Der private Agentenspeicher wird jeweils für einen Agenten zugeordnet und enthält Speicherbereiche, die ausschließlich von diesen bestimmten Agenten verwendet werden, wie z. B. den Zwischenspeicher für Sortierlisten und den Anwendungszwischenspeicher.

  Wenn eine Datenbank bereits von einer Anwendung verwendet wird, werden für nachfolgende Anwendungen, die die Verbindung herstellen, nur privater Agentenspeicher und gemeinsamer globaler Anwendungsspeicher zugeordnet.

Abb. 22 auf Seite 284 zeigt, wie die Einstellungen von Konfigurationsparametern den Hauptspeicher beeinflussen. Insbesondere können die Parameter in der folgenden Liste die Menge an Speicher begrenzen, die für spezielle Zwecke zugeordnet wird. (In Umgebungen mit partitionierten Datenbanken ist dieser Speicher in jeder Datenbankpartition erforderlich.)

- Der Parameter *numdb* definiert die maximale Anzahl gleichzeitig aktiver Datenbanken (die von verschiedenen Anwendungen verwendet werden). Da jede Datenbank über ihren eigenen globalen Speicherbereich verfügt, wächst die Menge an Speicher, die potenziell zugeordnet wird, wenn der Wert dieses Parameters erhöht wird.
- Der Parameter *maxappls* definiert die maximale Anzahl von Anwendungen, die gleichzeitig mit einer einzigen Datenbank verbunden sein können. Dieser Parameter wirkt sich auf die Menge an Speicher aus, die potenziell für „privaten Agentenspeicher" und „globalen Agenten-/Anwendungsspeicher" für diese Datenbank zugeordnet wird. (Beachten Sie, dass dieser Parameter für jede Datenbank unterschiedlich eingestellt werden kann.)
- (Nicht im obigen Diagramm dargestellt:) Der Parameter *maxagents* (und *max_coordagents* bei Parallelverarbeitung) definiert die maximale Anzahl von Datenbankmanageragenten, die gleichzeitig für alle aktiven Datenbanken in einem Exemplar vorhanden sein können. Zusammen mit dem Parameter *maxappls* begrenzen diese Parameter die Menge an Speicher, der für „priva-

ten Agentenspeicher" und „globalen Anwendungsspeicher" zugeordnet wird. (Informationen zu Agenten finden Sie in „Datenbankagenten" auf Seite 320.)

Abb. 23 fasst zusammen, wie viel Speicher für die Unterstützung von Anwendungen verwendet wird. Die folgenden Konfigurationsparameter ermöglichen Ihnen, die Größe dieses Speichers zu steuern, indem Sie die Anzahl von "Speichersegmenten" (logischen Speicherbereichen) und ihre Größe begrenzen.

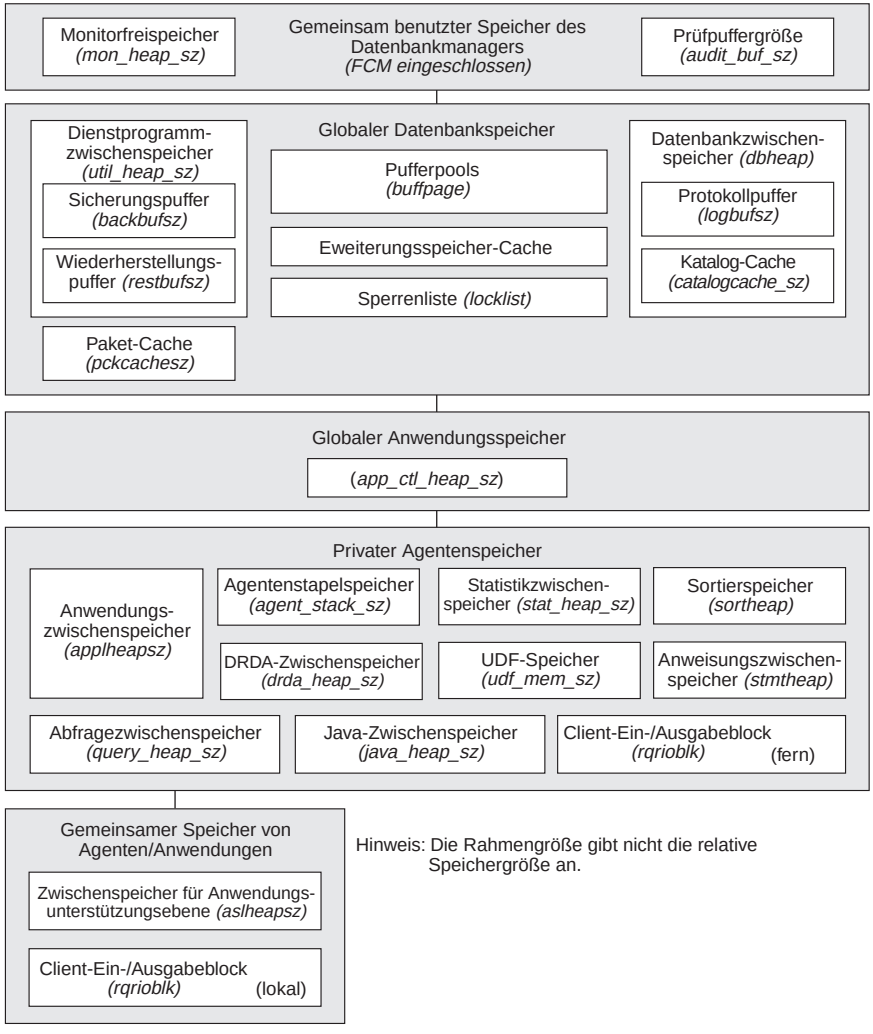


Abbildung 23. Verwendung von Speicher durch den Datenbankmanager

Gemeinsamer Speicher des Datenbankmanagers

Für die Ausführung des Datenbankmanagers ist Speicherbereich erforderlich. Dieser Speicherbereich kann, besonders in Umgebungen mit partitionsinterner und partitionsübergreifender Parallelität, sehr groß sein. Anhand der Informationen der folgenden Abschnitte können Sie die Größe dieses Speicherbereichs abschätzen und steuern:

- „Datenbankagenten" auf Seite 320. Agenten, die für Anwendungen aktiv sind, benötigen einen beträchtlichen Speicherbereich, besonders wenn der Wert des Parameters *maxagents* ungeeignet ist.
- „FCM-Anforderungen" auf Seite 291. Bei partitionierten Datenbanksystemen benötigt FCM (Fast Communications Manager) einen beträchtlichen Speicherbereich, besonders wenn der Wert des Parameters *fcm_num_buffers* ungeeignet ist.

  Der benötigte FCM-Speicher wird entweder aus dem FCM-Pufferpool oder aus dem gemeinsamen Speicher des Datenbankmanagers und dem FCM-Pufferpool gemeinsam zugeordnet, je nachdem, ob ein partitioniertes Datenbanksystem mehrere logische Knoten verwendet oder nicht. Einzelheiten finden Sie in der folgenden Beschreibung des FCM-Pufferpools.

FCM-Pufferpool

Wenn Sie ein partitioniertes Datenbanksystem haben, das nicht mehrere logische Knoten besitzt, werden der gemeinsame Speicher des Datenbankmanagers und der FCM-Pufferpool wie in Abb. 24 gezeigt eingesetzt.

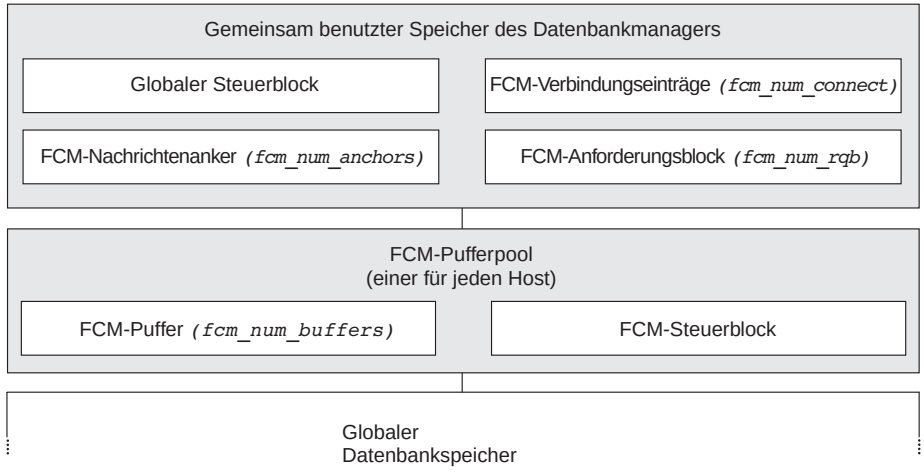


Abbildung 24. FCM-Pufferpool ohne Verwendung mehrerer logischer Knoten

Wenn Sie ein partitioniertes Datenbanksystem haben, das über mehrere logische Knoten verfügt, werden der gemeinsame Speicher des Datenbankmanagers und der FCM-Pufferpool wie in Abb. 25 gezeigt genutzt.

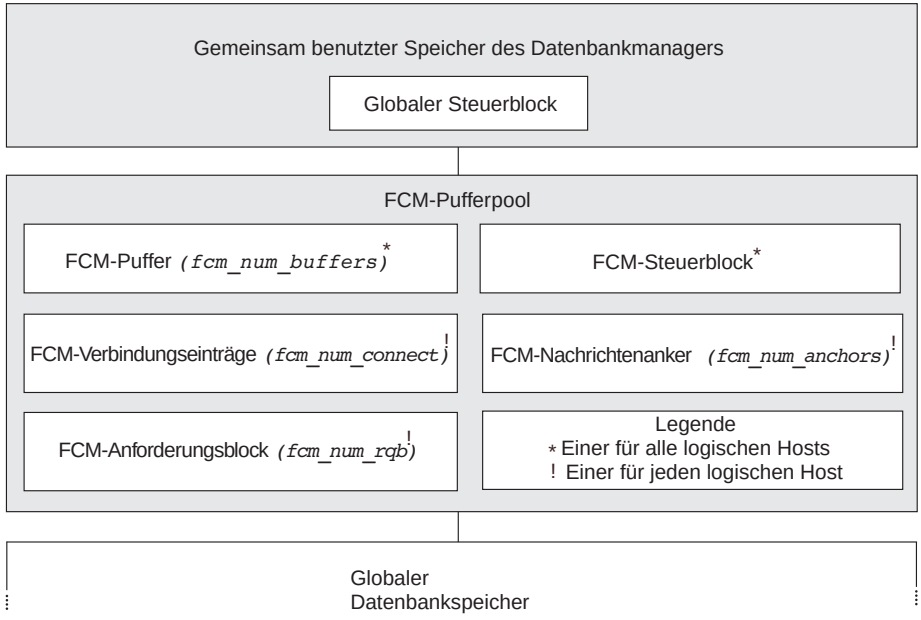


Abbildung 25. FCM-Pufferpool bei Verwendung mehrerer logischer Knoten

Globaler Datenbankspeicher

Der globale Datenbankspeicher wird durch folgende Konfigurationsparameter beeinflusst:

- Die Anzahl der Speichersegmente wird durch den Parameter *numdb* begrenzt (siehe „Maximale Anzahl gleichzeitig aktiver Datenbanken (numdb)" auf Seite 550).
- Die maximale Größe von Speichersegmenten wird von den Werten der folgenden Parameter bestimmt:
  - „Größe des Pufferpools (buffpage)" auf Seite 404 (wenn eine Pufferpoolgröße mit -1 angegeben wird) oder die bei der Erstellung oder Änderung der Pufferpools explizit angegebenen Größen
  - „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
  - „Zwischenspeicher für Datenbank (dbheap)" auf Seite 408
  - „Zwischenspeicher für Dienstprogramme (util_heap_sz)" auf Seite 412

- „Segmentgröße für erweiterten Speicher (estore_seg_sz)" auf Seite 460
- „Segmentanzahl für erweiterten Speicher (num_estore_segs)" auf Seite 460 .
- „Größe des Paket-Cache (pckcachesz)" auf Seite 418.

Globaler Anwendungsspeicher

Der globale Anwendungsspeicher wird durch den folgenden Konfigurationsparameter beeinflusst: „Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz)" auf Seite 420. Für parallele Systeme ist außerdem Speicherbereich für den Zwischenspeicher für Anwendungssteuerung erforderlich, der von den Agenten, die für dieselbe Anwendung in einer Datenbankpartition aktiv sind, gemeinsam benutzt wird. Der Zwischenspeicher wird zugeordnet, wenn der erste Agent, der eine Anforderung von der Anwendung empfangen soll, eine Verbindung anfordert. Der Agent kann entweder ein koordinierender Agent oder ein Subagent sein (siehe „Datenbankagenten" auf Seite 320).

Privater Agentenspeicher

- Die Anzahl von Speichersegmenten wird durch den niedrigeren der folgenden Werte begrenzt:
  - Der Gesamtwert von *maxappls* für alle aktiven Datenbanken (siehe „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461).
  - Der Wert des Parameters *maxagents* (siehe „Maximale Anzahl von Agenten (maxagents)" auf Seite 468).
- Die maximale Größe von Speichersegmenten wird von den Werten der folgenden Parameter bestimmt:
  - „Zwischenspeicher für Anwendungen (applheapsz)" auf Seite 426
  - „Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422
  - „SQL-Anweisungszwischenspeicher (stmtheap)" auf Seite 425
  - „Größe des Statistikzwischenspeichers (stat_heap_sz)" auf Seite 427
  - „Größe des Abfragezwischenspeichers (query_heap_sz)" auf Seite 428
  - „DRDA-Zwischenspeichergröße (drda_heap_sz)" auf Seite 429
  - „Größe des gemeinsamen UDF-Speichers (udf_mem_sz)" auf Seite 430
  - „Größe des Agentenstapelspeichers (agent_stack_sz)" auf Seite 432.

Gemeinsamer Agenten-/Anwendungsspeicher

- Die Gesamtanzahl der Segmente für gemeinsamen Agenten-/Anwendungsspeicher (für lokale Clients) wird durch den niedrigeren der folgenden Werte begrenzt:
  - Der Gesamtwert von *maxappls* für alle aktiven Datenbanken (siehe „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461).
  - Der Wert des Parameters *maxagents* (siehe „Maximale Anzahl von Agenten (maxagents)" auf Seite 468), oder (auf parallelen Systemen) *max_coordagents* (siehe „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470).
- Der gemeinsame Agenten-/Anwendungsspeicher wird außerdem durch folgende Konfigurationsparameter beeinflusst:
  - „Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)" auf Seite 436.
  - „E/A-Blockgröße für Clients (rqrioblk)" auf Seite 440.

Einstellen von Parametern mit Auswirkung auf die Speicherbelegung

Parameter, durch die die Zuordnung von Speicher gesteuert wird, sollten selbst auf Systemen mit maximalen Speicherkapazitäten *niemals* auf den höchstmöglichen Wert gesetzt werden, sofern ein solcher Wert nicht sorgfältig gerechtfertigt wird. Viele der Parameter können zulassen, dass der Datenbankmanager sehr leicht und schnell den gesamten auf einer Maschine verfügbaren Speicher belegt. Darüber hinaus kann die Verwaltung einer großen Speichermenge wesentlich mehr Zusatzaufwand auf Seiten des Datenbankmanagers verursachen, so dass der Gesamtaufwand noch einmal steigt.

Einige auf UNIX basierende Betriebssysteme ordnen Auslagerungsspeicher zu, wenn ein Prozess Speicher zuordnet, und nicht, wenn der Prozess selbst ausgelagert wird. In diesen Fällen sollten Sie sicherstellen, dass die Gesamtgröße des gemeinsamen Speichers durch eine äquivalente Menge an Auslagerungsspeicher unterstützt wird.

Bei der Mehrzahl der Konfigurationsparameter wird der entsprechende Speicher erst zugeordnet, wenn er angefordert wird. Diese Parameter spiegeln die maximale Größe eines speziellen Zwischenspeicherbereichs wider. Die besondere Ausnahme von dieser Regel bilden die folgenden Parameter, für die der Speicher gemäß den Parameterwerten vollständig zugeordnet wird:

- „Größe des Pufferpools (buffpage)" auf Seite 404 (wenn eine Pufferpoolgröße mit -1 angegeben wird) oder die bei der Erstellung oder Änderung der Pufferpools explizit angegebenen Größen
- „Schwellenwert für Sortierspeicher (sheapthres)" auf Seite 423
- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
- „Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)" auf Seite 436

- „Anzahl der FCM-Nachrichtenanker (fcm_num_anchors)" auf Seite 536
- „Anzahl der FCM-Puffer (fcm_num_buffers)" auf Seite 536
- „Anzahl der FCM-Verbindungseinträge (fcm_num_connect)" auf Seite 538
- „Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb)" auf Seite 539.

Der geeignete Wert dieser Arten von Parametern kann am besten mit Hilfe von Vergleichstests (Benchmarking) bestimmt werden, bei denen repräsentatives SQL und Extremfall-SQL für den Server ausgeführt werden und der Wert des Parameters solange geändert wird, bis der Punkt gefunden wird, an dem die Leistung wieder sinkt. Wenn die Leistung gegen die Parameterwerte in einem Diagramm aufgezeichnet würde, zeigt die Stelle, an dem die Kurve nicht weiter steigt bzw. wieder zu sinken beginnt, den Punkt an, an dem eine weitere Speicherzuordnung keinen weiteren Vorteil für die Anwendung bringt und daher nur zu einer Verschwendung von Speicher führen würde. (Siehe „Kapitel 12. Durchführen von Vergleichstests" auf Seite 371.)

Die oberen Grenzen der Speicherzuordnung einiger Parameter können über den Speicherkapazitäten der vorhandenen Hardware und des Betriebssystems liegen. Diese Grenzen wurden mit Blick auf zukünftige steigende Anforderungen gewählt.

Die gültigen Wertebereiche für die Parameter finden Sie in den Parameterbeschreibungen in „Kapitel 13. Konfigurieren von DB2" auf Seite 387.

FCM-Anforderungen

Beginnen Sie bei der Konfiguration der folgenden FCM-Konfigurationsparameter (Fast Communications Manager) mit den Standardwerten:
- „Anzahl der FCM-Puffer (fcm_num_buffers)" auf Seite 536
- „Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb)" auf Seite 539
- „Anzahl der FCM-Verbindungseinträge (fcm_num_connect)" auf Seite 538
- „Anzahl der FCM-Nachrichtenanker (fcm_num_anchors)" auf Seite 536.

Verwenden Sie zur Optimierung dieser Parameter den Datenbanksystemmonitor, um die untere Grenze für die freien Puffer, die freien Nachrichtenanker, die freien Verbindungseinträge und die freien Anforderungsblöcke zu überwachen. Wenn die untere Grenze (Low Water Mark) weniger als 10 Prozent des Werts des entsprechenden freien Datenelements beträgt, erhöhen Sie den Wert des entsprechenden Parameters. Informationen zum Datenbanksystemmonitor finden Sie in „Verwenden des Datenbanksystemmonitors" auf Seite 328.

Informationen zur Aktivierung der FCM-Kommunikation finden Sie im Handbuch *Systemverwaltung: Konzept*.

Verwalten des Datenbankpufferpools

Ein Pufferpool ist ein Speicherbereich, in den Datenbankseiten (mit Tabellenzeilen oder Indexeinträgen) temporär eingelesen und geändert werden. Zweck des Pufferpools ist die Verbesserung der Leistung des Datenbanksystems. Auf Daten im Hauptspeicher kann wesentlich schneller zugegriffen werden als auf Daten auf einer Platte. Daher ist die Leistung umso höher, je seltener der Datenbankmanager Daten von der Platte lesen bzw. auf Platte schreiben muss.

Die Konfiguration eines oder mehrerer Pufferpools ist der wichtigste Einzelbereich bei der Optimierung, da hier die meisten Datenoperationen, abgesehen von Operationen mit großen Objekten (LOB) und Langfeldern (LONG), für Anwendungen durchgeführt werden, die mit der Datenbank verbunden sind.

Wenn eine Anwendung auf eine Zeile einer Tabelle zum ersten Mal zugreift, liest der Datenbankmanager die Seite, die die Zeile enthält, in den Pufferpool ein. Wenn in der Folge eine Anwendung Daten anfordert, wird zuerst der Pufferpool nach den angeforderten Daten durchsucht. Werden die angeforderten Daten auf Seiten, die sich im Pufferpool befinden, gefunden, braucht der Datenbankmanager diese Daten nicht vom Plattenspeicher abzurufen. Durch Vermeidung von Datenabrufen vom Plattenspeicher wird eine schnellere Verarbeitungsleistung erzielt.

Der Speicherbereich für den Pufferpool wird zugeordnet, wenn eine Datenbank aktiviert wird oder wenn die erste Anwendung die Verbindung zur Datenbank herstellt. Anwendungen sind die Hauptnutznießer des Pufferpools. Wenn alle Anwendungen getrennt sind, wird der Speicherbereich für den Pufferpool wieder freigegeben.

Seiten verbleiben im Pufferpool, bis die Datenbank gestoppt wird oder der von einer Seite belegte Speicherbereich im Pufferpool für eine andere Seite benötigt wird. Die Auswahl des Bereichs, der im Pufferpool zum Einlesen einer anderen Seite ausgesucht wird, richtet sich nach Kriterien wie den folgenden:

- Der letzte Verweis auf eine Seite
- Die Wahrscheinlichkeit, mit der auf die Seite von dem letzten Agenten, der die Seite las, erneut zugegriffen wird
- Die Art der Daten auf der Seite
- Der Änderungsstatus der Seite, d. h. ob die Seite im Speicher geändert wurde, aber noch nicht auf der Festplatte gespeichert wurde oder nicht (Geänderte Seiten werden immer auf der Festplatte gespeichert, bevor sie im Pufferpool überschrieben werden.)

Anmerkung: Nachdem geänderte Seiten auf Platte geschrieben wurden, werden sie nicht aus dem Pufferpool entfernt, sofern nicht der Speicherbereich, den sie belegen, von anderen Seiten benötigt wird. Bis zu ihrer Überschreibung kann auf die Seiten immer wieder zugegriffen werden, wenn die Daten benötigt werden.

Beim Erstellen eines Pufferpools wird standardmäßig eine Seitengröße von 4 KB verwendet. Sie können die Seitengröße beim Erstellen des Pufferpools aber auf 4 KB, 8 KB, 16 KB oder 32 KB setzen. Wenn Pufferpools mit einer bestimmten Seitengröße erstellt werden, können ihnen nur Tabellenbereiche zugeordnet werden, die mit einer identischen Seitengröße erstellt wurden. Die Seitengröße des Pufferpools kann nach seiner Erstellung nicht mehr geändert werden. Stattdessen muss ein neuer Pufferpool mit der gewünschten Seitengröße erstellt werden.

Nutzung großer Speicher bei Windows-Systemen

Unter Windows 2000 werden Pufferpools bis zu einer Größe von 64 GB abzüglich des Umfangs von DB2 und des Betriebssystems unterstützt. (Dazu muss DB2 das primäre Produkt des Systems sein.) Diese Unterstützung wird über Microsoft Address Windowing Extensions (AWE) bereitgestellt.

AWE kann zwar für Pufferpools beliebiger Größe verwendet werden, wenn Sie AWE jedoch für größere Pufferpools benötigen, werden andere Windows-Produkte empfohlen. Windows 2000 Advanced Server unterstützt bis zu 8 GB Hauptspeicher. Windows 2000 Data Center Server unterstützt bis zu 64 GB Hauptspeicher.

Die Unterstützung von AWE-Puffpools erfordert eine entsprechende Konfigurierung von DB2 und Windows 2000. Der Pufferpool, der AWE nutzen soll, muss in der Datenbank vorhanden sein.

Verwenden Sie zur Zuordnung eines Benutzeradressbereichs von 3 GB die Windows 2000-Boot-Option /3GB. Hierauf kann eine größere AWE-Fenstergröße verwendet werden. Um den Zugriff auf mehr als 4 GB Hauptspeicher über die AWE-Hauptspeicherschnittstelle zu ermöglichen, verwenden Sie die Windows 2000-Boot-Option /PAE. Um sicherzustellen, dass Sie die richtige Boot-Option ausgewählt haben, wählen Sie unter *Steuerung* die Option *System* und dann *Autostart und Wiederherstellung* aus. Die Dropdown-Liste zeigt die verfügbaren Boot-Optionen. Wenn die gewünschte Boot-Option (/3GB oder /PAE) ausgewählt ist, können Sie mit dem nächsten Schritt zur Einrichtung der AWE-Unterstützung fortfahren. Wenn die gewünschte Option in der Liste fehlt, müssen Sie die Option der Datei boot.ini im Systemlaufwerk hinzufügen. Die Datei boot.ini enthält eine Liste der Aktionen, die beim Starten des Betriebssystems auszuführen sind.

Fügen Sie am Ende der Liste der verfügbaren Parameter die Option /3GB oder /PAE oder auch beide Optionen (durch Leerzeichen getrennt) ein. Nachdem Sie die geänderte Datei gesichert haben, können Sie die korrekte Boot-Option wie oben beschrieben kontrollieren und auswählen.

Außerdem muss in Windows 2000 dem Benutzer, für den DB2 installiert ist, das Recht „Seiten im Speicher sperren" zugeordnet werden. Melden Sie sich dazu bei Windows 2000 als der Benutzer an, der DB2 installiert hat. Wählen Sie dann im Menü *Start* den Ordner *Verwaltung* und dann das Programm *Lokale Sicherheitsrichtlinie* aus. Innerhalb der lokalen Richtlinien können Sie die Option zur Zuordnung des Benutzerrechts „Seiten im Speicher sperren" auswählen.

DB2 verlangt, dass die Registrierungsvariable DB2_AWE gesetzt wird. Damit Sie den richtigen Wert für diese Variable einstellen können, benötigen Sie die Pufferpool-ID des Pufferpools, für den Sie die AWE-Unterstützung aktivieren wollen. Die Pufferpool-ID können Sie der Spalte BUFFERPOOLID in der Systemkatalogsicht SYSCAT.BUFFERPOOLS entnehmen. Außerdem müssen Sie die Anzahl der zuzuordnenden physischen Seiten und Adressfensterseiten kennen. Die Anzahl der zuzuordnenden physischen Seiten sollte etwas unter der Gesamtzahl der verfügbaren physischen Seiten liegen. Welche Anzahl schließlich gewählt wird, hängt von Ihrer Arbeitsumgebung ab. Wenn z. B. auf Ihrem System nur DB2 und Datenbankanwendungen verwendet werden, können Sie als Wert für die Variable DB2_AWE einen Wert wählen, der zwischen einem halben und einem GB unter dem Gesamtumfang der physischen Seiten liegt. Wenn in Ihrer Umgebung auch andere Anwendungen das System nutzen, bei denen es sich nicht um Datenbankanwendungen handelt, müssen Sie einen höheren Wert vom Gesamtumfang abziehen, damit für diese anderen Anwendungen mehr physische Seiten bereitstehen. Die für die Datenbankregistriervariable DB2_AWE angegebene Anzahl entspricht der Anzahl der physischen Seiten, die zur Unterstützung von AWE und für DB2 verwendet werden sollen. Der Maximalwert für die Adressfensterseiten beträgt 1,5 GB bzw. 2,5 GB, wenn die Windows 2000-Boot-Option /3GB ausgewählt wurde.

Informationen zur Einstellung der DB2-Registrierungsvariablen DB2_AWE finden Sie in „Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen".

Funktionsweise von Pufferpool-Seiten

Seiten im Pufferpool können verschiedene Attribute haben:

- Seiten, die im Gebrauch sind, werden momentan gelesen oder aktualisiert. Sie können von anderen Agenten gelesen, aber nicht aktualisiert werden.
- „Benutzte" Seiten sind Seiten, auf denen Daten geändert, die aber noch nicht auf Platte geschrieben wurden. Wenn eine Seite auf Platte geschrieben wurde, wird sie als „sauber" betrachtet und verbleibt im Pufferpool. Der Bereich, der von sauberen Seiten belegt wird, kann für neue Seiten verwendet werden und ist für einen zugeordneten erweiterten Speicher-Cache (falls definiert) verfügbar.

Seiten können aus dem Pufferpool auf Platte geschrieben werden, wenn der Prozentsatz für den Bereich, der von geänderten Seiten im Pufferpool belegt wird, den Wert, der für den Konfigurationsparameter *chngpgs_thresh* angegeben ist, überschreitet. Außerdem müssen Sie eventuell die Datenbank so konfigurieren, dass sie über mehr als einen Agenten für eine Seitenlöschfunktion verfügt. Diese Agenten schreiben geänderte Seiten auf Platte, so dass die Datenbankagenten verwendbaren Speicherbereich im Pufferpool vorfinden.

Agenten von Seitenlöschfunktionen führen E/A-Operationen durch, die ansonsten von den Datenbankagenten durchgeführt werden müssen. Dadurch können die Anwendungen schneller ausgeführt werden, weil Transaktionen nicht warten müssen, während ihre Datenbankagenten Seiten auf Platte schreiben. (Agenten für Seitenlöschfunktionen können auch als *asynchrone Seitenlöschfunktionen* oder *asynchrone Funktionen zum Auslagern von Pufferseiten* bezeichnet werden, da sie zur selben Zeit wie die Datenbankagenten Operationen ausführen können.)

Zum Ändern der Anzahl der Agenten für Seitenlöschfunktionen wird der Konfigurationsparameter *num_iocleaners* verwendet (standardmäßig wird ein Agent für Seitenlöschfunktionen erstellt). Weitere Informationen hierzu finden Sie im Abschnitt „Anzahl asynchroner Seitenlöschfunktionen (num_iocleaners)" auf Seite 454. Legen Sie diesen Parameter auf einen Wert zwischen eins und der Zahl der physischen Platten in der Datenbank fest. Je größer diese Zahl ist, desto besser wird die Leistung bei der Ausführung von Auslastungen, die aufwendige Aktualisierungen erfordern. Dies trifft auch zu, wenn es eine große Zahl von Schreiboperationen für Daten oder Indexseiten in Bezug auf die Zahl der asynchronen Schreiboperationen für Daten oder Indexseiten gibt.

Das Schreiben von Seiten auf Platte ermöglicht eine schnellere Wiederherstellung der Datenbank im Falle eines Systemabsturzes, da der Datenbankmanager in der Lage ist, größere Bereiche des Pufferpools von der Platte ohne Zuhilfenahme der Datenbankprotokolldateien wiederherzustellen. Infolgedessen wird die Seitenlöschfunktion angefordert, wenn die Größe des Protokolls, das während der Wiederherstellung gelesen werden müsste, den folgenden Maximalwert überschreitet:

```
logfilsiz * softmax
```

Dabei gilt folgendes:

- *logfilsiz* ist die Größe der Protokolldateien (siehe „Protokolldateigröße (logfilsiz)" auf Seite 478).
- *softmax* ist der Prozentsatz der Protokolldateien, die nach einem Datenbankabsturz wiederherzustellen sind (siehe „Vor dem bedingten Prüfpunkt zu schreibende Protokollsätze (softmax)" auf Seite 487).

  Wenn der Parameter *softmax* beispielsweise den Wert 250 hat, enthalten 2,5 Protokolldateien die Änderungen, die wiederhergestellt werden müssen, wenn ein Systemabsturz auftritt.

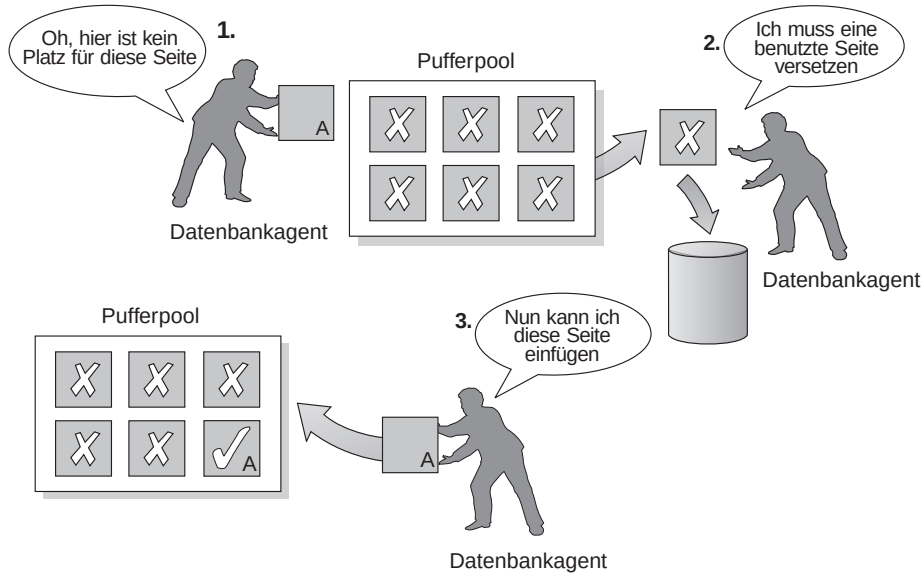
Mit dem Datenbanksystemmonitor können Sie die Anzahl von Anforderungen zum Seitenlöschen verfolgen, um die Zeit, die das Protokoll während einer Wiederherstellung gelesen wird, zu minimieren. Weitere Informationen finden Sie in der Beschreibung des Monitorelements *pool_lsn_gap_clns (für Pufferpool-protokollbereich ausgelöste Seitenlöschfunktionen)* im Handbuch *System Monitor Guide and Reference*.

Die Größe des Protokolls, das während der Wiederherstellung gelesen werden müsste, errechnet sich aus der Differenz der Positionen der folgenden Sätze im Protokoll:

- Der zuletzt geschriebene Protokollsatz
- Der Protokollsatz, der die älteste Änderung an Daten im Pufferpool beschreibt

Die folgende Abbildung veranschaulicht, wie die Arbeit zur Verwaltung des Pufferpools zwischen den Agenten für die Seitenlöschfunktionen und den Datenbankagenten aufgeteilt werden kann im Vergleich zu der Situation, in der die Datenbankagenten die gesamte Ein-/Ausgabe selbst durchführen.

Ohne Seitenlöschfunktionen



Mit Seitenlöschfunktionen

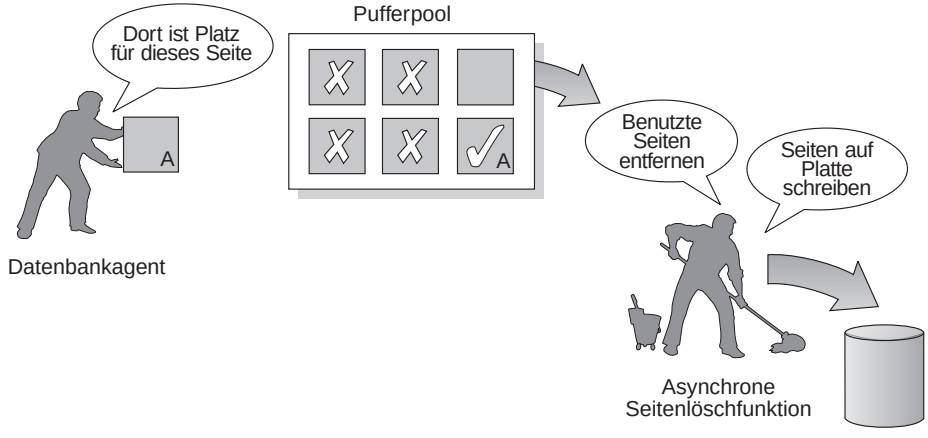


Abbildung 26. Asynchrone Seitenlöschfunktion. Benutzte Seiten werden auf Platte geschrieben.

Verwalten mehrerer Datenbankpufferpools

Für jede Datenbank ist mindestens ein Pufferpool erforderlich. Jedoch können je nach individuellen Anforderungen auch mehrere Pufferpools unterschiedlicher Größen für eine einzelne Datenbank erstellt werden. Mit den Anweisungen CREATE, ALTER und DROP BUFFERPOOL können Sie einen Pufferpool erstellen, ändern bzw. löschen. Mit den Anweisungen CREATE TABLESPACE und ALTER TABLESPACE können Sie angeben, welche Daten in einem Pufferpool zwischengespeichert werden.

Der Konfigurationsparameter *buffpage* definiert die Größe jedes Pufferpools, dessen Größe in der Katalogtabelle SYSCAT.BUFFERPOOLS mit dem Wert -1 angegeben ist. (Ansonsten wird dieser Parameter ignoriert.) Eine Pufferpoolgröße kann mit den DDL-Anweisungen ALTER BUFFERPOOL oder CREATE BUFFERPOOL definiert werden.

Eine neue Datenbank besitzt einen Standardpufferpool namens IBMDEFAULTBP mit einer Größe, die von der Plattform festgelegt wird. Wenn eine Datenbank einmal erstellt oder umgestellt ist, können andere Pufferpools für sie erstellt werden.

Bei der Arbeit an Ihrem Datenbankentwurf haben Sie möglicherweise festgestellt, dass Tabellen mit 8-KB-Seiten am besten geeignet waren. Darum sollten Sie einen Pufferpool mit 8-KB-Seiten (sowie einen oder mehrere Tabellenbereiche mit derselben Seitengröße) erstellen.

In einer Umgebung mit partitionierten Datenbanken hat jeder Pufferpool für eine Datenbank in allen Datenbankpartitionen dieselbe Standarddefinition (sofern dies nicht in der Anweisung CREATE BUFFERPOOL anders angegeben oder die Größe des Pufferpools für eine bestimmte Datenbankpartition mit der Anweisung ALTER BUFFERPOOL geändert wurde).

Wenn Sie einen Tabellenbereich mit 4-KB-Seiten erstellen und keinem bestimmten Pufferpool zuordnen, wird der Tabellenbereich dem Standardpufferpool zugeordnet. Wenn Sie einen Tabellenbereich mit einer Seitengröße über 4 KB (d. h. 8 KB, 16 KB oder 32 KB) erstellen, sollten Sie diesen Tabellenbereich einem Pufferpool mit derselben Seitengröße zuordnen. Wenn dieser Pufferpool momentan nicht aktiv ist, versucht DB2, den Tabellenbereich einem aktiven Pufferpool zuzuordnen, der eine identische Seitengröße besitzt (falls ein solcher Tabellenbereich verfügbar ist). Die Zuordnung (falls sie vorgenommen wird) ist temporär. Ist der ursprünglich angegebene Pufferpool beim nächsten Aktivieren der Datenbank aktiv, ordnet DB2 den Tabellenbereich diesem Pufferpool zu.

Mit der Anweisung ALTER TABLESPACE können Sie den Tabellenbereich keinem Pufferpool zuordnen, der eine andere Seitengröße verwendet.

Bei der Erstellung oder Änderung von Pufferpools muss der gesamte Speicherbereich, der von allen Pufferpools benötigt wird, dem Datenbankmanager zur Verfügung stehen, damit alle Pufferpools beim Starten der Datenbank zugeordnet werden können. Wenn dieser Speicherbereich beim Starten der Datenbank nicht verfügbar ist, versucht der Datenbankmanager, den Standardpufferpool (IBMDEFAULTBP) und jeweils einen von den mit einer anderen Seitengröße definierten Pufferpools, jedoch jeweils nur mit einer minimalen Größe von 16 Seiten zu starten. Die Größe dieses minimalen Pufferpools kann mit Hilfe der Registrierungsvariablen DB2_OVERRIDE_BPF überschrieben werden. Weitere Informationen zu dieser und anderen Registrierungs- und Umgebungsvariablen finden Sie in „Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen" auf Seite 571. Eine Warnung wird für jeden fehlgeschlagenen Versuch zurückgegeben, einen Pufferpool zu starten. Die Datenbank verbleibt in diesem Betriebszustand, bis ihre Konfiguration geändert wird und die Datenbank in vollem Umfang erneut gestartet werden kann.

Der Grund dafür, dass der Datenbankmanager mit Minimalwerten gestartet werden kann, ist, dass es ermöglicht werden soll, die Verbindung zur Datenbank herzustellen. Anschließend können Sie die Pufferpoolgrößen neu konfigurieren oder andere kritische Aufgaben mit dem Ziel ausführen, die Datenbank mit den richtigen Pufferpoolgrößen erneut zu starten. Es ist jedoch nicht empfehlenswert, die Datenbank über eine längere Zeit in diesem Status zu betreiben.

Anmerkung: Obwohl die Größe und die Attribute, die dem Standardpufferpool zugeordnet sind, geändert werden können, kann er nicht gelöscht werden. Außerdem gibt es eine Minimalgröße für jeden Pufferpool, der sich nach der verwendeten Plattform richtet.

Die Zuordnung großer Speichermengen für Pufferpools hat bestimmte Vorteile. Zum Beispiel dienen große Pufferpoolgrößen folgenden Zwecken:

- Sie ermöglichen, dass häufig angeforderte Datenseiten im Pufferpool behalten werden können. Dadurch kann schneller auf sie zugegriffen werden. Durch weniger E/A-Operationen können E/A-Konkurrenzsituationen besser vermieden werden, so dass die Antwortzeiten verbessert und die für E/A-Operationen benötigten Prozessorressourcen reduziert werden.
- Sie bieten die Möglichkeit, höhere Transaktionsgeschwindigkeiten mit derselben Antwortzeit zu erzielen.
- Sie vermeiden E/A-Konkurrenzsituationen für häufig verwendete Plattenspeichereinheiten zum Beispiel für Katalogtabellen, häufig verwendete Benutzertabellen und Indizes. Auch Sortierungen durch Abfragen profitieren von verminderten E/A-Konkurrenzsituationen auf Plattenspeichereinheiten, die die temporären Tabellenbereiche enthalten.

Auswählen eines oder mehrerer Pufferpools

Wenn eine der folgenden Bedingungen auf Ihr System zutrifft, sollten Sie nur einen einzigen Pufferpool verwenden:

- Der gesamte Pufferspeicherbereich beträgt weniger als 10 000 4-KB-Seiten.
- Es stehen keine Fachleute mit Anwendungskenntnissen für die spezialisierte Optimierung zur Verfügung.
- Sie arbeiten auf einem Testsystem.

Wenn auf Ihr System keine dieser Einschränkungen zutrifft, können Sie den Einsatz mehrerer Pufferpools zum Zweck der folgenden potenziellen Leistungsverbesserungen in Erwägung ziehen:

- Sie können temporäre Tabellenbereiche in einen separaten Pufferpool einlesen, um eine bessere Leistung für Abfragen zu erzielen, die temporären Speicherbereich benötigen, zum Beispiel für sortierintensive Abfragen.
- Wenn Sie Daten haben, auf die wiederholt und rasch durch viele kleine Transaktionsanwendungen zum Aktualisieren zugegriffen werden muss, sollten Sie überlegen, den Tabellenbereich mit diesen Daten in einen getrennten Pufferpool einzulesen. Wenn dieser Pufferpool eine geeignete Größe hat, ist die Wahrscheinlichkeit höher, dass die Seiten im Pufferpool gefunden werden. Dies trägt zur Verkürzung der Antwortzeiten und zur Reduzierung der Transaktionskosten bei.
- Sie können Daten in getrennten Pufferpools isolieren, um eine bevorzugte Verarbeitung bestimmter Anwendungen, Daten und Indizes zu erreichen. Zum Beispiel könnte es sinnvoll sein, Tabellen und Indizes, die häufig aktualisiert werden, in einen Pufferpool einzulesen, der von anderen Tabellen und Indizes, die zwar häufig abgefragt, aber nicht häufig aktualisiert werden, getrennt ist. Durch diese Änderung werden die Auswirkungen der häufigen Aktualisierungen (an der ersten Gruppe von Tabellen) auf die häufigen Abfragen (für die zweite Gruppe von Tabellen) verringert.
- Sie können kleinere Pufferpools für die Daten verwenden, auf die Anwendungen zugreifen, die sehr selten ausgeführt werden, besonders wenn eine solche Anwendung einen sehr wahlfreien Zugriff auf eine sehr umfangreiche Tabelle benötigt. In diesem Fall gibt es keinen Bedarf, die Daten länger als für eine einzige Abfrage im Pufferpoolspeicher zu behalten. Es ist günstiger, einen kleinen Pufferpool für diese Daten zu verwenden und zusätzlichen Speicher für andere Zwecke freizugeben (z. B. für andere Pufferpools).
- Nach der Trennung der verschiedenen Aktivitäten und Daten in verschiedene Pufferpools können gute und relativ unaufwendige Daten zur Leistungsdiagnose aus den Statistiken und aus Funktionen zur Ablaufverfolgung von Benutzeraktivitäten gewonnen werden.

Vorablesen von Daten in den Pufferpool

Durch das Vorablesen von Index- und Datenseiten in den Pufferpool kann die Leistung verbessert werden, da die Zeit, die auf die Ausführung der Ein-/Ausgabeoperationen gewartet werden muss, verringert wird. *Vorablesen* bedeutet, dass eine oder mehrere Seiten von der Platte abgerufen werden, wenn ihre Verwendung wahrscheinlich erscheint. Es gibt zwei Arten des Vorablesezugriffs:

- Der *sequenzielle Vorablesezugriff* ist eine Methode, mit der aufeinander folgende Seiten in den Pufferpool gelesen werden, bevor die Seiten von der Anwendung angefordert werden. (Siehe „Sequenzieller Vorablesezugriff".)
- Der *Vorablesezugriff über Listen* oder sequenzielle Vorablesezugriff über Listen ist eine effiziente Methode, auf Daten zuzugreifen, auch wenn die benötigten Datenseiten nicht in aufeinander folgender Reihenfolge vorliegen. (Siehe „Vorablesezugriff über Listen" auf Seite 303.)

Diese beiden Methoden zum Lesen von Datenseiten erfolgen zusätzlich zum normalen Lesen. Normale Lesevorgänge werden verwendet, wenn nur eine oder wenige aufeinander folgende Seiten abgerufen werden. Bei einem normalen Lesevorgang wird jeweils nur eine Seite von Daten übertragen.

Weitere Informationen zur Aktivierung des Vorablesezugriffs finden Sie in „Konfigurieren von E/A-Servern für Vorablesezugriff und parallele E/A" auf Seite 304.

Sequenzieller Vorablesezugriff

Durch das Lesen mehrerer aufeinander folgender Seiten in den Pufferpool in einer einzigen E/A-Operation kann der Systemaufwand, der mit der Ausführung der Anwendung verbunden ist, wesentlich reduziert werden. Darüber hinaus kann durch das Ausführen mehrerer E/A-Operationen parallel zum Einlesen mehrerer Seitenbereiche in den Pufferpool die Zeit verringert werden, die Anwendungen auf die Beendigung von E/A-Operationen warten müssen.

Der Vorablesezugriff wird gestartet, wenn der Datenbankmanager bestimmt, dass sequenzielle E/A-Operationen zweckmäßig sind und dass durch den Vorablesezugriff die Leistung verbessert werden könnte. In solchen Fällen wie Tabellensuchen und Sortierungen in Tabellen kann der Datenbankmanager leicht feststellen, dass durch den sequenziellen Vorablesezugriff die E/A-Leistung verbessert wird. In diesen Fällen startet der Datenbankmanager den sequenziellen Vorablesezugriff automatisch. Das folgende Beispiel zeigt eine Abfrage, die eine Tabellensuche erforderlich macht und für die deshalb der sequenzielle Vorablesezugriff die nahe liegende Methode wäre:

```
SELECT NAME FROM EMPLOYEE
```

Die Anzahl der Seiten, die vom Datenbankmanager vorab gelesen werden, kann für jeden Tabellenbereich mit Hilfe der Klausel PREFETCHSIZE in der Anweisung CREATE TABLESPACE bzw. ALTER TABLESPACE definiert werden. Der definierte Wert wird in der Spalte PREFETCHSIZE der Systemkatalogtabelle SYSCAT.TABLESPACES gespeichert.

Es empfiehlt sich, den Wert für PREFETCHSIZE als ein Vielfaches des Werts für EXTENTSIZE für Ihren Tabellenbereich und die Anzahl der zugehörigen Tabellenbereichsbehälter explizit zu definieren. (Der Wert für EXTENTSIZE definiert die Anzahl von Seiten, die der Datenbankmanager in einen Behälter schreibt, bevor er zu einem anderen Behälter wechselt. Siehe „Entwerfen und Auswählen von Tabellenbereichen" im Handbuch *Systemverwaltung: Konzept*.) Ist zum Beispiel für EXTENTSIZE ein Wert von 16 Seiten festgelegt und hat der Tabellenbereich zwei Behälter, könnte der Wert für die Menge der vorab gelesenen Datenseiten (PREFETCHSIZE) auf den Wert 32 gesetzt werden.

Der Datenbankmanager überwacht die Verwendung des Pufferpools, um sicherzustellen, dass durch den Vorablesezugriff auf Daten keine Seiten aus dem Pufferpool entfernt werden, wenn diese Seiten noch von einer anderen Arbeitseinheit verwendet werden. Zur Vermeidung von Problemen begrenzt der Datenbankmanager die Anzahl der vorab gelesenen Seiten möglicherweise auf einen Wert, der kleiner ist als die für den Tabellenbereich angegebene Menge.

Die Einstellung für PREFETCHSIZE kann erhebliche Auswirkungen auf die Leistung besonders bei Suchoperationen in großen Tabellen haben. Mit Unterstützung durch den Datenbanksystemmonitor und andere Systemüberwachungsprogramme kann die Einstellung von PREFETCHSIZE für die Tabellenbereiche optimiert werden. Zum Beispiel können folgende Arten von Informationen gesammelt werden:

- Mit Überwachungsprogrammen für Ihr Betriebssystem können Sie feststellen, ob E/A-Wartezeiten für die Abfrage auftreten.
- Mit Hilfe des Datenelements *pool_async_data_reads (asynchrone Leseoperationen für Pufferpooldaten)*, das vom Datenbanksystemmonitor bereitgestellt wird, können Sie feststellen, ob Vorablesezugriffe stattfinden. Weitere Informationen finden Sie im Handbuch *System Monitor Guide and Reference*.

Wenn E/A-Wartezeiten auftreten und für die Abfrage der Vorablesezugriff aktiv ist, können Sie versuchen, durch Erhöhen des Werts für PREFETCHSIZE eine Leistungssteigerung zu erzielen. Es ist möglich, dass die E/A-Wartezeiten nicht durch den Vorablesezugriff verursacht werden. In diesem Fall wird durch Erhöhen des Werts für PREFETCHSIZE keine Verbesserung für die Abfrage erreicht.

Bei allen Arten des Vorablesezugriffs können mehrere E/A-Operationen parallel ausgeführt werden, wenn für PREFETCHSIZE ein Vielfaches des Werts für EXTENTSIZE für den Tabellenbereich angegeben ist und sich die durch EXTENTSIZE definierten Bereiche des Tabellenbereichs in separaten Behältern befinden. Für eine optimale Leistung sollten die Behälter so konfiguriert werden, dass sie verschiedene physische Einheiten verwenden. Weitere Informationen zum parallelen Vorablesezugriff finden Sie in „Konfigurieren von E/A-Servern für Vorablesezugriff und parallele E/A" auf Seite 304.

Sequenzerkennung

Es gibt Fälle, in denen es nicht von vornherein offensichtlich ist, ob durch einen sequenziellen Vorablesezugriff eine Leistungsverbesserung erreicht werden kann. In diesen Fällen kann der Datenbankmanager die E/A-Operationen überwachen und, wenn sequenzielles Lesen von Seiten auftritt, den Vorablesezugriff aktivieren. Der Vorablesezugriff kann in diesem Fall vom Datenbankmanager aktiviert und inaktiviert werden, je nachdem ob er zweckmäßig erscheint oder nicht. Diese Art des sequenziellen Vorablesens ist die *Sequenzerkennung*, die für Index- und Datenseiten angewandt wird. Mit Hilfe des Konfigurationsparameters *seqdetect* (siehe „Markierung für Sequenzerkennung (seqdetect)" auf Seite 457) können Sie steuern, ob der Datenbankmanager die Sequenzerkennung durchführen soll. Wenn die Sequenzerkennung aktiviert ist, könnte mit ihrer Hilfe festgestellt werden, dass für die folgende SQL-Anweisung ein sequenzieller Vorablesezugriff von Vorteil wäre:

```
SELECT NAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

In diesem Beispiel hat das Optimierungsprogramm vielleicht entschieden, die Tabelle mit Hilfe eines Index für die Spalte EMPNO zu durchsuchen. Wenn die Tabelle eine hohe Clusterbildung in Bezug auf diesen Index aufweist, dann sind die Leseoperationen für die Datenseiten beinahe sequenziell, so dass ein Vorablesezugriff zu einer Leistungsverbesserung führen würde. In diesem Fall würde ein Vorablesezugriff auf die Datenseiten durchgeführt.

In diesem Beispiel könnte auch ein Vorablesezugriff auf Indexseiten auftreten. Wenn eine große Anzahl von Indexseiten durchsucht werden muss und der Datenbankmanager feststellt, dass ein Lesen sequenzieller Seiten der Indexseiten auftritt, wird ein Vorablesezugriff auf die Indexseiten durchgeführt.

Vorablesezugriff über Listen

Der *Vorablesezugriff über Listen* oder sequenzielle Vorablesezugriff über Listen ist eine effiziente Methode, auf Daten zuzugreifen, auch wenn die benötigten Datenseiten nicht in aufeinander folgender Reihenfolge vorliegen. Der Vorablesezugriff über Listen kann in Verbindung mit dem Zugriff über einen oder mehrere Indizes verwendet werden.

Vorablesezugriff und partitionsinterne Parallelität

Das Vorablesen ist ein sehr wesentlicher Gesichtspunkt im Hinblick auf die Leistung der partitionsinternen Parallelität, bei der mehrere Subagenten beim Durchsuchen eines Index oder einer Tabelle verwendet werden. Diese parallelen Suchoperationen können zu größerem Datenverarbeitungsdurchsatz führen, wodurch höhere Vorablesegeschwindigkeiten erforderlich werden.

Der Nachteil durch ungeeignetes Vorablesen ist bei parallelen Suchoperationen höher als bei seriellen Suchoperationen. Wenn bei der Ausführung einer seriellen Suchoperation kein Vorablesezugriff stattfindet, arbeit die Abfrage langsamer, weil der Agent immer auf E/A-Operationen warten muss. Wenn bei der Ausführung einer parallelen Suchoperation kein Vorablesezugriff stattfindet, müssen eventuell alle Subagenten auf einen Subagenten warten, der auf eine E/A-Operation wartet.

Aufgrund seiner Bedeutung wird der Vorablesezugriff bei partitionsinterner Parallelität intensiver durchgeführt. Die Funktion zur Sequenzerkennung toleriert größere Lücken zwischen benachbarten Seiten, so dass die Seiten als sequenziell betrachtet werden können. Die Breite dieser Lücken erhöht sich mit der Anzahl der an der Suchoperation beteiligten Subagenten.

Konfigurieren von E/A-Servern für Vorablesezugriff und parallele E/A

Zur Aktivierung des Vorablesezugriffs startet der Datenbankmanager separate Steuer-Threads, die als *E/A-Server* bezeichnet werden, um die Seitenleseoperationen durchzuführen. Infolgedessen gliedert sich die Verarbeitung einer Abfrage in zwei parallele Aktivitäten: Datenverarbeitung (CPU) und E/A-Operationen für Datenseiten. Die E/A-Server warten auf Vorableseanforderungen aus der CPU-Verarbeitungsaktivität. Diese Vorableseanforderungen enthalten eine Beschreibung der benötigten E/A-Operationen, um den erwarteten Datenbedarf zu befriedigen. Der Grund für den Vorablesezugriff bestimmt, wann und wie der Datenbankmanager Vorableseanforderungen generiert. (Weitere Informationen enthalten die Abschnitte „Sequenzieller Vorablesezugriff" auf Seite 301 und „Vorablesezugriff über Listen" auf Seite 303.)

Die folgende Abbildung veranschaulicht, wie E/A-Server zum Vorablesen von Daten in den Pufferpool verwendet werden.

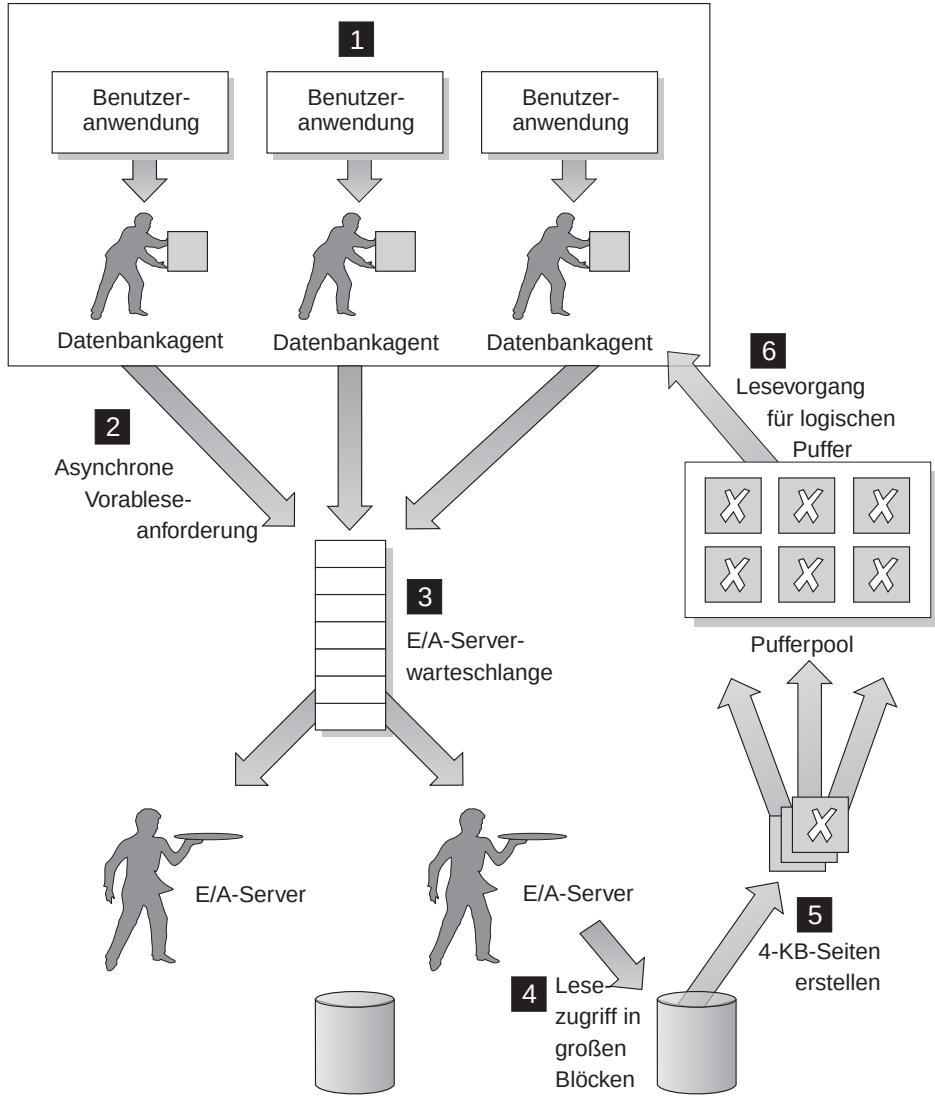


Abbildung 27. Vorablesen von Daten mit Hilfe von E/A-Servern

Die folgenden Schritte werden in Abb. 27 gezeigt:

1 Die Benutzeranwendung übergibt die SQL-Anforderung an den Datenbankagenten, der der Benutzeranwendung vom Datenbankmanager zugeordnet wurde.

2, **3**
Der Datenbankagent stellt fest, dass ein Vorablesezugriff erfolgen soll,

um die angeforderten Daten abzurufen und so die SQL-Anforderung zu erfüllen, und schreibt eine Vorableseanforderung an die E/A-Server-Warteschlange.

4 , **5**

Der erste verfügbare E/A-Server liest die Vorableseanforderung aus der Warteschlange und liest die Daten aus dem Tabellenbereich in den Pufferpool. In Abhängigkeit von der Anzahl der Vorableseanforderungen in der Warteschlange und der Anzahl der E/A-Server, die mit dem Konfigurationsparameter *num_ioservers* konfiguriert ist, können mehrere E/A-Server gleichzeitig Daten aus dem Tabellenbereich abrufen.

6 Der Datenbankagent führt die erforderlichen Aktionen an den Datenseiten im Pufferpool durch, um das Ergebnis der SQL-Anforderung an die Benutzeranwendung zurückzuliefern.

Durch Konfigurieren einer ausreichenden Anzahl von E/A-Servern mit Hilfe des Konfigurationsparameters *num_ioservers* kann die Leistung von Abfragen, für die der Vorablesezugriff auf Daten verwendet werden kann, erheblich gesteigert werden. Einige über den Bedarf hinaus konfigurierte E/A-Server beeinträchtigen die Leistung nicht, da überschüssige E/A-Server nicht verwendet werden und ihre Speicherseiten ausgelagert werden. Jeder E/A-Server-Prozess hat eine Nummer. Der Datenbankmanager verwendet immer den verfügbaren Prozess mit der niedrigsten Nummer, so dass einige der Prozesse mit höheren Nummern eventuell nie zum Einsatz kommen.

Bei der Festlegung, wie viele E/A-Server geeigneterweise konfiguriert werden sollten, sind folgende Punkte zu beachten:

- Der Umfang der gleichzeitig anfallenden Aktivitäten für eine Datenbank. Das heißt, die Anzahl der Datenbankagenten, die Vorableseanforderungen an die E/A-Server-Warteschlange zu einem gegebenen Zeitpunkt schreiben könnten.
- Der höchste Grad, bis zu dem die E/A-Server parallel arbeiten können. Weitere Informationen finden Sie in „Aktivieren paralleler E/A".

Legen Sie den Wert des Parameters *num_ioservers* mindestens auf die Zahl der physischen Platten in der Datenbank fest, um die Möglichkeit für parallele E/A-Operationen zu maximieren.

Aktivieren paralleler E/A

In Umgebungen, in denen mehrere Behälter für einen Tabellenbereich vorhanden sind, kann der Datenbankmanager die *parallele Ein-/Ausgabe* initialisieren. Parallele E/A bezieht sich auf die Fähigkeit des Datenbankmanagers, mehrere E/A-Server zur Verarbeitung der E/A-Anforderungen einer einzelnen Abfrage zu verwenden. Jedem E/A-Server werden die E/A-Operationen für einen anderen Behälter zugeordnet, so dass mehrere Behälter parallel gelesen wer-

den können. Die parallele Durchführung der E/A-Operationen kann zu bedeutenden Verbesserungen beim E/A-Durchsatz führen.

Obwohl für jeden Behälter ein getrennter E/A-Server die E/A-Operationen durchführt, ist die tatsächliche Anzahl der E/A-Server, die parallele E/A-Operationen durchführen können, auf die Anzahl der physischen Einheiten begrenzt, über die die angeforderten Daten verteilt sind. Dies bedeutet gleichzeitig, dass Sie so viele E/A-Server wie die Anzahl der physischen Einheiten benötigen.

Wie die parallele E/A initialisiert und verwendet wird, hängt von dem Grund für die Durchführung der parallelen E/A ab:

- **Sequenzieller Vorablesezugriff**

  Beim sequenziellen Vorablesezugriff wird die parallele E/A initialisiert, wenn der Wert für PREFETCHSIZE (Menge der vorab gelesenen Daten) ein Vielfaches des Werts für EXTENTSIZE eines Tabellenbereichs ist. Jede Vorableseanforderung wird dann in mehrere kleinere Anforderungen aufgeteilt, die sich an den Grenzen der durch EXTENTSIZE definierten Bereiche orientieren. Diese kleineren Anforderungen werden dann verschiedenen E/A-Servern zugeordnet.

- **Vorablesezugriff über Listen**

  Beim Vorablesezugriff über Listen wird jede Liste von Seiten in Abhängigkeit von den Behältern, in denen die Datenseiten gespeichert sind, in kleinere Listen unterteilt. Diese kleineren Listen werden dann verschiedenen E/A-Servern zugeordnet.

- **Sicherung und Wiederherstellung von Datenbanken oder Tabellenbereichen**

  Für Sicherungen oder Wiederherstellungen von Daten ist die Anzahl paralleler E/A-Anforderungen gleich der Größe des Sicherungspuffers dividiert durch den Wert von EXTENTSIZE. Der Maximalwert ist gleich der Anzahl von Behältern.

- **Wiederherstellung einer Datenbank oder eines Tabellenbereichs**

  Für die Wiederherstellung von Daten werden parallele E/A-Anforderungen in gleicher Weise wie beim sequenziellen Vorablesezugriff initialisiert und aufgeteilt. Die Daten werden nicht im Pufferpool wiederhergestellt, sondern direkt aus dem Wiederherstellungspuffer auf die Platte versetzt.

- **LOAD**

  Beim Laden von Daten können Sie den Grad der E/A-Parallelität mit der Option DISK_PARALLELISM des Befehls LOAD angeben. (Wenn diese Option nicht angegeben wird, wird ein Standardwert basierend auf der kumulativen Anzahl von Tabellenbereichsbehältern für alle Tabellenbereiche, die der Tabelle zugeordnet sind, verwendet.)

Für eine optimale Leistung bei paralleler E/A sollten Sie folgende Voraussetzungen erfüllen:

- Es ist eine ausreichende Anzahl E/A-Server vorhanden. Die Anzahl der E/A-Server sollte geringfügig höher als die Anzahl der Behälter sein, die für alle Tabellenbereiche innerhalb der Datenbank verwendet werden, konfiguriert werden.
- Die Werte für EXTENTSIZE und PREFETCHSIZE sind für den Tabellenbereich angemessen. Der Wert für PREFETCHSIZE sollte nicht zu groß sein, um eine exzessive Belastung des Pufferpools zu vermeiden. (Eine ideale Größe ist ein Vielfaches des Werts für EXTENTSIZE und der Anzahl der Tabellenbereichsbehälter.) Der Wert für EXTENTSIZE sollte relativ klein sein, wobei sich ein Wert zwischen 8 und 32 Seiten empfiehlt.
- Die Behälter sind so konfiguriert, dass sie sich auf separaten physischen Laufwerken befinden.
- Alle Behälter haben dieselbe Größe, um einen konsistenten Grad an Parallelität zu gewährleisten.

  Wenn ein oder mehrere Behälter kleiner als die anderen sind, verringern sie das Potenzial für das optimierte parallele Vorablesen. Zum Beispiel:

  – Wenn ein kleinerer Behälter vollständig gefüllt ist, werden weitere Daten in den übrigen Behältern gespeichert, wodurch sich eine ungleichmäßige Auslastung der Behälter ergibt. Ungleichmäßig ausgelastete Behälter beeinträchtigen die Leistung des parallelen Vorablesens, da die Anzahl von Behältern, aus denen Daten vorab gelesen werden können, eventuell kleiner ist als die Gesamtanzahl von Behältern.
  – Wenn ein kleinerer Behälter zu einem späteren Zeitpunkt hinzugefügt wird und die Daten neu verteilt werden, enthält der kleinere Behälter weniger Daten als die anderen Behälter. Diese im Verhältnis zu den anderen Behältern kleine Menge von Daten führt nicht zu einer Optimierung des parallelen Vorablesens.
  – Wenn ein Behälter größer ist und alle anderen Behälter vollständig gefüllt werden, wird dieser Behälter zum einzigen Behälter, in dem weitere Daten gespeichert werden. Beim Zugriff auf diese weiteren Daten ist der Datenbankmanager nicht in der Lage, einen parallelen Vorablesezugriff durchzuführen.
- Es ist eine angemessene E/A-Kapazität vorhanden, wenn partitionsinterne Parallelität verwendet wird. Partitionsinterne Parallelität kann auf SMP-Maschinen verwendet werden, um die abgelaufene Zeit für eine Abfrage zu verkürzen, indem die Abfrage auf mehreren Prozessoren ausgeführt wird. Es ist eine ausreichende E/A-Kapazität erforderlich, um jeden Prozessor auszulasten. Dabei sind in der Regel zusätzliche physische Laufwerke erforderlich, um diese E/A-Kapazität bereitzustellen.

  Der Vorablesezugriff muss in höheren Raten durchgeführt werden, um die E/A-Kapazität effektiv zu nutzen. Der Wert für PREFETCHSIZE sollte

höher sein, um das Vorablesen in höheren Raten zu ermöglichen. Der Wert für PREFETCHSIZE sollte ein Vielfaches des Werts für EXTENTSIZE und der Anzahl der Tabellenbereichsbehälter sein. Im Idealfall sollten die Behälter so konfiguriert werden, dass sie sich auf getrennten physischen Laufwerken befinden.

Die Anzahl der erforderlichen physischen Laufwerke kann von der Geschwindigkeit und der Kapazität der Laufwerke und des E/A-Busses sowie von der Geschwindigkeit der Prozessoren abhängig sein.

Gleichzeitiges Zuordnen mehrerer Seiten

SMS-Tabellenbereiche werden bei Bedarf erweitert. Diese Erweiterung erfolgt standardmäßig um jeweils eine Seite gleichzeitig. Sie können aber bei bestimmten Auslastungen (z. B. bei umfangreichen Einfügeoperationen (Bulk Insert)) die Leistung erhöhen, indem Sie das Programm *db2empfa* verwenden, um DB2 anzuweisen, die Tabellenbereiche jeweils um Gruppen von Seiten oder Gruppen durch EXTENTSIZE definierter Bereiche zu erweitern. Das Programm *db2empfa* befindet sich im Unterverzeichnis *bin* des Verzeichnisses *sqllib*. Durch Ausführen des Programms wird der Konfigurationsparameter *multipage_alloc* der Datenbank auf den Wert "Yes" gesetzt. Weitere Informationen zu diesem Programm finden Sie im Handbuch *Command Reference*.

Eine andere Möglichkeit zur optimalen Verwendung des verfügbaren Speichers wird im Abschnitt „Erweitern von Speicher" auf Seite 330 behandelt.

Sortieren

Sortieren ist für eine Abfrage häufig erforderlich, wobei die richtige Konfiguration der Sortierzwischenspeicherbereiche für die Leistung der Abfrage eine wichtige Rolle spielen kann. Sortieren ist in folgenden Fällen erforderlich:

- Es gibt keinen Index, um eine angeforderte Reihenfolge (z. B. durch eine Anweisung SELECT mit einer Klausel ORDER BY) der Daten herzustellen.
- Es gibt einen Index, aber Sortieren ist effizienter als der Zugriff über den Index.
- Es wird ein Index erstellt (und der Konfigurationsparameter *indexsort* ist aktiviert).

Verschiedene Arten der Sortierung

Ein Sortiervorgang umfasst zwei Phasen:

1. Die Sortierphase
2. Die Rückgabe der Ergebnisse der Sortierphase

Die Art, wie der Sortiervorgang innerhalb dieser beiden Phasen verarbeitet wird, führt zu verschiedenen Kategorien oder Arten von Sortierung, mit deren Hilfe sich der Sortiervorgang beschreiben lässt. Innerhalb der Sortierphase kann der Sortiervorgang in die Kategorien „mit Überlauf" oder „ohne Über-

lauf" eingeordnet werden. Bei der Rückgabe der Ergebnisse der Sortierphase lässt sich der Sortiervorgang als „über eine Pipe geleitet" oder „nicht über eine Pipe geleitet" klassifizieren.

Mit Überlauf und ohne Überlauf
Wenn die Daten, die sortiert werden, nicht vollständig in den Sortierzwischenspeicher (ein Speicherblock, der jedes Mal zugeordnet wird, wenn eine Sortierung durchgeführt wird) passen, laufen die Daten in temporäre Datenbanktabellen über. Sortierungen, die keinen Überlauf verursachen, zeigen stets eine bessere Leistung als solche, bei denen ein Überlauf auftritt.

Über Pipe geleitet und nicht über Pipe geleitet
Wenn sortierte Daten direkt zurückgegeben werden können, ohne dass eine temporäre Tabelle zum Speichern der endgültigen sortierten Liste von Daten erforderlich ist, wird dies als ein „über Pipe geleiteter Sortiervorgang" bezeichnet. Wenn die sortierten Daten in einer temporären Tabelle zurückgegeben werden müssen, wird dies als „nicht über Pipe geleiteter Sortiervorgang" bezeichnet. Ein über Pipe geleiteter Sortiervorgang ist immer effizienter als ein nicht über Pipe geleiteter Sortiervorgang.

Optimieren der Parameter mit Auswirkung auf Sortierungen

Die folgenden Elemente wirken sich auf die Sortierleistung aus:

- Die Einstellungen für die folgenden Konfigurationsparameter:

  „Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422
  Mit diesem Parameter wird die Speichermenge definiert, die für Sortiervorgänge verwendet wird.

  „Schwellenwert für Sortierspeicher (sheapthres)" auf Seite 423
  Mit diesem Parameter wird die Gesamtspeichermenge gesteuert, die für alle Sortierungen im gesamten Exemplar zur Verfügung steht.
- Anweisungen, die eine große Menge Sortierungen verursachen
- Fehlende Indizes, die zur Vermeidung unnötiger Sortiervorgänge dienen könnten
- Anwendungslogik, die das Sortieraufkommen nicht minimiert
- Paralleles Sortieren, das die Leistung der Sortierungen erhöht, aber nur auftreten kann, wenn die Anwendung partitionsinterne Parallelität verwendet (siehe „Aktivieren paralleler E/A" auf Seite 306).

Anzeichen für Probleme bei der Sortierleistung

Ein Vergleich der gesamten CPU-Zeit für die Sortierung mit der Zeit, die für die Ausführung der gesamten Anwendung benötigt wurde, kann entscheidende Hinweise darauf geben, ob ein allgemeines Problem in Bezug auf das Sortieren vorliegt. Der Datenbanksystemmonitor kann Hilfestellung geben (siehe „Verwenden des Datenbanksystemmonitors" auf Seite 328). Insbeson-

dere zeigt der Performance Monitor (der aus „Snapshot Monitor" und „ Event Monitor" (Ereignismonitor) besteht und über die Steuerzentrale verfügbar ist), standardmäßig neben den Zeiten für Ein-/Ausgaben (*I/O*) und Wartezeiten auf Sperren (*lock wait*) auch die Gesamtsortierzeit (*total sort time* an.

Wenn die Gesamtsortierzeit gegenüber den anderen Zeiten relativ groß ist, achten Sie auf folgende Werte, die ebenfalls standardmäßig angezeigt werden:

Prozentsatz von Sortierungen mit Überlauf
Diese Variable (auf der Anzeige mit den Leistungsdaten von Snapshot Monitor) zeigt den Prozentsatz von Sortierungen, bei denen ein Überlauf auftrat (Percentage of overflowed sorts). Wenn der Prozentsatz für Sortierungen mit Überlauf hoch ist, erhöhen Sie den Wert für den Konfigurationsparameter *sortheap* und/oder *sheapthres*, wenn es Sortiervorgänge nach Überschreiten des Schwellenwerts gab. (Mit Hilfe von Snapshot Monitor können Sie feststellen, ob es Sortiervorgänge nach Überschreiten des Schwellenwerts gab.)

Sortierungen nach Überschreiten des Schwellenwerts
Wenn Sortierungen nach Überschreiten des Schwellenwerts (Post Threshold Sorts) häufig sind, erhöhen Sie den Wert für den Parameter *sheapthres* und/oder verringern den Wert für den Parameter *sortheap*.

Im allgemeinen sollten Sie den für das gesamte Exemplar verfügbaren Sortierspeicher (*sheapthres*) so groß wie möglich definieren, ohne übermäßiges Seitenauslagern zu verursachen. Es ist möglich, dass eine Sortierung vollständig im Sortierspeicher durchgeführt wird. Wenn dies jedoch dazu führt, dass das Betriebssystem übermäßig viele Auslagerungen von Seiten vornehmen muss, um diesen Sortierspeicher unterzubringen, können Sie den Vorteil eines großen Sortierspeichers einbüßen. Bei einer Anpassung der Konfigurationsparameter für die Sortierungen sollten Sie einen Betriebssystemmonitor verwenden, um alle Änderungen bei der Auslagerung zu verfolgen.

Anmerkung: Nachdem die DB2-Binärsortierung mit Teilschlüssel verbessert wurde, so dass auch nicht ganzzahlige Datentypschlüssel unterstützt werden, ist beim Sortieren langer Schlüssel zusätzlicher Speicherbereich erforderlich. Wenn Sie annehmen, dass lange Schlüssel verwendet werden, erhöhen Sie den Wert des Konfigurationsparameters *sortheap*.

Beachten Sie außerdem, dass bei einer über Pipe geleiteten Sortierung der Sortierzwischenspeicher nicht freigegeben wird, bevor die Anwendung den dieser Sortierung zugeordneten Cursor schließt. Auf diese Weise kann eine über Pipe geleitete Sortierung Speicher belegen, bis der Cursor geschlossen wird.

Techniken zur Optimierung der Sortierleistung

Sie können den Datenbanksystemmonitor und Vergleichstesttechniken bei der Einstellung der Konfigurationsparameter *sortheap* und *sheapthres* zur Unterstützung heranziehen. Gehen Sie für jeden Datenbankmanager und den zugehörigen Datenbanken folgendermaßen vor:

- Erstellen Sie eine repräsentative Auslastung und führen Sie sie aus.
- Sammeln Sie für jede betroffene Datenbank Durchschnittswerte für die folgenden Leistungsvariablen über den Auslastungszeitraum der Vergleichstests:
  - Gesamter verwendeter Sortierspeicher
  - Aktive Sortiervorgänge

  Diese Leistungsvariablen werden in der Detailsicht der Leistung von Snapshot Monitor angezeigt.
- Setzen Sie den Parameter *sortheap* auf den Durchschnittswert für den *gesamten verwendeten Sortierspeicher* für jede Datenbank.
- Führen Sie Folgendes aus, um den Wert für *sheapthres* festzulegen:
  1. Stellen Sie fest, welche Datenbank in dem Exemplar über den größten Wert für *sortheap* verfügt.
  2. Ermitteln Sie die durchschnittliche Größe des Sortierspeichers für diese Datenbank.

     Wenn die Ermittlung des Durchschnittswerts zu aufwendig ist, verwenden Sie als Wert 80% des maximalen Sortierspeichers.
  3. Setzen Sie den Wert für *sheapthres* auf die Durchschnittsanzahl der aktiven Sortiervorgänge multipliziert mit der oben berechneten Durchschnittsgröße des Sortierspeichers.

     Dies ist die empfohlene Anfangseinstellung. Anschließend können Sie mit Hilfe von Vergleichstests diesen Wert optimieren.

Sie können darüber hinaus bestimmte Anwendungen und Anweisungen identifizieren, bei denen die Sortierung ein wesentliches Leistungsproblem darstellt:

- Richten Sie Ereignismonitore (Event Monitors) auf Anwendungs- und Anweisungsebene ein, um Unterstützung bei der Identifikation von Anwendungen mit der längsten Gesamtsortierzeit zu erhalten.
- Ermitteln Sie innerhalb dieser Anwendungen die Anweisungen mit der längsten *Gesamtsortierzeit*.
- Optimieren Sie diese Anweisungen mit Hilfe eines Programms wie Visual Explain.
- Stellen Sie sicher, dass geeignete Indizes vorhanden sind. Sie können mit Hilfe von Visual Explain alle Sortieroperationen für eine bestimmte

Anweisung ermitteln. Stellen Sie anschließend fest, ob ein geeigneter Index für jede Tabelle vorhanden ist, auf die von dieser Anweisung zugegriffen wird.

Anmerkung: Sie können die EXPLAIN-Tabellen durchsuchen, um herauszufinden, welche Abfragen mit Sortieroperationen verbunden sind. (Siehe „Anhang C. EXPLAIN-Programme (SQL)" auf Seite 645.)

Reorganisieren von Katalogen und Benutzertabellen

Die Leistung von SQL-Anweisungen, die Indizes verwenden, kann beeinträchtigt werden, nachdem zahlreiche Daten aktualisiert, gelöscht oder eingefügt wurden. Im allgemeinen können neu eingefügte Zeilen nicht in der gleichen physischen Reihenfolge angeordnet werden wie die logische Reihenfolge, die durch den Index definiert ist (es sei denn, Sie verwenden geclusterte Indizes). Das heißt, dass der Datenbankmanager zusätzliche Leseoperationen zum Zugriff auf die Daten durchführen muss, da sich logisch sequenzielle Daten auf verschiedenen physischen Datenseiten befinden können, die nicht sequenziell sind.

In der Regel nimmt die Reorganisation einer Tabelle mehr Zeit in Anspruch als die Ausführung des Dienstprogramms RUNSTATS zur Erhebung der Statistiken. Die Leistung kann vielleicht schon ausreichend verbessert werden, wenn aktuelle Statistiken für die enthaltenen Daten erstellt und die Anwendungen erneut gebunden werden. Daher sollte dies zuerst versucht werden. Wenn sich dadurch keine Leistungsverbesserung ergibt, sind die Daten in den Tabellen und Indizes wahrscheinlich nicht effizient angeordnet, so dass eine Reorganisation Abhilfe schaffen könnte. Die Informationen dieses Abschnitts beziehen sich nicht nur auf die Reorganisation Ihrer eigenen Datentabellen, sondern auch auf die Reorganisation der Systemkatalogtabellen, für die diese Maßnahme ebenfalls erforderlich werden kann.

Bei typisierten Tabellen muss der angegebene Tabellenname der Name der Stammtabelle der Hierarchie sein.

Der Befehl REORGCHK liefert Informationen über die physischen Merkmale einer Tabelle und gibt Anhaltspunkte, ob eine Reorganisation der Tabelle vorteilhaft wäre. Dieser Befehl kann über den Befehlszeilenprozessor ausgeführt werden. Im Handbuch *Command Reference* finden Sie weitere Informationen, einschließlich Hinweisen, wie die Ausgabe des Befehls zu interpretieren ist.

Anmerkung: Der Befehl REORGCHK zeigt keine Daten für erweiterte Indizes oder deklarierte temporäre Tabellen.

Das Dienstprogramm REORG ordnet wahlweise Daten anhand eines angegebenen Index in einer physischen Reihenfolge neu an. REORG verfügt über

eine Option, mit der die Reihenfolge von Zeilen in einer Tabelle mit Hilfe eines Index angegeben werden kann. Dadurch werden die Tabellendaten gemäß dem Index in Clustern angeordnet, so dass sich für die statistischen Daten der Spalte CLUSTERRATIO (Clusterverhältnis) bzw. CLUSTERFACTOR (Clusterfaktor), die vom Dienstprogramm RUNSTATS gesammelt werden, bessere Werte ergeben. Infolgedessen können SQL-Anweisungen, die Zeilen in der Reihenfolge des Index anfordern, effektiver verarbeitet werden. Durch das Dienstprogramm REORG werden die Tabellen außerdem kompakter gespeichert, indem nicht benötigter, leerer Speicherbereich wieder freigegeben wird (wenn Sie PCTFREE in ALTER TABLE angegeben haben, bleibt dieser Speicherbereich jedoch ungenutzt).

Verwenden Sie die Befehle REORG und REORGCHK nicht mit Kurznamen.

Das Dienstprogramm REORG setzt voraus, dass alle anderen Anwendungen, die normalerweise mit den betroffenen Tabellendaten und Indizes arbeiten, offline sind. Vielleicht ist es in Ihrer Arbeitsumgebung wünschenswert, den Zeitraum, den die Anwendungen nicht mit den Daten arbeiten können, möglichst begrenzt zu halten. In diesem Fall kann die Verwendung des Dienstprogramms zur Online-Indexreorganisation in Betracht kommen.

Der Protokollspeicherbereich, der für die Neuerstellung des Index bei der Indexreorganisation benötigt wird, wird wie folgt berechnet:

```
2 * (10500 + ((Anzahl der Indexseiten / Größe des Speicherbereichs) * 110) +
(Anzahl der Indexseiten * 45) +
(Anzahl der Indexseiten / 16000) * 64 ))
```

Mit den verschiedenen Teilen der Berechnungen sollen die unterschiedlichen Aufwandsarten im Zusammenhang mit der Erstellung und der Protokollaufzeichnung im Rahmen der Indexneuerstellung bestimmt werden.

Bei der Entscheidung, wann die Tabellendaten reorganisiert werden sollten, sind folgende Faktoren zu beachten:

- Der Umfang der INSERT-, UPDATE- und DELETE-Aktivitäten
- Jede wesentliche Änderung an der Leistung von Abfragen, die einen Index mit hohem Clusterverhältnis verwenden
- Die Ausführung des Dienstprogramms RUNSTATS verbessert die Leistung von Abfragen nicht.
- Der Befehl REORGCHK zeigt den Bedarf einer Reorganisation der Tabelle an.
- Der Aufwand für die Reorganisation der Tabelle, einschließlich der CPU-Zeit, der abgelaufenen Zeit und des eingeschränkten gemeinsamen Zugriffs aufgrund der Tatsache, dass das Dienstprogramm REORG die Tabelle bis zum Abschluss der Reorganisation sperrt

Zur Ausführung des Dienstprogramms REORG sind die Berechtigungen SYSADM, SYSMAINT, SYSCTRL oder DBADM bzw. das Zugriffsrecht CONTROL für die Tabelle erforderlich.

Das Dienstprogramm REORG verwendet temporäre Tabellen, die bedeutend größer sein können als die ursprüngliche Tabelle, wenn Spalten der Tabelle hinzugefügt wurden oder eine Tabelle LOB-Spalten enthält. Wenn diese temporären Tabellen größer sind, wird die vom Dienstprogramm REORG neu erstellte permanente Tabelle ebenfalls größer.

Anmerkung: Sie können das Dienstprogramm REORG nicht zum Reorganisieren von deklarierten temporären Tabellen verwenden.

Das Dienstprogramm ermöglicht die Angabe eines temporären Tabellenbereichs, der zur Erstellung der temporären REORG-Tabellen verwendet wird. Wenn kein temporärer Tabellenbereich angegeben wird, erstellt das Dienstprogramm die temporären REORG-Tabellen in dem Tabellenbereich, der die zu reorganisierende Tabelle enthält. Beachten Sie folgende Richtlinien bei der Entscheidung, ob ein temporärer Tabellenbereich zu verwenden ist:

- Wenn Sie einen temporären Tabellenbereich angeben, empfiehlt es sich im Allgemeinen, einen temporären, vom System verwalteten Tabellenbereich (SMS-Tabellenbereich) anzugeben. Die Verwendung eines temporären DMS-Tabellenbereichs empfiehlt sich nicht, da mit dieser Art von Tabellenbereich immer nur eine REORG-Verarbeitung gleichzeitig ausgeführt werden kann.
- Es wird im Allgemeinen empfohlen, einen temporären vom System verwalteten Tabellenbereich (SMS-Tabellenbereich) anzugeben. Bei Verwendung desselben Tabellenbereichs für die Reorganisation von Tabellen wird zwar weniger Zeit benötigt, doch ist der Protokollierungsaufwand größer und es muss genügend Speicherplatz für die reorganisierte Tabelle verfügbar sein. Wenn Sie einen temporären Tabellenbereich angeben, empfiehlt es sich im Allgemeinen, einen temporären vom System verwalteten Tabellenbereich (SMS-Tabellenbereich) anzugeben. Die Verwendung eines temporären DMS-Tabellenbereichs empfiehlt sich nicht, da mit dieser Art von Tabellenbereich immer nur eine REORG-Verarbeitung gleichzeitig ausgeführt werden kann.

Das Dienstprogramm REORG schließt implizit alle geöffneten Cursor.

Achten Sie darauf, dass Sie möglicherweise eine Tabelle in einem Tabellenbereich reorganisieren, der Seiten mit einer Größe über 4 KB (d. h. 8 KB, 16 KB oder 32 KB) verwendet. Während der Reorganisation muss der verwendete temporäre Tabellenbereich Seiten derselben Größe wie der Basistabellenbereich verwenden.

Wenn das Dienstprogramm REORG nicht erfolgreich beendet wird, dürfen die temporären Dateien, Tabellen oder Tabellenbereiche **nicht** gelöscht werden.

Mit Hilfe dieser Dateien und Tabellen kann der Datenbankmanager die Änderungen des Dienstprogramms REORG rückgängig machen oder die Reorganisation abschließen, je nachdem wie weit die Reorganisation vor dem Auftreten des Fehlers vorangeschritten war.

In einer partitionierten Datenbank organisiert das Dienstprogramm REORG Daten in jeder Partition neu. Wenn das Dienstprogramm in einer Partition fehlschlägt, werden nur die REORG-Änderungen in dieser Partition rückgängig gemacht. Wenn Sie einen Verzeichnispfad zum Speichern der temporären Tabellen angeben, wird dieser Pfad vom Datenbankmanager auf jeder Datenbankpartition erweitert. Daher werden die temporären Dateien, wenn Sie einen Pfad angeben, der von anderen Datenbankpartitionen mit benutzt wird, in verschiedenen (durch den Knotennamen identifizierten) Unterverzeichnissen unter diesem Pfad gespeichert.

Online-Indexreorganisation

Eine Online-Reorganisation ist dadurch möglich, dass ein benutzerdefinierbarer Schwellenwert für die Maximalgröße des freien Speicherbereichs auf einer Indexseite (Blattseite) zur Verfügung gestellt wird. Wenn ein Indexschlüssel aus einer Blattseite gelöscht und dabei der Schwellenwert überschritten wird, werden die benachbarten Indexseiten daraufhin überprüft, ob zwei Blattseiten zusammengefügt werden können. Wenn auf einer Seite ausreichend Platz zum Zusammenfügen zweier benachbarter Seiten vorhanden ist, erfolgt die Zusammenfügung, ohne dass die Datenbank offline genommen werden muss.

Diese Online-Reorganisation des Index ist nur möglich für Indizes, die in Version 6 und den nachfolgenden Releases erstellt werden. Vorhandene Indizes, für die die Möglichkeit der Online-Reorganisation in dieser Weise erforderlich ist, müssen gelöscht und anschließend erneut erstellt werden, um die nötigen internen Änderungen an den Indexseiten durchzuführen. Die Online-Indexreorganisation für einen bestimmten Index wird durch die Angabe eines Werts `MINPCTUSED` bei der Erstellung des Index aktiviert. Der Wert für `MINPCTUSED` sollte auf weniger als einhundert (100) gesetzt werden. Dieser Wert stellt die Reorganisationsschwelle dar und wird als Prozentsatz des auf einer Indexseite belegten Speicherbereichs angegeben, der der letzte akzeptable Wert ist, bevor eine Zusammenfügung der Indexseite mit der Nachbarseite versucht wird. Der empfohlene Wert für `MINPCTUSED` ist einer, der unter 50 % liegt, da der Zweck darin besteht, zwei benachbarte Indexseiten zusammenzufügen. Durch den Wert 0 für `MINPCTUSED`, der gleichzeitig der Standardwert ist, wird die Online-Reorganisation inaktiviert.

Indexseiten, die nach einer Online-Indexreorganisation zur Verwendung frei geworden sind, stehen zur Wiederverwendung zur Verfügung. Allerdings sind diese freien Seiten nur für andere Indizes in derselben Tabelle verfügbar. Eine vollständige Reorganisation der Tabelle gibt Seiten für andere Objekte frei, wenn mit einem DMS-Speichermodell gearbeitet wird. Bei Verwendung eines SMS-Speichermodells hingegen wird der entsprechende Plattenspeicherplatz freigegeben.

Nichtblattseiten von Indizes werden infolge einer Online-Indexreorganisation nicht freigegeben. Aber durch eine vollständige Reorganisation der Tabelle wird die Größe des Index minimiert. Die Blattseiten und Nichtblattseiten werden ebenso wie die Indexstufen zahlenmäßig reduziert.

Verringern der Notwendigkeit, Tabellen zu reorganisieren

Damit die Reorganisation einer Tabelle weniger häufig erforderlich wird, führen Sie nach dem Erstellen der Tabelle folgende Schritte aus:

- Ändern Sie die Tabelle (ALTER), um PCTFREE hinzuzufügen.
- Erstellen Sie einen Clusterungsindex mit PCTFREE für Index.
- Sortieren Sie die Daten.
- Laden Sie die Daten.

Nach Ausführung dieser Schritte für eine vorhandene Tabelle verfügen Sie über eine Tabelle mit Clusterungsindex. Der Clusterungsindex sorgt in Verbindung mit PCTFREE für die Tabelle dafür, dass die ursprünglich sortierte Reihenfolge erhalten bleibt. Bei ausreichend Platz auf den Seiten können neue Daten auf den richtigen Seiten eingefügt und so die Clusterungsmerkmale des Clusterungsindex erhalten bleiben. Wenn mit wachsender Menge eingefügter Daten die Seiten der Tabelle voll werden, werden Datensätze an das Ende der Tabelle angehängt und die Tabelle verliert allmählich ihre Clusterung.

Es ist empfehlenswert, eine REORG-Operation oder eine Sortierung und LOAD-Operation nach der Erstellung eines Clusterungsindex durchzuführen. Ein Clusterungsindex versucht, eine bestimmte Reihenfolge der Daten zu erhalten, um so die statistischen Werte für CLUSTERRATIO bzw. CLUSTERFACTOR zu verbessern, die vom Dienstprogramm RUNSTATS gesammelt werden.

Die Größe des Speicherbereichs, der bei einer REORG-Operation auf jeder Seite freizulassen ist, wird durch den Wert für PCTFREE der Tabelle bestimmt. Wenn dieser Wert nicht definiert ist, füllt REORG die Seiten während der Reorganisation mit Daten auf.

Überlegungen zur Leistung bei DMS-Einheiten

Wenn Sie von der Datenbank verwaltete Speichereinheitenbehälter (DMS-Einheitenbehälter) für Ihre Tabellenbereiche verwenden, müssen Sie mit folgenden Punkten vertraut sein, um Ihre Umgebung effektiv verwalten zu können:

- **Zwischenspeichern von Dateisystemen im Cache**

  Das Zwischenspeichern von Dateisystemen wird wie folgt ausgeführt:
  - Seiten aus DMS-Dateibehältern (und aus allen SMS-Behältern) können vom Betriebssystem im Dateisystem-Cache zwischengespeichert werden.
  - Seiten aus den Tabellenbereichen von DMS-Einheitenbehältern werden vom Betriebssystem nicht im Dateisystem-Cache zwischengespeichert.

  Anmerkung: Wenn Sie unter Windows NT arbeiten, definiert die Registrierungsvariable DB2NTNOCACHE, ob DB2 Datenbankdateien mit einer Option NOCACHE öffnet. Wenn DB2NTNOCACHE=ON definiert ist, wird das Zwischenspeichern im System-Cache inaktiviert. Wenn DB2NTNOCACHE=OFF definiert ist, speichert das Betriebssystem DB2-Dateien im Cache. Dies gilt für alle Daten außer für Dateien, die Langfelddaten oder LOB-Daten enthalten. Durch Inaktivieren der Cache-Funktion des Betriebssystems steht mehr Speicher für die Datenbank zur Verfügung, damit der Pufferpool oder der Sortierspeicher vergrößert werden kann.

- **Puffern von Daten**

  Von der Platte gelesene Tabellendaten sind in der Regel im Pufferpool der Datenbank verfügbar (siehe „Verwalten des Datenbankpufferpools" auf Seite 292). In einigen Fällen kann es geschehen, dass eine Datenseite aus dem Pufferpool entfernt wird, bevor sie von der Anwendung verwendet wurde. (Dies kann zum Beispiel geschehen, wenn Pufferpoolbereich für andere Datenseiten angefordert wird.) Informationen zu SMS-Tabellenbereichen und DMS-Dateibehältern finden Sie in der obigen Beschreibung zum Zwischenspeichern von Dateisystemen. Dadurch können E/A-Operationen vermieden werden, die sonst erforderlich gewesen wären.

  Tabellenbereiche, die DMS-Einheitenbehälter verwenden, verwenden das Dateisystem oder den Cache des Dateisystems **nicht**. Daher könnte es sinnvoll sein, die Größe des Pufferpools der Datenbank zu erhöhen und die Größe des Cache des Dateisystems zu verringern, um den Umstand auszugleichen, dass bei DMS-Tabellenbereichen, die Einheitenbehälter verwenden, keine doppelte Pufferung durchgeführt wird.

Wenn Sie anhand von Monitor-Tools auf Systemebene feststellen, dass das Aufkommen an E/A-Operationen für einen DMS-Tabellenbereich im Vergleich zu einem äquivalenten SMS-Tabellenbereich höher liegt, kann sich dieser Unterschied durch die oben beschriebene doppelte Pufferung erklären.

- **Verwenden von LOB- oder LONG-Daten**

  Wenn eine Anwendung LOB- oder LONG-Daten abruft, verwendet der Datenbankmanager keine Puffer, um die Daten zwischenzuspeichern. Jedesmal, wenn eine Anwendung eine dieser Seiten benötigt, muss der Datenbankmanager sie von der Platte abrufen.

  Wenn LOB- oder LONG-Daten jedoch in SMS- oder DMS-Dateibehältern gespeichert werden, kann die Pufferung durch den Dateisystem-Cache erfolgen und dadurch eine bessere Leistung erreicht werden.

  Da die Systemkataloge einige LOB-Spalten enthalten, ist es empfehlenswert, sie in einem SMS-Tabellenbereich (bzw., alternativ dazu, in einem DMS-Dateibehälter) zu speichern.

Verwalten des Initialisierungsaufwands

Durch den Befehl ACTIVATE DATABASE werden ausgewählte Datenbanken gestartet. Bei der Verwendung dieses Befehls in einer partitionierten Datenbank wird versucht, die ausgewählte partitionierte Datenbank in allen Partitionen zu aktivieren. Mit Hilfe dieses Befehls kann vermieden werden, dass Anwendungszeit zur Initialisierung oder zum Starten einer Datenbank verbraucht wird. Datenbanken, die Sie mit dem Befehl ACTIVATE DATABASE initialisiert haben, müssen mit dem Befehl DEACTIVATE DATABASE wieder inaktiviert werden. Die Datenbank wird nicht dadurch inaktiviert, dass die letzte Anwendung ihre Verbindung zu ihr trennt. Weitere Informationen zu den Befehlen ACTIVATE und DEACTIVATE finden Sie im Handbuch *Command Reference*.

Wenn eine Datenbank noch nicht gestartet wurde und eine Anweisung CONNECT TO (bzw. eine implizite Anweisung CONNECT) in einer Anwendung festgestellt wird, dann muss diese Anwendung warten, bis der Datenbankmanager die erforderliche Datenbank gestartet hat, bevor mit dieser Datenbank gearbeitet werden kann. Dies ist ein Startaufwand, der von der ersten Anwendung, die auf eine bestimmte Datenbank zugreift, aufgefangen werden muss. In einer partitionierten Datenbank entsteht dieser Startaufwand in jeder Datenbankpartition. Wenn die Datenbank gestartet ist, können alle anderen Anwendungen die Verbindung zu dieser Datenbank herstellen und sie verwenden, ohne dass der Zeitaufwand zum Starten der Datenbank erneut entsteht.

Datenbankagenten

DB2-Server müssen die Kommunikation zwischen dem Datenbankmanager und Client-Anwendungen sowie lokalen Anwendungen unterstützen. Auf UNIX basierende Umgebungen verwenden eine mit *Prozessen* arbeitende Architektur. Zum Beispiel werden die kommunikationsbereiten DB2-Agenten (Listeners) als Prozesse erstellt. Intel-Betriebssysteme wie OS/2 und Windows NT verwenden eine Architektur, die auf *Threads* basiert. Zum Beispiel werden die kommunikationsbereiten DB2-Agenten innerhalb des Systemsteuerprozesses des DB2-Servers als Threads erstellt. Für jede Datenbank, auf die zugegriffen wird, werden verschiedene Prozesse/Threads gestartet, um die verschiedenen Datenbankoperationen (z. B. Vorablesen, Übertragung und Protokollieren) durchzuführen.

Eine der wichtigsten Arten von Prozessen/Threads sind die Datenbankagenten, die die Operationen von Anwendungen mit Datenbanken vollziehen.

Ein *logischer Agent* stellt eine mit dem Datenbankmanager verbundene Anwendung dar. Der logische Agent verfügt über alle Informationen und Steuerblöcke, die von einer Anwendung benötigt werden. Die maximale Anzahl logischer Agenten wird durch den Konfigurationsparameter *max_logicagents* des Datenbankmanagers gesteuert. Da alle Anwendungen über einen logischen Agenten verfügen, steuert dieser Parameter die maximale Anzahl von Anwendungen, von denen aus eine Verbindung zum Exemplar hergestellt werden kann.

Ein *Verarbeitungsagent* führt Anwendungsanforderungen aus, ist aber nicht dauerhaft mit einer bestimmten Anwendung verbunden. Der Verarbeitungsagent verfügt über alle Informationen und Steuerblöcke, die zum Beenden von Aktionen im Datenbankmanager erforderlich sind, die von der Anwendung angefordert wurden.

Es gibt vier Arten von Verarbeitungsagenten: *aktive Koordinationsagenten* (Active Coordinator Agents), *Subagenten* (Subagents), *inaktive Agenten* (Inactive Agents) sowie *Agenten im Bereitschaftsmodus* (Idle Agents).

Der Agent im Bereitschaftsmodus ist die einfachste Form eines Verarbeitungsagenten: Es ist nicht mit einem logischen Agenten verbunden, hat keine abgehende Verbindung und verfügt nicht über eine Verbindung zu einer lokalen Datenbank oder eine Exemplarverbindung.

Der inaktive Agent ist ein Verarbeitungsagent, der sich nicht in einer aktiven Transaktion befindet, nicht mit einem logischen Agenten verbunden ist, keine abgehende Verbindung hat und nicht über eine Verbindung zu einer lokalen Datenbank oder eine Exemplarverbindung verfügt. Ein inaktiver Agent kann sich mit einem anderen logischen Agenten verbinden und dadurch für die Anwendung arbeiten, die durch diesen logischen Agenten repräsentiert wird.

Jeder Prozess/Thread einer Client-Anwendung verfügt über einen einzigen *aktiven Koordinationsagenten*, der mit einer Datenbank arbeitet. Wenn der Koordinationsagent gestartet ist, führt er alle Datenbankanforderungen für die zugehörige Anwendung aus und kommuniziert mit anderen Agenten über die Interprozesskommunikation (IPC) oder über Protokolle zur Fernverbindung. Jeder Agent arbeitet mit seinem eigenen privaten Speicher und benutzt Ressourcen des Datenbankmanagers und globale Datenbankressourcen, wie z. B. den Pufferpool, gemeinsam mit anderen Agenten. Wenn eine Transaktion vollständig abgeschlossen wird, kann sich der aktive Koordinationsagent vom logischen Agenten lösen und so ein inaktiver Agent werden.

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität verteilt der Koordinationsagent die Datenbankanforderungen an *Subagenten*, die ihrerseits die Anforderungen für die Anwendungen ausführen. Wenn der koordinierende Agent erstellt ist, verarbeitet er alle Datenbankanforderungen für seine Anwendung, indem er die Subagenten koordiniert, die die Anforderungen in der Datenbank ausführen.

Wenn ein Client die Verbindung zu einer Datenbank oder einem Exemplar beendet, geschieht mit dem koordinierenden Agenten folgendes:

- Er wird ein aktiver Agent. Wenn andere logische Agenten warten, wird der Verarbeitungsagent ein aktiver Koordinationsagent.
- Er wird freigegeben und als 'im Bereitschaftsmodus' markiert, wenn keine anderen logischen Agenten warten und die maximale Zahl von Poolagenten noch nicht erreicht wurde.
- Er wird beendet, und der von ihm verwendete Speicher wird freigegeben, wenn keine anderen logischen Agenten warten und die maximale Zahl von Poolagenten erreicht wurde.

Die Agenten, die keine Arbeit für Anwendungen ausführen und darauf warten, zugeordnet zu werden, gelten als Agenten im Bereitschaftsmodus und befinden sich in einem *Agentenpool*. Diese Agenten sind für Anforderungen von Koordinationsagenten, die für Client-Programme aktiv sind, oder für Subagenten, die für vorhandene Koordinationsagenten aktiv sind, verfügbar. Die Anzahl der verfügbaren Agenten hängt vom Wert der Konfigurationsparameter *maxagents* und *num_poolagents* des Datenbankmanagers ab.

Agenten aus dem Agentenpool (*num_poolagents*) werden als Koordinationsagenten für einen der folgenden Zwecke erneut verwendet:

- Für ferne Anwendungen auf TCP/IP-Basis
- Für lokale Anwendungen in Betriebssystemen auf UNIX-Basis
- Für lokale sowie ferne Anwendungen in den Betriebssystemen Windows NT und OS/2

Andernfalls erstellen ferne Anwendungen immer einen neuen Agenten.

Falls kein Agent im Bereitschaftsmodus vorhanden ist, wenn ein Agent angefordert wird, muss ein neuer Agent dynamisch erstellt werden. Die Erstellung eines neuen Agenten verursacht einen bestimmten Systemaufwand, so dass die Leistung bei CONNECT- und ATTACH-Anweisungen merklich besser ist, wenn bereits ein Agent im Bereitschaftsmodus vorhanden ist, der für einen Client aktiviert werden kann.

Wenn ein Subagent für eine Anwendung aktiv ist, wird er als dieser Anwendung *zugeordnet* betrachtet. Nach Beendigung der angeforderten Operationen kann er in den Agentenpool gesetzt werden, bleibt jedoch der ursprünglichen Anwendung zugeordnet. Wenn die Anwendung eine weitere Operation anfordert, überprüft der Datenbankmanager bei der Suche nach einem Agenten für die Anwendung zunächst den Agentenpool nach zugeordneten freien Agenten.

Die Möglichkeit, die Zahl der verbundenen Anwendungen (unter Verwendung der Zahl der logischen Agenten, die in *max_logicagents* definiert wird) und die Zahl der Anwendungsanforderungen, die verarbeitet werden können (unter Verwendung der Zahl der aktiven Koordinationsagenten, die in *max_coordagents* definiert wird), getrennt zu steuern, schafft Flexibilität in den in der Datenbank verarbeiteten Auslastungen. Eine Eins-zu-eins-Beziehung zwischen der Zahl der verbundenen Anwendungen und der Zahl der Anwendungsanforderungen, die verarbeitet werden können, ist die normale Methode, in der Anwendungen mit der Datenbank arbeiten. Es kann jedoch sein, dass Sie aufgrund Ihrer Arbeitsumgebung eine Viele-zu-eins-Beziehung zwischen der Zahl der verbundenen Anwendungen und der Zahl der Anwendungsanforderungen benötigen, die verarbeitet werden können.

Da der globale Ressourcensystemaufwand der Datenbank den aktiven Koordinationsagenten zugeordnet ist, bedeutet eine größere Zahl dieser Agenten, dass eine größere Möglichkeit dafür besteht, dass die obere Begrenzung der verfügbaren globalen Ressourcen der Datenbank erreicht wird. Sie können mehr verbundene Anwendungen als aktive Koordinationsagenten zulassen, damit die obere Begrenzung der verfügbaren globale Ressourcen der Datenbank nicht erreicht werden. Wenn Sie für *max_logicagents* einen größeren Wert als für *max_coordagents* festlegen, konzentrieren Sie Ihre Datenbankarbeit.

Weitere Informationen und Beispiele zur Verwendung von DB2 Connect als XA-Transaktionsunterstützungskonzentrator finden Sie im Handbuch *DB2 Connect Benutzerhandbuch*.

In einer Umgebung, für die die Verwendung von DB2 Connect zur Verbindung mit fernen Systemen erforderlich ist, gibt es einen *Pool abgehender Verbindungen (Outbound Connect Pool)*. Dieser Verbindungspool verringert die Zeit, die (im Anschluss an die erste Verbindung) zur Herstellung der Verbindung zu einem Host benötigt wird. Wenn eine Trennung von einem Host angefordert wird, beendet DB2 Connect die eingehende Verbindung (Inbound Connection), behält die abgehende Verbindung (Outbound Connection) zum Host jedoch in einem Pool. Wenn eine neue Anforderung zur Herstellung einer Verbindung zu diesem Host erfolgt, greift DB2 Connect auf eine vorhandene abgehende Verbindung (falls verfügbar) aus dem Pool zurück.

Anmerkung: Beim Einsatz der Poolfunktion für Verbindungen ist DB2 Connect auf eingehende TCP/IP- und abgehende TCP/IP- und SNA-Verbindungen beschränkt. Bei Verwendung von SNA muss die Sicherheitsart auf den Wert `NONE` (Keine) eingestellt sein, damit die Verbindung in den Pool gesetzt werden kann.

Bei der Verwendung des Verbindungspools schließt der aktive Agent seine abgehende Verbindung nach einer Trennung nicht, sondern wird mit einer aktiven Verbindung zum fernen Host in den Agentenpool versetzt. Diese Art von Agent wird als *inaktiver DRDA-Agent* bezeichnet. Der Pool für inaktive DRDA-Agenten ist mit dem Pool für abgehende Verbindungen identisch-.„DRDA" steht für „verteilte relationale Datenbankarchitektur" (Distributed Relational Database Architecture).

Betrachten Sie die folgenden Beispiele für vier verschiedene Verwendungs- und Auslastungsanforderungen:

1. Im ersten Beispiel stellen durchschnittlich 40 Benutzer über DB2 Connect gleichzeitig eine Verbindung zu fernen Host-Datenbanken her. Zuweilen erreicht die Anzahl gleichzeitig bestehender Verbindungen Spitzenwerte um 50, überschreitet 55 jedoch nie. Die Transaktionen sind von kurzer Dauer, und Benutzer stellen häufig Verbindungen her und trennen sie wieder.

   Unter diesen Bedingungen sollte der Systemadministrator den Parameter *max_coordagents* mit dem Wert 55 konfigurieren, da er weiß, dass nie mehr als 55 Benutzer gleichzeitig versuchen werden, eine Verbindung über DB2 Connect herzustellen. Der Wert für *num_poolagents*, die Größe des Agentenpools, sollte auf 40 gesetzt werden, da dies zu einem beliebigen Zeitpunkt die durchschnittliche Anzahl von Benutzern ist, die verbunden sind bzw. versuchen, eine Verbindung herzustellen. Diese Poolgröße sorgt für ausreichend vorhandene Verbindungen zur fernen Datenbank, um alle

eingehenden Client-Anforderungen zu erfüllen, ohne, abgesehen von den Spitzenauslastungszeiten, neue Verbindungen erstellen zu müssen.

2. In diesem zweiten Beispiel ist die Auslastung mit ca. 1 000 eingehenden Client-Verbindungen wesentlich höher. Die Benutzerverbindungen sind ebenfalls von kurzer Dauer. Der Systemadministrator will außer diesen gleichzeitig bestehenden Verbindungen keine weiteren zulassen. Daher setzt der Systemadministrator sowohl *max_coordagents* als auch *num_poolagents* auf den Wert 1 000. Dies bedeutet, dass die maximale Anzahl eingehender Clients, die gleichzeitig mit der fernen Datenbank verbunden sein können, 1 000 ist. Wenn alle Clients die Verbindung trennen, enthält der Pool genau 1 000 verbundene Agenten, die alle darauf warten, neue eingehende Clients zu bedienen.
3. Das dritte Beispiel betrachtet eine einzelne Anwendung, die eine Verbindung zu nur einer fernen Datenbank über DB2 Connect herstellt. Die Anwendung bleibt über lange Zeiträume hinweg verbunden. In diesem Beispiel besteht die beste Konfiguration für den Agenten- und den Verbindungspool darin, *max_coordagents* auf den Wert 1 zu setzen, da bekannt ist, dass höchstens ein Client eine Verbindung herstellt. Der Parameter *num_poolagents* kann in diesem Fall auf den Wert 0 gesetzt werden, da die Herstellung und Trennung der Verbindung zum fernen Host nicht häufig durchgeführt wird. Durch die Einstellung des Parameters *num_poolagents* auf den Wert 0 wird die Poolfunktion für Verbindungen effektiv inaktiviert, da keine Agenten mit aktiven Verbindungen zur fernen Datenbank in den Pool gesetzt werden. Für jeden neuen eingehenden Client, der die Verbindung herstellt, wird ein neuer Agent erstellt und eine neue Fernverbindung hergestellt.
4. Das vierte Beispiel zeigt eine Variante, die von den drei vorigen Auslastungsbeispielen ausgeht. In diesem Beispiel will der Systemadministrator die gleichzeitigen Zugriffe auf ferne Datenbanken auf nur 100 beschränken. Aus diesem Grund wird *max_coordagents* und, um eine maximale Leistung bei der Verbindungsherstellung zu ermöglichen, *num_poolagents* auf den Wert 100 gesetzt. Später indes könnte sich die Notwendigkeit ergeben, lokal eine Verbindung herzustellen, um die Auslastung auf dem System, auf dem DB2 Connect installiert ist, zu überwachen. Es wird geschätzt, dass zu einem beliebigen Zeitpunkt nicht mehr als fünf Momentaufnahmen von Überwachungsprogrammen gleichzeitig erfolgen, so dass *max_coordagents* auf den Wert 105 gesetzt wird. Dieser neue Konfigurationswert ermöglicht, dass die maximale Anzahl gleichzeitig verbundener Anwendungen über die frühere Obergrenze von 100 hinaus anwachsen kann, um die gelegentlichen Verbindungen für Momentaufnahmen von Überwachungsprogrammen und/oder Verbindungen zum Exemplar aufzunehmen.

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität besitzt jede Partition (d. h. jeder Datenbank-Server oder Knoten) einen eigenen Pool von Agenten, aus dem Subagenten entnommen werden. Durch die Verwendung dieses Pools müssen Subagenten nicht jedes Mal erstellt und wieder gelöscht werden, wenn ein Subagent benötigt wird oder seine Arbeit beendet hat. Die Subagenten können als zugeordnete Agenten im Pool bleiben und vom Datenbankmanager für neue Anforderungen von der Anwendung, der sie zugeordnet sind, verwendet werden.

Die folgenden Konfigurationsparameter des Datenbankmanagers beeinflussen die Anzahl von Datenbankagenten:

- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468. Wenn die Anzahl der Verarbeitungsagenten diesen Wert erreicht, werden alle nachfolgenden Anforderungen, die einen neuen Agenten benötigen, zurückgewiesen, bis die Anzahl der Agenten wieder unter den Wert gesunken ist. Dieser Wert bezieht sich auf die Gesamtanzahl von Agenten, einschließlich Koordinationsagenten, Subagenten, inaktiven Agenten und Agenten im Bereitschaftsmodus, die für alle Anwendungen zusammen aktiv sind.
- „Größe des Agentenpools (num_poolagents)" auf Seite 472. Die Zahl der inaktiven Agenten, der Agenten im Bereitschaftsmodus und der zugeordneten Subagenten im Agentenpool kann diesen Wert nicht überschreiten.
- „Anfangswert für die Anzahl Agenten im Pool (num_initagents)" auf Seite 473. Wenn der Datenbankmanager gestartet wird, wird ein Pool von Verarbeitungsagenten nach diesem Wert erstellt. Dadurch wird die Leistung für Erstabfragen erhöht. Die Verarbeitungsagenten beginnen alle als Agenten im Bereitschaftsmodus.
- „Maximale Anzahl logischer Agenten (max_logicagents)" auf Seite 471. Die maximale Zahl logischer Agenten. Da alle Anwendungen über einen logischen Agenten verfügen, steuert dieser Parameter die maximale Zahl von Anwendungen, von denen aus eine Verbindung zum Exemplar hergestellt werden kann.
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470. In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität begrenzt dieser Wert die Anzahl koordinierender Agenten.

- „Maximale Anzahl gleichzeitig aktiver Agenten (maxcagents)" auf Seite 469. Mit diesem Wert wird die Anzahl von *Token* gesteuert, die der Datenbankmanager zulässt. Für jede Datenbanktransaktion (Arbeitseinheit), die auftritt, wenn ein Client mit einer Datenbank verbunden ist, muss ein koordinierender Agent die Berechtigung zur Verarbeitung (als Verarbeitungstoken bezeichnet) vom Datenbankmanager einholen. Nur Agenten mit einem Verarbeitungstoken haben vom Datenbankmanager die Berechtigung, eine Arbeitseinheit an einer Datenbank auszuführen. Wenn kein Token verfügbar ist, wartet der Agent, bis wieder eines verfügbar ist, und verarbeitet anschließend die angeforderte Arbeitseinheit.

  Dieser Parameter kann in Umgebungen nützlich sein, in denen der Leistungsbedarf in Zeiten hoher Auslastung die Kapazität der vorhandenen Systemressourcen (Hauptspeicher, CPU und Plattenspeicher) übersteigt. In einer solchen Umgebung kann zu Spitzenzeiten z. B. durch Seitenwechselvorgänge ein starker Leistungsabfall auftreten. Mit diesem Parameter können Sie die Auslastung steuern und den Leistungsabfall verhindern. Er kann sich jedoch auf den gleichzeitigen Zugriff und/oder die verschiedenen Wartezeiten auswirken.

In Umgebungen mit partitionierten Datenbanken und Umgebungen mit aktivierter partitionsinterner Parallelität besteht eine enge Beziehung zwischen dem Einfluss auf die Leistung und den Speicherbedarf innerhalb des Systems und der Optimierung des Agentenpools:

- Der Konfigurationsparameter des Datenbankmanagers für die Größe des Agentenpools (*num_poolagents*) betrifft die Gesamtanzahl von Subagenten, die Anwendungen in einer Partition (d. h. Knoten) zugeordnet bleiben können. Wenn die Poolgröße zu klein ist (und der Pool voll ist), wird ein Subagent aus der Zuordnung mit der Anwendung, für die er aktiv war, gelöst und beendet. Eine solche Bedingung führt zu einer Leistungsbeeinträchtigung, da Subagenten ständig erstellt und den Anwendungen neu zugeordnet werden müssen.

  Außerdem kann eine Anwendung den Pool mit zugeordneten Subagenten füllen, wenn der Wert für den Parameter *num_poolagents* zu klein ist. Das bedeutet, wenn eine andere Anwendung einen neuen Subagenten benötigt und über keine Agenten im zugeordneten Pool verfügt, „stiehlt" sie Subagenten aus den Agentenpools anderer Anwendungen. In diesem Fall ist der Systemaufwand recht hoch und die Leistung gering.

- Die oben beschriebenen Fälle müssen gegen den Ressourcenaufwand für zu viele Agenten, die zu einem gegebenen Zeitpunkt aktiv sein können, abgewogen werden.

  Wenn der Wert des Parameters *num_poolagents* zum Beispiel zu groß ist, werden zugeordnete Subagenten lange Zeit ungenutzt im Pool behalten. Diese Subagenten verbrauchen Ressourcen des Datenbankmanagers, die dann nicht für andere Funktionen zur Verfügung stehen.

Neben den Datenbankagenten gibt es auch noch andere asynchrone Aktivitäten des Datenbankmanagers, die als eigene Prozesse (bzw. Threads) ausgeführt werden. Dazu gehören:

- E/A-Server (oder E/A-Vorableseprozesse) der Datenbank (siehe „Vorablesen von Daten in den Pufferpool" auf Seite 301)
- Asynchrone Seitenlöschfunktionen der Datenbank (siehe „Verwalten des Datenbankpufferpools" auf Seite 292)
- Protokollfunktionen der Datenbank
- Detektoren für gegenseitige Sperren in der Datenbank
- Ereignismonitoren
- Übertragungs- und IPC-Empfangsprozesse (Listeners)
- Prozesse zur gleichmäßigen Verteilung von Daten auf Tabellenbereichsbehälter

Weitere Informationen zur Identifizierung der verschiedenen DB2-Prozesse finden Sie im Handbuch *Troubleshooting Guide*.

Verwenden des Datenbanksystemmonitors

Der DB2-Datenbankmanager pflegt Daten über den Betrieb, die Leistung und die Anwendungen, die ihn verwenden. Diese Daten werden während des Betriebs des Datenbankmanagers verwaltet und können wichtige Informationen über die Leistung und zur Fehlerbehebung liefern. Zum Beispiel erhalten Sie Anhaltspunkte über:

- Die Anzahl von Anwendungen, die mit einer Datenbank verbunden sind, ihren Status und welche SQL-Anweisungen (falls überhaupt) von jeder Anwendung ausgeführt werden.
- Informationen, die zeigen, wie gut der Datenbankmanager und die Datenbank konfiguriert sind, und bei der Optimierung helfen.
- Aufgetretene gegenseitige Sperren für eine angegebene Datenbank, beteiligte Anwendungen und welche Sperren in der Konkurrenzsituation waren.
- Die Liste von Sperren, die von einer Anwendungen oder Datenbank aktiviert wurden. Wenn die Anwendung die Verarbeitung nicht fortsetzen kann, weil sie auf eine Sperre wartet, werden weitere Informationen zur Sperre, einschließlich der Anwendung, die sie aktiviert hat, bereitgestellt.

Da die Erfassung einiger dieser Daten mit Systemaufwand für den Betrieb von DB2 verbunden ist, sind **Monitorschalter** verfügbar, mit denen gesteuert werden kann, welche Informationen erfasst werden. Zum expliziten Einstellen der Monitorschalter wird der Befehl UPDATE MONITOR SWITCHES oder die API sqlmon() verwendet. (Sie benötigen hierzu die Berechtigung SYSADM, SYSCTRL oder SYSMAINT.)

Es gibt zwei Methoden, auf die vom Datenbankmanager gepflegten Daten zuzugreifen:

- **Erstellen einer Momentaufnahme**

  Sie haben drei Möglichkeiten, eine Momentaufnahme zu erstellen. Sie können den Befehl GET SNAPSHOT über die Befehlszeile oder die Steuerzentrale unter OS/2- bzw. Windows-Betriebssystemen verwenden, um mit einer grafischen Schnittstelle zu arbeiten, oder unter Verwendung des API-Aufrufs sqlmonss() eine eigene Anwendung schreiben.

  Die Steuerzentrale, die über den DB2-Ordner oder mit Hilfe des Befehls `db2cc` aufgerufen werden kann, stellt ein Programm eines Leistungsmonitors (Performance Monitor) bereit, der Probedaten zur Überwachung in regelmäßigen Intervallen durch Erstellen einer Momentaufnahme erfasst. Diese grafische Schnittstelle bietet entweder Diagramme oder Textanzeigen von den Daten der Momentaufnahmen sowohl in Detailsicht als auch in Übersichtsform an. Sie können außerdem Leistungsvariablen mit Hilfe von Datenelementen definieren, die vom Datenbankmonitor zurückgegeben werden.

Das Tool Snapshot Monitor der Steuerzentrale ermöglicht Ihnen auch, Ausnahmebedingungen zu definieren, indem Sie Schwellenwerte für Leistungsvariablen angeben können. Für den Fall, dass ein Schwellenwert erreicht wird, können Sie eine der folgenden Aktionen vordefinieren: Benachrichtigung über ein Fenster oder durch ein akustisches Signal und/oder Ausführung einer Prozedur oder eines Programms.

Wenn Sie eine Momentaufnahme über die Steuerzentrale erstellen, können Sie eine Aktion ausführen, die ein Datenbankobjekt (z. B. ein Exemplar oder eine Datenbank) ändert oder löscht, während Sie mit Snapshot Monitor entweder dieses Objekt oder eines der davon abhängigen Objekte überwachen. (Außerdem können Sie bei der Überwachung eines partitionierten Datenbanksystems die Anzeige der Objekte der partitionierten Datenbank nicht aktualisieren.) Das heißt zum Beispiel, dass Sie nicht Datenbank A überwachen können, wenn Sie das zugehörige Exemplar löschen wollen. Wenn Sie jedoch nur das Exemplar überwachen, können Sie Datenbank A ändern.

Um sämtliche Überwachungsfunktionen für ein Exemplar (einschließlich der zugehörigen abhängigen Objekte) zu stoppen, wählen Sie die Option **Monitor stoppen** im Kontextmenü des Exemplars aus. Sie sollten immer die Überwachung vom Exemplar aus stoppen, da dadurch alle Sperren, die von Performance Monitor aktiviert wurden, freigegeben werden.

- **Verwenden eines Ereignismonitors**

  Ein Ereignismonitor (Event Monitor) erfasst Systemmonitordaten nach Eintreten bestimmter Ereignisse, wie z. B. das Ende einer Transaktion, das Ende einer Anweisung oder die Erkennung einer gegenseitigen Sperre. Diese Informationen können in Dateien oder benannte Pipes geschrieben werden.

  Ein Ereignismonitor wird wie folgt verwendet:

  1. Erstellen Sie die Definition des Ereignismonitors mit Hilfe der Steuerzentrale oder der SQL-Anweisung CREATE EVENT MONITOR. Diese Anweisung speichert die Definition in den Systemkatalogen der Datenbank.
  2. Aktivieren Sie den Ereignismonitor über die Steuerzentrale oder mit der folgenden SQL-Anweisung:

     ```
     SET EVENT MONITOR ergname STATE 1
     ```

     Wenn die Ergebnisse in eine benannte Pipe geschrieben werden, starten Sie die Anwendung, die die Daten aus der benannten Pipe liest, bevor Sie den Ereignismonitor aktivieren. Sie können entweder eine eigene Anwendung zum Lesen schreiben oder den Befehl **db2evmon** verwenden. Sobald der Ereignismonitor aktiv ist und mit dem Schreiben von Ereignissen in die Pipe beginnt, liest **db2evmon** die Ereignisse, wie sie generiert werden, und schreibt Sie in die Standardausgabe.

3. Lesen Sie die Ablaufverfolgungsdaten. Wenn Sie einen Ereignismonitor verwenden, der in eine Datei schreibt, können Sie die binären Ablaufverfolgungsdaten, die er erstellt, mit einer der folgenden Methoden anzeigen:
   - Verwenden Sie das Tool **db2evmon** zum Formatieren der Ablaufverfolgungsdaten in der Standardausgabe.
   - Klicken Sie das Symbol **Event Analyzer** in der Steuerzentrale (bei Windows- oder OS/2-Systemen) an, um eine grafische Schnittstelle zum Anzeigen der Ablaufverfolgungsdaten, zum Suchen nach Schlüsselwörtern und zum Herausfiltern nicht erwünschter Daten aufzurufen.

     Anmerkung: Wenn das Datenbanksystem, das Sie überwachen, nicht auf derselben Maschine wie die Steuerzentrale ausgeführt wird, müssen Sie die Ereignismonitordatei auf dieselbe Maschine wie die Steuerzentrale kopieren, damit die Ablaufverfolgung angezeigt werden kann. Sie können die Datei stattdessen auch in ein gemeinsam benutztes Dateisystem stellen, auf das beide Maschinen zugreifen können.

Informationen zum Systemdatenbankmonitor und zum Ereignismonitor finden Sie im Handbuch *System Monitor Guide and Reference*.

Erweitern von Speicher

Ihre Maschine verfügt eventuell über mehr adressierbaren Realspeicher als die maximal adressierbare virtuelle Speichermenge (z. B. liegt der adressierbare virtuelle Speicher auf den meisten Plattformen in der Regel zwischen 2 GB und 4 GB). Sie können den zusätzlichen adressierbaren Realspeicher oberhalb des adressierbaren virtuellen Speichers als *erweiterten Speicher-Cache* konfigurieren. Ein solcher erweiterter Speicher-Cache kann von jedem der definierten Pufferpools verwendet werden und sollte die Leistung des Datenbankmanagers erhöhen. Der Erweiterungsspeicher-Cache wird in Speichersegmenten definiert.

Wenn Sie einen Teil des adressierbaren Realspeichers als Erweiterungsspeicher-Cache verwenden, sollte Ihnen bewusst sein, dass dieser Speicher dann nicht mehr für andere Zwecke auf der Maschine, wie beispielsweise als JFS-Cache oder als privater Adressraum für Prozesse, verwendet werden kann. Das Zuordnen von zusätzlichem adressierbaren Realspeicher zum Erweiterungsspeicher-Cache führt möglicherweise zu höherer Systemauslagerung.

DB2 nutzt den adressierbaren Speicher in Ihrer Maschine mit Hilfe von Pufferpools (siehe „Verwalten des Datenbankpufferpools" auf Seite 292). Der Erweiterungsspeicher-Cache wird von den Pufferpools als sekundäre Cache-Ebene genutzt (wobei die Pufferpools das Caching der ersten Ebene durchführen). Im Idealfall können die Pufferpools die Daten enthalten, auf die am häufigsten zugegriffen wird, während der Erweiterungsspeicher-Cache Daten enthält, auf die zwar zugegriffen, aber nicht so häufig zugegriffen wird.

Wenn Sie in Windows 2000 über die Registrierungsvariable DB2_AWE AWE-Pufferpools zuordnen (AWE, Address Windowing Extensions), kann der Erweiterungsspeicher-Cache nicht verwendet werden.

Die folgenden Datenbankkonfigurationsparameter beeinflussen die Menge und die Größe des für den Erweiterungsspeicher-Cache verfügbaren Speichers:

- *num_estore_segs* definiert die Anzahl der Segmente für den Erweiterungsspeicher-Cache. Der Standardwert für diesen Konfigurationsparameter ist null, d. h., es ist kein Erweiterungsspeicher-Cache vorhanden. (Siehe „Segmentanzahl für erweiterten Speicher (num_estore_segs)" auf Seite 460.)
- *estore_seg_sz* definiert die Größe der Segmente des Erweiterungsspeichers. Diese Größe wird durch die Plattform begrenzt, auf der der Erweiterungsspeicher-Cache verwendet wird. (Siehe „Segmentgröße für erweiterten Speicher (estore_seg_sz)" auf Seite 460.)

Da der Erweiterungsspeicher-Cache eine Erweiterung eines Pufferpools darstellt, muss er immer einem oder mehreren bestimmten Pufferpools zugeordnet werden. Das bedeutet, dass Sie definieren müssen, welche Pufferpools einen Cache, wenn er erstellt ist, nutzen können. Die Anweisungen CREATE und ALTER BUFFERPOOL verfügen über die Attribute NOT EXTENDED STORAGE und EXTENDED STORAGE, die die Verwendung des Cache steuern. Standardmäßig verwenden weder der Pufferpool IBMDEFAULTBP noch irgendein neu erstellter Pufferpool den erweiterten Speicher.

Anmerkung: Eventuell verwenden Sie Pufferpools, für die unterschiedliche Seitengrößen definiert sind. Für einige oder auch für alle Pufferpools ist u. U. die Verwendung des erweiterten Speichers definiert. Die für die Erweiterungsspeicherunterstützung verwendete Seitengröße ist die größte definierte Seitengröße.

Der Datenbankmanager kann Daten im Erweiterungsspeicher-Cache nicht direkt bearbeiten. Jedoch kann er Daten aus dem Erweiterungsspeicher-Cache viel schneller als von der Platte in den Pufferpool übertragen.

Wenn eine Zeile von Daten einer Seite im Erweiterungsspeicher-Cache benötigt wird, wird die gesamte Seite in den entsprechenden Pufferpool eingelesen.

Ein Pufferpool und der zugeordnete Erweiterungsspeicher-Cache (falls definiert) werden zugeordnet, wenn eine Datenbank aktiviert oder die erste Verbindung zu ihr hergestellt wird.

Kapitel 9. Verwenden von Governor

Governor wird zum Überwachen und Ändern des Verhaltens von Anwendungen verwendet, die gegen eine Datenbank ausgeführt werden.

Governor enthält zwei Komponenten:
- Ein Front-End-Dienstprogramm
- Einen Dämonen

Beim Start von Governor setzen Sie im Front-End-Dienstprogramm von Governor einen Startbefehl ab, mit dem der Dämon gestartet wird. Standardmäßig wird ein Dämon auf sämtlichen Partitionen einer partitionierten Datenbank gestartet; mit Hilfe des Front-End-Dienstprogramms können Sie jedoch auch einen einzelnen Dämonen auf einer bestimmten Partition starten, um die Aktivität gegen die vorgefundene Datenbankpartition zu überwachen. Außerdem kann ein Dämon die Aktivität in einer Datenbank mit einer Partition überwachen. Weitere Informationen finden Sie in „Starten und Stoppen von Governor" auf Seite 334.

Ein Governor-Dämon sammelt Statistik zu den gegen eine Datenbank ausgeführten Anwendungen. Dann vergleicht er diese Statistik mit den Regeln, die Sie in einer Governor-Konfigurationsdatei angegeben haben, die für die betreffende Datenbank gilt. (Einzelheiten hierzu finden Sie in „Erstellen der Governor-Konfigurationsdatei" auf Seite 337.) Governor handelt dann entsprechend diesen Regeln. Beispielsweise könnte eine Regel anzeigen, dass eine Anwendung zu viel Ressourcen verwendet. Dann könnte Governor die Priorität der Anwendung ändern oder ihre Abmeldung von der Datenbank erzwingen, je nach Ihren Anweisungen in der Governor-Konfigurationsdatei.

Beinhaltet eine Regel die Änderung der Priorität einer Anwendung, so ändert Governor die Priorität der Agenten auf derjenigen Datenbankpartition, auf der er die Ressourcenverletzung festgestellt hat. Beinhaltet eine Regel das erzwungene Abmelden einer Anwendung, so wird die Anwendung abgemeldet, selbst wenn der Governor, der die Ressourcenverletzung festgestellt hat, auf dem Koordinatorknoten dieser Anwendung oder in einer partitionierten Datenbank ausgeführt wird.

Governor protokolliert alle seine Aktionen. Mit einer Abfrage der Protokolldateien können Sie die Aktionen von Governor prüfen. Weitere Informationen finden Sie in „Governor-Protokolldateien" auf Seite 346 und „Abfragen auf Governor-Protokolldateien" auf Seite 348.

Starten und Stoppen von Governor

Verwenden Sie das Front-End-Dienstprogramm von Governor `db2gov` zum Starten und Stoppen von Governor (auf allen bzw. auf einzelnen Datenbankpartitionen). Zur Verwendung des Dienstprogramms ist die Berechtigung SYSADM bzw. SYSCTRL erforderlich.

`db2gov` hat folgende Syntax:

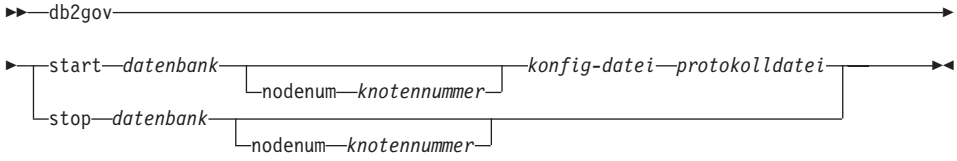


Abbildung 28. Syntax von db2gov

Der Befehl hat die folgenden Parameter:

start **datenbank**
: Startet den Governor-Dämon zur Überwachung der angegebenen Datenbank. Bei *datenbank* können Sie den Datenbanknamen oder den Aliasnamen der Datenbank angeben.

  Der angegebene Datenbankname muss mit dem in der Governor-Konfigurationsdatei angegebenen Namen übereinstimmen. Governor überprüft diese beiden Namen, um sicherzustellen, dass Sie die richtige Konfigurationsdatei verwenden. Wenn das Front-End-Dienstprogramm mit einem Aliasnamen gestartet wird, der sich von dem Aliasnamen der Governor-Konfigurationsdatei unterscheidet, wird eine Fehlernachricht angezeigt, weil Governor nicht feststellen kann, ob die Namen Aliasnamen für dieselbe Datenbank sind.

  In einer Umgebung mit partitionierten Datenbanken prüft das Front-End-Dienstprogramm beim Starten von Governor zunächst die Konfigurationsdatei auf Fehler. Danach liest es die Knotenkonfigurationsdatei und sendet auf alle Datenbankpartitionen den Startbefehl für das Front-End-Dienstprogramm von Governor auf allen Partitionen mit der Startoption (dadurch wird der Dämon auf jeder Datenbankpartition gestartet).

  Anmerkung: Da Governor die Überwachung auf Datenbankebene übernimmt, ist auf jeder überwachten Datenbank ein Dämon aktiv. (In einer Umgebung mit partitionierten Datenbanken ist auf jeder Datenbankpartition ein Dämon aktiv.) Wird Governor für mehrere Datenbanken ausgeführt, so sind mehrere Dämonen auf dem entsprechenden Datenbank-Server aktiv.

nodenum **knotennummer**
Gibt die Datenbankpartition an, auf welcher der Governor-Dämon gestartet werden soll. Die Nummer ist mit der in der Knotenkonfigurationsdatei angegebenen identisch.

Beim Starten von Governor auf einer einzelnen Datenbankpartition erstellt das Front-End-Dienstprogramm einen Dämon zur Auswertung der Konfigurationsdatei von Governor. Der Governor-Dämon stellt sicher, dass noch kein anderer Dämon auf dieser Partition aktiv ist.

konfig-datei
Gibt an, welche Konfigurationsdatei bei der Überwachung der Datenbank verwendet werden soll.

Standardmäßig befindet sich die Konfigurationsdatei im Verzeichnis `sqllib`. Ist die angegebene Datei nicht dort, geht Front-End davon aus, dass der angegebene Name der vollständige Name der Datei ist.

protokolldatei
Gibt den Namen der Datei an, in die Governor die Protokollsätze schreibt. Die Protokolldatei wird im Unterverzeichnis `log` des Verzeichnisses `sqllib` gespeichert. (Unter Windows NT befindet sich das Unterverzeichnis `log` im Exemplarverzeichnis.) Die Nummer der Datenbankpartition, auf der Governor ausgeführt wird, wird dem Namen der Protokolldatei automatisch hinzugefügt (zum Beispiel `mylog.0, mylog.1, mylog.2`).

stop **datenbank**
Stoppt den Governor-Dämon, der die angegebene Datenbank überwacht.

Wenn Sie in einer Umgebung mit partitionierten Datenbanken sind, stoppt das Front-End-Dienstprogramm den Governor auf allen Datenbankpartitionen, indem es die Knotenkonfigurationsdatei liest und den Befehl auf alle Datenbankpartitionen sendet, das Front-End-Dienstprogramm Governor mit dem Stopp-Parameter aufzurufen. Dadurch wird der Dämon auf allen Datenbankpartitionen gestoppt.

nodenum **knotennummer**
Gibt die Datenbankpartition an, auf welcher der Governor-Dämon gestoppt werden soll. Die Nummer ist mit der in der Knotenkonfigurationsdatei angegebenen identisch.

Wenn das Front-End-Dienstprogramm den Governor-Dämon auf einer einzelnen Datenbankpartition stoppt, kommuniziert er mit dem Dämon auf dieser Partition, indem er Dateien im Unterverzeichnis `tmp` des Verzeichnisses `sqllib` erstellt, verschiebt oder löscht. Versuchen Sie nicht, diese Dateien zu löschen oder zu ändern.

Der Governor-Dämon

Wenn der Governor-Dämon gestartet wird (durch das Front-End-Dienstprogramm `db2gov` oder durch Aufwecken), arbeitet er eine Ausführungsschleife ab. Als Erstes überprüft er, ob seine Governor-Konfigurationsdatei geändert bzw. noch nicht gelesen worden ist. Trifft eine der beiden Bedingungen zu, dann liest der Dämon die Regeln in der Datei. Dadurch können Sie das Verhalten des Governor-Dämons ändern, während er aktiv ist.

Danach setzt der Governor-Dämon eine Anforderung für eine Momentaufnahme ab, um Statistik für jede(n) in der Datenbank arbeitende(n) Anwendung bzw. Agenten zu erhalten.

Anmerkung: Auf einigen Plattformen ist keine CPU-Statistik von DB2 Monitor verfügbar. Ist dies der Fall, so sind weder Benutzereintragsregel noch CPU-Begrenzung verfügbar.

Anschließend überprüft Governor die Statistik aller Anwendungen anhand der Regeln in der Governor-Konfigurationsdatei. Wenn eine Regel für eine Anwendung greift, so hat Governor folgende Möglichkeiten: Er kann die Anwendung zum Abmelden zwingen, ihre Priorität ändern (dadurch werden indirekt alle Agentenprioritäten der Agenten und Subagenten geändert, die in dieser Datenbankpartition arbeiten) oder den Plan der Anwendung ändern, wodurch je nach Art der von der Regel angegebenen Aktion die Prioritäten der in der Anwendung arbeitenden Agenten indirekt geändert werden. Governor schreibt jede von ihm ausgeführte Aktion in eine Protokolldatei.

Anmerkung: Governor kann nicht als Alternative für die Anpassung der Agentenprioritäten verwendet werden, wenn der Konfigurationsparameter *agentpri* für den Datenbankmanager nicht den Systemstandardwert enthält. (Dies gilt nicht für Windows NT-Plattformen.)

Wenn Governor die Überprüfung aller Anwendungen abgeschlossen hat, wird er für die in der Konfigurationsdatei angegebene Zeitdauer inaktiviert. Nach Ablauf dieser Frist wacht Governor auf und führt dieselbe Schleife erneut aus.

Trifft Governor auf einen Fehler oder eine Stoppsignal, bereinigt er das Problem vor der Inaktivierung. Bei der Bereinigung werden die Prioritäten aller Anwendungsagenten zurückgesetzt (mit Hilfe einer Liste von Anwendungen, deren Prioritäten festgelegt wurden). Danach werden die Prioritäten derjenigen Agenten zurückgesetzt, die nicht mehr in einer Anwendung arbeiten. Dadurch wird sichergestellt, dass nach Beendigung von Governor nicht mit anderen als den Standardprioritäten aktiv bleiben. Wenn ein Fehler auftritt, wird eine Nachricht in die Datei `db2diag.log` geschrieben, die anzeigt, dass Governor abnormal beendet wurde.

Anmerkung: Governor-Dämon ist keine Datenbankanwendung; daher erhält er keine Verbindung zu der Datenbank aufrecht. (Er hat jedoch eine Exemplarverbindung (Attach).) Governor-Dämon kann feststellen, wenn der Datenbankmanager beendet wird, da er Anforderungen für Momentaufnahmen absetzen kann.

Erstellen der Governor-Konfigurationsdatei

Beim Start von Governor geben Sie den Namen der Konfigurationsdatei an, die die Regeln zum Überprüfen der in einer Datenbank aktiven Anwendungen enthält. Governor handelt auf der Grundlage dieser Regeln.

Wenn Ihre Anforderungen zur Verwaltung der Datenbank sich ändern, können Sie die Konfigurationsdatei editieren, ohne Governor zu stoppen. Jeder Governor-Dämon findet die geänderte Datei und liest sie erneut.

Sie müssen die Konfigurationsdatei in einem Verzeichnis erstellen, das über alle Datenbankknoten angehängt ist, denn alle Governor-Dämonen der einzelnen Partitionen müssen in der Lage sein, dieselbe Konfigurationsdatei zu lesen.

Die Konfigurationsdatei enthält Regeln und Kommentare. Die meisten Einträge können in Großbuchstaben, Kleinbuchstaben oder gemischt angegeben werden. Die Ausnahme ist `applname`, wo Groß- und Kleinschreibung unterschieden wird.

Kommentare werden mit geschweiften Klammern { } begrenzt. Die Regeln umfassen folgendes:

- Die Datenbank, für die die Regeln gelten
- Die Zeitdauer, während der Governor inaktiviert ist, bevor er aktiv wird und die Anwendungen überprüft
- Die Regeln, die angeben, wie die Anwendungen zu behandeln sind. Diese Regeln bestehen aus kleineren Komponenten, den so genannten Regelklauseln.

Jede Regel in der Datei muss mit einem Semikolon (;) abgeschlossen werden.

Die folgenden Regeln dienen zur Angabe der überwachten Datenbank und der Zeitdauer nach Beendigung der Aktivitäten des Dämons (beschrieben in „Der Governor-Dämon" auf Seite 336) bis zu seinem erneuten Erwachen. Diese Regeln werden nur einmal in der Datei angegeben.

dbname
Der Name oder Aliasname der zu überwachenden Datenbank

account *nnn*

Benutzereintragsdatensätze, die statistische Daten zur CPU-Auslastung für jede Verbindung enthalten, werden jeweils nach Ablauf der angegebenen Anzahl Minuten geschrieben.

Anmerkung: Diese Option ist in der Windows NT-Umgebung nicht verfügbar.

Liegt eine kurze Verbindung ganz innerhalb einer Aufzeichnungsperiode, so wird kein Protokollsatz geschrieben. Wenn Protokollsätze geschrieben werden, enthalten Sie CPU-Statistik, die Aufschluss über die CPU-Verwendung der Verbindung seit dem vorangegangenen Protokollsatz geben. Wird Governor gestoppt und erneut gestartet, wird die CPU-Verwendung möglicherweise in zwei Protokollsätzen aufgezeichnet; diese können über die Anwendungs-IDs in den Protokollsätzen erkannt werden. Weitere Informationen zu Governor-Protokolldateien finden Sie in „Governor-Protokolldateien" auf Seite 346.

interval

Das Intervall in Sekunden, in dem der Dämon aktiv wird. Wenn kein Intervall angegeben wird, wird ein Intervall von 120 Sekunden verwendet.

Sie verbinden die folgenden Regelklauseln zu einer Regel (d.h. nicht jede einzelne Klausel, sondern nur die gesamte Regel wird mit einem Semikolon abgeschlossen). In den Klausel geben Sie die Zeitspanne für die Anwendung einer Regel und die Obergrenze für die verwendbaren Betriebsmittel an. Optional können Sie außerdem noch bestimmte Benutzer oder Anwendungen und Aktionen angeben, die Governor ausführen soll, wenn in einer Regel angegebne Grenzen überschritten werden. Dies Klauseln können innerhalb einer Regel nur einmal, in unterschiedlichen Regeln jedoch mehrmals angegeben werden. Die Klauseln müssen in der angezeigten Reihenfolge angegeben werden. In der folgenden Beschreibung wird eine wahlfreie Klausel durch [ ] angezeigt.

[desc] Gibt eine Textbeschreibung der Regel an. Die Beschreibung muss in einfachen oder doppelten Anführungszeichen stehen.

[time] Gibt die Zeitspanne an, während der die Regel auswertet werden soll.

Die Zeitspanne muss im Format `time hh:mm hh:mm` angegeben werden, z. B. `time 8:00 18:00`. Wird diese Klausel nicht angegeben, ist die Regel 24 Stunden am Tag gültig.

[SQL-Berechtigungs-ID]

Gibt eine oder mehrere Berechtigungs-IDs (authid) an, mit denen die Anwendung ausgeführt wird. Mehrere Berechtigungs-IDs müssen

durch ein Komma (,) voneinander getrennt werden, z. B. `authid gene, michael, james`. Wird diese Klausel in einer Regel nicht angezeigt, wird sie auf alle Berechtigungs-IDs angewendet.

[applname]
Gibt den Namen der ausführbaren Datei (oder Objektdatei) an, welche die Verbindung zu der Datenbank herstellt.

Mehrere Anwendungsnamen müssen durch ein Komma (,) voneinander getrennt werden, z. B. `applname db2bp, batch, geneprog`. Wird diese Klausel in einer Regel nicht angezeigt, wird sie auf alle Anwendungsnamen angewendet.

Anmerkungen:

1. Bei Anwendungsnamen muss Groß-/Kleinschreibung beachtet werden.
2. Der Datenbankmanager schneidet alle Anwendungsnamen auf 20 Zeichen ab. Stellen Sie sicher, dass die zu prüfende Anwendung mit den ersten 20 Zeichen ihres Anwendungsnamens eindeutig identifiziert wird; ist das nicht der Fall, wird möglicherweise unbeabsichtigt eine andere Anwendung überprüft.

   Anwendungsnamen in der Governor-Konfigurationsdatei werden auf 20 Zeichen abgeschnitten, damit sie mit ihrer internen Darstellung übereinstimmen.

setlimit
Gibt Begrenzungen ein, die Governor überprüfen soll. Die Begrenzungen dürfen nur die Werte -1 oder größer als 0 annehmen (z. B. `cpu -1 locks 1000 rowssel 10000`). Sie müssen mindestens eine der Begrenzungen (cpu, locks, rowsread, uowtime) angeben; jede nicht angegebene Begrenzung wird nicht der entsprechenden Regel nicht eingeschränkt. Governor kann folgende Begrenzungen überprüfen:

cpu *nnn*
Gibt die Anzahl der CPU-Sekunden an, die eine Anwendung nutzen kann. Wenn Sie -1 angeben, schränkt Governor die CPU-Verwendung der Anwendung nicht ein.

Anmerkung: Diese Option ist in der Windows NT-Umgebung nicht verfügbar.

locks *nnn*
Gibt die Anzahl der Sperren an, die eine Anwendung enthalten kann. Wenn Sie -1 angeben, schränkt Governor die Anzahl der in der Anwendung enthaltenen Sperren nicht ein.

rowssel *nnn*
Gibt die Anzahl von Zeilen an, die an die Anwendung zurückgegeben werden. Dieser Wert ist nur auf dem

Koordinatorknoten ungleich Null. Wenn Sie -1 angeben, schränkt Governor die Anzahl der auswählbaren Zeilen nicht ein.

uowtime *nnn*
Gibt die Zeitspanne in Sekunden an, die verstreichen kann, nachdem eine Arbeitseinheit (UOW) zum ersten Mal aktiv wird. Wenn Sie -1 angeben, wird die abgelaufene Zeit nicht eingeschränkt.

Anmerkung: Wenn Sie die API sqlmon (den Datenbanksystemmonitor-Schalter) zum Inaktivieren des Schalters der Arbeitseinheit verwendet haben, wird die Fähigkeit von Governor beeinträchtigt, Anwendungen auf der Grundlage der abgelaufenen Zeit der Arbeitseinheit zu überprüfen. Governor verwendet das Überwachungsprogramm zum Sammeln von Systeminformationen. Wenn Sie die Schalter in der Konfigurationsdatei des Datenbankmanagers ausschalten, wird dieses Programm für das gesamte Exemplar abgeschaltet; dadurch erhält Governor diese Informationen nicht mehr. Weitere Informationen finden Sie in „Parameter des Datenbanksystemmonitors" auf Seite 547.

idle *nnn*
Gibt die für eine Verbindung zulässige Leerlaufzeit in Sekunden an, bevor eine Aktion ausgeführt wird. Wenn Sie -1 angeben, wird die Leerlaufzeit der Verbindung nicht eingeschränkt.

rowsread *nnn*
Gibt die Anzahl der von einer Anwendung auswählbaren Zeilen an. Wenn Sie -1 angeben, kann die Anwendung eine unbegrenzte Anzahl an Zeilen auswählen.

Anmerkung: Diese Begrenzung ist nicht mit rowssel identisch. Der Unterschied besteht darin, dass sich rowsread auf die Anzahl der Zeilen bezieht, die gelesen werden mussten, um eine Ergebnismenge zurückgeben zu können. Die Anzahl der gelesenen Zeilen schließt Lesevorgänge der Katalogtabellen durch die Steuerkomponente ein; sie kann durch die Verwendung von Indizes verringert werden.

[action]

Gibt die Aktion an, die ausgeführt werden muss, wenn eine angegebene Begrenzung überschritten wird. Sie können folgende Aktionen angeben:

Anmerkung: Wenn eine Begrenzung überschritten wird und die Aktionsklausel nicht angegeben ist, verringert Governor die Priorität der in der Anwendung aktiven Agenten um 10.

priority *nnn*

Gibt eine Änderung der Priorität der in der Anwendung aktiven Agenten an. Gültige Werte: –20 bis +20.

Dieser Parameter ist unter folgenden Voraussetzungen wirksam:

- Auf UNIX-Plattformen muss der Parameter *agentpri* des Datenbankmanagers den Standardwert aufweisen; andernfalls überschreibt er die Prioritätsklausel.
- Auf OS/2- und Windows NT-Plattformen kann der Datenbankmanagerparameter *agentpri* gemeinsam mit der Aktion *priority* verwendet werden.

force Gibt an, dass der Agent, der die Anwendung wartet, zum Abmelden gezwungen wird. (Setzt die Anweisung FORCE APPLICATION ab, um den koordinierenden Agenten zu beenden.)

schedule [class]

Durch Zeitzuweisung werden die Prioritäten der in den Anwendungen aktiven Agent verbessert und dadurch die durchschnittlichen Antwortzeiten minimiert, ohne irgendeine Anwendung tz benachteiligen.

Governor legt mit Hilfe von Schätzwerten für Abfrageaufwände aus dem internen DB2-Abfragecompiler Prioritäten für die in den Anwendungen aktiven Agenten fest und setzt so seine Zeitzuweisung durch. Wenn die Option class angegeben wird, erfolgt die Zeitzuweisung nur innerhalb der durch die Regel ausgewählten Anwendungen. Wird diese Option nicht angegeben, verwendet Governor eine oder mehrere Klassen, wobei die Zeitzuweisung innerhalb jeder Klasse erfolgt.

Innerhalb einer Klasse werden die Prioritäten für Anwendungen nach folgenden Kriterien vergeben:

- Anzahl der Sperren der Anwendung innerhalb der Klasse (Eine Anwendung, die viele andere Anwendungen durch Sperren aufhält, erhält ein hohe Priorität.)

- Alter der Anwendung (Eine Anwendung, die schon lange auf einem System ist, erhält eine hohe Priorität.)
- Voraussichtliche Restlaufzeit der Anwendung (Eine Anwendung, die kurz vor dem Abschluss steht, erhält eine hohe Priorität.)

Anwendungen, die in keiner Zeitzuweisung enthalten sind, werden mit der höchsten Berechtigung ausgeführt.

Anmerkung: Wenn Sie die API `sqlmon` (den Datenbanksystemmonitor-Schalter) zum Inaktivieren des Schalters der Anweisung verwendet haben, wird die Fähigkeit von Governor beeinträchtigt, Anwendungen auf der Grundlage der abgelaufenen Zeit der Anweisung zu überprüfen. Governor verwendet das Überwachungsprogramm zum Sammeln von Systeminformationen. Wenn Sie die Schalter in der Konfigurationsdatei des Datenbankmanagers ausschalten, wird dieses Programm für das gesamte Exemplar abgeschaltet; dadurch erhält Governor diese Informationen nicht mehr.

Eine Aktion für die Zeitzuweisung kann Folgendes erreichen:

- Anwendungen in unterschiedlichen Gruppen erhalten Zeit zugewiesen, ohne dass diese Zeit gleichmäßig unter allen Anwendungen aufgeteilt wird.

  Wenn beispielsweise 12 Anwendungen (3 kurze, 5 mittlere und 6 lange) gleichzeitig aktiv sind, haben möglicherweise alle eine schlechte Antwortzeit, weil sie die CPU teilen. Der Datenbankadministrator kann zwei Gruppen einrichten, mittlere Anwendungen und lange Anwendungen. Mit Hilfe der Prioritäten kann Governor alle kurzen Anwendungen ausführen und sicherstellen, dass höchstens drei mittlere und drei lange Anwendungen gleichzeitig aktiv sind. Zu diesem Zweck enthält die Governor-Konfigurationsdatei je eine Regel für mittlere bzw. lange Anwendungen. Das folgende Beispiel zeigt einen Abschnitt einer Governor-Konfigurationsdatei, der diesen Punkt verdeutlicht:

```
desc "Mittellange Anwendungen in 1
Zeitzuweisungsklasse
zusammenfassen"
applname medq1,  medq2, medq3, medq4, medq5
setlimit cpu -1  action schedule class;

desc "Lange Anwendungen in 1 Zeitzuweisungsklasse
```

```
zusammenfassen"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;
```

- Unterschiedliche Benutzergruppen (z. B. Abteilungen in Unternehmen) erhalten identische Kriterien für die Vergabe von Prioritäten.

  Wenn eine Gruppe eine Vielzahl von Anwendungen ausführt, kann der Administrator sicherstellen, dass andere Gruppen dennoch akzeptable Antwortzeiten für ihre Anwendungen erhalten. Sind zum Beispiel drei Abteilungen beteiligt (Finanzen, Lagerbestand und Planung), kann man alle Benutzer der Finanzabteilung in eine Gruppe legen, alle Benutzer aus dem Lager in eine andere und alle Planer in die dritte. Dann werden die Verarbeitungskapazitäten etwa gleichmäßig unter den drei Abteilungen aufgeteilt. Das folgende Beispiel zeigt einen Abschnitt einer Governor-Konfigurationsdatei, der diesen Punkt verdeutlicht:

  ```
  desc "Benutzer der
  Finanzwesenabteilung in Gruppe zusammenfassen"
  authid tom, dick, harry, mo, larry, curly
  setlimit cpu -1
  action schedule class;
  desc "Benutzer der Lagerabteilung in Gruppe zusammenfassen"
  authid pat, chris, jack, jill
  setlimit cpu -1
  action schedule class;

  desc "Benutzer der Planungsabteilung in Gruppe zusammenfassen"
  authid tara, dianne, henrietta, maureen, linda, candy
  setlimit cpu -1
  action schedule class;
  ```

- Governor bindet alle Anwendungen in eine Zeitzuweisung ein.

  Wenn die Option class nicht mit der Aktion angegeben wird, erstellt Governor eigene Klassen nach der Anzahl der Anwendungen, für die diese Aktion gilt, und ordnet die Anwendungen in verschiedene Klassen ein. Dies erfolgt entsprechend dem Kostenvoranschlag des DB2-Abfragecompilers für die Abfrage, die eine Anwendung gerade ausführt. Der Administrator kann alle Anwendungen für die Zeitzuweisung (Scheduling) auswählen, indem er nicht angibt, welche Anwendungen ausgewählt sind. Die Klauseln *applname* und *authid* werden also nicht angegeben, und die Klausel *setlimit* verursacht keine Einschränkungen.

Anmerkung: Wenn eine Begrenzung überschritten wird und die Aktionsklausel nicht angegeben ist, verringert Governor die Priorität der in der Anwendung aktiven Agenten.

Wenn mehrere Regeln auf eine Anwendung zutreffen, werden alle Regeln angewendet. Je nach Regel und je nachdem, welche Begrenzungen definiert wurden, ist zuerst die Aktion auszuführen, die der zuerst gefundenen Regelbegrenzung zugeordnet ist. Eine Ausnahme wird gemacht, wenn -1 für eine Klausel in einer Regel angegeben ist. In diesem Fall kann ein in der Klausel der nachfolgenden Regel angegebener Wert nur den zuvor in *derselben* Klausel angegebenen Wert überschreiben: die übrigen Klauseln in der vorhergehenden Regel sind weiterhin gültig. Beispielsweise gibt eine Regel an, dass die Priorität einer Anwendung verringert werden muss, wenn Ihre abgelaufene Zeit 1 Stunde überschritten bzw. wenn Sie mehr als 100 000 Zeilen auswählt (d. h. `rowssel 100000 uowtime 3600`). Eine nachfolgende Regel besagt dagegen, dass diese Anwendung keine Einschränkung bei abgelaufener Zeit hat (d. h. `uowtime -1`). In diesem Fall wird die Priorität der Anwendung nicht geändert, wenn sie länger als eine Stunde aktiv ist (d. h. `uowtime -1` überschreibt `uowtime 3600`). Wählt die Anwendung jedoch mehr als 100 000 Zeilen aus, wird ihre Priorität verringert (da `rowssel 100000` weiterhin gültig ist).

Abb. 29 auf Seite 345 zeigt ein Beispiel für eine Konfigurationsdatei.

```
{ Einmal pro Sekunde aktiv werden, Datenbankname lautet ibmsampl
 Benutzereintragsdatensätze alle 30 Minuten }
interval 1; dbname ibmsampl; account 30;

desc "CPU-Einschränkungen gelten 24 Stunden pro Tag für alle"
setlimit cpu 600 rowssel 1000000 rowsread 5000000;

desc "Keine Arbeitseinheit darf länger als 1 Stunde dauern"
setlimit uowtime 3600 action force;

desc 'Verlangsamen einer Untermenge von Anwendungen'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;

desc "Governor soll diese 6 langen Anwendungen in 1
Prioritätenklasse einordnen"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;

desc "Zeitzuweisung für alle Anwendungen von der Planungsabteilung"
authid planid1, planid2, planid3, planid4, planid5
setlimit cpu -1
action schedule;

desc "Einordnen (Scheduling) aller CPU-Fresser in eine Klasse zur
      Verbrauchssteuerung"
setlimit cpu 3600
action schedule class;

desc "Beschränken der Nutzung von db2 CLP durch neuen Benutzer novice"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;

desc "Während der Tagesarbeitszeit darf keine Anwendung länger als 10"
      Sekunden aktiv sein"
time 8:30 17:00 setlimit cpu 10 action force;

desc "Einige Benutzer, die mit Leistungsoptimierung befasst sind, dürfen
      einige Ihrer Anwendungen in der Mittagspause durchführen"
time 12:00 13:00 authid ming, geoffrey, john, bill
applname tpcc1, tpcc2, tpcA, tpvG setlimit cpu 600 rowssel 120000 action force;

desc "Einige Benutzer sollten nicht eingeschränkt werden -- Datenbankadministrator
  und einige andere. Da dies die letzte Angabe in der Datei ist,
    wird zuvor Angegebenes hierdurch überschrieben. "
authid gene, hershel, janet setlimit cpu -1 locks -1 rowssel -1 uowtime -1;

desc "Erhöhen der Priorität einer wichtigen Anwendung, so dass sie immer
      schnell verarbeitet wird"
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;
```

Abbildung 29. Beispielkonfigurationsdatei (Governor)

Governor-Protokolldateien

Wenn ein Governor-Dämon das Abmelden einer Anwendung erzwingt, die Governor- Konfigurationsdatei liest, die Priorität einer Anwendung ändert, einen Fehler oder eine Warnung entdeckt, gestartet oder beendet wird, zeichnet er das in der Protokolldatei auf. Jeder Governor-Dämon besitzt eine eigene Protokolldatei. Dadurch werden Engpässe vermieden, die zur Sperrung von Dateien führen könnten, wenn viele Governor-Dämonen gleichzeitig auf dieselbe Datei schreiben würden. Mit dem Dienstprogramm `db2govlg` können Sie die Protokolldateien zusammenfügen und Abfragen ausführen. Dieses Dienstprogramm wird im Abschnitt „Abfragen auf Governor-Protokolldateien" auf Seite 348 beschrieben.

Die Protokolldateien werden im Unterverzeichnis `log` des Verzeichnisses `sqllib` gespeichert. (Unter Windows NT befindet sich das Unterverzeichnis `log` im Exemplarverzeichnis.) Durch Absetzen des Befehls `db2gov` geben Sie den Basisdateinamen für die Protokolldatei an. Stellen Sie sicher, dass der Datenbankname im Namen der Protokolldatei enthalten ist, da sich auf allen Knoten aller überwachten Datenbanken eine Protokolldatei befindet. In einer Umgebung mit partitionierten Datenbanken wird die Knotennummer der Datenbankpartition, auf der Governor ausgeführt wird, dem Namen der Protokolldatei automatisch hinzugefügt, um sicherzustellen, dass der Dateiname für jeden Governor eindeutig ist.

Ein Eintrag in der Protokolldatei hat folgendes Format:

```
Datum Uhrzeit Knoten-Nr. Satzart Nachricht
```

Die Felder *Datum* und *Uhrzeit* haben das Format jjjj-mm-tt hh.mm.ss; Sie können die Protokolldateien auf jeder Datenbankpartition durch Sortieren nach diesem Feld zusammenfügen.

Das Feld *Knoten-Nr.* gibt die Nummer der Datenbankpartition an, auf der Governor aktiv ist.

Das Feld *Satzart* enthält unterschiedliche Werte, je nach Art des in das Protokoll geschriebenen Eintrags. Folgende Werte können eingetragen werden:

- `START` gibt an, dass Governor gestartet wurde.
- `FORCE` gibt an, dass eine Anwendung zum Abmelden gezwungen wurde.
- `PRIORITY` gibt an, dass die Priorität einer Anwendung geändert wurde.
- `ERROR` gibt einen Fehler an.
- `WARNING` gibt eine Warnung an.
- `READCFG` gibt an, dass Governor die Konfigurationsdatei gelesen hat.
- `STOP` gibt an, dass Governor gestoppt wurde.
- `ACCOUNT` gibt die statistischen Daten für den Benutzereintrag der Anwendung an

  Folgende Datenfelder werden protokolliert:

  – `SQL-Berechtigungs-ID (authid)`
  – `Anwendungs-ID (appl_id)`
  – `Benutzer-CPU-Zeit (written_usr_cpu)`
  – `System-CPU-Zeit (written_sys_cpu)`
  – `Anwendungsverbindungszeit (appl_con_time)`
- `SCHEDULE` gibt an, dass die Prioritäten innerhalb der Agenten geändert wurden.

Da Standardwerte geschrieben werden, können Sie die Protokolldateien nach unterschiedlichen Aktionstypen abfragen. Das Feld *Nachricht* enthält andere vom Standard abweichende Informationen, die sich nach dem im Feld *Satzart* angegebenen Wert ändern. Der Eintrag `FORCE` oder `NICE` beispielsweise zeigt Informationen zu der Anwendung im Feld *Nachricht* an, während der Eintrag `ERROR` eine Fehlernachricht enthält.

Eine Beispielprotokolldatei könnte folgendermaßen aussehen:

```
1995-12-11 14.54.52    0 START      Database = TQTEST
1995-12-11 14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
1995-12-11 14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
1995-12-11 14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

Abfragen auf Governor-Protokolldateien

Jeder Governor-Dämon schreibt in eine eigene Protokolldatei. Mit dem Dienstprogramm `db2govlg` können Sie die Protokolldatei abfragen. Sie können die Protokolldateien einer einzelnen oder aller Datenbankpartitionen auflisten, sortiert nach Datum und Uhrzeit. Außerdem können Sie Abfragen auf der Grundlage des Protokollfelds *RecType* durchführen. `db2govlg` hat folgende Syntax:

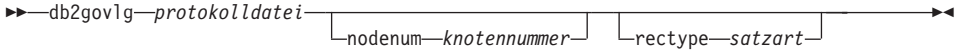


Abbildung 30. Syntax von db2govlg

Der Befehl hat die folgenden Parameter:

protokolldatei
Der Basisdateiname der Protokolldatei (en), die Sie abfragen möchten

nodenum **knotennummer**
Die Knotennummer der Datenbankpartition, auf der Governor aktiv ist

rectype **satzart**
Die Satzart, die Sie abfragen möchten. Es gibt folgende Satzarten:
- START
- READCFG
- STOP
- FORCE
- NICE
- ERROR
- WARNING
- ACCOUNT

Es gibt bei der Verwendung dieses Dienstprogramms keinerlei Berechtigungseinschränkungen. Dadurch können alle Benutzer abfragen, ob Governor ihre Anwendung geändert hat. Wenn Sie den Zugriff auf dieses Dienstprogramm beschränken möchten, können Sie die Gruppenberechtigungen für die Datei `db2govlg` ändern.

Ausführen von Governor und Leistung des Datenbankmanagers

Governor kann die Leistung des Datenbankmanagers beeinträchtigen, da er Momentaufnahmen des Datenbankmanagers anfordert. Wenn Governor zu viel CPU-Zeit in Anspruch nimmt, können Sie das Aktivierungsintervall verlängern, um den CPU-Bedarf von Governor zu verringern.

Kapitel 10. Skalieren der Konfiguration über das Hinzufügen von Prozessoren

Möglicherweise stellen Sie fest, dass die Kenndaten Ihrer Konfiguration für Ihre momentanen und geplanten Erfordernisse nicht ausreichen. Aus diesem Grund sollten Sie Aktionen in Betracht ziehen, die die Kapazität oder Leistung bzw. beide Werte in Ihrer Konfiguration vergrößern. Wenn Sie beispielsweise Behälter zu Ihrer Konfiguration hinzufügen, wird dadurch Ihre Kapazität zum Speichern von Daten gesteigert. Zudem kann dadurch die Leistung während der Verwendung von Dienstprogrammen verbessert werden (z. B. beim Laden von Daten). Andere Methoden zum Steigern der Kapazität oder der Leistung sind u. a.: Hinzufügen von Speicher und Prozessoren in einer Mehrprozessorumgebung oder einer Umgebung mit partitionierten Datenbanken.

In diesem Kapitel liegt der Schwerpunkt auf der Verbesserung der Leistung durch Vergrößern der Zahl an Prozessoren in Ihrer Konfiguration.

Sie sollten Ihre Konfiguration entsprechend der Beschreibung in diesem Kapitel skalieren, wenn Folgendes vorliegt:

- Sie hatten bisher eine Konfiguration mit einer Partition und einem Einzelprozessor, der bis zu seiner Kapazitätsgrenze ausgelastet wurde. Nun haben Sie beschlossen, die Konfiguration zu ändern, und Folgendes festgestellt:
  - Die beste Variante bei der Wahl einer neuen Umgebung ist eine symmetrische Mehrprozessorenkonfiguration (SMP). Möglicherweise haben Sie sich dafür entschieden, weil Sie die hohe Verarbeitungsleistung von mehreren Prozessoren nutzen möchten. Alle Prozessoren haben gemeinsam benutzte Speicher und Speichersystemressourcen. Sämtliche Prozessoren arbeiten in demselben System, so dass Sie sich keine Gedanken über Übertragungsleitungen zwischen Systemen, zusätzliches Personal zur Unterstützung neuer Systeme oder die Koordinierung von systemübergreifenden Aufgaben zu machen brauchen. DB2 Universal Database unterstützt diese Umgebung.
  - Die beste Variante bei der Wahl einer neuen Umgebung ist eine Konfiguration mit partitionierten Datenbanken. Möglicherweise haben Sie sich dafür entschieden, weil Sie die hohe Verarbeitungsleistung von mehreren Prozessoren nutzen möchten, die physisch voneinander getrennt sind. Jeder Prozessor hat seine eigenen Speicher und eigene Speichersystemressourcen, die er mit keinem anderen Prozessor gemeinsam benutzen muss. Zwar müssen Sie die oben genannten Fragen lösen (Leitungen, Personal und Aufgabenkoordinierung), jedoch bietet diese Variante

bestimmte Vorteile, unter anderem einen ausgewogenen Daten- und Benutzerzugriff über mehrere Systeme hinweg. DB2 Universal Database unterstützt diese Umgebung.

- Sie haben derzeit eine SMP-Konfiguration und haben vor, einen oder mehrere Prozessor(en) hinzuzufügen. In diesem Fall sind Ihnen die Überlegungen bereits vertraut, die bei einer solchen Umgebung anzustellen sind. Durch das Hinzufügen von zusätzlichen Prozessoren wird nur die Computerleistung Ihrer Umgebung gesteigert, ohne dass jedoch neue Überlegungen angestellt werden müssen. DB2 Universal Database unterstützt diese Umgebung.
- Sie haben derzeit eine Konfiguration mit partitionierten Datenbanken und möchten eine oder mehrere Datenbankpartition(en) hinzufügen. In diesem Fall sind Ihnen die Überlegungen bereits vertraut, die bei einer solchen Umgebung anzustellen sind. Durch das Hinzufügen von zusätzlichen Datenbankpartitionen wird nur die Computerleistung Ihrer Umgebung gesteigert, ohne dass jedoch neue Überlegungen außer zur Durchführung des Übergangs zu einer größeren Zahl an Partitionen angestellt werden müssen. DB2 Universal Database unterstützt diese Umgebung.

  Eine Art der Konfiguration mit einer partitionierten Datenbank besteht darin, SMP-Maschinen als Datenbankpartitionen zu verwenden. DB2 Universal Database unterstützt diese Umgebung.

Wenn Sie Ihr System durch Änderung der Umgebung skalieren, sollten Sie sich über die Auswirkungen dieser Änderung auf die Prozeduren in Ihrer Datenbank, beispielsweise auf das Laden von Daten sowie das Sichern oder Wiederherstellen der Datenbank, im Klaren sein.

Wenn Sie eine neue Datenbankpartition hinzufügen, können Sie eine Datenbank, die die neue Partition nutzt, erst dann wieder löschen oder erstellen, wenn die Aktion erledigt und der neue Server erfolgreich in das System integriert ist.

Hinzufügen von Prozessoren zu einer Maschine

Wenn die vorhandenen Prozessoren für den größten Teil der Zeit vollständig belegt sind, sollten Sie in Betracht ziehen, einen oder mehrere zusätzliche Prozessoren auf Ihrer Maschine zu installieren. Sie sollten die Konfigurationsparameter prüfen und gegebenenfalls aktualisieren, damit DB2-Datenbankmanager die neuen Prozessoren nutzen kann (Bestimmte Betriebssysteme, darunter Solaris, können Prozessoren online und offline variieren). Folgende Parameter werden zur Ermittlung der Prozessorenanzahl verwendet und müssen möglicherweise aktualisiert werden:

- „Grad der Parallelität (dft_degree)" auf Seite 516
- „Maximaler Grad der Parallelität bei Abfragen (max_querydegree)" auf Seite 541

- „Partitionsinterne Parallelität aktivieren (intra_parallel)” auf Seite 543

Prüfen Sie auch die Parameter für Anwendungen, die gegebenenfalls aktualisiert werden müssen. Weitere Informationen finden Sie in „Parallele Verarbeitung von Anwendungen” auf Seite 100.

Wenn Sie in einer Umgebung arbeiten, in der TCP/IP für die Kommunikation verwendet wird, sollten Sie sich Gedanken über den Wert der Registrierungsvariablen DB2TCPCONNMGRS machen. Weitere Informationen zu dieser Variablen finden Sie in „Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen” auf Seite 571.

Hinzufügen von Datenbankpartitionen zu einem System mit partitionierten Datenbanken

Sie können einem System mit einer partitionierten Datenbank Datenbankpartitionen hinzufügen, wenn es aktiv oder gestoppt ist. In den folgenden Abschnitten wird diese Aufgabe beschrieben. Da das Hinzufügen eines neuen Servers sehr zeitaufwendig sein kann, ist es vielleicht wünschenswert, dies zu tun, wenn der Datenbankmanager bereits aktiv ist. Diese Prozedur wird im Abschnitt „Hinzufügen von Datenbankpartitionen zu einem aktiven System” auf Seite 352 beschrieben.

Mit Hilfe des Befehls ADD NODE wird einem System eine Datenbankpartition hinzugefügt. Dieser Befehl kann folgendermaßen aufgerufen werden:

- als Option bei `db2start`
- über
  - den Befehl ADD NODE des Befehlszeilenprozessors
  - `sqleaddn`
  - `sqlepstart`

Wie Sie den Befehl aufrufen, hängt davon ab, ob Ihr System gestoppt ist (mit Hilfe von `db2start`) oder aktiv (mit Hilfe der übrigen Optionen).

Wenn Sie den Befehl ADD NODE zum Hinzufügen einer neuen Datenbankpartition verwenden, werden alle vorhandenen Datenbanken dieses Exemplars auf der neuen Datenbankpartition erstellt. Außerdem können Sie angeben, welche Behälter für temporäre Tabellenbereiche mit den erstellten Datenbanken verwendet werden sollen. Für die Behälter gibt es folgende Möglichkeiten:

- Es sind dieselben wie die für den Katalogknoten für jede Datenbank definierten Behälter (Standardeinstellung)
- Es sind dieselben wie die für eine andere Datenbankpartition definierten Behälter

- Sie sind noch nicht erstellt worden. Sie müssen jeder Datenbank mit Hilfe der Anweisung ALTER TABLESPACE Behälter für temporäre Tabellenbereiche hinzufügen, damit die Datenbank verwendet werden kann.

Eine Datenbank auf der neuen Partition kann erst dann Daten aufnehmen, wenn eine oder mehrere Knotengruppe(n) so geändert wird (werden), dass die neue Partition darin enthalten ist. Weitere Informationen zum Ändern von Knotengruppen finden Sie in „Hinzufügen und Entfernen von Datenbankpartitionen" auf Seite 365.

Anmerkung: Wenn auf dem System keine Datenbanken definiert sind und Sie DB2 Enterprise - Extended Edition auf einem UNIX-System ausführen, editieren Sie die Datei `db2nodes.cfg`, um eine neue Datenbankpartitionsdefinition hinzuzufügen. Verwenden Sie keine der folgenden Prozeduren, da sie nur beim Vorhandensein einer Datenbank verwendet werden können. Weitere Informationen zum Aktualisieren der Knotenkonfigurationsdatei finden Sie in „Ändern einer Knotengruppe" im Handbuch *Systemverwaltung: Konzept*.

Überlegungen zu Windows NT: Wenn Sie DB2 Enterprise - Extended Edition unter Windows NT verwenden und keine Datenbanken in dem Exemplar haben, sollten Sie das Datenbanksystem mit Hilfe des Befehls DB2NCRT skalieren. Informationen zu diesem Befehl finden Sie im Handbuch *Command Reference*. Wenn Sie jedoch bereits Datenbanken haben, sollten Sie den Befehl DB2START ADDNODE verwenden. Dadurch stellen Sie sicher, dass beim Skalieren des Systems für jede vorhandene Datenbank eine Datenbankpartition erstellt wird. Informationen zu dem Befehl DB2START und den unter Windows NT zu verwendenden Parametern finden Sie im Handbuch *Command Reference*. Unter Windows NT sollten Sie die Knotenkonfigurationsdatei (`db2nodes.cfg`) nie manuell editieren, da dies zu Inkonsistenzen in der Datei führen kann.

Hinzufügen von Datenbankpartitionen zu einem aktiven System

Sie können einem System mit partitionierte Datenbanken neue Partitionen hinzufügen, während es aktiv ist und Anwendungen mit Datenbanken verbunden sind. Allerdings steht der hinzugefügte Server erst dann allen Datenbanken zur Verfügung, wenn der Datenbankmanager gestoppt und erneut gestartet wird.

Gehen Sie wie folgt vor, um einem System mit mehreren Servern eine Datenbankpartition hinzuzufügen:

1. Wenn Sie die Datenbankpartition auf einem bereits in dem System vorhandenen Server erstellen möchten, fahren Sie mit dem nächsten Schritt fort. Ansonsten gehen Sie wie folgt vor:
   - Auf UNIX-Plattformen:

a. Installieren Sie den neuen Server. Dazu müssen Sie den Zugriff auf ausführbare Dateien herstellen (mit Hilfe von SFS-Mounts oder lokalen Kopien), Betriebssystemdateien mit den Dateien vorhandener Prozessoren synchronisieren, sicherstellen, dass auf das Verzeichnis `sqllib` als Shared File System zugegriffen werden kann und dafür sorgen, dass die relevanten Betriebssystemparameter (beispielsweise die maximale Prozessanzahl) auf die richtigen Werte eingestellt sind.

b. Registrieren Sie den Host-Namen auf dem Namens-Server oder in der Datei `hosts` im Verzeichnis `etc` auf allen Datenbankpartitionen.

- Auf Windows NT-Plattformen:

  a. Installieren Sie den neuen Server.

  b. Führen Sie den Befehl ADD NODE auf dem neuen Server aus. Mit diesem Befehl erstellen Sie lokal für jede bereits in dem System vorhandene Datenbank eine Datenbankpartition. Die Datenbankparameter für die neuen Datenbankpartitionen enthalten die Standardwerte, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Werte der Datenbankkonfigurationsparameter so, dass sie mit den in den anderen Datenbankpartitionen gefundenen Werten übereinstimmen.

  c. Fahren Sie mit Punkt 3 fort.

2. Führen Sie den Befehl DB2START für alle Datenbankpartitionen aus; geben Sie dabei die neuen Partitionswerte für die Parameter NODENUM, ADDNODE, HOSTNAME, PORT und NETNAME an. Auf einer Windows NT-Plattform müssen Sie außerdem die Parameter COMPUTER, USER und PASSWORD angeben. Weitere Informationen zm Befehl DB2START finden Sie im Handbuch *Command Reference*.

   Sie können auch die Quelle für die Definition von Behältern für temporäre Tabellenbereiche angeben, die mit den Datenbanken erstellt werden müssen. Werden keine Tabellenbereichsinformationen angegeben, werden die Definitionen von Behältern für temporäre Tabellenbereiche für jede Datenbank vom Katalogknoten abgerufen.

   Wenn der Befehl abgeschlossen ist, wird der neue Server gestoppt. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Server-Informationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD NODE (der durch die Angabe des Parameters ADDNODE aufgerufen wird) auf der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm abgeschlossen ist, wird der neue Server gestoppt.

3. Stoppen Sie den Datenbankmanager durch Ausführen des Befehls DB2STOP.

   Wenn Sie alle Datenbankpartitionen in einem System stoppen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition.

4. Starten Sie den Datenbankmanager durch Ausführen des Befehls DB2START.

   Nun wird die hinzugefügte Datenbankpartition gemeinsam mit dem restlichen System gestartet.

   Wenn alle Datenbankpartitionen des Systems aktiv sind, können Vorgänge durchgeführt werden, die das gesamte System betreffen, wie das Erstellen und Löschen einer Datenbank.

   Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal für alle Datenbankpartitions-Server ausführen, um auf die neue Datei `db2nodes.cfg` zuzugreifen.
5. Wahlfrei: Sichern Sie alle Datenbanken der neuen Datenbankpartition.
6. Wahlfrei: Verteilen Sie die Daten auf der neuen Datenbankpartition. Weitere Informationen finden Sie in „Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen" auf Seite 363.

Hinzufügen von Datenbankpartitionen zu einem gestoppten System

Sie können einem System mit partitionierte Datenbanken neue Datenbankpartitionen hinzufügen, während es gestoppt ist. Die neu hinzugefügte Datenbankpartition wird erst nach einem Neustart des Datenbankmanagers für alle Datenbanken verfügbar. Sie haben zwei Möglichkeiten. Entweder lassen Sie den Datenbankmanager die Knotenkonfigurationsdatei aktualisieren, oder Sie tun dies manuell. Die Vorbereitung ist in beiden Fällen identisch.

Anmerkung: Aktualisieren Sie die Knotenkonfigurationsdatei nicht manuell, während Sie auf Windows NT arbeiten. Verwenden Sie in diesem Fall den Datenbankmanager zum Aktualisieren dieser Datei (siehe nachfolgende Beschreibung).

Gehen Sie wie folgt vor, um einem System mit mehreren Servern eine Datenbankpartition hinzuzufügen:

1. Setzen Sie den Befehl DB2STOP ab, um alle Datenbankpartitionen zu stoppen.
2. Wenn Sie den Server auf einem bereits in dem System vorhandenen Prozessor erstellen möchten, fahren Sie mit dem nächsten Schritt fort. Ansonsten gehen Sie wie folgt vor:
   a. Auf UNIX-Plattformen:
      1) Installieren Sie den neuen Server. Dazu müssen Sie den Zugriff auf ausführbare Dateien herstellen (mit Hilfe von SFS-Mounts oder lokalen Kopien), Betriebssystemdateien mit den Dateien vorhandener Prozessoren synchronisieren, sicherstellen, dass auf das Verzeichnis `sqllib` als Shared File System zugegriffen werden kann

und dafür sorgen, dass die relevanten Betriebssystemparameter (beispielsweise die maximale Prozessanzahl) auf die richtigen Werte eingestellt sind.

2) Registrieren Sie den Host-Namen auf dem Namens-Server oder in der Datei `hosts` im Verzeichnis `etc` auf allen Datenbankpartitionen.

b. Auf Windows NT-Plattformen:

1) Installieren Sie den neuen Server.

2) Führen Sie den Befehl ADD NODE auf dem neuen Server aus. Mit diesem Befehl erstellen Sie lokal für jede bereits in dem System vorhandene Datenbank eine Datenbankpartition. Die Datenbankparameter für die neuen Datenbankpartitionen enthalten die Standardwerte, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Werte der Datenbankkonfigurationsparameter so, dass sie mit den in den anderen Datenbankpartitionen gefundenen Werten übereinstimmen.

3) Führen Sie den Befehl DB2START aus, um das Datenbanksystem zu starten. Beachten Sie, dass die Knotenkonfigurationsdatei (`db2nodes.cfg`) bei der Installation des neuen Servers aktualisiert worden ist und den Server bereits enthält.

4) Wahlfrei: Verteilen Sie Daten auf dem neuen Server. Weitere Informationen dazu finden Sie in „Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen" auf Seite 363.

c. Wenn der Datenbankmanager die Datei `db2nodes.cfg` für Sie aktualisieren soll, fahren Sie mit den Anweisungen in „Aktualisieren der Knotenkonfigurationsdatei durch den Datenbankmanager" fort.

Anmerkung: Unter Windows NT sollten Sie die Datei `db2nodes.cfg` nicht manuell editieren, da dies zu Inkonsistenzen in der Datei führen kann. Verwenden Sie in diesem Fall den Datenbankmanager zum Aktualisieren dieser Datei.

Wenn Sie die Datei `db2nodes.cfg` selbst aktualisieren wollen, fahren Sie mit den Anweisungen in „Manuelles Aktualisieren der Knotenkonfigurationsdatei" auf Seite 356 fort.

Aktualisieren der Knotenkonfigurationsdatei durch den Datenbankmanager

Nach dem Hinzufügen einer oder mehrerer neuer Datenbankpartitionen zu Ihrem partitionierten Datenbanksystem müssen Sie die Datei `db2nodes.cfg` aktualisieren, um den Prozess zum Verfügbarmachen der neuen Partition vollständig abzuschließen. Falls Sie beschlossen haben, die Aktualisierung der Knotenkonfigurationsdatei vom Datenbankmanager durchführen zu lassen, finden Sie nachfolgend detaillierte Informationen zur Vorgehensweise.

Anmerkung: Wenn Sie beschlossen haben, die Knotenkonfigurationsdatei manuell zu aktualisieren, können Sie die Informationen im verbleibenden Teil dieses Abschnitts ignorieren.

Fahren Sie mit der Prozedur folgendermaßen fort:

1. Führen Sie den Befehl DB2START auf der neuen Datenbankpartition aus, indem Sie die Parameter NODENUM, ADDNODE, HOSTNAME, PORT und NETNAME angeben. Auf einer Windows NT-Plattform müssen Sie außerdem die Parameter COMPUTER, USER und PASSWORD angeben. Weitere Informationen zm Befehl DB2START finden Sie im Handbuch *Command Reference*. Die Werte, die Sie für diese Parameter angeben, werden zum Aktualisieren der Knotenkonfigurationsdatei verwendet.

   Wenn der Befehl abgeschlossen ist, wird der neue Server gestoppt. Die Knotenkonfigurationsdatei wird erst dann mit den neuen Server-Informationen aktualisiert, wenn DB2STOP ausgeführt wird. Dadurch wird sichergestellt, dass der Befehl ADD NODE (der durch die Angabe des Parameters ADDNODE aufgerufen wird) auf der richtigen Datenbankpartition ausgeführt wird. Wenn das Dienstprogramm abgeschlossen ist, wird die neue Datenbankpartition gestoppt.
2. Setzen Sie den Befehl DB2STOP ab.

   Wenn Sie den Befehl DB2STOP absetzen, wird die Knotenkonfigurationsdatei aktualisiert und enthält nun die neue Datenbankpartition.
3. Setzen Sie den Befehl DB2START ab, um das Datenbanksystem zu starten.

   Anmerkung: Möglicherweise müssen Sie den Befehl DB2START zweimal absetzen, bis alle Datenbankpartitionen auf die neue Knotenkonfigurationsdatei zugreifen.
4. Wahlfrei: Sichern Sie alle Datenbanken der neuen Datenbankpartition.
5. Wahlfrei: Verteilen Sie Daten auf dem neuen Server. Weitere Informationen finden Sie in „Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen” auf Seite 363.

Manuelles Aktualisieren der Knotenkonfigurationsdatei

Nach dem Hinzufügen einer oder mehrerer neuer Datenbankpartitionen zu Ihrem partitionierten Datenbanksystem müssen Sie die Datei `db2nodes.cfg` aktualisieren, um den Prozess zum Verfügbarmachen der neuen Partition vollständig abzuschließen. Wenn Sie beschlossen haben, die Knotenkonfigurationsdatei manuell zu aktualisieren, finden Sie nachfolgend detaillierte Informationen zur Durchführung der manuellen Aktualisierung der Knotenkonfigurationsdatei. (Denken Sie daran, dass Sie die Knotenkonfigurationsdatei nicht manuell aktualisieren sollten, wenn Sie unter Windows NT arbeiten.)

Anmerkung: Wenn Sie beschlossen haben, die Aktualisierung der Knotenkonfigurationsdatei vom Datenbankmanager durchführen zu lassen, fahren Sie mit dem Abschnitt „Aktualisieren der Knotenkonfigurationsdatei durch den Datenbankmanager" auf Seite 355 fort.

Fahren Sie mit der Prozedur folgendermaßen fort:

1. Editieren Sie die Datei `db2nodes.cfg` und fügen Sie ihr die neue Datenbankpartition hinzu.
2. Setzen Sie folgenden Befehl ab, um den neuen Knoten zu starten:

   ```
   DB2START NODENUM nodenum
   ```

   Geben Sie die Nummer, die Sie der neuen Datenbankpartition zuordnen wollen, als Wert für *nodenum* an.
3. Wenn der neue Server eine logische Datenbankpartition sein soll (also nicht Knoten 0), müssen Sie mit Hilfe des Befehls **db2set** die Registrierungsvariable DB2NODE aktualisieren und dabei die Nummer der Datenbankpartition angeben, die Sie hinzufügen.
4. Führen Sie den Befehl ADD NODE in der neuen Datenbankpartition aus.

   Mit diesem Befehl erstellen Sie außerdem lokal für jede bereits in dem System vorhandene Datenbank eine Datenbankpartition. Die Datenbankparameter für die neuen Datenbankpartitionen enthalten die Standardwerte, und jede Datenbankpartition bleibt leer, bis Sie Daten dorthin versetzen. Aktualisieren Sie die Werte der Datenbankkonfigurationsparameter so, dass sie mit den in den anderen Datenbankpartitionen gefundenen Werten übereinstimmen.
5. Wenn der Befehl ADD NODE abgeschlossen ist, setzen Sie den Befehl DB2START ab, um die übrigen Datenbankpartitionen des Systems zu starten.

   Versuchen Sie nicht, systemweite Vorgänge wie das Erstellen und Löschen einer Datenbank durchzuführen, bevor alle Datenbankpartitionen erfolgreich gestartet worden sind.
6. Wahlfrei: Sichern Sie alle neuen Datenbankpartitionen auf dem neuen Server.
7. Wahlfrei: Verteilen Sie die Daten auf der neuen Datenbankpartition. Weitere Informationen finden Sie in „Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen" auf Seite 363.

Löschen einer Datenbankpartition von einem System

Sie können eine Datenbankpartition mit Hilfe des Befehls DB2STOP und dem Parameter DROP NODENUM oder über die API `sqlepstp` löschen. Zuvor müssen Sie sicherstellen, dass die zu löschende Datenbankpartition nicht gerade von einer Datenbank verwendet wird. Setzen Sie dazu den Befehl DROP NODE VERIFY ab.

Stellen Sie sicher, dass alle Transaktionen, die von der betreffenden Datenbankpartition koordiniert wurden, festgeschrieben oder zurückgesetzt wurden. Dazu ist möglicherweise eine Wiederherstellung nach einem Systemabsturz auf anderen Servern erforderlich.

Wenn Sie z. B. die Koordinatordatenbankpartition (d. h. den Koordinatorknoten) löschen und vor dem Löschen des Koordinatorknotens eine andere an einer Transaktion beteiligte Datenbankpartition abgestürzt war, ist die abgestürzte Datenbankpartition nicht in der Lage, den Koordinatorknoten nach dem Ergebnis unbestätigter Transaktionen abzufragen.

Gehen Sie wie folgt vor, um eine Datenbankpartition von einem System mit einer partitionierten Datenbank zu löschen:

1. Verteilen Sie die Daten aller Datenbanken auf diesem Knoten um. Dadurch wird die Anforderung erfüllt, dass die zu löschende Datenbankpartition nicht gerade von einer Datenbank verwendet wird. Weitere Informationen finden Sie in „Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen" auf Seite 363.
2. Setzen Sie den Befehl DROP NODE VERIFY oder die API `sqledrpn` ab, um sicherzustellen, dass der Server nicht im Gebrauch ist.

Fahren Sie je nach der angezeigten Nachricht mit Schritt 3 oder 4 fort.

3. Wird die Nachricht SQL6034W (Der Knoten wird von keiner Datenbank verwendet.) angezeigt, gehen Sie folgendermaßen vor:
   a. Setzen Sie den Befehl DB2STOP mit dem Parameter DROP NODENUM ab, um die Datenbankpartition zu löschen. Nachdem der Befehl erfolgreich abgeschlossen ist, wird das System gestoppt.
   b. Wenn Sie den Datenbankmanager starten möchten, tun Sie das mit Befehl DB2START.
4. Wird die Nachricht SQL6035W (Der Knoten wird von der Datenbank verwendet.) angezeigt, gehen Sie folgendermaßen vor:
   a. Verwenden Sie den Befehl REDISTRIBUTE NODEGROUP zum Umverteilen der Daten von der zu löschenden Datenbankpartition auf andere Datenbankpartitionen vom Aliasnamen der Datenbank, wie in der Nachricht SQL6035W angezeigt. Solange dies nicht geschehen ist, können Sie die Datenbankpartition nicht löschen.
   b. Löschen Sie die auf der Datenbankpartition definierten Ereignismonitore.
   c. Kehren Sie zu Schritt 2 zurück, und fahren Sie fort.

Probleme beim Hinzufügen von Knoten zu einer partitionierten Datenbank

Wenn einer partitionierten Datenbank, die mindestens einen temporären Systemtabellenbereich mit einer anderen Seitengröße als der Standardseitengröße (4 KB) hat, Knoten hinzugefügt werden, können die Fehlernachricht „SQL6073N Das Hinzufügen von Knoten ist fehlgeschlagen" und ein SQLCODE-Wert auftreten. Dies ist darauf zurückzuführen, dass bei der Erstellung des Knotens nur der Puffpool IBMDEFAULTBP mit einer Seitengröße von 4 KB existiert.

Sie können z. B. den Befehl **db2start** verwenden, um der aktuellen partitionierten Datenbank einen Knoten hinzuzufügen:

```
DB2START NODENUM 2 ADDNODE HOSTNAME newhost PORT 2
```

Wenn die partitionierte Datenbank über temporäre Systemtabellenbereiche mit der Standardseitengröße verfügt, wird die folgende Nachricht zurückgegeben:

```
SQL6075W Die Operation START DATABASE MANAGER wurde erfolgreich zum
Knoten hinzugefügt.
Der Knoten ist erst aktiv, nachdem alle Knoten gestoppt und erneut
gestartet wurden.
```

Wenn jedoch die partitionierte Datenbank temporäre Systemtabellenbereiche mit einer anderen Größe als der Standardseitengröße hat, wird die folgende Nachricht zurückgegeben:

```
SQL6073N Das Hinzufügen von
Knoten ist fehlgeschlagen. SQLCODE = "<-902>"
```

In einem ähnlichen Beispiel können Sie den Befehl ADD NODE verwenden, nachdem die Datei `db2nodes.cfg` manuell mit der neuen Knotenbeschreibung aktualisiert wurde. Nachdem die Datei editiert und der Befehl ADD NODE für eine partitionierte Datenbank ausgeführt wurde, die temporäre Systemtabellenbereiche mit der Standardseitengröße hat, wird folgende Nachricht zurückgegeben:

```
DB20000I Der Befehl ADD NODE wurde erfolgreich ausgeführt.
```

Wenn jedoch die partitionierte Datenbank temporäre Systemtabellenbereiche mit einer anderen Größe als der Standardseitengröße hat, wird die folgende Nachricht zurückgegeben:

```
SQL6073N Das Hinzufügen von
Knoten ist fehlgeschlagen. SQLCODE = "<-902>"
```

Eine Möglichkeit, die oben dargestellten Probleme zu vermeiden, ist die Ausführung des folgenden Befehls:

```
DB2SET DB2_HIDDENBP=16
```

Dieser Befehl muss vor dem Befehl **db2start** oder ADD NODE abgesetzt werden. Durch diese Registrierungsvariable kann DB2 verdeckte Pufferpools von je 16 Seiten zuordnen, die jeweils eine von der Standardgröße abweichende Seitengröße verwenden. Dadurch kann die Operation ADD NODE erfolgreich durchgeführt werden.

Eine weitere Möglichkeit zur Umgehung dieses Problems ist die Angabe der Klausel WITHOUT TABLESPACES beim Befehl ADD NODE oder **db2start**. Danach müssen Sie die Pufferpools mit der Anweisung CREATE BUFFERPOOL erstellen und die temporären Systemtabellenbereiche mit der Anweisung ALTER TABLESPACE dem Pufferpool zuordnen.

Wenn einer vorhandenen Knotengruppe, die mindestens einen Tabellenbereich mit einer anderen Seitengröße als der Standardseitengröße (4 KB) hat, Knoten hinzugefügt werden, kann folgende Fehlernachricht angezeigt werden: „SQL0647N Pufferpool "" ist zur Zeit nicht aktiv.". Dieser Fehler tritt auf, weil die auf dem neuen Knoten erstellten Pufferpools mit einer vom Standard abweichenden Seitengröße für die Tabellenbereiche nicht aktiviert wurden.

Sie können z. B. die Anweisung ALTER NODEGROUP verwenden, um einen Knoten zu einer Knotengruppe hinzuzufügen:

```
DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2)
```

Wenn die Knotengruppe Tabellenbereiche mit der Standardseitengröße hat, wird die folgende Nachricht zurückgegeben:

```
Es ist erforderlich, die Knotengruppe umzuverteilen, um die Datenpartitionierung für
Objekte in der Knotengruppe "<name-der-knotengruppe>" zu ändern, damit hinzugefügte
Knoten aufgenommen oder gelöschte Knoten entfernt werden können.
```

Wenn die Knotengruppe jedoch Tabellenbereiche mit einer anderen Größe als der Standardseitengröße hat, wird die folgende Nachricht zurückgegeben:

```
SQL0647N Pufferpool "" ist zur Zeit nicht aktiv.
```

Eine Möglichkeit zum Umgehen dieses Problems besteht darin, Pufferpools für jede Seitengröße zu erstellen und sie dann wieder mit der Datenbank zu verbinden, bevor Sie die Anweisung ALTER NODEGROUP absetzen:

```
DB2START
CONNECT TO mpp1
CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
CONNECT RESET
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2)
```

Eine zweite Umgehungsmöglichkeit für dieses Problem besteht in der Ausführung des folgenden Befehls:

```
DB2SET DB2_HIDDENBP=16
```

Dies muss erfolgen, bevor Sie den Befehl **db2start** und die Anweisungen CONNECT und ALTER NODEGROUP absetzen.

Ein weiteres Problem kann auftreten, wenn die Anweisung ALTER TABLESPACE verwendet wird, um einem Tabellenbereich einen Knoten hinzuzufügen. Beispiel:

```
DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2) WITHOUT TABLESPACES
ALTER TABLESPACE ts1 ADD ('ts1') ON NODE (2)
```

Diese Reihe von Befehlen und Anweisungen generiert die Fehlernachricht SQL0647N (nicht die erwartete Nachricht SQL1759W).

Damit diese Änderung korrekt abgeschlossen wird, sollten Sie die Verbindung zur Datenbank wiederherstellen, nachdem die Anweisung ALTER NODEGROUP... WITHOUT TABLESPACES ausgeführt wurde.

```
DB2START
CONNECT TO mpp1
ALTER NODEGROUP ng1 ADD NODE (2) WITHOUT TABLESPACES
CONNECT RESET
CONNECT TO mpp1
ALTER TABLESPACE ts1 ADD ('ts1') ON NODE (2)
```

Eine weitere Umgehungsmöglichkeit für dieses Problem besteht in der Ausführung des folgenden Befehls:

```
DB2SET DB2_HIDDENBP=16
```

Dies muss erfolgen, bevor Sie den Befehl **db2start** und die Anweisungen CONNECT, ALTER NODEGROUP und ALTER TABLESPACE absetzen.

Kapitel 11. Umverteilen von Daten auf Datenbankpartitionen

Die Umverteilung von Daten spielt nur in einer Umgebung mit partitionierten Datenbanken eine Rolle. Wenn Sie eine Datenbankumgebung mit einer Einzelpartition betreiben, benötigen Sie die in diesem Kapitel enthaltenen Informationen nicht.

Zur Versetzung von Daten zwischen den Datenbankpartitionen in einer vorhandenen Knotengruppe verwenden Sie das Dienstprogramm zur Datenumverteilung (Data Redistribution). Das Dienstprogramm dient zu folgenden Zwecken:

- Gleichmäßiges Verteilen von Datenvolumen und Verarbeiten von Ladeoperationen zwischen Datenbankpartitionen

  Dies ist sinnvoll, wenn Sie eine Datenbanktabelle haben, in der regelmäßig auf alle Daten zugegriffen wird.
- Ungleichmäßiges Verteilen der Daten auf die Datenbankpartitionen

  Dies ist sinnvoll, wenn Sie eine Datenbanktabelle haben, in der nur auf einige Daten regelmäßig zugegriffen wird. In diesem Fall sollten Sie die Daten der Tabelle so umverteilen, dass die Daten, auf die seltener zugegriffen wird, in einer kleinen Anzahl von Datenbankpartitionen in der Knotengruppe gespeichert werden, und die Daten, auf die häufig zugegriffen wird, auf eine größere Anzahl von Partitionen verteilt werden. Dadurch würden die Zugriffsleistung und der Durchsatz für die am häufigsten ausgeführten Anwendungen verbessert.

Der Befehl REDISTRIBUTE NODEGROUP dient zum Aufrufen des Dienstprogramms zur Datenumverteilung. Ausführliche Angaben zur Syntax dieses Befehls finden Sie im Handbuch *Command Reference*.

Zur Erhaltung der Kollokation (Zusammenfassung) von Tabellen wird diese Operation auf alle Tabellen in einer Knotengruppe angewandt, so dass die Umverteilung auf Knotengruppenebene und nicht auf Tabellenebene stattfindet.

Zur Herstellung der gewünschten Datenverteilung verwendet das Dienstprogramm eine Partitionierungszuordnung, um die Zeilen der Tabellen zwischen den Datenbankpartitionen der Knotengruppe zu versetzen. Abhängig von der angegebenen Option kann das Dienstprogramm eine Zielpartitionierungszuordnung generieren oder eine vorhandene Partitionierungszuordnung als Eingabe verwenden.

Anmerkungen:

1. Sie sollten einen Wert für die Größe der Protokolldateien angeben, die den Anforderungen an Protokollspeicherbereich, die Sie für die Umverteilungsoperation veranschlagen, entspricht. Außerdem müssen Sie sicherstellen, dass die Größe des Protokolls ausreicht, um die INSERT- und DELETE-Operationen aufzunehmen, die in jeder von der Umverteilung betroffenen Datenbankpartition ausgeführt werden.
2. Wenn Sie die Daten in einer Knotengruppe umverteilen wollen, die replizierte Übersichtstabellen enthält, müssen Sie diese Tabellen zunächst löschen (DROP), die Knotengruppe umverteilen und anschließend die Tabellen erneut erstellen. Eine Knotengruppe, die replizierte Übersichtstabellen enthält, kann nicht umverteilt werden.

Partitionierung von Daten

Standardmäßig nimmt das Dienstprogramm zur Datenumverteilung an, dass jeder Hash-Partition dieselbe Anzahl von Zeilen durch das Hash-Verfahren zugeordnet wird. Daher partitioniert es die Hash-Partitionen gleichmäßig auf alle Datenbankpartitionen der Knotengruppe. Wenn durch das Hash-Verfahren nicht dieselbe Anzahl von Zeilen jeder Hash-Partition zugeordnet wird, können Sie mit Hilfe einer *Verteilungsdatei* die aktuelle Verteilung angeben. Diese Datei enthält für jede der 4 096 Hash-Partitionen einen Wert. Jeder Wert wird als *Gewichtung* für die entsprechende Hash-Partition verwendet. Das Dienstprogramm zur Datenumverteilung generiert eine Zielpartitionierungszuordnung, in der alle Datenbankpartitionen annähernd dieselbe Gewichtung haben. Das heißt, mit Hilfe der Verteilungsdatei kann eine gleichmäßige Datenverteilung erreicht werden, auch wenn die Werte der Daten eine ungleichmäßige Verteilung aufweisen.

Das Programm `AutoLoader` kann mit der Option ANALYZE zur Erstellung einer Datenverteilungsdatei verwendet werden. Sie können diese Datei als Eingabe für das Dienstprogramm zur Datenumverteilung verwenden. Weitere Informationen zum Dienstprogramm `AutoLoader` finden Sie im Handbuch *Versetzen von Daten Dienstprogramme und Referenz*.

Alternativ können Sie auch die SQL-Funktionen PARTITION und NODENUMBER verwenden, um die aktuelle Datenverteilung über die Hash-Partitionen bzw. Datenbankpartitionen zu bestimmen. (Die Funktion PARTITION wird zur Bestimmung der Verteilung über Hash-Partitionen verwendet.) Aus diesen Informationen können Sie sowohl eine Verteilungsdatei als auch eine Zielpartitionierungszuordnung ableiten.

Gehen Sie wie folgt vor, um beispielsweise festzustellen, ob es Datenbankpartitionen gibt, die aufgrund von ungleichmäßiger Datenverteilung eine außergewöhnlich große Zahl von Zeilen enthalten, und welche Partitionen dies sind:

```
SELECT PARTITION(column_name), COUNT(*) FROM table_name
   GROUP BY PARTITION(column_name)
   ORDER BY PARTITION(column_name) DESC
FETCH FIRST 100 ROWS ONLY
```

Sie sollten sicherstellen, dass `table_name` (Tabellenname) die größte Tabelle und `column_name` (Spaltenname) eine geeignete Spalte in dieser Tabelle ist.

Hinzufügen und Entfernen von Datenbankpartitionen

Datenbankpartitionen werden mit Hilfe der Anweisung ALTER NODEGROUP einer Knotengruppe hinzugefügt bzw. aus ihr entfernt. Wenn Datenbankpartitionen hinzugefügt werden, müssen diese Partitionen bereits in der Knotenkonfigurationsdatei definiert sein.

Nach Verwendung der Anweisung ALTER NODEGROUP wird eine neue Partitionierungszuordnung erstellt. Diese neue Partitionierungszuordnung kann zur Zielpartitionierungszuordnung bei der Verwendung des Dienstprogramms zur Datenumverteilung werden. (Eine andere Möglichkeit wäre, die Partitionierungszuordnung selbst zu erstellen.)

Wenn Sie die Anweisung ALTER NODEGROUP mit der Klausel WITHOUT TABLESPACES verwenden, müssen Sie vor der Umverteilung der Daten einer neuen Datenbankpartition (bzw. Partitionen) neue Tabellenbereichsbehälter hinzufügen. Weitere Informationen zur Anweisung ALTER NODEGROUP finden Sie im Handbuch *SQL Reference*.

Angeben einer Zielpartitionierungszuordnung

Das Dienstprogramm zur Datenumverteilung verwendet zur Erfüllung seiner Funktion eine Partitionierungszuordnung. Es kann eine eigene Zielpartitionierungszuordnung erstellen, oder Sie können selbst eine zur Verwendung durch das Dienstprogramm bereitstellen. Wenn Sie eine Partitionierungszuordnung erstellen, bestimmt der Eintrag bzw. die Einträge die Art der Knotengruppe, die sich aus der Datenumverteilung ergibt:

- 1 Eintrag für eine Knotengruppe mit einer Einzelpartition
- 4 096 Einträge für eine Knotengruppe mit mehreren Partitionen

Wenn die Zielpartitionierungszuordnung mehr als eine Datenbankpartition enthält, muss für alle Tabellen in der Knotengruppe derselbe Partitionierungsschlüssel definiert sein.

Die Zielpartitionierungszuordnung kann nur Datenbankpartitionsnummern enthalten, die in der Katalogtabelle SYSCAT.NODEGROUPDEF definiert sind, ausgenommen solche Datenbanken, die einen IN_USE-Wert von 'T' haben. ('T' bedeutet, dass die Partition nicht in der Zielpartitionierungszuordnung ist.) Alle Datenbankpartitionen, die einen IN_USE-Wert von 'D' (d. h. 'zu löschen') haben und nicht in der Zielpartitionierungszuordnung auftreten, werden gelöscht (DROP), wenn die Umverteilungsoperation erfolgreich beendet wurde.

Verfahren der Datenumverteilung auf Datenbankpartitionen

Die Operation der Datenumverteilung findet in der Tabellengruppe der angegebenen Knotengruppe einer Datenbank statt. (Die Anwendung muss mit der Datenbank in der Katalogdatenbankpartition verbunden werden, bevor die Operation ausgeführt werden kann.) Das Dienstprogramm verwendet die Quellenpartitionierungszuordnung und die Zielpartitionierungszuordnung, um zu ermitteln, welche Hash-Partitionen einer neuen Speicherposition (d. h. einer neuen Datenbankpartitionsnummer) zugeordnet wurden. Alle Zeilen, die zu einer Partition mit einer neuen Speicherposition gehören, werden von der in der Quellenpartitionierungszuordnung angegebenen Datenbankpartition in die in der Zielpartitionierungszuordnung angegebenen Datenbankpartition versetzt.

Das Dienstprogramm zur Datenumverteilung führt folgende Operationen aus:

1. Es empfängt eine neue Partitionierungszuordnungs-ID für die Zielpartitionierungszuordnung und fügt diese in die Katalogsicht SYSCAT.PARTITIONMAPS ein.
2. Es aktualisiert die Spalte REBALANCE_PMAP_ID in der Katalogsicht SYSCAT.NODEGROUPS für die Knotengruppe mit der neuen Partitionierungszuordnungs-ID.
3. Es fügt die neuen Datenbankpartitionen der Katalogsicht SYSCAT.NODEGROUPDEF hinzu.
4. Es setzt den Wert der Spalte IN_USE in der Katalogsicht SYSCAT.NODEGROUPDEF auf 'D' (DROP) für jede Datenbankpartition, die zu löschen ist.
5. Es führt eine COMMIT-Operation für die Katalogaktualisierungen aus.
6. Es erstellt Datenbankdateien für alle neuen Datenbankpartitionen.
7. Es verteilt die Daten tabellenweise für jede Tabelle in der Knotengruppe um. Dies wird im Abschnitt „Verfahren der Datenverteilung in Tabellen" auf Seite 367 beschrieben.
8. Es löscht Datenbankdateien und Einträge in der Katalogsicht SYSCAT-.NODEGROUPDEF für Datenbankpartitionen, die zuvor als zu löschen ('D') markiert wurden.

9. Es aktualisiert den Knotengruppendatensatz in der Katalogsicht SYSCAT-.NODEGROUPS, um den Wert von PMAP_ID auf den Wert von REBALANCE_PMAP_ID und den Wert von REBALANCE_PMAP_ID auf NULL zu setzen.
10. Es löscht die alte Partitionierungszuordnung aus der Katalogsicht SYSCAT.PARTITIONMAPS.
11. Es führt eine COMMIT-Operation für alle Änderungen aus.

Verfahren der Datenverteilung in Tabellen

Bei der Datenumverteilung in einer Tabelle arbeitet das Dienstprogramm in folgenden Schritten:

1. Es sperrt die Zeile für die Tabelle in der Katalogtabelle SYSTABLES.
2. Es macht alle Pakete, die mit dieser Tabelle arbeiten, ungültig. Die der Tabelle zugeordnete Partitionierungszuordnungs-ID wird geändert, da die Tabelle neu verteilt wird. Da die Pakete ungültig gemacht werden, muss der Compiler die neuen Partitionierungsinformationen für die Tabelle abrufen und entsprechende Pakete generieren.
3. Es sperrt die Tabelle im Exklusivmodus (Exclusive).
4. Es verteilt die Daten in der Tabelle mit Hilfe von DELETE- und INSERT-Anweisungen neu.
5. Wenn die Umverteilungsoperation erfolgreich ist, geschieht folgendes:
   a. Es setzt einen COMMIT-Befehl für die Tabelle ab.
   b. Es setzt die Operation mit der nächsten Tabelle in der Knotengruppe fort.

   Wenn die Operation fehlschlägt, bevor die Daten der Tabelle vollständig umverteilt sind, reagiert das Dienstprogramm wie folgt:
   a. Es setzt einen ROLLBACK-Befehl für Aktualisierungen an der Tabelle ab.
   b. Es beendet die gesamte Umverteilungsoperation und gibt eine Fehlernachricht aus.

Das Einschätzen des Platzbedarfs für Protokolle beim Verteilen von Daten ist wichtig. Die Größe des Protokolls muss ausreichend sein, damit die INSERT- und DELETE-Operationen darin aufgenommen werden können, die in jeder von der Umverteilung betroffenen Datenbankpartition ausgeführt werden. Die größten Protokollierungsanforderungen gelten entweder für die Datenbankpartition, die die meisten Daten verliert, oder für die Datenbankpartition, die die meisten Daten dazuerhält. Wenn Sie eine Veränderung hin zu einer größeren Zahl an Datenbankpartitionen vornehmen, dient das Verhältnis der momentanen Datenbankpartitionen zur neuen Zahl an Datenbankpartitionen zum Ermitteln der Zahl der INSERT- und DELETE-Operationen.

Wenn Sie beispielsweise die Zahl der Datenbankpartitionen von vier auf fünf erhöhen, werden von ca. zwanzig Prozent der vier ursprünglichen Datenbankpartitionen Daten in die neue Datenbankpartition verschoben. Das bedeutet, dass für die vier ursprünglichen Datenbankpartitionen zwanzig Prozent an DELETE-Operationen auf der Basis der Gesamtdatenmenge in jeder Datenbankpartition durchgeführt werden. In der neuen Datenbankpartition werden alle INSERT-Operationen durchgeführt (d. h. ein Wert, der gleichen Anzahl von DELETE-Operationen in allen vier ursprünglichen Datenbankpartitionen entspricht).

Beim oben genannten Beispiel wird von einer gleichmäßigen Verteilung der Daten ausgegangen. Es kann aber auch Fälle geben, in denen die Daten nicht gleichmäßig verteilt sind, wie beispielsweise, wenn im Partitionierungsschlüssel eine große Zahl an NULL-Werten enthalten ist. In diesem Fall würden sich alle diese Zeilen im alten Partitionierungsschema in einer bestimmten Datenbankpartition und im neuen Partitionierungsschema in einer anderen Datenbankpartition befinden. Dadurch kann die Menge an Protokollspeicherbereich, die für diese beiden Datenbankpartitionen erforderlich ist, möglicherweise die Menge deutlich übersteigen, die unter der Annahme einer gleichmäßigen Verteilung berechnet wurde.

Beim Durchführen der Berechnungen müssen Sie den Prozentsatz der Änderungen (z. B. zwanzig Prozent) mit der Größe der größten Tabelle multiplizieren. Dies ist erforderlich, weil die Umverteilung aller Tabellen jeweils als einzige Transaktion erfolgt.

Anmerkung: Die größte Tabelle kann jedoch über eine gleichmäßige Verteilung verfügen, während die zweitgrößte (beispielsweise) eine oder mehrere stark vergrößerte Datenbankpartitionen enthält. In einem solchen Fall sollten Sie die zweite Tabelle anstatt der ersten verwenden.

Wenn Sie die Höchstmenge an Daten berechnet haben, für die in einer Datenbankpartition INSERT- und DELETE-Operationen durchgeführt werden, verdoppeln Sie diese Zahl, um den Spitzenwert für die Größe der aktiven Protokolldatei zu ermitteln. Wenn dieser Wert die Begrenzung von 32 GB für die aktive Protokolldatei überschreitet, muss die Umverteilung von Daten schrittweise erfolgen. Es gibt ein Dienstprogramm mit dem Namen „makepmap", das zum Generieren einer Reihe von Zielpartitionszuordnungen für jeden Schritt verwendet werden kann.

Beheben von Fehlern bei der Umverteilung

Nach dem Starten der Umverteilungsoperation wird eine Datei in das Unterverzeichnis `redist` des Verzeichnisses `sqllib` geschrieben. In dieser Statusdatei werden alle Operationen, die in Datenbankpartitionen ausgeführt werden, die Namen der Tabellen, die umverteilt wurden, und der Beendigungsstatus der Operation aufgelistet. Wenn eine Tabelle nicht umverteilt werden kann, wird der Name der Tabelle und der betreffende SQLCODE in der Datei aufgeführt. Wenn die Umverteilungsoperation aufgrund eines falschen Eingabeparameters nicht beginnen kann, wird die Datei nicht geschrieben und ein SQLCODE zurückgegeben.

Für die Datei gilt folgende Namenskonvention:

```
Datenbankname.Knotengruppenname.Zeitmarke   (für UNIX-Plattformen)
Datenbankname\Knotengruppenname\Datum\Uhrzeit  (für andere Plattformen)
```

Anmerkung: Auf Nicht-UNIX-Plattformen werden nur die ersten acht (8) Byte des `Knotengruppennamens` verwendet.

Wenn die Operation zur Datenumverteilung fehlschlägt, sind die Daten einiger Tabellen eventuell bereits neu verteilt, während andere Tabellen noch nicht bearbeitet wurden. Dies hat seine Ursache darin, dass die Datenumverteilung jeweils für eine Tabelle gleichzeitig durchgeführt wird. Sie haben zwei Möglichkeiten der Fehlerbehandlung:

- Verwenden der Option CONTINUE, um die Operation zur Datenumverteilung für die übrigen Tabellen fortzusetzen.
- Verwenden der Option ROLLBACK, um die Datenumverteilung rückgängig zu machen und die umverteilten Tabellen in ihren ursprünglichen Zustand zurückzuversetzen. Die ROLLBACK-Operation kann dabei ebenso lange dauern wie die ursprüngliche Umverteilungsoperation.

Bevor Sie eine dieser Optionen verwenden können, muss zuvor eine Operation zur Datenumverteilung fehlgeschlagen sein, so dass die Spalte REBALANCE_PMID in der Katalogtabelle SYSNODEGROUPS auf einen Nicht-NULL-Wert gesetzt ist.

Falls Sie die Statusdatei versehentlich löschen, haben Sie immer noch die Möglichkeit, eine CONTINUE-Operation auszuführen.

Datenumverteilung und andere Operationen

Die folgenden Operationen können an Objekten der Knotengruppe ausgeführt werden, während das Dienstprogramm aktiv ist. Sie können jedoch nicht an der Tabelle ausgeführt werden, die momentan umverteilt wird. Mögliche Operationen:

- Erstellen von Indizes für andere Tabellen. Die Anweisung CREATE INDEX verwendet die Partitionierungszuordnung der betroffenen Tabelle.
- Löschen anderer Tabellen. Die Anweisung DROP TABLE verwendet die Partitionierungszuordnung der betroffenen Tabelle.
- Löschen von Indizes für andere Tabellen. Die Anweisung DROP INDEX verwendet die Partitionierungszuordnung der betroffenen Tabelle.
- Abfragen anderer Tabellen
- Aktualisieren anderer Tabellen
- Erstellen neuer Tabellen in einem in der Knotengruppe definierten Tabellenbereich. Die Anweisung CREATE TABLE verwendet die Zielpartitionierungszuordnung.
- Erstellen von Tabellenbereichen in der Knotengruppe

Folgende Operationen stehen während der Ausführung des Dienstprogramms nicht zur Verfügung:

- Eine weitere Umverteilungsoperation für die Knotengruppe
- Eine Anweisung ALTER TABLE für irgendeine Tabelle in der Knotengruppe
- Löschen (DROP) der Knotengruppe
- Ändern der Knotengruppe

Abschließen der Datenumverteilung

Nach dem Abschluss der Umverteilung von Daten in einer Knotengruppe ist die Ausführung des Befehls RUNSTATS sehr zu empfehlen, um die zu den eventuell umverteilten Tabellen gehörenden Statistikdaten zu aktualisieren.

Weitere Informationen zum Befehl RUNSTATS finden Sie im Handbuch *Command Reference*.

Kapitel 12. Durchführen von Vergleichstests

Vergleichstests (Benchmark-Tests) bilden einen natürlichen Bestandteil des Entwicklungszyklus für Anwendungen. Sie erfordern die Zusammenarbeit von Anwendungsentwicklern und Datenbankadministratoren (DBAs) und sollten für eine Anwendung durchgeführt werden, um Daten über die Leistung zu erhalten und Ansätze zur Leistungsoptimierung zu ermitteln. Unter der Annahme, dass der Code einer Anwendung bereits mit größtmöglicher Effizienz arbeitet, können weitere Leistungsvorteile durch die Optimierung der Konfigurationsparameter der Datenbank und des Datenbankmanagers und sogar der Anwendungsparameter realisiert werden, um den Anforderungen der Anwendung entgegenzukommen.

Es gibt mehrere verschiedene Arten von Vergleichstests. Ein Vergleichstest der Art *Transaktion pro Sekunde* könnte Anhaltspunkte über die Leistungskapazität des Datenbankmanagers unter bestimmten, eingeschränkten Laborbedingungen liefern. Ein *Anwendungsvergleichstest* würde dieselbe Leistungskapazität testen, aber unter Bedingungen, die denen, unter denen die Anwendungen ausgeführt wird, wenn sie implementiert ist, weit näher kommen. Vergleichstests zum Zweck der Optimierung von Konfigurationsparametern werden unter diesen „Realbedingungen" durchgeführt und machen die wiederholte Ausführung von SQL aus der Anwendung mit variierenden Parameterwerten erforderlich, bis die Anwendung mit der höchstmöglichen Effizienz arbeitet.

Die in diesem Abschnitt beschriebenen Vergleichstestmethoden wurden speziell auf die Optimierung der Konfigurationsparameter ausgerichtet. Darüber hinaus kann derselbe Grundansatz auch für die Optimierung anderer, die Leistung beeinflussender Faktoren herangezogen werden, wie zum Beispiel:
- SQL-Anweisungen
- Indizes
- Tabellenbereichskonfiguration
- Anwendungscode
- Hardwarekonfiguration

Vergleichstests geben Aufschluss darüber, wie der Datenbankmanager unter unterschiedlichen Bedingungen reagiert. Es könnten Szenarios entwickelt werden, um die Behandlung gegenseitiger Sperren, die verschiedenen Methoden zum Laden von Daten, die Transaktionsgeschwindigkeit bei wachsenden Benutzerzahlen und auch die Auswirkungen der Verwendung eines neuen Produkt-Release auf die Anwendung zu testen.

Die folgenden Themen werden behandelt:
- „Methoden für Vergleichstests"
- „Vorbereiten von Vergleichstests" auf Seite 373
- „Erstellen eines Vergleichstestprogramms" auf Seite 375
- „Ausführen der Vergleichstests" auf Seite 382.

Methoden für Vergleichstests

Dieser Ansatz für Vergleichstests basiert auf der wissenschaftlichen Methode. Es wird eine reproduzierbare Umgebung erstellt, in der derselbe Test unter gleichen Ausführungsbedingungen vergleichbare Ergebnisse liefert.

Vergleichstests können auch mit der Ausführung einer Testanwendung in einer Normalumgebung beginnen. In dem Maße, wie ein Leistungsproblem eingegrenzt wird, können spezialisierte Anwendungsbeispiele entwickelt werden, um den Wirkungsbereich einer Funktion, die diesem Test unterzogen wird und unter Beobachtung steht, weiter einzuschränken. Die spezialisierten Anwendungsbeispiele brauchen nicht eine gesamte Anwendung zu emulieren, um wertvolle Informationen liefern zu können. Es empfiehlt sich, mit einfachen Messungen zu beginnen und die Komplexität der Methoden nur dann zu erhöhen, wenn dies gerechtfertigt ist.

Gute Vergleichstests (oder Messungen) besitzen folgende Merkmale:
- Jeder Test ist wiederholbar.
- Jede Wiederholung eines Tests wird in einem identischen Systemstatus gestartet.
- Es werden keine anderen Funktionen bzw. Anwendungen im System ausgeführt außer denen, die dem aktuellen Test unterzogen werden (sofern vom Testszenario nicht vorgesehen ist, dass bestimmte andere Systemaktivitäten gleichzeitig stattfinden).

  Anmerkung: Gestartete Anwendungen belegen Speicher, selbst wenn Sie auf Symbolgröße verkleinert wurden oder momentan inaktiv sind. Die erhöht die Wahrscheinlichkeit, dass die Seitenauslagerung zu einer Verzerrung der Vergleichstestergebnisse und zu einer Verletzung der Wiederholbarkeitsregel führt.
- Die Hardware und die Software, die für die Vergleichstests verwendet werden, entsprechen der realen Anwendungsumgebung.

Wie bei allen Vergleichstests, muss ein Szenario entworfen und anschließend mehrmals ausgeführt werden. Die Sammlung von Schlüsseldaten im Anschlusss an jede Ausführung ist von großer Bedeutung für die Ermittlung der Änderungen, die zur Verbesserung der Leistung von Anwendung und Datenbank durchgeführt werden müssen.

Vorbereiten von Vergleichstests

Das logische Modell der Datenbank für die Anwendung sollte vollständig sein, bevor die Leistungsvergleichstests durchgeführt werden. Tabellen, Sichten und Indizes müssen definiert und mit Informationen gefüllt sein. Die Tabellen sollten normalisiert und mit realistischen Daten gefüllt sein, und die Anwendungspakete sollten gebunden sein.

Das endgültige physische Modell der Datenbank sollte bereits feststehen. Die Objekte des Datenbankmanagers sollten sich an ihren endgültigen Speicherpositionen befinden, die Protokolldateien sollten in Bezug auf ihre Größe festgelegt, die Positionen der Arbeitsdateien und Sicherungskopien bestimmt, sowie die Sicherungsprozeduren gestestet sein. Darüber hinaus sollten die Pakete überprüft worden sein, um sicherzustellen, dass Leistungsoptionen wie Zeilenblockung aktiviert werden, wenn dies möglich ist.

Sie sollten einen Stand der Programmier- und Testphasen für die Anwendung erreicht haben, der es Ihnen ermöglicht, die Vergleichstestprogramme zu erstellen (siehe „Erstellen eines Vergleichstestprogramms" auf Seite 375). Während der Vergleichstests können die praktischen Grenzen einer Anwendungen zutage treten. Jedoch ist das Ziel der Vergleichstests, die hier beschrieben werden, die Messung der Leistung und nicht die Feststellung von Fehlern oder abnormalen Beendigungsbedingungen.

Das Vergleichstestprogramm muss in einer möglichst wirklichkeitsnahen Nachbildung der tatsächlichen Umgebung ausgeführt werden. Im Idealfall sollte ein Server des gleichen Modells mit derselben Speicher- und Festplattenkonfiguration verwendet werden. Dies ist besonders dann von Bedeutung, wenn die Anwendung letztendlich eine große Anzahl von Benutzern und große Mengen von Daten verarbeiten soll. Das Betriebssystem und mögliche Einrichtungen der Datenübertragung oder des Dateiservice sollten auch bereits optimiert worden sein.

Darüber hinaus ist es wichtig, dass die Vergleichstests mit einer Datenbank durchgeführt werden, deren Größe tatsächlichen Geschäftsumgebungen entspricht. Eine einzelne SQL-Anweisung sollte die gleiche Menge an Daten liefern und den gleichen Aufwand für Sortiervorgänge bewirken wie später in der implementierten Geschäftsumgebung. Bei Beachtung dieser Regel ist sichergestellt, dass die Anwendung repräsentative Speicheranforderungen verursachen wird.

Die Art der zu testenden SQL-Anweisungen sollte entweder zur Kategorie *repräsentatives SQL* oder zur Kategorie *Extremfall-SQL* (Worst-Case) gehören, wie im Folgenden erläutert wird:

Repräsentatives SQL

Zu repräsentativem SQL werden solche Anweisungen gezählt, die während eines typischen Einsatzes der zu testenden Anwendung ausgeführt werden. Die Anweisungen, die zum Test ausgewählt werden, sind von der Art der Anwendung abhängig. Beispielsweise kann für eine Dateneingabeanwendung eine Anweisung INSERT getestet werden, während für eine Banktransaktion eine Anweisung FETCH, eine Anweisung UPDATE und mehrere Anweisungen INSERT getestet werden können. Die Häufigkeit, mit der die Anwendung ausgeführt wird, und der Umfang der von den ausgewählten Anweisungen verarbeiteten Daten sollte als durchschnittlich angesehen werden können. Wenn die aufgrund der Anweisungen zu verarbeitenden Datenmengen sehr umfangreich sind, sollten diese Anweisungen unter der Kategorie *Extremfall-SQL* betrachtet werden, selbst wenn es sich um typische SQL-Anweisungen handelt.

Extremfall-SQL

Zu dieser Kategorie gehören Anweisungen mit folgenden Merkmalen:

- Anweisungen, die häufig ausgeführt werden
- Anweisungen, durch die umfangreiche Datenmengen verarbeitet werden
- Anweisungen, die zeitkritisch sind

  Ein Beispiel hierfür ist eine Anwendung, die ausgeführt wird, wenn ein Telefonanruf von einem Kunden eingeht, und deren Anweisungen ausgeführt werden müssen, um Daten des Kunden abzurufen oder zu aktualisieren, während der Kunde wartet.
- Anweisungen mit der größten Anzahl zu verknüpfender Tabellen, oder mit dem komplexesten SQL in der Anwendung

  Ein Beispiel wäre eine Bankanwendung, die kombinierte Kontoauszüge über die monatlichen Kontobewegungen für sämtliche verschiedene Arten von Konten eines Kunden erstellt. Eine allgemeine Tabelle könnte vielleicht die Kundenadressen und die Kontonummern enthalten; jedoch müssen mehrere andere Tabellen verknüpft werden, um alle benötigten Daten über Kontotransaktionen verarbeiten und zusammenstellen zu können. Die Multiplikation des Aufwands, der für ein Konto erforderlich ist, mit den mehreren Tausend Konten, die im gleichen Zeitraum verarbeitet werden müssen, macht deutlich, das jede potenzielle Zeiteinsparung die Leistungsanforderungen erhöht.

- Anweisungen, die einen ungünstigen Zugriffspfad verwenden, z. B. eine Anweisung, die nicht sehr oft ausgeführt wird und nicht von den Indizes unterstützt wird, die für die betroffenen Tabellen erstellt wurden
- Anweisungen, die über einen langen Zeitraum hinweg laufen
- Eine Anweisung, die nur bei der Initialisierung einer Anwendung ausgeführt wird, jedoch überproportionale Ressourcenanforderungen stellt

  Ein Beispiel wäre eine Anwendung, die eine Liste der Arbeiten für Konten erstellt, die während des Arbeitstages auszuführen sind. Wenn die Anwendung gestartet wird, löst die erste größere SQL-Anweisung eine Verknüpfung über zahlreiche Tabellen aus, um eine sehr umfangreiche Liste aller Konten zu erstellen, für die der Benutzer der Anwendung verantwortlich ist. Die Anweisung wird vielleicht nur wenige Male am Tag ausgeführt, aber ihre Ausführung nimmt einige Minuten in Anspruch, wenn sie nicht ausreichend optimiert wurde.

Erstellen eines Vergleichstestprogramms

Es gibt eine Reihe von Faktoren, die bei der Entwicklung und Implementierung eines Vergleichstestprogramms in Betracht gezogen werden sollten. Da der Hauptzweck des Programms darin besteht, eine Benutzeranwendung zu simulieren, kann die allgemeine Struktur des Programms unterschiedlich sein. Es kann die gesamte Anwendung zum Vergleichstest verwendet werden, so dass nur die entsprechenden Mittel zur Erfassung der Zeiten für die zu analysierenden SQL-Anweisungen eingefügt werden müssen. Bei großen oder komplexen Anwendungen ist es eventuell praktischer, nur die Blöcke mit den wichtigen Anweisungen in das Vergleichstestprogramm aufzunehmen.

Zum Testen der Leistung bestimmter SQL-Anweisungen gibt es auch den Ansatz, nur diese Anweisungen allein in das Vergleichstestprogramm aufzunehmen und die benötigten Anweisungen CONNECT, PREPARE, OPEN und andere sowie Mechanismen zur Erfassung der Zeitdaten hinzuzufügen.

Ein weiterer wichtiger Gesichtspunkt ist die Art des Vergleichs, die verwendet werden sollte. Eine Möglichkeit ist die, eine Gruppe von SQL-Anweisungen über ein Zeitintervall wiederholt auszuführen. Das Verhältnis der Anzahl der ausgeführten Anweisungen zu diesem Zeitintervall ergäbe einen Wert für den Durchsatz für die Anwendung. Eine andere Möglichkeit ist die, einfach die für die Ausführung einzelner SQL-Anweisungen erforderliche Zeit zu bestimmen.

Ungeachtet der Art des Vergleichsprogramms ist ein effizientes Zeiterfassungssystem erforderlich, um die während der Verarbeitung entweder einzelner SQL-Anweisungen oder der gesamten Anwendung abgelaufene Zeit zu berechnen. Bei der Simulation von Anwendungen, in denen einzelne SQL-Anweisungen isoliert ausgeführt werden, kann es sehr sinnvoll sein, die Zeiten für die Anweisungen CONNECT, PREPARE und COMMIT zu ermitteln. Bei Programmen jedoch, die viele verschiedene Anweisungen verarbeiten, wird vielleicht nur eine einzige Anweisung CONNECT oder COMMIT benötigt, so dass die Erfassung der Ausführungszeit für eine einzelne Anweisung Priorität haben könnte.

Obwohl die Erfassung der benötigten Zeit für jede Abfrage einen wichtigen Faktor bei der Leistungsanalyse darstellt, werden durch sie nicht unbedingt potenzielle Leistungsengpässe offen gelegt. Zum Beispiel könnten Informationen über die CPU-Auslastung, über aktive Sperren und Pufferpooleinbzw. -ausgaben Hinweise darauf geben, dass eine Anweisung durch die Ein-/Ausgabeaktivitäten gebremst wird, anstatt die CPU mit voller Kapazität auszunutzen. Ein Vergleichstestprogramm sollte es ermöglichen, diese Art von Daten für eine detailliertere Analyse bei Bedarf zu erfassen.

Nicht alle Anwendungen müssen die gesamte Menge der durch eine Abfrage abgerufenen Zeilen an eine Ausgabeeinheit senden. Einige Anwendungen können beispielsweise die als Ergebnis der Abfrage abgerufenen Zeilen als Eingabe für ein anderes Programm verwenden (d. h., keine der Zeilen aus der ersten Anwendung werden als Ausgabe verwendet). Die Formatierung von Daten zur Ausgabe auf der Anzeige verursacht in der Regel einen hohen CPU-Aufwand und spiegelt den Benutzerbedarf nicht unbedingt wider. Um eine genaue Simulation zu erhalten, sollte ein Vergleichstestprogramm die Zeilenbehandlung der bestimmten Anwendung berücksichtigen. Wenn Zeilen an eine Ausgabeeinheit gesendet werden, könnte ineffizientes Formatieren den Hauptanteil der CPU-Verarbeitungszeit in Anspruch nehmen und so zu einer Verzerrung der tatsächlichen Leistungsdaten für die SQL-Anweisung als solche führen.

Das Vergleichstest-Tool db2batch: Es steht ein Vergleichstest-Tool (`db2batch`) im Unterverzeichnis `bin` des Verzeichnisses `sqllib` Ihres Exemplars zur Verfügung. In diesem Tool werden viele der oben genannten Punkte zur Erstellung eines Vergleichstestprogramms berücksichtigt. Dieses Tool liest SQL-Anweisungen entweder aus einer unstrukturierten Datei oder von der Standardeingabeeinheit, beschreibt die Anweisungen dynamisch und bereitet sie vor und liefert eine Antwortmenge zurück. Es verfügt außerdem über die zusätzliche Flexibilität, dass die Größe der Antwortmenge und die Anzahl der Zeilen, die aus dieser Anwortmenge an eine Ausgabeeinheit gesendet werden sollen, gesteuert werden können.

Darüber hinaus kann die Stufe der leistungsbezogenen Daten, die geliefert werden sollen, einschließlich der benötigten Zeit, CPU- und Pufferpool-auslastung, Sperren sowie anderer statistischer Daten aus dem Datenbank-monitor, angegeben werden. Bei der Zeitmessung für eine Gruppe von SQL-Anweisungen erstellt `db2batch` auch eine Ergebnisübersicht und berechnet arithmetische und geometrische Mittelwerte. Weitere Informationen zur Syntax des Aufrufs und zu den Optionen finden Sie im Rahmen der Informationen zu `db2batch` im Handbuch *Command Reference*. Die verfügbaren Informationen zu Syntax und Optionen können Sie auch abrufen, indem Sie in einer Befehlszeile `db2batch -h` eingeben.

Das folgende Beispiel zeigt, wie das Tool `db2batch` mit einer Eingabedatei `db2batch.sql` verwendet werden könnte:

```
-- db2batch.sql
-- ------------
--#SET PERF_DETAIL 3 ROWS_OUT 5

-- This query lists employees, the name of their department
-- and the number of activities to which they are assigned for
-- employees who are assigned to more than one activity less than
-- full-time.
--#COMMENT Query 1
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2;
--#SET PERF_DETAIL 1 ROWS_OUT 5
--#COMMENT Query 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2;
```

Abbildung 31. Beispieleingabedatei für das Vergleichstest-Tool: db2batch.sql

Geben Sie zum Beispiel den folgenden Aufruf für das Vergleichstest-Tool ein:

```
db2batch -d sample -f db2batch.sql
```

Dadurch wird die folgende Ausgabe erstellt:

```
--#SET PERF_DETAIL 3 ROWS_OUT 5
Query 1

Statement number: 1

select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2
```

Abbildung 32. Beispielausgabe des Programms db2batch (Teil 1)

```
LASTNAME         FIRSTNME      DEPTNAME                        NUM_ACT
-------------------------------------------------------------------------
JEFFERSON        JAMES         ADMINISTRATION SYSTEMS                3
JOHNSON          SYBIL         ADMINISTRATION SYSTEMS                4
NICHOLLS         HEATHER       INFORMATION CENTER                    4
PEREZ            MARIA         ADMINISTRATION SYSTEMS                4
SMITH            DANIEL        ADMINISTRATION SYSTEMS                7
Number of rows retrieved is:       5
Number of rows sent to output is:  5
Elapsed Time is:           0.074      seconds
Locks held currently                        = 0
Lock escalations                            = 0
Total sorts                                 = 5
Total sort time (ms)                        = 0
Sort overflows                              = 0
Buffer pool data logical reads              = 13
Buffer pool data physical reads             = 5
Buffer pool data writes                     = 0
Buffer pool index logical reads             = 3
Buffer pool index physical reads            = 0
Buffer pool index writes                    = 0
Total buffer pool read time (ms)            = 23
Total buffer pool write time (ms)           = 0
Asynchronous pool data page reads           = 0
Asynchronous pool data page writes          = 0
Asynchronous pool index page reads          = 0
Asynchronous pool index page writes         = 0
Total elapsed asynchronous read time        = 0
Total elapsed asynchronous write time       = 0
Asynchronous read requests                  = 0
LSN Gap cleaner triggers                    = 0
Dirty page steal cleaner triggers           = 0
Dirty page threshold cleaner triggers       = 0
Direct reads                                = 8
Direct writes                               = 0
Direct read requests                        = 4
Direct write requests                       = 0
Direct read elapsed time (ms)               = 0
Direct write elapsed time (ms)              = 0
Rows selected                               = 5
Log pages read                              = 0
Log pages written                           = 0
Catalog cache lookups                       = 3
Catalog cache inserts                       = 3
Buffer pool data pages copied to ext storage    = 0
Buffer pool index pages copied to ext storage   = 0
Buffer pool data pages copied from ext storage  = 0
Buffer pool index pages copied from ext storage = 0
Total Agent CPU Time (seconds)              = 0.02
Post threshold sorts                        = 0
Piped sorts requested                       = 5
Piped sorts accepted                        = 5
```

Abbildung 33. Beispielausgabe des Programms db2batch (Teil 1)

```
--#SET PERF_DETAIL 1 ROWS_OUT 5
Query 2
Statement number: 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2
LASTNAME         FIRSTNME      DEPTNAME                       NUM_ACT
------------------------------------------------------------------------
GEYER            JOHN          SUPPORT SERVICES                     2
GOUNOT           JASON         SOFTWARE SUPPORT                     2
HAAS             CHRISTINE     SPIFFY COMPUTER SERVICE DIV.         2
JONES            WILLIAM       MANUFACTURING SYSTEMS                2
KWAN             SALLY         INFORMATION CENTER                   2
Number of rows retrieved is:       8
Number of rows sent to output is:  5
Elapsed Time is:           0.037      seconds
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                     0.074               0.020         5         5
2                     0.037       Not Collected         8         5
Arith. mean     0.055
Geom.  mean     0.052
```

Abbildung 34. Beispielausgabe des Programms db2batch (Teil 2)

Die gezeigte Beispielausgabe umfasst spezifische Datenelemente, die vom Datenbanksystemmonitor geliefert werden. Weitere Informationen zu diesen und anderen Monitorelementen finden Sie im Handbuch *System Monitor Guide and Reference*.

Im folgenden Beispiel (unter UNIX) wird lediglich die Ergebnisübersicht (Summary of Results) erstellt.

```
db2batch -d sample -f db2batch.sql -r /dev/null
```

Mit diesem Aufruf wird nur die Ergebnisübersicht angezeigt. Durch die Option `-r` wird die Ausgabedatei 1 (`outfile1`) durch `/dev/null` ersetzt und die Ausgabedatei 2 (`outfile2`), die nur die Ergebnistabelle enthält, bleibt leer, so dass die Ausgabe von `db2batch` an die Anzeige gesendet wird:

```
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                    0.074                0.020          5         5
2                    0.037       Not Collected          8         5
Arith. mean     0.055
Geom.  mean     0.052
```

Abbildung 35. Beispielausgabe des Programms db2batch - nur Ergebnistabelle

Dieses Vergleichstest-Tool verfügt darüber hinaus über die Option CLI. Mit dieser Option können Sie die Größe eines Cache angeben. Im folgenden Beispiel wird `db2batch` im CLI-Modus mit einer Cache-Größe von 30 Anweisungen ausgeführt:

```
db2batch -d sample -f db2batch.sql -cli 30
```

Auch die Fernausführung von db2batch ist möglich. Wenn Sie entweder den Befehlsparameter

```
-f <dateiname>
```

oder

```
-o <optionen>
```

des Vergleichstest-Tools verwenden, gilt Folgendes:

- Die Steuerungsoptionen

  ```
  perf_detail
  ```

  und

  ```
  -p <perf_detail>
  ```

  (die die Stufe der zuzurückzugebenden Leistungsdaten angeben) werden bei der Fernausführung nicht unterstützt, wenn sie auf einen Wert größer 1 gesetzt werden.
- Die Steuerungsoptionen

  ```
  perf_detail
  ```

  und

  ```
  -p <perf_detail>
  ```

  (die die Stufe der zuzurückzugebenden Leistungsdaten angeben) sind für DB2 Universal Database für Windows 3.x- oder DOS-Plattformen nicht gültig.

Anders als diese beiden Optionen werden die Steuerungsoptionswerte
`perf_detail`

und
`-p <perf_detail>`

unterstützt und sind für alle DB2 Universal Database-Plattformen gültig.

Ausführen der Vergleichstests

Eine Art von Datenbankvergleichstest besteht darin, einen Konfigurationsparameter auszuwählen und den Test mit verschiedenen Werten für den gewählten Parameter auszuführen, bis die maximale Leistungssteigerung erzielt ist. Ein einzelner Testlauf sollte die mehrmalige Ausführung der Anwendung (z. B. zwanzig- oder dreißigmal) mit demselben Parameterwert beinhalten, um einen Durchschnittswert für die benötigte Zeit zu ermitteln, der die Auswirkungen einer Änderung des Parameterwertes exakter wiedergibt.

Bei der Ausführung des Vergleichstests sollte der erste Durchlauf (Warm-up Run, Aufwärmdurchlauf) von den nachfolgenden Iterationen (Normal Run, normaler Durchlauf) getrennt betrachtet werden. Dies ist notwendig, weil der Aufwärmdurchlauf stets einige Startaktivitäten (z. B. Initialisierung des Pufferpools) enthält. Folglich dauert der Aufwärmdurchlauf etwas länger als die normalen Durchläufe. Obwohl die Daten aus dem Aufwärmdurchlauf durchaus realistische Werte darstellen *können*, sind sie für die statistische Auswertung nicht relevant. Zur Berechnung des Durchschnittswerts für die Taktung oder die CPU für eine bestimmte Menge von Parameterwerten sollten Sie daher die Ergebnisse von normalen Durchläufen verwenden.

Sie können auch in Betracht ziehen, mit dem *Assistent: Leistungskonfiguration* den Aufwärmdurchlauf des Vergleichstests zu erstellen. Die im *Assistent: Leistungskonfiguration* gestellten Fragen geben einen Einblick in die Gesichtspunkte, die bei der Anpassung der Konfiguration Ihrer Umgebung für normale Durchläufe in der Vergleichstestphase zu beachten sind. Wenn Sie den *Assistent: Leistungskonfiguration* verwenden möchten, geben Sie `db2cc` ein, um auf die Steuerzentrale zuzugreifen, und fahren von dort aus fort. Wenn die Vergleichstests mit einzelnen Abfragen durchgeführt werden, muss sichergestellt werden, dass die potenziellen Auswirkungen früherer Abfragen minimal bleiben. Dies kann durch Füllen des Pufferpools (Flushing) erreicht werden, d. h. eine Anzahl von Seiten, die für die zu testende Abfrage irrelevant sind, wird in den Pufferpool eingelesen.

Nach der Ausführung der Durchläufe für eine Menge von Parameterwerten kann ein einzelner Parameter geändert werden. Zwischen den einzelnen

Durchläufen sollten jedoch folgende Maßnahmen durchgeführt werden, um die Vergleichstestumgebung wieder in den Ausgangszustand zurückzusetzen:

- Setzen Sie die Anwendungsdaten und die Statistiken des Datenbankmanagers in den Ausgangszustand zurück. Wenn die Katalogstatistiken für den Test aktualisiert wurden, stellen Sie sicher, dass für jeden Durchlauf dieselben Werte für die Statistiken verwendet werden. Die Daten, die im Test verwendet werden, müssen konsistent sein, wenn sie im Laufe des Tests aktualisiert werden. Dies kann folgendermaßen sichergestellt werden:
  - Durch Verwenden des Dienstprogramms RESTORE, um die gesamte Datenbank wieder herzustellen. Die Sicherungskopie der Datenbank wäre in ihrem früheren Zustand und für den nächsten Test bereit.
  - Durch Verwenden des Dienstprogramms IMPORT oder LOAD, um eine exportierte Kopie der Daten wiederherzustellen. Diese Methode erlaubt die Wiederherstellung nur der Daten, die vom Test betroffen waren. Die Dienstprogramme REORG und RUNSTATS sollten für die Tabellen und Indizes, die diese Daten enthalten, ausgeführt werden.
- Setzen Sie die Anwendung in ihren Ausgangszustand zurück, indem Sie sie erneut an die Datenbank binden.

Die folgenden Zusatzmaßnahmen gelten für Vergleichstests, die auf OS/2-Systemen durchgeführt werden:

- Wenn während des Testablaufs Auslagerungen auftraten, stellen Sie sicher, dass die Datei SWAPPER.DAT auf die Ausgangsgröße zurückgesetzt wird.
- Starten Sie das System bei Bedarf erneut, um die Wiederholgenauigkeit zu gewährleisten.

Die Ausgabe des Vergleichstestprogramms sollte eine Kennung für jeden Test (Test Number), die Iteration (Durchlauf) der Programmausführung (Iteration Number), die Anweisungsnummer (Statement Number) und die für die Ausführung erfasste Zeit (Timing) enthalten. Eine Übersicht über Vergleichstestergebnisse nach einer Reihe von Messungen könnte wie folgt aussehen:

```
Test     Iter.   Stmt    Timing         SQL Statement
Numbr    Numbr   Numbr   (hh:mm:ss.ss)
 002      05      01     00:00:01.34    CONNECT TO SAMPLE
 002      05      10     00:02:08.15    OPEN cursor_01
 002      05      15     00:00:00.24    FETCH cursor_01
 002      05      15     00:00:00.23    FETCH cursor_01
 002      05      15     00:00:00.28    FETCH cursor_01
 002      05      15     00:00:00.21    FETCH cursor_01
 002      05      15     00:00:00.20    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      15     00:00:00.22    FETCH cursor_01
 002      05      20     00:00:00.84    CLOSE cursor_01
 002      05      99     00:00:00.03    CONNECT RESET
```

Abbildung 36. Beispielergebnisse eines Vergleichstestprogramms

Anmerkung: Die Daten im gezeigten Bericht dienen nur der Illustration. Sie stellen **keine** durch Tests ermittelten Ergebnisse dar.

Bei der Auswertung dieses Berichts ergäbe sich, dass die Anweisung CONNECT (Anweisung 01) 1,34 Sekunden dauerte, die Anweisung OPEN CURSOR (Anweisung 10) 2 Minuten und 8,15 Sekunden, die Anweisungen FETCH (Anweisung 15) sieben Zeilen mit der längsten Verzögerung von 0,28 Sekunden lieferten, die Anweisung CLOSE CURSOR (Anweisung 20) 0,84 Sekunden benötigte und die Anweisung CONNECT RESET (Anweisung 99) 0,03 Sekunden in Anspruch nahm.

Es könnte sich als vorteilhaft erweisen, wenn das Programm die Daten in einem DEL-Format (Delimited ASCII) ausgeben könnte, so dass diese später in eine Datenbanktabelle oder ein Arbeitsblatt zur weiteren statistischen Analyse importiert werden könnten.

Eine Beispielausgabe für einen Vergleichstestbericht könnte folgendermaßen aussehen:

```
      PARAMETER        VALUES FOR EACH BENCHMARK TEST
      TEST NUMBER      001     002     003    004     005
      locklist         63      63      63      63      63
>>    buffpage         1000    1175    1250    1325    1400     <<
      maxappls         8       8       8       8       8
      applheapsz       48      48      48      48      48
      dbheap           128     128     128     128     128
      sortheap         256     256     256     256     256
      maxlocks         22      22      22      22      22
      stmtheap         1024    1024    1024    1024    1024
      SQL STMT         AVERAGE TIMINGS (seconds)
        01             01.34   01.34   01.35   01.35   01.36
        10             02.15   02.00   01.55   01.24   01.00
        15             00.22   00.22   00.22   00.22   00.22
        20             00.84   00.84   00.84   00.84   00.84
        99             00.03   00.03   00.03   00.03   00.03
```

Abbildung 37. Beispielbericht zu den vom Vergleichstest ermittelten Ausführungszeiten

Anmerkung: Die Daten im gezeigten Bericht dienen nur der Illustration. Sie stellen **keine** durch Tests ermittelten Ergebnisse dar.

Eine Untersuchung der Daten in diesem Beispiel ergibt, dass durch die Änderung des Parameters *buffpage* die Zeiten für die Anweisung OPEN CURSOR erfolgreich von 2,15 Sekunden auf 1,00 Sekunden herabgesetzt wurden. (Hierbei wird angenommen, dass es nur einen Pufferpool gibt, dessen Größe (NPAGES) auf den Wert -1 gesetzt ist. Das heißt, die Größe des Pufferpools wird vom Parameter *buffpage* gesteuert.)

Zusammenfassend können folgende Schritte/Iterationen genannt werden, die bei Vergleichstests einer Datenbankanwendung ausgeführt werden können:

Schritt 1

Belassen Sie alle Parameter zur Optimierung der Datenbank und des Datenbankmanagers außer folgenden auf ihren **Standardwerten**:

- Die Parameter, die für die Belastung durch den Test und die Zielsetzung des Tests von Bedeutung sind. (Sie werden selten genügend Zeit haben, um Vergleichstests zur Optimierung aller Parameter durchzuführen, so dass es empfehlenswert ist, zu Beginn gute Schätzwerte für einige Parameter festzulegen und von diesen ausgehend die Optimierung vorzunehmen.)
- Protokolldateigrößen, die während der Einheiten- oder Systemtests für Ihre Anwendung festgelegt werden sollten. (Der Abschnitt „Protokolldateigröße (logfilsiz)" auf Seite 478 enthält weitere Informationen.)
- Alle die Parameter, die geändert werden müssen, damit die Anwendung ausgeführt werden kann (d. h., Änderungen zur Verhinderung negativer SQL-Rückkehrcodes aufgrund von Ereignissen wie Speicherknappheit für den Anweisungszwischenspeicher).

Führen Sie Ihre Anzahl von Durchläufen (Iterationen) für diesen Anfangszustand aus, und berechnen Sie den Durchschnittswert für die Taktung oder die CPU.

Schritt 2

Wählen Sie einen und nur einen für die Optimierung zu testenden Parameter aus, und ändern Sie seinen Wert.

Schritt 3

Führen Sie eine weitere Anzahl von Durchläufen (Iterationen) aus, und berechnen Sie den Durchschnittswert für die Taktung oder die CPU.

Schritt 4

Ergreifen Sie in Abhängigkeit von den Ergebnissen des Vergleichstests eine der folgenden Maßnahmen:

- Wenn die Leistung besser wird, ändern Sie den Wert desselben Parameters und kehren zu Schritt 3 zurück. Ändern Sie diesen Parameter so lange, bis der maximale Leistungswert gezeigt wird.
- Wenn die Leistung sinkt oder unverändert bleibt, setzen Sie den Parameter auf seinen vorigen Wert zurück, kehren zu Schritt 2 zurück und wählen einen anderen Parameter. Wiederholen Sie diese Prozedur, bis alle Parameter getestet sind.

Anmerkung: Wenn Sie die Leistungsergebnisse in grafischer Darstellung festhalten würden, müssten Sie nach Stellen suchen, an denen die Kurve einen Maximalwert erreicht und dort verbleibt bzw. wieder absinkt.

Sie können ein Treiberprogramm schreiben, das Sie bei der Durchführung der Vergleichstests unterstützt. Dieses Treiberprogramm könnte in einer Sprache wie REXX bzw. bei auf UNIX basierenden Systemen mit Hilfe von Shell-Prozeduren (Skripts) erstellt werden.

Dieses Treiberprogramm könnte das Vergleichstestprogramm ausführen, ihm dabei die richtigen Parameter übergeben, den Test durch mehrere Durchläufe führen, die Umgebung in einen konsistenten Zustand zurückversetzen, den nächsten Test mit neuen Parameterwerten vorbereiten und die Testdaten sammeln bzw. konsolidieren. Diese Treiberprogramme können so flexibel gestaltet werden, dass sie zur Ausführung einer ganzen Reihe von Vergleichstests, zur Analyse der Ergebnisse und zur Erstellung eines Berichts über die endgültigen und optimalen Parameterwerte für einen bestimmten Test verwendet werden könnten.

Kapitel 13. Konfigurieren von DB2

Konfigurationsparameter sind Werte, die die Betriebsmerkmale einer Datenbank oder eines Datenbankverwaltungssystems beeinflussen.

Die Konfigurationsparameter des Datenbankmanagers sind auf Servern und Clients vorhanden. Auf dem Client können jedoch nur bestimmte Konfigurationsparameter des Datenbankmanagers eingestellt werden. Diese Parameter sind eine Untergruppe der Konfigurationsparameter für die Datenbankverwaltung, die auf dem Server eingestellt werden können. Je nach der Art des verwendeten Produkts DB2 Universal Database gibt es spezifische Fragen in Bezug auf Konfigurationsparameter. Zum Beispiel wird bei der DB2 Enterprise - Extended Edition eine Konfigurationsdatei des Datenbankmanagers von allen Datenbankpartitions-Servern im Exemplar gemeinsam benutzt. Und jede Datenbankpartition verfügt über eine eigene Datenbankkonfigurationsdatei.

DB2 bietet zahlreiche Optimierungs- und Konfigurationsparameter. Diese Parameter werden in zwei Hauptkategorien unterteilt:

- „Parameter des Datenbankmanagers" auf Seite 389
- „Datenbankparameter" auf Seite 396.

Neben den Beschreibungen der einzelnen Parameter sind folgende Themen enthalten, die in starkem Maß von Konfigurationsparametern beeinflusst werden:

- „Optimieren der Konfigurationsparameter" auf Seite 388.
- „Einzelheiten zu Parametern nach Funktion" auf Seite 403 (jeder Funktionsbereich hat seine eigene Liste von Konfigurationsparametern).
- „Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen" auf Seite 571.

  Es kann für Ihre Plattform leistungsbezogene Umgebungs- oder Registrierungsvariablen geben. Sie sollten dann erwägen, ob Sie diese zusätzlich zu den leistungsbezogenen Konfigurationsparametern verwenden.
- „Kapitel 8. Leistung bei der Ausführung" auf Seite 283.
- „Kapitel 12. Durchführen von Vergleichstests" auf Seite 371.

Sie sollten zunächst alle Kurzbeschreibungen der Parameter in Tabelle 17 auf Seite 391 und Tabelle 19 auf Seite 398 lesen und sich dann auf die Beschreibungen und Optimierung der Parameter konzentrieren, mit denen sich in Ihrer Arbeitsumgebung die größten Leistungsgewinne erzielen lassen.

Optimieren der Konfigurationsparameter

Der vom Datenbankmanager auf der Basis der Standardwerte für die Parameter zugeordnete Plattenspeicherplatz und Hauptspeicher können für Ihre Anforderungen in einigen Fällen ausreichend sein. In einigen Situationen wird die maximale Leistung bei Verwendung dieser Standardwerte jedoch vielleicht nicht erreicht.

Da die Standardwerte für Systeme mit relativ kleinem Hauptspeicher, die als Datenbank-Server eingesetzt werden, ausgelegt sind, müssen Sie die Parameterwerte ändern, wenn Ihre Umgebung folgende Merkmale aufweist:

- Umfangreiche Datenbanken
- Große Anzahl von Verbindungen
- Hohe Leistungsanforderungen für eine bestimmte Anwendung
- Spezifische Abfrage-/Transaktionsladevorgänge bzw. -arten
- Verschiedene Arten der Konfiguration und des Einsatzes von Systemen

Jede Umgebung, die Transaktionen verarbeitet, hat einen oder mehrere spezifische, nur für sie geltende Aspekte. Diese Unterschiede können sich tief greifend auf die Leistung des Datenbankmanagers auswirken, wenn die Standardkonfiguration verwendet wird. Aus diesem Grund wird ausdrücklich empfohlen, die Konfiguration für die Umgebung zu optimieren.

Verschiedene Arten von Anwendungen und Benutzern unterscheiden sich in ihren Anforderungen und Erwartungen an die Antwortzeiten. Die Bandbreite der Anwendungen kann von einfachen Dateneingabebildschirmen bis hin zu strategischen Anwendungen zur Verarbeitung zahlreicher komplexer SQL-Anweisungen mit Zugriff auf Dutzende von Tabellen pro Arbeitseinheit reichen. Zum Beispiel können die Anforderungen an die Antwortzeiten zwischen einer Serviceanwendung für Telefonkunden und einer Stapelanwendung zur Erstellung von Berichten beträchtliche Unterschiede aufweisen.

Weitere zugehörige Themen enthalten hilfreiche Informationen zum Testen der Anwendung und Optimierung der Konfigurationsparameter:

- „Parameter des Datenbankmanagers" auf Seite 389
- „Datenbankparameter" auf Seite 396
- „Einzelheiten zu Parametern nach Funktion" auf Seite 403 (jeder Funktionsbereich hat seine eigene Liste von Konfigurationsparametern)
- „Kapitel 8. Leistung bei der Ausführung" auf Seite 283
- „Kapitel 12. Durchführen von Vergleichstests" auf Seite 371
- Beschreibungen der Datenbanksystemmonitorelemente im Handbuch *System Monitor Guide and Reference*

Parameter des Datenbankmanagers

Die Parameter des Datenbankmanagers sind in einer Datei mit dem Namen `db2systm` gespeichert. Diese Datei wird erstellt, wenn das Exemplar des Datenbankmanagers erstellt wird. In auf UNIX basierenden Umgebungen befindet sich diese Datei im Unterverzeichnis `sqllib` für das Exemplar des Datenbankmanagers. In allen anderen Umgebungen finden Sie diese Datei standardmäßig im Unterverzeichnis des Exemplars im Verzeichnis `sqllib`. Wenn die Variable DB2INSTPROF aktiviert ist, befindet sich die Datei im Unterverzeichnis `instance` des in der Variable DB2INSTPROF angegebenen Verzeichnisses.

In einer Umgebung mit partitionierten Datenbanken befindet sich diese Datei auf einem gemeinsam benutzten Dateisystem, so dass alle Datenbankpartitions-Server auf dieselbe Datei zugreifen können. Die Konfiguration des Datenbankmanagers ist auf allen Datenbankpartitions-Servern gleich.

Die meisten Parameter haben entweder Einfluss darauf, wie viel Systemressourcen einem einzelnen Datenbankmanagerexemplar zugeordnet werden, oder sie konfigurieren die Einrichtung des Datenbankmanagers und der verschiedenen Kommunikationssubsysteme nach umgebungsspezifischen Überlegungen. Darüber hinaus gibt es noch weitere Parameter, die nur der Information dienen und deren Werte nicht geändert werden können. Alle diese Parameter sind allgemein gültig und unabhängig von einzelnen Datenbanken, die unter diesem Datenbankmanagerexemplar gespeichert sind.

Die Datei `db2systm` kann nicht direkt editiert werden. Sie kann nur mit Hilfe einer bereitgestellten Anwendungsprogrammierschnittstelle (API) oder einem Tool, das diese API aufruft, geändert oder angezeigt werden.

Achtung: Wenn Sie die Datei mit anderen Methoden als denen, die vom Produkt vorgesehen sind, editieren, wird Ihr System möglicherweise unbrauchbar. **Es ist äußerst ratsam**, diese Datei nur mit den von DB2 unterstützten und dokumentierten Methoden zu ändern.

Sie können eine der folgenden Methoden zum Anzeigen, Aktualisieren und Zurücksetzen der Konfigurationsparameter des Datenbankmanagers verwenden:

- Die Verwendung der DB2-Steuerzentrale. In der DB2-Steuerzentrale steht das Notizbuch **Exemplar konfigurieren** zur Verfügung, mit dem Sie die Konfigurationsparameter des Datenbankmanagers auf einem Client oder Server einstellen können. Die DB2-Steuerzentrale stellt auch den *Assistent: Leistungskonfiguration* zum Ändern der Werte der Konfigurationsparameter auf einem Server bereit. Dieser Assistent generiert Werte für Parameter auf der Grundlage Ihrer Antworten zu einer Reihe von Fragen, z. B. Auslas-

tung und Transaktionsart der Datenbank. In der Online-Hilfefunktion der Steuerzentrale finden Sie Informationen zur Verwendung dieser Schnittstellen.

- Die Verwendung des Befehlszeilenprozessors. Befehle zum Ändern der Einstellungen können schnell und bequem eingegeben werden. Weitere Informationen zu den folgenden Befehlen finden Sie im Handbuch *Command Reference*:
  - GET DATABASE MANAGER CONFIGURATION (oder GET DBM CFG)
  - UPDATE DATABASE MANAGER CONFIGURATION (oder UPDATE DBM CFG)
  - RESET DATABASE MANAGER CONFIGURATION (oder RESET DBM CFG)
- Die Verwendung von Anwendungsprogrammierschnittstellen (APIs). Die APIs können leicht aus einer Anwendung aufgerufen werden. Weitere Informationen finden Sie im Handbuch *Administrative API Reference*.
- Die Verwendung von **Client-Konfiguration - Unterstützung**. Mit **Client-Konfiguration - Unterstützung** können Sie nur die Konfigurationsparameter des Datenbankmanagers auf einem Client einstellen.

Nach dem Ändern der Parameter muss der Datenbankmanager gestoppt (`db2stop`) und wieder gestartet (`db2start`) werden, damit die neuen Parameterwerte in Kraft treten. Bei Clients werden Änderungen an den Konfigurationsparametern des Datenbankmanagers wirksam, wenn der Client das nächste Mal die Verbindung zu einem Server herstellt. Obwohl die neuen Parameterwerte nicht sofort in Kraft treten, werden beim Anzeigen der Parametereinstellungen stets die zuletzt aktualisierten Werte angezeigt.

Anmerkung: Sie müssen den Datenbankmanager nicht erneut starten, wenn Sie den Wert des Parameters *dft_monswitches* aktualisieren; dieser Parameter wird automatisch aktualisiert, wenn Sie seinen Wert ändern.

Überblick über die Konfigurationsparameter des Datenbankmanagers

In der folgenden Tabelle sind die Parameter der Konfigurationsdatei des Datenbankmanagers für Datenbank-Server aufgeführt. Beachten Sie beim Ändern der Konfigurationsparameter des Datenbankmanagers die detaillierten Informationen zu jedem Parameter. Informationen zu spezifischen Betriebsumgebungen mit Standardwerten sind in jeder Parameterbeschreibung enthalten.

In der Spalte „Leistungsrelevanz" der folgenden Tabelle ist vermerkt, in welchem relativen Ausmaß sich jeder Parameter in Bezug auf die Systemleistung auswirken kann. Diese Spalte kann allerdings nicht für alle Arten von Umgebungen gültige Angaben enthalten. Sie sollten die Informationen als allgemeine Hinweise betrachten.

- **Hoch** — gibt an, dass ein Parameter wesentliche Auswirkungen auf die Leistung haben kann. Die Werte für diese Parameter müssen sehr eingehend überlegt werden. In einigen Fällen kann das bedeuten, dass Sie die vorgegebenen Standardwerte übernehmen sollten.
- **Mittel** — gibt an, dass der Parameter unter Umständen Auswirkungen auf die Leistung haben kann. Sie sollten von Ihrer spezifischen Umgebung und Ihren Anforderungen ausgehend entscheiden, wie viel Aufwand in die Optimierung dieser Parameter investiert werden sollte.
- **Niedrig** — gibt an, dass der Parameter weniger allgemeine bzw. weniger bedeutende Auswirkungen auf die Leistung hat.
- **Keine** — gibt an, dass der Parameter keine direkten Auswirkungen auf die Leistung hat. Da diese Parameter nicht leistungsrelevant sind, müssen sie nicht optimiert werden, sie können jedoch in anderer Hinsicht für die Systemkonfiguration von Bedeutung sein, z. B. zur Einrichtung der Übertragungsunterstützung.

Tabelle 17. Konfigurierbare Konfigurationsparameter des Datenbankmanagers

Parameter	Leistungsrelevanz	Zusätzliche Informationen
agentpri	Hoch	„Agentenpriorität (agentpri)" auf Seite 466
agent_stack_sz	Niedrig	„Größe des Agentenstapelspeichers (agent_stack_sz)" auf Seite 432
aslheapsz	Hoch	„Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)" auf Seite 436
audit_buf_sz	Hoch	„Prüfpuffergröße (audit_buf_sz)" auf Seite 446
authentication	Niedrig	„Authentifizierungsart (authentication)" auf Seite 561
backbufsz	Mittel	„Standardgröße für Sicherungspuffer (backbufsz)" auf Seite 413
catalog_noauth	Keine	„Katalogisieren ohne Berechtigung zulässig (catalog_noauth)" auf Seite 563
comm_bandwidth	Mittel	„Übertragungsbandbreite (comm_bandwidth)" auf Seite 549
conn_elapse	Mittel	„Antwortzeit für Knotenverbindung (conn_elapse)" auf Seite 535
cpuspeed	Niedrig (s. Anm.)	„CPU-Geschwindigkeit (cpuspeed)" auf Seite 549
datalinks	Niedrig	„Unterstützung der Aktivierung von Data Links (datalinks)" auf Seite 510
dft_account_str	Keine	„Standardzeichenfolge für Abrechnung (dft_account_str)" auf Seite 555
dft_client_adpt	Keine	„Standard-Client-Adapternummer (dft_client_adpt)" auf Seite 531

Tabelle 17. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Leistungsrelevanz	Zusätzliche Informationen
dft_client_comm	Keine	„Standard-Client-Übertragungsprotokoll (dft_client_comm)" auf Seite 530
dft_monswitches • *dft_mon_bufpool* • *dft_mon_lock* • *dft_mon_sort* • *dft_mon_stmt* • *dft_mon_table* • *dft_mon_uow*	Mittel	„Standardschalter für den Datenbanksystem-monitor (dft_monswitches)" auf Seite 547
dftdbpath	Keine	„Standarddatenbankpfad (dftdbpath)" auf Seite 563
diaglevel	Niedrig	„Aufzeichnungsebene bei Fehlerdiagnose (diaglevel)" auf Seite 544
diagpath	Keine	„Verzeichnispfad für Diagnosedaten (diagpath)" auf Seite 545
dir_cache	Mittel	„Verzeichnis-Cache-Unterstützung (dir_cache)" auf Seite 444
dir_obj_name	Keine	„Verzeichnisobjektname (dir_obj_name)" auf Seite 528
dir_path_name	Keine	„Verzeichnispfadname (dir_path_name)" auf Seite 527
dir_type	Keine	„Verzeichnisserviceart (dir_type)" auf Seite 526
discover	Mittel	„Discovery-Modus (discover)" auf Seite 532
discover_comm	Niedrig	„Discovery-Kommunikationsprotokoll (discover_comm)" auf Seite 533
discover_inst	Niedrig	„Discover-Server-Exemplar (discover_inst)" auf Seite 534
dos_rqrioblk	Hoch	„E/A-Blockgröße für DOS-Requester (dos_rqrioblk)" auf Seite 441
drda_heap_sz	Niedrig	„DRDA-Zwischenspeichergröße (drda_heap_sz)" auf Seite 429
fcm_num_anchors	Hoch	„Anzahl der FCM-Nachrichtenanker (fcm_num_anchors)" auf Seite 536
fcm_num_buffers	Hoch	„Anzahl der FCM-Puffer (fcm_num_buffers)" auf Seite 536
fcm_num_connect	Hoch	„Anzahl der FCM-Verbindungseinträge (fcm_num_connect)" auf Seite 538

Tabelle 17. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Leistungsrelevanz	Zusätzliche Informationen
fcm_num_rqb	Hoch	„Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb)" auf Seite 539
federated	Mittel	„Unterstützung für Systeme mit zusammengeschlossenen Datenbanken (federated)" auf Seite 556
fileserver	Keine	„Name des IPX/SPX-Datei-Servers (fileserver)" auf Seite 524
indexrec	Mittel	„Zeitpunkt für Indexneuerstellung (indexrec)" auf Seite 493
initdari_jvm	Mittel	„DARI-Prozess mit JVM initialisieren (initdari_jvm)" auf Seite 477
intra_parallel	Hoch	„Partitionsinterne Parallelität aktivieren (intra_parallel)" auf Seite 543
ipx_socket	Keine	„IPX/SPX-Socket-Nummer (ipx_socket)" auf Seite 525
java_heap_sz	Hoch	„Maximaler Zwischenspeicher für Java-Interpreter (java_heap_sz)" auf Seite 447
jdk11_path	Keine	„Java Development Kit 1.1: Installationspfad (jdk11_path)" auf Seite 556
keepdari	Mittel	„DARI-Prozess beibehalten (keepdari)" auf Seite 474
maxagents	Mittel	„Maximale Anzahl von Agenten (maxagents)" auf Seite 468
maxcagents	Mittel	„Maximale Anzahl gleichzeitig aktiver Agenten (maxcagents)" auf Seite 469
max_connretries	Mittel	„Maximale Anzahl Wiederholungen für Knotenverbindungen (max_connretries)" auf Seite 540
max_coordagents	Mittel	„Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470
maxdari	Mittel	„Maximale Anzahl von DARI-Prozessen (maxdari)" auf Seite 475
max_logicagents	Mittel	„Maximale Anzahl logischer Agenten (max_logicagents)" auf Seite 471
max_querydegree	Hoch	„Maximaler Grad der Parallelität bei Abfragen (max_querydegree)" auf Seite 541
max_time_diff	Mittel	„Maximale Zeitdifferenz zwischen Knoten (max_time_diff)" auf Seite 540

Tabelle 17. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Leistungsrelevanz	Zusätzliche Informationen
maxtotfilop	Mittel	„Maximale Anzahl offener Dateien (maxtotfilop)" auf Seite 465
min_priv_mem	Mittel	„Minimaler reservierter privater Speicher (min_priv_mem)" auf Seite 433
mon_heap_sz	Niedrig	„Zwischenspeicher für den Datenbanksystem-monitor (mon_heap_sz)" auf Seite 443
nname	Keine	„NetBIOS-Knotenname (nname)" auf Seite 521
notifylevel	Niedrig	„Aufzeichnungsebene (notifylevel)" auf Seite 546
numdb	Niedrig	„Maximale Anzahl gleichzeitig aktiver Datenbanken (numdb)" auf Seite 550
num_initagents	Mittel	„Anfangswert für die Anzahl Agenten im Pool (num_initagents)" auf Seite 473
num_initdaris	Mittel	„Anfangsanzahl der abgeschirmten DARI-Prozesse im Pool (num_initdaris)" auf Seite 477
num_poolagents	Hoch	„Größe des Agentenpools (num_poolagents)" auf Seite 472
objectname	Keine	„Objektname für IPX/SPX-DB2-Server (objectname)" auf Seite 525
priv_mem_thresh	Mittel	„Schwellenwert für privaten Speicher (priv_mem_thresh)" auf Seite 434
query_heap_sz	Mittel	„Größe des Abfragezwischenspeichers (query_heap_sz)" auf Seite 428
restbufsz	Mittel	„Standardgröße für Wiederherstellungspuffer (restbufsz)" auf Seite 414
resync_interval	Keine	„Intervall für Transaktionsresynchronisation (resync_interval)" auf Seite 500
route_obj_name	Keine	„Name des Leitweginformationsobjekts (route_obj_name)" auf Seite 529
rqrioblk	Hoch	„E/A-Blockgröße für Clients (rqrioblk)" auf Seite 440
sheapthres	Hoch	„Schwellenwert für Sortierspeicher (sheapthres)" auf Seite 423
spm_log_file_sz	Niedrig	„Protokolldateigröße für SPM (spm_log_file_sz)" auf Seite 502
spm_log_path	Mittel	„Protokolldateipfad für SPM (spm_log_path)" auf Seite 501
spm_max_resync	Niedrig	„SPM-Maximum für Resynchronisationsagenten (spm_max_resync)" auf Seite 503

Tabelle 17. Konfigurierbare Konfigurationsparameter des Datenbankmanagers (Forts.)

Parameter	Leistungsrelevanz	Zusätzliche Informationen
spm_name	Keine	„Name des Synchronisationspunktmanagers (spm_name)" auf Seite 502
ss_logon	Keine	„LOGON für DB2START/DB2STOP erforderlich (ss_logon)" auf Seite 565
start_stop_time	Niedrig	„db2start/db2stop-Zeitlimit (start_stop_time)" auf Seite 541
svcename	Keine	„TCP/IP-Servicename (svcename)" auf Seite 522
sysadm_group	Keine	„SYSADM-Gruppenname (sysadm_group)" auf Seite 557
sysctrl_group	Keine	„SYSCTRL-Gruppenname (sysctrl_group)" auf Seite 559
sysmaint_group	Keine	„SYSMAINT-Gruppenname (sysmaint_group)" auf Seite 560
tm_database	Keine	„Name für Transaktionsmanagerdatenbank (tm_database)" auf Seite 499
tp_mon_name	Keine	„Name des TP-Monitors (tp_mon_name)" auf Seite 552
tpname	Keine	„APPC-Transaktionsprogrammname (tpname)" auf Seite 523
trust_allclnts	Keine	„Alle Clients akzeptieren (trust_allclnts)" auf Seite 565
trust_clntauth	Keine	„Identifikationsüberprüfung für gesicherte Clients (trust_clntauth)" auf Seite 566
udf_mem_sz	Niedrig	„Größe des gemeinsamen UDF-Speichers (udf_mem_sz)" auf Seite 430
Anmerkung: Der Parameter *cpuspeed* kann sich wesentlich auf die Leistung auswirken, jedoch sollten Sie den Standardwert verwenden, sofern nicht spezielle Umstände vorliegen, wie sie in der Parameterbeschreibung dokumentiert sind.		

Tabelle 18. Informative Konfigurationsparameter des Datenbankmanagers

Parameter	Zusätzliche Informationen
nodetype	„Knotenart des Systems (nodetype)" auf Seite 554
release	„Release-Stand der Datenbankkonfiguration (release)" auf Seite 505

Datenbankparameter

Parameter für eine einzelne Datenbank werden in einer Konfigurationsdatei namens SQLDBCON gespeichert. Diese Datei wird zusammen mit anderen Steuerdateien für die Datenbank im Verzeichnis SQLnnnnn gespeichert, wobei nnnnn eine Nummer ist, die bei der Erstellung der Datenbank zugeordnet wurde. (Weitere Informationen zur Speicherposition dieses Verzeichnisses finden Sie im Abschnitt zu physischen Datenbankverzeichnissen im Handbuch *Systemverwaltung: Konzept*.) Jede Datenbank hat eine eigene Konfigurationsdatei, und die meisten Parameter in der Datei geben an, wie viele Ressourcen der betreffenden Datenbank zugeordnet werden. Die Datei enthält außerdem beschreibende Informationen sowie Markierungen, die den Status der Datenbank angeben.

Die Datei SQLDBCON kann nicht direkt editiert werden, sondern kann lediglich mit Hilfe einer bereitgestellten Anwendungsprogrammierschnittstelle (API) oder mit einem Tool, das diese API aufruft, geändert oder angezeigt werden.

Achtung: Wenn Sie die Datei mit anderen Methoden als denen, die von DB2 vorgesehen sind, editieren, wird die Datenbank möglicherweise unbrauchbar. **Es ist äußerst ratsam**, diese Datei nur mit den von DB2 unterstützten und dokumentierten Methoden zu ändern.

Die folgenden drei Methoden können zum Anzeigen, Aktualisieren und Zurücksetzen der Konfigurationsparameter von Datenbanken verwendet werden:

- Die Verwendung der Steuerzentrale. Die DB2-Steuerzentrale bietet das Notizbuch zum Konfigurieren einer Datenbank und den *Assistent: Leistungskonfiguration* zum Ändern der Werte der Konfigurationsparameter. Dieser Assistent generiert Werte für Parameter auf der Grundlage Ihrer Antworten zu einer Reihe von Fragen, z. B. Auslastung und Transaktionsart der Datenbank. In der Online-Hilfefunktion der Steuerzentrale finden Sie Informationen zur Verwendung dieser Schnittstellen.

  In einer Umgebung mit partitionierten Datenbanken ist die Datei SQLDBCON für jede Datenbankpartition vorhanden. Das Notizbuch zum Konfigurieren einer Datenbank in der Steuerzentrale ändert den Wert aller Partitionen, wenn Sie es vom Datenbankobjekt in der Baumstruktursicht der Steuerzentrale aus starten. Wenn Sie das Notizbuch von einem Datenbankpartitionsobjekt aus starten, ändert es nur die Werte für die betreffende Partition. (Es wird jedoch empfohlen, dass die Werte der Konfigurationsparameter in allen Partitionen übereinstimmen.)

  Anmerkung: Der *Assistent: Leistungskonfiguration* ist in der Umgebung mit partitionierten Datenbanken nicht verfügbar.

- Die Verwendung des Befehlszeilenprozessors. Befehle zum Ändern der Einstellungen können schnell und bequem eingegeben werden. Weitere Informationen zu den folgenden Befehlen finden Sie im Handbuch *Command Reference*:
  - GET DATABASE CONFIGURATION (oder GET DB CFG)
  - UPDATE DATABASE CONFIGURATION (oder UPDATE DB CFG)
  - RESET DATABASE CONFIGURATION (oder RESET DB CFG)
- Die Verwendung von Anwendungsprogrammierschnittstellen (APIs). Die APIs können leicht aus einem Programm in der Host-Programmiersprache aufgerufen werden. Weitere Informationen finden Sie im Handbuch *Administrative API Reference*.

Aktualisierte Werte der meisten änderbaren Parameter treten nicht in Kraft, während Anwendungen mit der Datenbank verbunden sind. Alle Anwendungen müssen zunächst ihre Verbindung zur Datenbank beenden. (Wenn die Datenbank aktiviert war, muss sie inaktiviert und anschließend erneut aktiviert werden.) Bei der Herstellung der ersten neuen Verbindung zur Datenbank werden die Änderungen wirksam. Beachten Sie, dass einige Parameteränderungen (z. B. für *newlogpath*, *logfilsiz* oder *logprimary*) aufgrund des Systemaufwands für die Speicherzuordnung merklich mehr Zeit benötigen, um in Kraft zu treten. Es kann sinnvoll sein, eine Testverbindung zur Datenbank herzustellen, so dass die Änderung zum Zeitpunkt dieser Testverbindung aktiviert wird und der entsprechende Systemaufwands andere Benutzer nicht beeinträchtigt. Wenn Sie wegen des hier besprochenen Systemaufwands Bedenken haben, sollten Sie erwägen, den Befehl ACTIVATE DATABASE (siehe das Handbuch *Command Reference*) zu verwenden.

Anmerkung: Sie müssen die Verbindung zur Datenbank nicht unterbrechen, wenn Sie den Parameter *mincommit* aktualisieren; dieser Parameter wird automatisch aktualisiert, wenn Sie seinen Wert ändern.

Das Ändern einiger Konfigurationsparameter der Datenbank kann den Zugriffsplan beeinflussen, der vom SQL-Optimierungsprogramm gewählt wird. Diese Datenbankparameter werden im Abschnitt „Konfigurationsparameter mit Auswirkung auf die Abfrageoptimierung" auf Seite 103 behandelt. Nach der Änderung eines der hier behandelten Parameter sollten Sie in Betracht ziehen, die Anwendungen erneut zu binden, um sicherzustellen, dass der beste Zugriffsplan für die SQL-Anweisungen verwendet wird. Weitere Informationen über den Befehl BIND finden Sie im Handbuch *Command Reference*.

Obwohl die neuen Parameterwerte möglicherweise nicht sofort in Kraft treten, werden beim Anzeigen der Parametereinstellungen stets die zuletzt aktualisierten Werte angezeigt.

Anmerkung: Eine Reihe von Konfigurationsparametern für Datenbanken (z. B. *userexit*) besitzen laut Beschreibung in der Hilfe und anderen DB2-Handbüchern die zulässigen Werte „Yes" oder „No" bzw. „On" oder „Off". Zur Vermeidung von Unklarheiten sei hier vermerkt, dass „Yes" als äquivalent zu „On" und „No" als äquivalent zu „Off" anzusehen sind.

Überblick über die Konfigurationsparameter der Datenbank

In der folgenden Tabelle sind die Parameter der Konfigurationsdatei für die Datenbank aufgeführt. Wenn Sie Konfigurationsparameter der Datenbank ändern möchten, lesen Sie die detaillierten Informationen zu den Parametern.

In der Spalte „Leistungsrelevanz" der folgenden Tabelle ist vermerkt, in welchem relativen Ausmaß sich jeder Parameter in Bezug auf die Systemleistung auswirken kann. Diese Spalte kann allerdings nicht für alle Arten von Umgebungen gültige Angaben enthalten. Sie sollten die Informationen als allgemeine Hinweise betrachten.

- **Hoch** — gibt an, dass ein Parameter wesentliche Auswirkungen auf die Leistung haben kann. Die Werte für diese Parameter müssen sehr eingehend überlegt werden. In einigen Fällen kann das bedeuten, dass Sie die vorgegebenen Standardwerte übernehmen sollten.
- **Mittel** — gibt an, dass der Parameter unter Umständen Auswirkungen auf die Leistung haben kann. Sie sollten von Ihrer spezifischen Umgebung und Ihren Anforderungen ausgehend entscheiden, wie viel Aufwand in die Optimierung dieser Parameter investiert werden sollte.
- **Niedrig** — gibt an, dass der Parameter weniger allgemeine bzw. weniger bedeutende Auswirkungen auf die Leistung hat.
- **Keine** — gibt an, dass der Parameter keine direkten Auswirkungen auf die Leistung hat. Da diese Parameter nicht leistungsrelevant sind, müssen sie nicht optimiert werden, sie können jedoch in anderer Hinsicht für die Systemkonfiguration von Bedeutung sein, z. B. zur Einrichtung der Übertragungsunterstützung.

Tabelle 19. Konfigurierbare Konfigurationsparameter für die Datenbank

Parameter	Leistungsrelevanz	Zusätzliche Informationen
app_ctl_heap_sz	Mittel	„Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz)" auf Seite 420
applheapsz	Mittel	„Zwischenspeicher für Anwendungen (applheapsz)" auf Seite 426
autorestart	Niedrig	„Automatischer Neustart aktiviert (autorestart)" auf Seite 492
avg_appls	Hoch	„Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)" auf Seite 463
buffpage	Hoch (wenn aktiv)	„Größe des Pufferpools (buffpage)" auf Seite 404

Tabelle 19. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Leistungsrelevanz	Zusätzliche Informationen
catalogcache_sz	Mittel	„Katalog-Cache-Größe (catalogcache_sz)" auf Seite 409
chngpgs_thresh	Hoch	„Schwellenwert für geänderte Seiten (chngpgs_thresh)" auf Seite 453
copyprotect	Keine	„Kopierschutz aktiviert (copyprotect)" auf Seite 508
dbheap	Mittel	„Zwischenspeicher für Datenbank (dbheap)" auf Seite 408
dft_degree	Hoch	„Grad der Parallelität (dft_degree)" auf Seite 516
dft_extent_sz	Mittel	„Standardwert für EXTENTSIZE bei Tabellenbereichen (dft_extent_sz)" auf Seite 459
dft_loadrec_ses	Mittel	„Standardanzahl von Sitzungen für Wiederherstellung (dft_loadrec_ses)" auf Seite 494
dft_prefetch_sz	Mittel	„Standardwert für PREFETCHSIZE (dft_prefetch_sz)" auf Seite 458
dft_queryopt	Mittel	„Standardabfrageoptimierungsklasse (dft_queryopt)" auf Seite 517
dft_refresh_age	Mittel	„Standardaktualisierungsalter (dft_refresh_age)" auf Seite 518
dft_sqlmathwarn	Keine	„Bei arithmetischen Ausnahmebedingungen fortsetzen (dft_sqlmathwarn)" auf Seite 515
dir_obj_name	Keine	„Verzeichnisobjektname (dir_obj_name)" auf Seite 528
discover_db	Mittel	„Discovery-Unterstützung für diese Datenbank (discover_db)" auf Seite 531
dlchktime	Mittel	„Intervall für Prüfung gegenseitiger Sperren (dlchktime)" auf Seite 448
dl_expint	Keine	„Ablaufintervall für Data Link-Zugriffs-Token (dl_expint)" auf Seite 508
dl_num_copies	Keine	„Anzahl Data Link-Kopien (dl_num_copies)" auf Seite 509
dl_time_drop	Keine	„Data Links-Zeit nach DROP (dl_time_drop)" auf Seite 509
dl_token	Niedrig	„Data Links-Token-Algorithmus (dl_token)" auf Seite 510
dl_upper	Keine	„Data Links-Token in Großbuchstaben (dl_upper)" auf Seite 510

Tabelle 19. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Leistungsrelevanz	Zusätzliche Informationen
dyn_query_mgmt	Niedrig	„Dynamische SQL-Abfrageverwaltung (dyn_query_mgmt)" auf Seite 504
estore_seg_sz	Mittel	„Segmentgröße für erweiterten Speicher (estore_seg_sz)" auf Seite 460
indexrec	Mittel	„Zeitpunkt für Indexneuerstellung (indexrec)" auf Seite 493
indexsort	Niedrig (s. Anm. auf Seite 401)	„Markierung für Indexsortierung (indexsort)" auf Seite 457
locklist	Hoch bei Einfluss auf die Eskalation	„Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
locktimeout	Mittel	„Zeitlimit für Sperren (locktimeout)" auf Seite 451
logbufsz	Hoch	„Protokollpuffergröße (logbufsz)" auf Seite 410
logfilsiz	Mittel	„Protokolldateigröße (logfilsiz)" auf Seite 478
logprimary	Mittel	„Anzahl primärer Protokolldateien (logprimary)" auf Seite 480
logretain	Niedrig	„Protokollspeicherung für Wiederherstellung (logretain)" auf Seite 490
logsecond	Mittel	„Anzahl sekundärer Protokolldateien (logsecond)" auf Seite 482
maxappls	Mittel	„Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461
maxfilop	Mittel	„Maximale Anzahl offener Datenbankdateien pro Anwendung (maxfilop)" auf Seite 464
maxlocks	Hoch bei Einfluss auf die Eskalation	„Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450
mincommit	Hoch	„Anzahl der Gruppenfestschreibungen (mincommit)" auf Seite 486
min_dec_div_3	Hoch	„3 Kommastellen bei Dezimaldivision (min_dec_div_3)" auf Seite 438
newlogpath	Niedrig	„Datenbankprotokollpfad ändern (newlogpath)" auf Seite 483
num_db_backups	Keine	„Anzahl der Datenbanksicherungen (num_db_backups)" auf Seite 495
num_estore_segs	Mittel	„Segmentanzahl für erweiterten Speicher (num_estore_segs)" auf Seite 460
num_freqvalues	Niedrig	„Anzahl der häufigsten Werte (num_freqvalues)" auf Seite 518

Tabelle 19. Konfigurierbare Konfigurationsparameter für die Datenbank (Forts.)

Parameter	Leistungsrelevanz	Zusätzliche Informationen
num_iocleaners	Hoch	„Anzahl asynchroner Seitenlöschfunktionen (num_iocleaners)" auf Seite 454
num_ioservers	Hoch	„Anzahl von E/A-Servern (num_ioservers)" auf Seite 456
num_quantiles	Niedrig	„Anzahl der Quantile für Spalten (num_quantiles)" auf Seite 519
pckcachesz	Hoch	„Größe des Paket-Cache (pckcachesz)" auf Seite 418
rec_his_retentn	Keine	„Aufbewahrungszeitraum für Wiederherstellungsprotokoll (rec_his_retentn)" auf Seite 496
seqdetect	Hoch	„Markierung für Sequenzerkennung (seqdetect)" auf Seite 457
softmax	Mittel	„Vor dem bedingten Prüfpunkt zu schreibende Protokollsätze (softmax)" auf Seite 487
sortheap	Hoch	„Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422
stat_heap_sz	Niedrig	„Größe des Statistikzwischenspeichers (stat_heap_sz)" auf Seite 427
stmtheap	Mittel	„SQL-Anweisungszwischenspeicher (stmtheap)" auf Seite 425
trackmod	Niedrig	„Geänderte Seiten protokollieren (trackmod)" auf Seite 497
tsm_mgmtclass	Keine	„Tivoli Storage Manager-Verwaltungsklasse (tsm_mgmtclass)" auf Seite 497
tsm_nodename	Keine	„Tivoli Storage Manager-Knotenname (tsm_nodename)" auf Seite 498
tsm_owner	Keine	„Tivoli Storage Manager-Eignername (tsm_owner)" auf Seite 498
tsm_password	Keine	„Tivoli Storage Manager-Kennwort (tsm_password)" auf Seite 497
userexit	Niedrig	„Benutzerausgang für Protokollierung aktivieren (userexit)" auf Seite 491
util_heap_sz	Niedrig	„Zwischenspeicher für Dienstprogramme (util_heap_sz)" auf Seite 412
Anmerkung: Das Ändern des Parameters *indexsort* auf einen vom Standardwert abweichenden Wert kann sich negativ auf die Leistung für das Erstellen von Indizes auswirken. Sie sollten immer versuchen, den Standardwert für diesen Parameter zu verwenden.		

Tabelle 20. Informative Konfigurationsparameter für die Datenbank

Parameter	Zusätzliche Informationen
backup_pending	„Sicherung anstehend (backup_pending" auf Seite 511
codepage	„Codepage für die Datenbank (codepage)" auf Seite 507
codeset	„Codierter Zeichensatz für die Datenbank (codeset)" auf Seite 506
collate_info	„Informationen zur Sortierfolge (collate_info)" auf Seite 507
country	„Landescode der Datenbank (country)" auf Seite 506
database_consistent	„Datenbank ist konsistent (database_consistent)" auf Seite 511
database_level	„Release-Stand der Datenbank (database_level)" auf Seite 505
log_retain_status	„Status der Protokollspeicherung für Wiederherstellung (log_retain_status)" auf Seite 512
loghead	„Erste aktive Protokolldatei (loghead)" auf Seite 485
logpath	„Pfad zu Protokolldateien (logpath)" auf Seite 485
multipage_alloc	„Zuordnung aus mehreren Seiten bestehender Datei aktiv (multipage_alloc)" auf Seite 513
numsegs	„Standardanzahl von SMS-Behältern (numsegs)" auf Seite 459
release	„Release-Stand der Datenbankkonfiguration (release)" auf Seite 505
restore_pending	„Wiederherstellung anstehend (restore_pending)" auf Seite 513
rollfwd_pending	„Aktualisierende Wiederherstellung anstehend (rollfwd_pending)" auf Seite 512
territory	„Datenbankgebiet (territory)" auf Seite 506
user_exit_status	„Status des Benutzerausgangs für Protokollierung (user_exit_status)" auf Seite 512

Einzelheiten zu Parametern nach Funktion

Die folgenden Abschnitte enthalten zusätzliche Einzelinformationen, die Ihnen das Verständnis der Parameter erleichtern und Sie bei der Optimierung der verschiedenen Konfigurationsparameter unterstützen sollen. Die Beschreibungen der einzelnen Parameter sind nach Funktion bzw. Zweck der Parameter geordnet:

- „Kapazitätsverwaltung" auf Seite 404
- „Protokollieren und Wiederherstellung" auf Seite 478
- „Datenbankverwaltung" auf Seite 504
- „Übertragung" auf Seite 521
- „Partitionierte Datenbank" auf Seite 535
- „Exemplarverwaltung" auf Seite 544.

Die Beschreibung der Parameter enthält jeweils folgende Informationen:

Konfigurationsart
Gibt an, welche Konfigurationsdatei die Einstellung für den Parameter enthält:

- Datenbankmanager (betrifft ein Exemplar des Datenbankmanagers und alle Datenbanken, die innerhalb dieses Exemplars definiert sind)
- Datenbank (betrifft eine bestimmte Datenbank)

Parameterart
Gibt an, ob der Parameterwert geändert werden kann oder nicht:

- *Konfigurierbar*

  Eine Bandbreite von Werten ist möglich, so dass der Parameter nach dem Kenntnisstand des Datenbankadministrators über die Anwendungen und/oder mit entsprechender Testerfahrung optimiert werden muss.
- *Informativ*

  Die Werte dieser Parameter werden nur durch den Datenbankmanager selbst geändert und enthalten Informationen, wie z.B. das Release von DB2, unter dem eine Datenbank erstellt wurde, oder die Angabe, dass eine erforderliche Sicherung ansteht.

Kapazitätsverwaltung

Es gibt eine Reihe von Konfigurationsparametern sowohl auf Datenbank- als auch auch Datenbankmanagerebene, die Auswirkungen auf die Leistung des Systems haben können. Diese Parameter können in folgende Kategorien eingeteilt werden:

- „Gemeinsam benutzter Speicher der Datenbank"
- „Gemeinsamer Anwendungsspeicher" auf Seite 420
- „Privater Agentenspeicher" auf Seite 422
- „Agenten-/Anwendungskommunikationsspeicher" auf Seite 436
- „Exemplarspeicher des Datenbankmanagers" auf Seite 443
- „Sperren" auf Seite 448
- „Ein-/Ausgabe und Speicher" auf Seite 453
- „Agenten" auf Seite 461
- „Gespeicherte Prozeduren (DARI)" auf Seite 474.

Eine Einführung in die Speicherverwaltung von DB2 finden Sie in „Verwendung des Speichers durch DB2" auf Seite 283.

Gemeinsam benutzter Speicher der Datenbank

Die folgenden Parameter wirken sich auf den globalen Datenbankspeicher aus, der auf dem System zugeordnet wird:

- „Größe des Pufferpools (buffpage)".
- „Zwischenspeicher für Datenbank (dbheap)" auf Seite 408.
- „Katalog-Cache-Größe (catalogcache_sz)" auf Seite 409.
- „Protokollpuffergröße (logbufsz)" auf Seite 410.
- „Zwischenspeicher für Dienstprogramme (util_heap_sz)" auf Seite 412.
- „Standardgröße für Sicherungspuffer (backbufsz)" auf Seite 413.
- „Standardgröße für Wiederherstellungspuffer (restbufsz)" auf Seite 414.
- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415.
- „Größe des Paket-Cache (pckcachesz)" auf Seite 418.

Lesen Sie im Abschnitt „Verwendung des Speichers durch DB2" auf Seite 283 die Informationen dazu, welches Verhältnis zwischen globalem Datenbankspeicher und dem übrigen Speicher besteht, der vom Datenbankmanager zugeordnet wird.

Größe des Pufferpools (buffpage)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX-Plattformen (32 Bit)
1 000 [2 — 524 288]

UNIX-Plattformen (64 Bit)
1 000 [2 — 2 147 483 647]

OS/2 und Windows NT
250 [2 — 524 288]

Maßeinheit
Seiten

Zuordnung
Wenn die erste Anwendung die Verbindung zur Datenbank herstellt

Freigabe
Wenn die letzte Anwendung die Verbindung zur Datenbank beendet

Zugehörige Parameter

- „Schwellenwert für geänderte Seiten (chngpgs_thresh)" auf Seite 453
- „Zwischenspeicher für Datenbank (dbheap)" auf Seite 408
- „Anzahl asynchroner Seitenlöschfunktionen (num_iocleaners)" auf Seite 454

Jede Datenbank verfügt über mindestens einen Pufferpool (IBMDEFAULTBP, der bei der Erstellung der Datenbank erstellt wird), sie kann jedoch auch mehrere haben. Alle Pufferpools befinden sich im globalen Speicher, der für alle Anwendungen, die die Datenbank verwenden, verfügbar ist. Der Speicher wird auf dem System zugeordnet, auf dem sich die Datenbank befindet. Wenn die Pufferpools groß genug sind, um die angeforderten Daten zu speichern, sind weniger Zugriffe auf die Platte erforderlich. Sind hingegen die Pufferpools nicht groß genug, kann die allgemeine Leistung der Datenbank erheblich beeinträchtigt werden. Außerdem kann die Verarbeitungsgeschwindigkeit des Datenbankmanagers infolge der umfangreichen Festplattenoperationen (E/A), die zur Verarbeitung der Daten für Ihre Anwendung erforderlich werden, auf die Geschwindigkeit der E/A-Operationen für die Festplatte sinken. Mit dem Parameter *buffpage* wird die Größe eines Pufferpools gesteuert, wenn die SQL-Anweisung CREATE BUFFERPOOL oder ALTER BUFFERPOOL mit der Angabe NPAGES -1 ausgeführt wurde. Andernfalls wird der Parameter *buffpage* ignoriert und der Pufferpool mit der Seitenanzahl erstellt, die im Parameter NPAGES angegeben wird.

Führen Sie die folgende Anweisung aus, wenn Sie feststellen möchten, ob der Parameter *buffpage* für einen Pufferpool aktiv ist:

```
SELECT * from SYSCAT.BUFFERPOOLS.
```

Jeder Pufferpool, der den Wert NPAGES -1 hat, verwendet den Parameter *buffpage*.

Zwischen der Größe des Pufferpools und der Menge an Speicher, die anderen Systembenutzern zur Verfügung gestellt wird, muss ein geeigneter Kompromiss gefunden werden. Der Speicherbedarf von Datenbank-Servern auf Mehrbenutzersystemen mit hohen Transaktionsgeschwindigkeiten ist so groß, dass Datenbank-Server und Datei- oder Kommunikations-Server häufig voneinander getrennt auf verschiedenen Systemen betrieben werden.

Wenn Ihre Abfragen auf Kurznamen zugreifen, erwägen Sie das Erhöhen der Pufferpoolgröße unter folgenden Umständen:
- Das Optimierungsprogramm entscheidet, dass die meisten oder alle Operationen lokal verarbeitet werden. Bei der Verarbeitung einer Abfrage verschiebt das Optimierungsprogramm in der Regel alle Operationen, bei denen dies möglich ist, an die Datenquelle. Zum Beispiel wird der Operator GROUP BY gewöhnlich an der Datenquelle ausgewertet. Es kann jedoch sein, dass die Erstellung der Tabelle unter DB2 und die lokale Ausführung einer Operation mit dem geringsten Aufwand verbunden ist. Dies kann der Fall sein, wenn die DB2-Server-Workstation leistungsfähiger als die Workstation der Datenquelle ist.
- Sortieroperationen müssen lokal ausgeführt werden. Abfragen mit Kurznamen werden nach der DB2-Sortierfolge sortiert. Wenn eine Datenquelle nicht die gleiche Sortierfolge aufweist, werden alle Sortieroperationen lokal ausgeführt.

Alle Pufferpools werden zugeordnet, wenn die erste Anwendung die Verbindung zur Datenbank herstellt oder die Datenbank explizit aktiviert wird. Wenn eine Anwendung Daten aus einer Datenbank abfragt, werden die Seiten mit diesen Daten von der Platte in einen der Pufferpools übertragen. (Beachten Sie, dass Datenbankdaten innerhalb der Tabellen auf der Platte in Seiten gespeichert werden.) Die Seiten werden erst wieder auf Platte geschrieben, wenn die Seite geändert wird und eines der folgenden Ereignisse eintritt:
- Alle Anwendungen trennen die Verbindung zur Datenbank.
- Die Datenbank wird explizit inaktiviert.
- Die Datenbank wird in den Wartemodus versetzt (d. h., alle verbundenen Anwendungen haben ihre Daten festgeschrieben).
- Der Speicherbereich ist für eine andere Seite erforderlich, die in den Pufferpool gelesen werden muss.
- Eine Seitenlöschfunktion ist verfügbar (*num_iocleaners*) und wurde vom Datenbankmanager aktiviert.

Empfehlungen:
- Anstelle des Konfigurationsparameters *buffpage* können Sie auch die SQL-Anweisungen CREATE BUFFERPOOL und ALTER BUFFERPOOL zum Erstellen und Ändern von Pufferpools und ihrer Größen verwenden.

- Die Größe des Pufferpools wird vom Optimierungsprogramm zur Festlegung des Zugriffsplans verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).
- Da die Größen aller Pufferpools die Leistung der Datenbank stark beeinträchtigen können, sollten Sie die folgenden Faktoren beachten, um sicherzustellen, dass Seiten nicht zu häufig ausgelagert werden müssen:
  - Die Größe des auf Ihrem System installierten Speichers
  - Die Menge an Speicher, die für andere Anwendungen erforderlich ist, die parallel zum Datenbankmanager auf demselben System ausgeführt werden.

  Eine *Seitenauslagerung* tritt auf, wenn für die Seite, auf die zugegriffen wird, nicht genügend Hauptspeicher verfügbar ist. Dies führt dazu, dass die Seite in einen temporären Speicher auf der Platte geschrieben („ausgelagert") wird, um Platz für eine andere Seite zu schaffen. Wird die Seite im temporären Speicher wieder benötigt, wird sie wieder in den Hauptspeicher „eingelagert".
- Sie können bis zu 75% des Hauptspeichers einer Maschine für die Pufferpools der Datenbank reservieren, wenn Folgendes zutrifft:
  - Sie haben ein Mehrbenutzersystem.
  - Sie haben ein System, das nur als Datenbank-Server verwendet wird.
  - Auf dem System wird häufig wiederholt auf dieselben Daten- und Indexseiten zugegriffen.
  - Auf dem System befindet sich nur eine Datenbank.
- Für jede zugeordnete Seite des Pufferpools wird ein Bereich im Datenbankzwischenspeicher für interne Steuerstrukturen verwendet.

  Wenn die Gesamtgröße des Pufferpools (oder der Pufferpools) erhöht wird, müssen Sie eventuell auch den Wert des Parameters *dbheap* erhöhen.
- Wenn Sie in einer Umgebung mit zusammengeschlossenen Datenbanken arbeiten und die Sortierfolge der Datenquelle mit der DB2-Sortierfolge übereinstimmt, stellen Sie sicher, dass die Serveroption *collating_sequence* entsprechend festgelegt ist. Das heißt, dass die Option *collating_sequence* auf „Y" gesetzt sein sollte. Sie können zusammengeschlossene Datenbanken mit einer bestimmten Sortierfolge erstellen, die der Sortierfolge der Datenquellen entspricht. Diese Lösung kann die Leistung erhöhen, wenn alle Datenquellen die gleiche Sortierfolge besitzen bzw. wenn die meisten oder alle Spaltenfunktionen auf Datenquellen gerichtet sind, die die gleiche Sortierfolge verwenden. Wenn eine Datenquelle eine Sortierfolge besitzt, die von der Sortierfolge von DB2 abweicht, können die meisten Operationen, die von der Sortierfolge von DB2 abhängig sind, nicht fern an einer Datenquelle ausgewertet werden. Weitere Informationen über diese Serveroption finden Sie im Handbuch *Systemverwaltung: Implementierung*.

Mit dem Datenbanksystemmonitor können Sie zur Optimierung Ihrer Pufferpools die Effektivität der Zugriffe auf Pufferpools ermitteln. Weitere Informationen finden Sie im Handbuch *System Monitor Guide and Reference*.

Zwischenspeicher für Datenbank (dbheap)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 1200 [32 – 524 288]

OS/2 und Windows NTDatenbank-Server mit lokalen und fernen Clients
600 [32 – 524 288]

OS/2 und Windows NTDatenbank-Server mit lokalen Clients
300 [32 – 524 288]

Maßeinheit Seiten (4 KB)

Zuordnung Bei der ersten Verbindung zu einer Datenbank

Freigabe Wenn die letzte Anwendung die Verbindung zur Datenbank beendet

Zugehörige Parameter

- „Katalog-Cache-Größe (catalogcache_sz)" auf Seite 409
- „Protokollpuffergröße (logbufsz)" auf Seite 410

Für jede Datenbank gibt es einen Datenbankzwischenspeicher, der vom Datenbankmanager für alle Anwendungen, die auf die Datenbank zugreifen, verwendet wird. Er enthält Steuerblockdaten für Tabellen, Indizes, Tabellenbereiche und Pufferpools. Er enthält außerdem Speicherbereich für den Protokollpuffer (*logbufsz*) und den Katalog-Cache (*catalogcache_sz*). Daher ist die Größe des Zwischenspeichers von der Anzahl der zu einem bestimmten Zeitpunkt in ihm gespeicherten Steuerblöcke abhängig. Die Steuerblockdaten werden im Zwischenspeicher gehalten, bis alle Anwendungen die Verbindung zur Datenbank getrennt haben.

Der Mindestspeicherbereich, den der Datenbankmanager zu Beginn benötigt, wird bei der Herstellung der ersten Verbindung zugeordnet. Der Datenbereich wird nach Bedarf bis zu der Maximalgröße erweitert, die durch den Parameter *dbheap* definiert ist.

Empfehlung: Dieser Wert muss erhöht werden, wenn eine Anwendung einen Fehler empfängt, der anzeigt, dass zur Verarbeitung einer Anweisung nicht genügend Speicher im Datenbankzwischenspeicher zur Verfügung steht.

Mit Hilfe des Datenbanksystemmonitors können Sie die größte Speichermenge ermitteln, die für den Datenbankzwischenspeicher verwendet wurde. In der Beschreibung zum Monitorelement *db_heap_top (Maximaler zugeordneter Datenbankzwischenspeicher)* im Handbuch *System Monitor Guide and Reference* finden Sie weitere Informationen.

Bei der Einstellung dieses Parameters sollten Sie auch folgende Werte berücksichtigen:

- Den Wert des Parameters *logbufsz*, da der Protokollpuffer aus dem Datenbankzwischenspeicher zugeordnet wird.
- Den Wert des Parameters *catalogcache_sz*, da der Katalog-Cache aus dem Datenbankzwischenspeicher zugeordnet wird.

Katalog-Cache-Größe (catalogcache_sz)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 64 [1 – 60 000]

OS/2 und Windows NTDatenbank-Server mit lokalen und fernen Clients
32 [1 – 60 000]

OS/2 und Windows NTDatenbank-Server mit lokalen Clients
16 [1 – 60 000]

Maßeinheit Seiten (4 KB)

Zugehörige Parameter

- „Zwischenspeicher für Datenbank (dbheap)" auf Seite 408
- „Protokollpuffergröße (logbufsz)" auf Seite 410
- „Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz)" auf Seite 420

Mit diesem Parameter wird der maximale Speicherbereich angegeben, den der Katalog-Cache aus dem Datenbankzwischenspeicher (*dbheap*) verwenden kann. Der Katalog-Cache dient zum Speichern der Tabellendeskriptorinforma-

tionen, die verwendet werden, wenn während der Kompilierung einer SQL-Anweisung auf eine Tabelle, eine Sicht oder einen Aliasnamen verwiesen wird.

Die Verwendung dieses Cache kann die Leistung beim Binden von SQL-Anweisungen (auch dynamisches SQL) verbessern, wenn auf dieselben Tabellen, Sichten oder Aliasnamen bereits in vorherigen Anweisungen verwiesen wurde. Deskriptorinformationen für deklarierte temporäre Tabellen werden nicht im Katalog-Cache gespeichert; stattdessen wird der Zwischenspeicher zur Anwendungssteuerung verwendet.

Durch die Ausführung von DDL-Anweisungen (Data Definition Language - Datendefinitionssprache) für eine Tabelle wird der Eintrag dieser Tabelle im Katalog-Cache gelöscht. Andernfalls wird ein Tabelleneintrag im Cache behalten, bis der Speicherplatz für eine andere Tabelle benötigt wird. Der Eintrag wird jedoch erst aus dem Cache entfernt, nachdem alle Arbeitseinheiten, die auf diese Tabelle verweisen, beendet wurden.

Empfehlung: Verwenden Sie zu Beginn den Standardwert, und optimieren Sie den Wert mit Hilfe des Datenbanksystemmonitors.

Informationen zu folgenden Monitorelementen finden Sie im Handbuch *System Monitor Guide and Reference*:
- *cat_cache_lookups* (Suchoperationen im Katalog-Cache)
- *cat_cache_inserts* (Einfügungen im Katalog-Cache)
- *cat_cache_overflows* (Überläufe des Katalog-Cache)
- *cat_cache_heap_full* (Voller Katalog-Cache-Zwischenspeicher)

Mit Hilfe dieser Elemente des Datenbanksystemmonitors können Sie ermitteln, ob Sie den Wert dieses Konfigurationsparameters anpassen sollten. Bei der Optimierung dieses Parameters sollten Sie den Wert nur in kleinen Schritten vergrößern, z. B. um jeweils zwei Seiten.

Im allgemeinen ist ein größerer Cache erforderlich, wenn eine Arbeitseinheit mehrere dynamische SQL-Anweisungen enthält oder wenn Pakete gebunden werden, die zahlreiche statische SQL-Anweisungen enthalten.

Berücksichtigen Sie bei der Einstellung des Katalog-Cache auch die Größe der Protokolldateien (*logbufsz*), da sowohl *catalogcache_sz* als auch *logbufsz* aus dem Datenbankzwischenspeicher (*dbheap*) zugeordnet werden.

Protokollpuffergröße (logbufsz)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar

Standardwert [Bereich]

UNIX-Plattformen (32 Bit)
8 [4 — 4 096]

UNIX-Plattformen (64 Bit)
8 [4 — 65 535]

OS/2 und Windows NT
8 [4 — 4 096]

Maßeinheit
Seiten (4 KB)

Zugehörige Parameter

- „Katalog-Cache-Größe (catalogcache_sz)" auf Seite 409
- „Zwischenspeicher für Datenbank (dbheap)" auf Seite 408
- „Anzahl der Gruppenfestschreibungen (mincommit)" auf Seite 486

Mit diesem Parameter kann die Menge an Datenbankzwischenspeicher (der durch den Parameter *dbheap* definiert wird) angegeben werden, die als Puffer für Protokollsätze verwendet werden soll, bevor diese Protokollsätze auf die Festplatte geschrieben werden. Die Protokollsätze werden bei Eintreten eines der folgenden Umstände auf die Festplatte geschrieben:

- Eine Transaktion oder eine Gruppe von Transaktionen wird entsprechend der Angaben für den Konfigurationsparameter *mincommit* festgeschrieben.
- Der Protokollpuffer ist voll.
- Ein anderes internes Ereignis im Datenbankmanager macht das Schreiben auf die Festplatte erforderlich.

Der Wert dieses Parameters muss außerdem kleiner oder gleich dem Wert des Parameters *dbheap* sein. Durch Puffern der Protokollsätze werden E/A-Operationen für die Protokolldateien effektiver, da weniger häufig Schreiboperationen ausgeführt und bei jeder Schreiboperation mehr Protokollsätze auf die Festplatte geschrieben werden.

Empfehlung: Erhöhen Sie den Wert für die Größe dieses Pufferbereichs, wenn umfangreiche Leseaktivitäten auf einer dedizierten Protokollplatte auftreten oder die Festplatte in erheblichem Maße beansprucht wird. Wenn Sie den Wert dieses Parameters erhöhen, sollten Sie auch den Parameter *dbheap* berücksichtigen, da der Protokollpuffer einen Speicherbereich verwendet, der mit dem Parameter *dbheap* gesteuert wird. Sie können mit dem Datenbanksystemmonitor feststellen, wie viel Speicherbereich des Protokollpuffers für eine bestimmte Transaktion (oder Arbeitseinheit) verwendet wird.

Weitere Informationen finden Sie in der Beschreibung des Monitorelements *log_space_used* (Protokollbereich für Arbeitseinheit) im Handbuch *System Monitor Guide and Reference*.

Berücksichtigen Sie bei der Einstellung der Protokollpuffergröße auch die Größe des Katalog-Cache (*catalogcache_sz*), da sowohl *logbufsz* als auch *catalogcache_sz* aus dem Datenbankzwischenspeicher (*dbheap*) zugeordnet werden.

Zwischenspeicher für Dienstprogramme (util_heap_sz)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	5000 [16 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn für die Dienstprogramme des Datenbankmanagers erforderlich
Freigabe	Wenn der Speicher nicht mehr vom Dienstprogramm benötigt wird
Zugehörige Parameter	• „Standardgröße für Sicherungspuffer (backbufsz)" auf Seite 413 • „Standardgröße für Wiederherstellungspuffer (restbufsz)" auf Seite 414

Dieser Parameter gibt die maximale Speichergröße an, die gleichzeitig von den Dienstprogrammen BACKUP, RESTORE und LOAD (einschließlich Wiederherstellung) verwendet werden kann.

Empfehlung: Verwenden Sie den Standardwert, bis Ihren Dienstprogrammen nicht mehr genügend Speicher zur Verfügung steht. Wenn der Speicher nicht ausreicht, müssen Sie den Wert erhöhen. Wenn der auf Ihrem System verfügbare Speicher eingeschränkt ist, können Sie den Wert für diesen Parameter herabsetzen, um den von den Dienstprogrammen der Datenbank verwendeten Speicher zu begrenzen. Wenn ein zu niedriger Parameterwert angegeben wurde, können Sie die Dienstprogramme eventuell nicht mehr gleichzeitig ausführen. Der von Ihnen angegebene Parameterwert muss so groß sein, dass alle Puffer eingerichtet werden können, die Sie den gleichzeitig ablaufenden Dienstprogrammen zuordnen möchten.

Standardgröße für Sicherungspuffer (backbufsz)

Konfigurationsart	Datenbankmanager
Gilt für	• Datenbank-Server mit lokalen und fernen Clients • Datenbank-Server mit lokalen Clients • Partitionierter Datenbank-Server mit lokalen und fernen Clients • Satellitendatenbank-Server mit lokalen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	1024 [8 — 16 384]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn das Sicherungsprogramm aufgerufen wird
Freigabe	Wenn die Verarbeitung des Sicherungsprogramms abgeschlossen ist
Zugehörige Parameter	• „Standardgröße für Wiederherstellungspuffer (restbufsz)" auf Seite 414 • „Zwischenspeicher für Dienstprogramme (util_heap_sz)" auf Seite 412

Mit diesem Parameter wird die Größe des Puffers angegeben, der zur Sicherung der Datenbank verwendet wird, wenn die Puffergröße nicht explizit beim Aufruf des Sicherungsprogramms angegeben wird. Weitere Informationen zum Sicherungsprogramm finden Sie in *Command Reference*.

Bei der Sicherung einer Datenbank werden die Daten zunächst in einen internen Puffer kopiert. Wenn der Puffer voll ist, werden die Daten aus diesem Puffer auf den Sicherungsdatenträger geschrieben.

Durch Optimieren dieser Puffergröße kann die Leistung des Sicherungsprogramms erhöht und die Beeinträchtigung der Leistung anderer, parallel ausgeführter Datenbankoperationen minimiert werden.

Standardgröße für Wiederherstellungspuffer (restbufsz)

Konfigurationsart	Datenbankmanager
Gilt für	• Datenbank-Server mit lokalen und fernen Clients • Datenbank-Server mit lokalen Clients • Partitionierter Datenbank-Server mit lokalen und fernen Clients • Satellitendatenbank-Server mit lokalen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	1024 [8 — 16 384]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn das Wiederherstellungsprogramm aufgerufen wird
Freigabe	Wenn die Verarbeitung des Wiederherstellungsprogramms abgeschlossen ist
Zugehörige Parameter	• „Standardgröße für Sicherungspuffer (backbufsz)" auf Seite 413 • „Zwischenspeicher für Dienstprogramme (util_heap_sz)" auf Seite 412

Mit diesem Parameter wird die Größe des Puffers angegeben, der zur Wiederherstellung der Datenbank verwendet wird, wenn die Puffergröße nicht explizit beim Aufruf des Datenbankwiederherstellungsprogramms angegeben wird. Weitere Informationen zum Wiederherstellungsprogramm finden Sie in *Command Reference*.

Bei der Wiederherstellung einer Datenbank werden die Daten zunächst vom Sicherungsdatenträger in einen internen Puffer kopiert. Wenn der Puffer voll ist, werden die Daten aus diesem Puffer auf den Datenträger der Zieldatenbank geschrieben.

Durch Optimieren dieser Puffergröße kann die Leistung des Wiederherstellungsprogramms erhöht und die Beeinträchtigung der Leistung anderer, parallel ausgeführter Datenbankoperationen minimiert werden.

Maximaler Speicher für Sperrenliste (locklist)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 100 [4 – 524 288]

OS/2- und NT-Datenbank-Server mit lokalen und fernen Clients
50 [4 – 524 288]

OS/2- und NT-Datenbank-Server mit lokalen Clients 25 [4 – 60 000]

Maßeinheit Seiten (4 KB)

Zuordnung Wenn die erste Anwendung die Verbindung zur Datenbank herstellt

Freigabe Wenn die letzte Anwendung die Verbindung zur Datenbank beendet

Zugehörige Parameter

- „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450
- „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461

Dieser Parameter gibt die Speichermenge an, die für die Sperrenliste zugeordnet wird. Für jede Datenbank gibt es eine Sperrenliste, die die Sperren aller gleichzeitig mit der Datenbank verbundenen Anwendungen enthält. Das Sperren ist eine Funktion des Datenbankmanagers zur Steuerung des gleichzeitigen Zugriffs auf Daten der Datenbank durch mehrere Anwendungen. Sowohl Zeilen als auch Tabellen können gesperrt werden. Der Datenbankmanager fordert eventuell auch Sperren für interne Verwendung an.

Weitere Informationen über Sperren finden Sie im Abschnitt „Sperren" auf Seite 57.

Jede Sperre nimmt 36 oder 72 Byte der Sperrenliste in Anspruch, je nachdem, ob für das Objekt andere Sperren aktiv sind:

- 72 Byte sind erforderlich, um eine Sperre für ein Objekt zu aktivieren, das keine anderen Sperren hat.
- 36 Byte sind erforderlich, um eine Sperre für ein Objekt einzutragen, für das bereits eine vorhandene Sperre aktiv ist.

Wenn der Prozentsatz der Sperrenliste, der von einer Anwendung verwendet wird, den Wert des Parameters *maxlocks* erreicht, führt der Datenbankmanager eine Sperreneskalation von Zeilenebene auf Tabellenebene für die Sperren aus, die von dieser Anwendung aktiviert wurden (siehe unten). Obwohl der Eskalationsprozess an sich nicht viel Zeit in Anspruch nimmt, wird durch Sperren ganzer Tabellen (im Vergleich zum Sperren einzelner Zeilen) der gemeinsame Zugriff eingeschränkt und die allgemeine Datenbankleistung kann bei nachfolgenden Zugriffen auf die betroffenen Tabellen beeinträchtigt werden. Es wird empfohlen, die Größe der Sperrenliste folgendermaßen klein zu halten:

- Führen Sie häufig Festschreibungen (COMMIT-Operationen) aus, um Sperren freizugeben.
- Wenn viele Aktualisierungen ausgeführt werden sollen, sperren Sie vor den Aktualisierungen die ganze Tabelle (mit der SQL-Anweisung LOCK TABLE). Dadurch wird nur eine Sperre verwendet und der Zugriff anderer auf die zu aktualisierenden Daten verhindert. Der gemeinsame Zugriff auf die Daten wird jedoch eingeschränkt.

  Sie können den Parameter LOCKSIZE der Anweisung ALTER TABLE auch verwenden, um das Sperren einer bestimmten Tabelle zu steuern. Genauere Informationen finden Sie im Handbuch *SQL Reference*.

  Die Verwendung der Isolationsstufe RR (Wiederholtes Lesen) kann zu einer automatischen Tabellensperre führen. Weitere Informationen zu Isolationsstufen finden Sie in „Kapitel 3. Überlegungen zu Anwendungen" auf Seite 47.
- Verwenden Sie die Isolationsstufe der Cursorstabilität (CS), wenn möglich, um die Anzahl der Sperren für gemeinsamen Zugriff zu verringern. Wenn dadurch die durch die Anwendung festgelegten Integritätsanforderungen nicht beeinträchtigt werden, verwenden Sie anstatt der Cursorstabilität die Isolationsstufe UR (Nicht festgeschriebener Lesevorgang), um die Anzahl der Sperren weiter zu verringern.

Wenn die Sperrenliste voll ist, kann die Leistung herabgesetzt werden, da die Sperreneskalation mehr Tabellensperren und weniger Zeilensperren erzeugt und auf diese Weise den gemeinsamen Zugriff auf gemeinsam benutzte Objekte in der Datenbank einschränkt. Zudem kann es mehr gegenseitige Sperren zwischen Anwendungen geben (da sie alle auf eine verringerte Anzahl von Tabellensperren warten), was dazu führt, dass Transaktionen zurückgesetzt werden. Ihre Anwendung empfängt einen SQLCODE-Wert -912, wenn die maximale Anzahl von Sperranforderungen für die Datenbank erreicht wurde.

Empfehlung: Wenn die Sperreneskalationen zu Leistungseinbußen führen, müssen Sie eventuell den Wert dieses Parameters oder den Wert des Parameters *maxlocks* erhöhen. Mit dem Datenbanksystemmonitor können Sie feststellen, ob Sperreneskalationen auftreten.

Weitere Informationen finden Sie in der Beschreibung des Monitorelements *lock_escals (Sperreneskalationen)* im Handbuch *System Monitor Guide and Reference*.

Anhand der folgenden Schritte können Sie die Anzahl der Seiten, die für Ihre Sperrenliste erforderlich sind, ermitteln:

1. Berechnen Sie eine Untergrenze für die Größe der Sperrenliste:

   ```
   (512 * 36 * maxappls) / 4096
   ```

   In dieser Formel ist 512 ein Schätzwert für die durchschnittliche Anzahl von Sperren pro Anwendung, und 36 ist die Anzahl der benötigten Byte für jede Sperre eines Objekts, für das bereits eine Sperre aktiv ist.
2. Berechnen Sie eine Obergrenze für die Größe der Sperrenliste:

   ```
   (512 * 72 * maxappls) / 4096
   ```

   In dieser Formel ist 72 die Anzahl der benötigten Byte für die erste Sperre eines Objekts.
3. Schätzen Sie das Aufkommen an gemeinsamem Zugriff durch Anwendungen auf Ihre Daten, und wählen Sie nach Ihren Schätzungen einen Anfangswert für *locklist*, der zwischen der berechneten Ober- und Untergrenze liegt.
4. Mit dem Datenbanksystemmonitor können Sie, wie unten beschrieben, den Wert dieses Parameters optimieren.

Sie können mit dem Datenbanksystemmonitor die maximale Anzahl der von einer bestimmten Transaktion aktivierten Sperren feststellen.

Weitere Informationen finden Sie in der Beschreibung des Monitorelements *locks_held_top (Maximale Anzahl aktiver Sperren)* in *System Monitor Guide and Reference*.

Anhand dieser Informationen können Sie die geschätzte Anzahl der Sperren pro Anwendung bestätigen oder anpassen. Um diese Auswertung durchzuführen, müssen Sie mehrere Probeanwendungen ausführen und dabei beachten, dass die Monitordaten auf einer Transaktionsebene und nicht auf einer Anwendungsebene geliefert werden.

Der Wert des Parameters *locklist* sollte eventuell auch dann heraufgesetzt werden, wenn der Parameter *maxappls* erhöht wird oder wenn die Anwendungen, die ausgeführt werden, nur relativ selten Festschreibungen (COMMIT-Operationen) ausführen.

Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

Weitere Informationen zur Leistung von Anwendungen und der Beeinflussung der Abfrageoptimierung finden Sie in „Teil 2. Optimieren der Anwendungsleistung" auf Seite 45.

Größe des Paket-Cache (pckcachesz)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX-Plattformen (32 Bit)
-1 [-1, 32 — 128 000]

UNIX-Plattformen (64 Bit)
-1 [-1, 32 — 524 288]

OS/2 und Windows NT
-1 [-1, 32 — 128 000]

Maßeinheit Seiten (4 KB)

Zuordnung Wenn die Datenbank initialisiert wird

Freigabe Wenn die Datenbank heruntergefahren wird

Der Paket-Cache wird aus dem globalen Datenbankspeicher zugeordnet und für das Caching (Zwischenspeichern) statischer und dynamischer SQL-Anweisungen für eine Datenbank verwendet. In einem partitionierten Datenbanksystem gibt es für jede Datenbankpartition einen Paket-Cache.

Das Caching von Paketen ermöglicht dem Datenbankmanager, den internen Systemaufwand zu verringern, da der Zugriff auf die Systemkataloge beim erneuten Laden eines Pakets oder bei dynamischem SQL eine Kompilierung nicht mehr erforderlich ist. Die Abschnitte werden im Paket-Cache behalten, bis einer der folgenden Umstände eintritt:
- Die Datenbank wird heruntergefahren.
- Das Paket oder die dynamische SQL-Anweisung wird ungültig gemacht.
- Im Cache ist nicht mehr genügend Platz.

Dieses Zwischenspeichern des Abschnitts für eine statische oder dynamische SQL-Anweisung kann die Leistung besonders dann verbessern, wenn dieselbe Anweisung mehrere Male von Anwendungen verwendet wird, die mit einer Datenbank verbunden sind. Dies ist insbesondere für eine Anwendung wichtig, die Transaktionen verarbeitet.

Durch Definieren des Standardwerts (-1) wird als Wert für die Berechnung der Seitenzuordnung das Achtfache des Werts für den Konfigurationsparameter *maxappls* genommen. Wenn das Achtfache von *maxappls* jedoch kleiner als 32 ist, gilt dies nicht. In diesem Fall entspricht der Standardwert -1 für *pckcachesz* dem Wert 32.

Empfehlung: Bei der Einstellung dieses Parameters sollten Sie überlegen, ob der zusätzliche Speicher, der für den Paket-Cache reserviert wird, günstiger für einen anderen Zweck zugeordnet werden sollte, z.B. für den Pufferpool. Aus diesem Grund sollten Sie zur Optimierung dieses Parameters Vergleichstests (Benchmark-Tests) durchführen.

Die optimale Einstellung dieses Parameters ist von besonderer Bedeutung, wenn zu Beginn mehrere Abschnitte verwendet werden und nur wenige Abschnitte wiederholt ausgeführt werden. Wenn der Cache zu groß ist, wird Speicher zum Behalten von Kopien der Anfangsabschnitte verschwendet.

Informationen zu folgenden Monitorelementen finden Sie im Handbuch *System Monitor Guide and Reference* :
- *pkg_cache_lookups* (Zugriffe auf Paket-Cache)
- *pkg_cache_inserts* (Einfügungen in Paket-Cache)
- *pkg_cache_size_top* (maximale Größe des Paket-Cache)
- *pkg_cache_num_overflows* (Anzahl der Paket-Cache-Überläufe)

Mit Hilfe dieser Elemente des Datenbanksystemmonitors können Sie ermitteln, ob Sie den Wert dieses Konfigurationsparameters anpassen sollten.

Anmerkung: Der Paket-Cache ist ein Arbeits-Cache, so dass dieser Parameter nicht auf den Wert 0 gesetzt werden kann. Diesem Cache muss ausreichend Speicherplatz für alle Abschnitte der SQL-Anweisungen zugeordnet sein, die momentan ausgeführt werden. Wenn mehr als der momentan benötigte Speicherplatz zugeordnet ist, werden Abschnitte zwischengespeichert. Diese Abschnitte können einfach ausgeführt werden, wenn sie das nächste Mal benötigt werden, und müssen nicht erneut geladen oder kompiliert werden.

Der durch den Parameter *pckcachesz* angegebene Grenzwert ist ein veränderlicher Grenzwert. Dieser Grenzwert kann, falls erforderlich, überschritten werden, wenn in dem von den

Datenbanken gemeinsam benutzten Speicher noch Speicherplatz verfügbar ist. Sie können mit dem Monitorelement *pkg_cache_size_top* den Höchstwert ermitteln, bis zu dem der Paket-Cache angewachsen ist. Mit dem Monitorelement *pkg_cache_num_overflows* können Sie ermitteln, wie häufig der durch den Parameter *pckcachesz* angegebene Grenzwert überschritten wurde.

Gemeinsamer Anwendungsspeicher

Der folgende Parameter gibt den Arbeitsbereich an, der von allen Agenten (sowohl koordinierende als auch Subagenten) verwendet wird, die für eine Anwendung arbeiten:

- „Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz)"

Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

Datenbank-Server mit lokalen und fernen Clients 128 [1 – 64 000]

Datenbank-Server mit lokalen Clients
64 [1–64 000] (Nicht-UNIX-Plattformen)

128 [1–64 000] (UNIX-Plattformen)

Partitionierter Datenbank-Server mit lokalen und fernen Clients
256 [1 – 64 000]

Maßeinheit Seiten (4 KB)

Zuordnung Beim Starten einer Anwendung

Freigabe Beim Beenden einer Anwendung

Zugehörige Parameter

- „Katalog-Cache-Größe (catalogcache_sz)" auf Seite 409
- „Zwischenspeicher für Anwendungen (applheapsz)" auf Seite 426
- „Partitionsinterne Parallelität aktivieren (intra_parallel)" auf Seite 543

Bei partitionierten und nicht partitionierten Datenbanken mit aktivierter partitionsinterner Parallelität (intra_parallel=ON) ist dies die Größe des gemeinsam benutzten Speicherbereichs, der dem Zwischenspeicher für Anwendungssteuerung zugeordnet ist. Bei nicht partitionierten Datenbanken, bei denen die partitionsinterne Parallelität nicht aktiviert ist (intra_parallel=OFF), ist dies der maximale private Speicher, der dem Zwischenspeicher zugeordnet wird. Pro Verbindung einer Partition gibt es einen Zwischenspeicher für Anwendungssteuerung.

Der Zwischenspeicher für Anwendungssteuerung wird in erster Linie für die gemeinsame Benutzung von Informationen durch Agenten benötigt, die für dieselbe Anforderung arbeiten, sowie, in einer Umgebung mit partitionierten Datenbanken, für die Speicherung ausführbarer Abschnitte, die SQL-Anweisungen repräsentieren. Die Auslastung dieses Zwischenspeichers ist bei nicht partitionierten Datenbanken minimal, wenn Abfragen mit einem Parallelitätsgrad kleiner-gleich 1 ausgeführt werden.

Dieser Zwischenspeicher wird auch zum Speichern von Deskriptorinformationen für deklarierte temporäre Tabellen verwendet. Die Deskriptorinformationen für alle deklarierten temporären Tabellen, die nicht explizit gelöscht wurden, werden in diesem Zwischenspeicher aufbewahrt und können erst nach dem Löschen der deklarierten temporären Tabelle gelöscht werden.

Empfehlung: Verwenden Sie zunächst den Standardwert. Sie müssen den Wert eventuell erhöhen, wenn Sie komplexe Anwendungen ausführen oder ein System mit zahlreichen Datenbankpartitionen oder deklarierte temporäre Tabellen verwenden. Die erforderliche Speicherkapazität erhöht sich mit der Anzahl deklarierter temporärer Tabellen, die gleichzeitig aktiv sind. Der Tabellendeskriptor einer deklarierten temporären Tabelle mit vielen Spalten ist größer als jener einer Tabelle mit wenigen Spalten. Daher erhöht eine große Anzahl Spalten in den deklarierten temporären Tabellen einer Anwendung auch den Bedarf an Zwischenspeicher zur Anwendungssteuerung.

Privater Agentenspeicher

Die folgenden Parameter wirken sich auf die Menge an Speicher aus, die für jeden Datenbankagenten verwendet wird:

- „Zwischenspeicher für Sortierlisten (sortheap)".
- „Schwellenwert für Sortierspeicher (sheapthres)" auf Seite 423.
- „SQL-Anweisungszwischenspeicher (stmtheap)" auf Seite 425.
- „Zwischenspeicher für Anwendungen (applheapsz)" auf Seite 426.
- „Größe des Statistikzwischenspeichers (stat_heap_sz)" auf Seite 427.
- „Größe des Abfragezwischenspeichers (query_heap_sz)" auf Seite 428.
- „DRDA-Zwischenspeichergröße (drda_heap_sz)" auf Seite 429.
- „Größe des gemeinsamen UDF-Speichers (udf_mem_sz)" auf Seite 430.
- „Größe des Agentenstapelspeichers (agent_stack_sz)" auf Seite 432.
- „Minimaler reservierter privater Speicher (min_priv_mem)" auf Seite 433.
- „Schwellenwert für privaten Speicher (priv_mem_thresh)" auf Seite 434.
- „Maximaler Zwischenspeicher für Java-Interpreter (java_heap_sz)" auf Seite 447. Auf UNIX-Plattformen wird *java_heap_sz* pro Agent zugeordnet.

Lesen Sie im Abschnitt „Verwendung des Speichers durch DB2" auf Seite 283 die Informationen dazu, welches Verhältnis zwischen privatem Agentenspeicher und dem übrigen Speicher besteht, der vom Datenbankmanager zugeordnet wird.

Zwischenspeicher für Sortierlisten (sortheap)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	256 [16 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wie zur Ausführung von Sortiervorgängen erforderlich
Freigabe	Wenn der Sortiervorgang abgeschlossen ist
Zugehörige Parameter	„Schwellenwert für Sortierspeicher (sheapthres)" auf Seite 423

Mit diesem Parameter wird die maximale Anzahl von Seiten des privaten Speichers definiert, die für private Sortiervorgänge verwendet werden soll, bzw. die maximale Anzahl von Seiten des gemeinsamen Speichers, die für gemeinsame Sortiervorgänge verwendet werden soll. Wenn es sich um einen privaten Sortiervorgang handelt, bezieht sich dieser Parameter auf den privaten Agentenspeicher. Handelt es sich um einen gemeinsamen Sortiervorgang,

bezieht sich dieser Parameter auf den gemeinsamen Datenbankspeicher. Jeder Sortiervorgang verwendet einen getrennten Sortierspeicher, der bei Bedarf vom Datenbankmanager zugeordnet wird. Dieser Sortierspeicher ist der Bereich, in dem Daten sortiert werden. Bei Steuerung durch das Optimierungsprogramm wird anhand der vom Optimierungsprogramm bereitgestellten Informationen ein kleinerer als der durch diesen Parameter angegebene Sortierspeicher zugeordnet.

Empfehlung:

Bei der Arbeit mit dem Sortierspeicher ist Folgendes zu beachten:
- Geeignete Indizes können die Verwendung des Sortierspeichers minimieren.
- Hash-Verknüpfungs-Puffer und dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND (Index ANDing) und Star Joins verwendet werden), greifen auf den Sortierspeicher zu. Erhöhen Sie den Wert dieses Parameters, wenn diese Methoden verwendet werden.
- Erhöhen Sie den Wert dieses Parameters, wenn häufig umfangreiche Sortiervorgänge ausgeführt werden müssen.
- Wenn Sie den Wert dieses Parameters erhöhen, sollten Sie überprüfen, ob auch der Wert des Parameters *sheapthres* in der Konfigurationsdatei des Datenbankmanagers angepasst werden muss.
- Die Größe des Zwischenspeichers für Sortierlisten wird vom Optimierungsprogramm zur Bestimmung der Zugriffspfade verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

Schwellenwert für Sortierspeicher (sheapthres)

Konfigurationsart Datenbankmanager

Gilt für
- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX-Plattformen (32 Bit)
20 000 [250 — 2 097 152]

UNIX-Plattformen (64 Bit)
20 000 [250 — 2 147 483 647]

	OS/2 und Windows NT 10 000 [250 — 2 097 152]
Maßeinheit	Seiten (4 KB)
Zugehörige Parameter	„Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422

Private und gemeinsame Sortiervorgänge verwenden Speicher aus verschiedenen Speicherquellen. Die Größe des gemeinsamen Sortierspeicherbereichs wird auf der Grundlage des Werts von *sheapthres* statisch vorherbestimmt, wenn die erste Verbindung zu einer Datenbank hergestellt wird. Die Größe des privaten Sortierspeicherbereichs ist nicht begrenzt.

Der Parameter *sheapthres* wird für private und gemeinsame Sortiervorgänge unterschiedlich verwendet:

- Bei privaten Sortiervorgängen ist dieser Parameter ein exemplarweiter *veränderlicher Grenzwert* für die maximale Speichermenge, die zu einem beliebigen Zeitpunkt von privaten Sortiervorgängen beansprucht werden kann. Wenn der gesamte private Sortierspeicherbereich für ein Exemplar diesen Grenzwert erreicht, wird der für zusätzliche eingehende private Sortieranforderungen zugeordnete Speicherbereich beträchtlich verkleinert.
- Bei gemeinsamen Sortiervorgängen ist dieser Parameter ein datenbankweiter fester Grenzwert für den gesamten Speicherbereich, der zu einem beliebigen Zeitpunkt von gemeinsamen Sortiervorgängen belegt werden kann. Wenn dieser Grenzwert erreicht wird, sind keine weiteren Speicheranforderungen für gemeinsame Sortiervorgänge mehr möglich (bis der Speicherbereich, der für gemeinsame Sortiervorgänge belegt ist, wieder unterhalb des von *sheapthres* angegebenen Grenzwerts liegt).

Der Sortierspeicher wird beispielsweise bei folgenden Operationen verwendet: Sortierungen, Hash-Verknüpfungen, dynamische Bitzuordnungen (die für logisches Verknüpfen von Indizes über AND (Index ANDing) und Star Joins verwendet werden) und Operationen, bei denen sich die Tabelle im Speicher befindet.

Durch die explizite Angabe des Schwellenwerts wird verhindert, dass der Datenbankmanager übermäßig große Speichermengen für sehr viele Sortiervorgänge verwendet.

Es gibt keinen Grund, den Wert dieses Parameter beim Versetzen von einer Umgebung mit einem Knoten in eine Umgebung mit mehreren Knoten zu erhöhen. Wenn Sie die Konfigurationsparameter der Datenbank und des Datenbankmanagers in einer Umgebung mit einem Knoten (DB2 EE) optimiert haben, funktionieren diese Werte in einer Umgebung mit mehreren Knoten (DB2 EEE) zumeist ebenfalls einwandfrei.

Der Parameter *Schwellenwert für Sortierspeicher*, ein Konfigurationsparameter des Datenbankmanagers, gilt für das gesamte DB2-Exemplar. Sie können diesen Parameter in anderen Knoten oder Partitionen nur auf unterschiedliche Werte setzen, indem Sie mehrere DB2-Exemplare erstellen. Dies erfordert die Verwaltung verschiedener DB2-Datenbanken in verschiedenen Knotengruppen. Ein solches Vorgehen macht viele Vorteile einer Umgebung mit partitionierten Datenbanken zunichte.

Empfehlung: Sie sollten den Wert dieses Parameters möglichst auf ein angemessenes Vielfaches des größten Werts für den Parameter *sortheap* setzen, der in Ihrem Datenbankmanagerexemplar definiert ist. Der Wert dieses Parameters sollte **mindestens** zweimal so groß sein wie der größte Sortierspeicher (*sortheap*), der für eine Datenbank im Exemplar definiert ist.

Wenn Sie private Sortiervorgänge ausführen und Ihr System über genügend Speicher verfügt, kann der Idealwert für diesen Parameter auf folgende Weise berechnet werden:

1. Berechnen Sie die normale Sortierspeicherbelegung für jede Datenbank:

   ```
   (normale Anzahl Agenten, die gleichzeitig für die Datenbank ausgeführt werden)
   * (sortheap, wie für die Datenbank definiert)
   ```

2. Berechnen Sie die Summe der obigen Ergebnisse. Dies ergibt einen Wert für den gesamten Sortierspeicher, der unter normalen Umständen für alle Datenbanken innerhalb des Exemplars verwendet werden kann.

Informationen zu Sortieroperationen in einer SMP-Umgebung finden Sie in „Strategien für paralleles Sortieren" auf Seite 229.

Sie sollten Vergleichstests (Benchmark-Tests) ausführen, um die optimale Einstellung für diesen Parameter zu ermitteln, die Sortierleistung und Speicherbedarf gleichermaßen berücksichtigt. Weitere Informationen finden Sie in „Kapitel 12. Durchführen von Vergleichstests" auf Seite 371.

Weitere Informationen zum Sortieren finden Sie auch in „Sortieren" auf Seite 309.

Mit dem Datenbanksystemmonitor können Sie die Sortieraktivitäten verfolgen.

Weitere Informationen finden Sie in der Beschreibung des Monitorelements Sortierungen nach Überschreiten des Schwellenwerts (*post_threshold_sorts*) im Handbuch *System Monitor Guide and Reference*.

SQL-Anweisungszwischenspeicher (stmtheap)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]	2048 [128 – 60 000]
Maßeinheit	Seiten (4 KB)
Zuordnung	Für jede Anweisung während des Vorkompilierens oder des Bindens
Freigabe	Wenn das Vorkompilieren oder Binden der betreffenden Anweisung abgeschlossen ist

Der Anweisungszwischenspeicher wird als Arbeitsbereich für den SQL-Compiler während des Kompilierens einer SQL-Anweisung verwendet. Mit diesem Parameter wird die Größe dieses Arbeitsbereichs angegeben.

Dieser Bereich bleibt nicht permanent zugeordnet, sondern wird für die Verarbeitung jeder SQL-Anweisung einzeln zugeordnet und anschließend wieder freigegeben. Beachten Sie, dass dieser Arbeitsbereich bei dynamischen SQL-Anweisungen während der Ausführung Ihres Programms, bei statischen SQL-Anweisungen hingegen während des Bindens, und nicht während der Ausführung des Programms, verwendet wird.

Empfehlung: In den meisten Fällen kann der Standardwert für diesen Parameter übernommen werden. Wenn Sie sehr große SQL-Anweisungen verwenden und der Datenbankmanager beim Versuch, eine Anweisung zu optimieren, einen Fehler ausgibt (dass die Anweisung zu komplex ist), sollten Sie den Wert dieses Parameters in gleichmäßigen Schritten (z. B. 256 oder 1024) erhöhen, bis die Fehlersituation bereinigt ist.

Zwischenspeicher für Anwendungen (applheapsz)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	128 [16 – 60 000] 64 [16 – 60 000] (Umgebung mit partitionierten Datenbanken)
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn ein Agent zur Arbeit für eine Anwendung initialisiert wird
Freigabe	Wenn ein Agent die Arbeit für eine Anwendung beendet
Zugehörige Parameter	„Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz)" auf Seite 420

Mit diesem Parameter wird die Anzahl der Seiten an privatem Speicher definiert, die zur Verwendung durch den Datenbankmanager für einen bestimmten Agenten oder Subagenten zur Verfügung stehen.

Der Zwischenspeicher wird zugeordnet, wenn ein Agent oder Subagent für eine Anwendung initialisiert wird. Die zugeordnete Speichermenge ist die Mindestmenge, die zur Verarbeitung der an den Agenten oder Subagenten übergebenen Anforderung benötigt wird. Wenn der Agent oder Subagent mehr Zwischenspeicher zur Verarbeitung größerer SQL-Anweisungen benötigt, ordnet der Datenbankmanager weiteren Speicher nach Bedarf bis zu der durch diesen Parameter angegebenen Obergrenze zu.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird der Zwischenspeicher für die Anwendungssteuerung (*app_ctl_heap_sz*) verwendet, um Kopien der in Ausführung befindlichen Abschnitte von SQL-Anweisungen für Agenten und Subagenten zu speichern. SMP-Subagenten verwenden hingegen den Parameter *applheapsz*, wie es auch Agenten in allen anderen Umgebungen tun.

Empfehlung: Erhöhen Sie den Wert dieses Parameters, wenn Ihre Anwendungen einen Fehler empfangen, weil im Anwendungszwischenspeicher nicht genügend Speicher verfügbar ist.

Der Anwendungszwischenspeicher (*applheapsz*) wird aus dem privaten Agentenspeicher heraus zugeordnet.

Größe des Statistikzwischenspeichers (stat_heap_sz)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	4384 [1096 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn das Dienstprogramm RUNSTATS gestartet wird
Freigabe	Wenn das Dienstprogramm RUNSTATS beendet ist
Zugehörige Parameter	• „Anzahl der häufigsten Werte (num_freqvalues)" auf Seite 518 • „Anzahl der Quantile für Spalten (num_quantiles)" auf Seite 519

Mit diesem Parameter wird die **maximale** Größe des Zwischenspeichers angegeben, der bei Erfassung statistischer Daten mit dem Befehl RUNSTATS verwendet wird.

Empfehlung: Der Standardwert ist geeignet, wenn keine Verteilungsstatistikdaten gesammelt werden oder wenn die Verteilungsstatistikdaten nur für relativ schmale Tabellen gesammelt werden. Der Minimalwert ist **nicht** zu empfehlen, wenn Verteilungsstatistikdaten gesammelt werden, da nur Tabellen mit einer oder zwei Spalten in den Zwischenspeicher passen.

Dieser Parameter sollte nach der Anzahl der Spalten, für die statistische Daten erfasst werden, angepasst werden. Schmale Tabellen mit relativ wenigen Spalten erfordern weniger Speicher für die zu sammelnden Verteilungsstatistikdaten. Breite Tabellen mit vielen Spalten machen erheblich mehr Speicher erforderlich. Wenn Sie Verteilungsstatistikdaten für sehr breite Tabellen sammeln und einen großen Statistikzwischenspeicher benötigen, sollten Sie die Statistikdaten in einer Phase geringer Systemaktivität sammeln, um die Speicheranforderungen anderer Benutzer nicht einzuschränken.

Größe des Abfragezwischenspeichers (query_heap_sz)

Konfigurationsart	Datenbankmanager
Gilt für	• Datenbank-Server mit lokalen und fernen Clients • Datenbank-Server mit lokalen Clients • Partitionierter Datenbank-Server mit lokalen und fernen Clients • Satellitendatenbank-Server mit lokalen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	1000 [2 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn eine Anwendung (lokal oder fern) eine Verbindung zur Datenbank herstellt
Freigabe	Wenn die Anwendung die Verbindung zur Datenbank oder zum Exemplar trennt
Zugehörige Parameter	„Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)" auf Seite 436

Mit diesem Parameter wird die **maximale** Speichermenge angegeben, die für den Abfragezwischenspeicher zugeordnet werden kann. Ein Abfragezwischenspeicher wird zur Speicherung jeder Abfrage im privaten Speicher des Agen-

ten verwendet. Die Informationen für jede Abfrage bestehen aus dem Eingabe- und Ausgabe-SQL-Deskriptorbereich, dem Text der Anweisung, dem SQL-Kommunikationsbereich, dem Paketnamen, dem Ersteller, der Abschnittsnummer und dem Konsistenz-Token. Mit diesem Parameter kann sichergestellt werden, dass eine Anwendung nicht unnötig große virtuelle Speicherbereiche innerhalb eines Agenten belegt.

Aus dem Abfragezwischenspeicher wird auch der Speicher für Blockcursor zugeordnet. Dieser Speicher besteht aus einem Cursorsteuerungsblock und einem vollständig formatierten Ausgabe-SQL-Deskriptorbereich.

Zu Anfang ist der zugeordnete Abfragezwischenspeicher ebenso groß wie der Zwischenspeicher für die Anwendungsunterstützungsebene, der durch den Parameter *aslheapsz* definiert ist. Der Abfragezwischenspeicher muss größer oder gleich 2 sowie größer oder gleich dem Wert des Parameters *aslheapsz* sein. Wenn dieser Abfragezwischenspeicher zur Verarbeitung einer Anforderung nicht ausreicht, wird neuer Speicher in der für die Anforderung erforderlichen Größe (nicht über den Wert *query_heap_sz* hinaus) zugeordnet. Wenn dieser neue Abfragezwischenspeicher mehr als anderthalbmal so groß wie der Wert für *aslheapsz* ist, wird der Abfragezwischenspeicher in der Größe von *aslheapsz* neu zugeordnet, wenn die Abfrage beendet ist.

Empfehlung: In den meisten Fällen ist der Standardwert ausreichend. Als Minimalwert sollte für *query_heap_sz* ein Wert angegeben werden, der mindestens fünfmal so groß wie der Wert für *aslheapsz* ist. Dadurch können Abfragen größer als *aslheapsz* sein, und es wird zusätzlicher Speicher für drei oder vier Blockcursor bereitgestellt, die gleichzeitig geöffnet sein können.

Wenn Sie sehr umfangreiche LOBs (große Objekte) haben, müssen Sie möglicherweise den Wert dieses Parameters erhöhen, damit der Abfragezwischenspeicher groß genug für diese LOBs ist.

DRDA-Zwischenspeichergröße (drda_heap_sz)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 128 [16 – 60 000]

Maßeinheit Seiten (4 KB)

Zuordnung

- Der DRDA-Anwendungs-Server (AS) ordnet einen DRDA-Zwischenspeicher jedes Mal zu, wenn ein DRDA-Anwendungs-Requester (AR) die Verbindung zu einer DB2-Datenbank herstellt.
- DB2 Connect ordnet einen DRDA-Zwischenspeicher jedes Mal zu, wenn eine Verbindung zu einem DRDA-Anwendungs-Server hergestellt wird.

Freigabe Wenn ein DRDA-Anwendungs-Requester die Verbindung zur Datenbank trennt

Mit diesem Parameter wird die Anzahl der Seiten angegeben, die für den von DB2 Connect und der Einrichtung zur Unterstützung von DRDA-Anwendungs-Servern (DRDA Application Server Support Feature) benutzten Speicher zugeordnet werden sollen. Die folgenden Größen haben Einfuß auf die Speicherkapazität, die aus diesem Zwischenspeicher zugeordnet wird:

- Die Anzahl der von einer Anwendung geöffneten Cursor
- Die Anzahl der Eingabe-Host-Variablen
- Die Anzahl der Einträge in der SELECT-Liste
- Die Größe der Ein- und Ausgabedaten
- Die Länge der SQL-Anweisungen, die gebunden (BIND) oder vorbereitet (PREPARE) werden

Empfehlung: Verwenden Sie den Standardwert, sofern Sie keinen Fehlercode empfangen, der auf unzureichenden DRDA-Zwischenspeicher hinweist.

Größe des gemeinsamen UDF-Speichers (udf_mem_sz)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 256 [128 – 60 000]

Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn eine UDF gestartet wird
Freigabe	Wenn eine UDF beendet wird

Dieser Parameter gilt sowohl für abgeschirmte als auch für nicht abgeschirmte benutzerdefinierte Funktionen (UDFs). Für eine abgeschirmte benutzerdefinierte Funktion gibt er die Standardzuordnung für Speicher an, der vom Datenbankprozess und der benutzerdefinierten Funktion gemeinsam benutzt werden soll. In einer Datenbankumgebung mit einer Partition gibt es nur einen gemeinsam benutzten Speicherbereich. In einer Umgebung mit partitionierten Datenbanken gibt es einen gemeinsam benutzten Speicherbereich für jeden Datenbankpartitions-Server, und alle Anwendungsagenten und Subagenten, die auf diesem Server aktiv sind, verwenden denselben gemeinsamen Speicherbereich.

Für nicht abgeschirmte benutzerdefinierte Funktionen gibt der Parameter die Größe des privaten Arbeitsspeichers an. In einer Datenbankumgebung mit einer Partition wird der Zwischenspeicher aus dem privaten Speicher zugeordnet. In einer Umgebung mit partitionierten Datenbanken wird der Zwischenspeicher aus dem globalen Anwendungsspeicher (Application Global Memory) für jeden Datenbankpartitions-Server zugeordnet, und alle Agenten und Subagenten, die auf diesem Datenbankpartitions-Server für diese Anwendung aktiv sind, verwenden denselben gemeinsamen Speicherbereich.

Bei abgeschirmten und nichtabgeschirmten benutzerdefinierten Funktionen wird dieser Speicher zur Übergabe von Daten zwischen der UDF und der Datenbank verwendet.

Wenn keine benutzerdefinierten Funktionen in Anwendungen verwendet werden, wird der Speicher nicht zugeordnet. Wenn sowohl abgeschirmte als auch nichtabgeschirmte UDFs in derselben Anwendung ausgeführt werden, erfolgen zwei Speicherzuordnungen: eine für abgeschirmte UDFs und eine für nichtabgeschirmte UDFs.

Weitere Informationen zu benutzerdefinierten Funktionen finden Sie in *Application Development Guide* und in *SQL Reference*.

Empfehlung: In allen Fällen, in denen keine LOB-Daten an eine benutzerdefinierte Funktion übergeben werden, sollte die Standardeinstellung angemessen sein. In Fällen, in denen LOB-Daten an eine benutzerdefinierte Funktion übergeben werden, müssen Sie eventuell den zuzuordnenden Speicherbereich vergrößern. Sie sollten den Wert dieses Parameters mindestens um zwei Seiten größer definieren als die Größe der Eingabeargumente und des Ergebnisses der externen Funktion.

Anmerkung: Der Speicherbedarf für UDFs ist häufig kumulativ, so dass sich die Anzahl von UDFs, die in einer Anwendung aufgerufen werden, auf die optimale Einstellung dieses Parameters auswirkt.

Größe des Agentenstapelspeichers (agent_stack_sz)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

OS/2	64 [8 – 1000]
Windows NT	16 [8 – 1000]

Maßeinheit Seiten (4 KB)

Zuordnung Wenn ein Agent zur Arbeit für eine Anwendung initialisiert wird

Freigabe Wenn ein Agent die Arbeit für eine Anwendung beendet

Der Agentenstapelspeicher ist der virtuelle Speicherbereich, der von DB2 für jeden Agenten zugeordnet wird. Dieser Speicher wird zugeordnet, wenn er zur Verarbeitung einer SQL-Anweisung angefordert wird. Sie können diesen Parameter zur Optimierung der Speicherauslastung des Servers für einen bestimmten Satz von Anwendungen verwenden. Komplexere Abfragen benötigen im Vergleich zu einfachen Abfragen mehr Stapelspeicherbereich.

Dieser Parameter gilt nicht für auf UNIX basierende Plattformen.

Empfehlung: In den meisten Fällen kann die Standardstapelspeichergröße verwendet werden. Sie sollten den Wert für diesen Parameter nur erhöhen, wenn in Ihrer Umgebung viele extrem komplexe Abfragen ausgeführt werden. Wenn die Stapelspeichergröße zur Verarbeitung Ihrer SQL-Anweisung nicht ausreicht, wird ein Fehlereintrag in der Datei db2diag.log protokolliert und ein SQL-Code zurückgegeben. In diesem Fall müssen Sie den Wert für *agent_stack_sz* erhöhen und das Datenbankexemplar erneut starten.

Sie können eventuell die Stapelspeichergröße verringern, um anderen Clients mehr Adressraum zur Verfügung zu stellen. Dies ist möglich, wenn Ihre Umgebung folgende Kriterien erfüllt:

- Die Umgebung enthält nur einfache Anwendungen (z. B. einfache Online-Transaktionsprogramme, OLTP), in denen es keine komplexen Abfragen gibt.
- Für die Umgebung sind relativ viele gleichzeitig aktive Clients erforderlich z. B. über 100).

Die Größe des Agentenstapelspeichers und die Anzahl gleichzeitig ausgeführter Clients stehen in einem umgekehrten Verhältnis zueinander: Durch eine Vergrößerung des Stapelspeichers wird die mögliche Anzahl der Clients verringert, die gleichzeitig ausgeführt werden können. Die Ursache hierfür ist, dass der Adressraum unter OS/2 und Windows NT begrenzt ist. Sie können beispielsweise unter OS/2 400 MB Adressraum haben. (Diese Größe hängt jedoch von der Angabe in der Datei `config.sys` ab). Wenn Sie *agent_stack_sz* auf 1 MB setzen, können Sie nur bis zu 400 Agenten haben. (Aufgrund anderen Bedarfs für Adressraum, z. B für Pufferpools, werden Sie wahrscheinlich sogar noch viel weniger Agenten verwenden können.) Das heißt, wenn *maxagents* auf einen größeren Wert (z. B. 5000) gesetzt ist, werden Sie diesen Grenzwert nie erreichen.

Minimaler reservierter privater Speicher (min_priv_mem)

Konfigurationsart	Datenbankmanager
Gilt für	• Datenbank-Server mit lokalen und fernen Clients • Datenbank-Server mit lokalen Clients • Partitionierter Datenbank-Server mit lokalen und fernen Clients • Satellitendatenbank-Server mit lokalen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	32 [32 – 112 000]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn der Datenbankmanager gestartet wird
Freigabe	Wenn der Datenbankmanager gestoppt wird
Zugehörige Parameter	„Schwellenwert für privaten Speicher (priv_mem_thresh)" auf Seite 434

Mit diesem Parameter wird die Anzahl der Seiten angegeben, die der Datenbank-Server-Prozess als privaten virtuellen Speicher reserviert, wenn ein

Datenbankmanagerexemplar gestartet wird (db2start). Wenn der Server mehr privaten Speicher benötigt, versucht er gegebenenfalls, mehr Speicher vom Betriebssystem zu erhalten.

Dieser Parameter gilt nicht für auf UNIX basierende Systeme.

Empfehlung: Verwenden Sie den Standardwert.

Sie sollten den Wert dieses Parameters nur ändern, wenn Sie mehr Speicher für den Datenbank-Server reservieren möchten. Dadurch wird der Zeitaufwand für die Zuordnung verringert. Allerdings sollte dieser Wert nicht zu hoch gesetzt werden, da die Leistung anderer Nicht-DB2-Anwendungen beeinträchtigt werden kann.

Schwellenwert für privaten Speicher (priv_mem_thresh)

Konfigurationsart	Datenbankmanager
Gilt für	• Datenbank-Server mit lokalen und fernen Clients • Datenbank-Server mit lokalen Clients • Partitionierter Datenbank-Server mit lokalen und fernen Clients • Satellitendatenbank-Server mit lokalen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	1296 [-1; 32 – 112 000] 32 [-1; 32 – 112 000] auf dem Satellitendatenbank-Server mit lokalen Clients
Maßeinheit	Seiten (4 KB)
Zugehörige Parameter	„Minimaler reservierter privater Speicher (min_priv_mem)" auf Seite 433

Dieser Parameter wird zur Festlegung des nicht benutzten privaten Agentenspeichers verwendet, der zugeordnet bleibt und zur Verwendung durch neue Agenten, die gestartet werden, zur Verfügung steht. Er gilt nicht für auf UNIX basierende Plattformen.

Nach Beendigung eines Agenten wird nicht automatisch der gesamte, von diesem Agenten benutzte Speicher freigegeben, sondern der Datenbankmanager gibt nur die *überschüssigen Speicherzuordnungen* frei, die nach der folgenden Formel berechnet werden:

```
zugeordneter privater Speicher -
(benutzter privater Speicher + priv_mem_thresh)
```

Wenn diese Formel ein negatives Ergebnis liefert, wird keine Aktion ausgeführt.

Die folgende Tabelle enthält ein Beispiel, das verdeutlicht, wann Speicher zugeordnet und freigegeben wird. In diesem Beispiel wird für den Parameter *priv_mem_thresh* der willkürliche Wert 100 verwendet.

Aktionsbeschreibung	Zugeordneter Speicher	Benutzter Speicher
Eine Anzahl von Agenten ist aktiv und verfügt über zugeordneten Speicher.	1000	1000
Ein neuer Agent wird gestartet und verwendet 100 Seiten Speicher.	1100	1100
Ein Agent, der 200 Seiten Speicher benutzte, wird beendet. (Beachten Sie, dass 100 Seiten Speicher freigegeben werden, während die anderen 100 Seiten zur möglichen zukünftigen Verwendung zugeordnet bleiben.)	1000	900
Ein Agent, der 50 Seiten Speicher benutzte, wird beendet. (Beachten Sie, dass 50 Seiten Speicher freigegeben werden und 100 überschüssige Seiten, die von den vorhandenen Agenten nicht verwendet werden, zugeordnet bleiben.)	950	850
Ein neuer Agent wird gestartet, für den 150 Seiten Speicher erforderlich sind. (100 der 150 Seiten sind bereits zugeordnet, so dass der Datenbankmanager nur noch weitere 50 Seiten für diesen Agenten zuordnen muss.)	1000	1000

Der Wert „-1" bewirkt, dass dieser Parameter den Wert des Parameters *min_priv_mem* verwendet.

Empfehlung: Bei der Einstellung dieses Parameters sollten die Abläufe, wann und wie Clients die Verbindung herstellen bzw. trennen, sowie der Speicherbedarf anderer Prozesse auf derselben Maschine berücksichtigt werden.

Wenn es nur eine kurze Phase gibt, während der viele Clients gleichzeitig mit der Datenbank verbunden sind, verhindert ein hoher Schwellenwert, dass ungenutzter Speicher freigegeben und für andere Prozesse verfügbar gemacht wird. Dies führt zu einer schlechten Speicherverwaltung, die andere Prozesse, für die Speicher erforderlich ist, beeinträchtigen kann.

Wenn die Anzahl gleichzeitig zugreifender Clients eher gleichmäßig hoch ist, aber zahlreiche Verbindungsänderungen auftreten, stellt ein hoher Schwellen-

wert sicher, dass Speicher für die Client-Prozesse verfügbar ist, und verringert so den Systemaufwand, der durch die Zuordnung und Freigabe von Speicher entsteht.

Agenten-/Anwendungskommunikationsspeicher

Die folgenden Parameter wirken sich auf die Menge an Speicher aus, der zum Übergeben von Daten zwischen den Anwendungs- und Agentenprozessen zugeordnet wird:

- „Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)"
- „3 Kommastellen bei Dezimaldivision (min_dec_div_3)" auf Seite 438
- „E/A-Blockgröße für Clients (rqrioblk)" auf Seite 440
- „E/A-Blockgröße für DOS-Requester (dos_rqrioblk)" auf Seite 441

Lesen Sie im Abschnitt „Verwendung des Speichers durch DB2" auf Seite 283 die Informationen dazu, welches Verhältnis zwischen diesem von Anwendungen und Agenten gemeinsam benutzten Speicher und dem übrigen Speicher besteht, der vom Datenbankmanager zugeordnet wird.

Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)

Konfigurationsart	Datenbankmanager
Gilt für	• Datenbank-Server mit lokalen und fernen Clients • Datenbank-Server mit lokalen Clients • Partitionierter Datenbank-Server mit lokalen und fernen Clients • Satellitendatenbank-Server mit lokalen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	15 [1 – 524 288]
Maßeinheit	Seiten (4 KB)
Zuordnung	Wenn der Agentenprozess des Datenbankmanagers für die lokale Anwendung gestartet wird
Freigabe	Wenn der Agentenprozess des Datenbankmanagers beendet wird
Zugehörige Parameter	„Größe des Abfragezwischenspeichers (query_heap_sz)" auf Seite 428

Der Zwischenspeicher für die Anwendungsunterstützungsebene ist ein Kommunikationspuffer zwischen der lokalen Anwendung und dem zugeordneten Agenten. Dieser Puffer wird als gemeinsam benutzter Speicher von jedem Datenbankmanageragenten, der gestartet wird, zugeordnet.

Wenn die Anforderung an den Datenbankmanager oder die zugehörige Antwort nicht in den Puffer passt, wird sie in zwei oder mehr Sende-/Empfangspufferpaare geteilt. Die Größe dieses Puffers sollte so eingestellt werden, dass die Mehrzahl der Anforderungen mit einem einzigen Sende-/Empfangspufferpaar verarbeitet werden kann. Die Größe der Anforderung hängt von der Speichermenge ab, die zur Speicherung folgender Daten erforderlich ist:

- Der Eingabe-SQL-Deskriptorbereich
- Alle zugeordneten Daten in den SQLVARs
- Der Ausgabe-SQL-Deskriptorbereich
- Andere Felder, die im Allgemeinen 250 Byte nicht überschreiten

Außer zur Steuerung dieses Kommunikationspuffers wird dieser Parameter auch dazu verwendet, die Größe des E/A-Blocks festzulegen, wenn ein Blockcursor geöffnet wird. Dieser Speicher für Blockcursor wird aus dem privaten Adressraum der Anwendung zugeordnet. Sie sollten daher die optimale Größe des privaten Speichers, der jedem Anwendungsprogramm zugeordnet werden soll, ermitteln. Wenn der Datenbank-Client keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.

Die Daten, die von der lokalen Anwendung gesendet werden, werden vom Datenbankmanager in einem Bereich zusammenhängenden Speichers empfangen, der aus dem Abfragezwischenspeicher zugeordnet wurde. Mit dem Parameter *aslheapsz* wird die Anfangsgröße des Abfragezwischenspeichers (für lokale und ferne Clients) festgelegt. Die maximale Größe des Abfragezwischenspeichers wird durch den Parameter *query_heap_sz* definiert.

Empfehlung: Wenn die Anforderungen Ihrer Anwendung im Allgemeinen klein sind und die Anwendung auf einem System mit eingeschränkter Speicherkapazität ausgeführt wird, können Sie den Wert dieses Parameters verringern. Wenn Ihre Abfragen im Allgemeinen sehr groß sind und mehr als eine Sende- und Empfangsanforderung erfordern und die Speicherkapazität Ihres Systems nicht eingeschränkt ist, können Sie den Wert dieses Parameters erhöhen.

Berechnen Sie anhand der folgenden Formel die Mindestanzahl der Seiten für *aslheapsz*:

```
aslheapsz >= ( Größe von(Eingabe-SQL-Deskriptorbereich)
             + Größe von(jeder Eingabe-SQLVAR)
             + Größe von(Ausgabe-SQL-Deskriptorbereich)
             + 250 ) / 4096
```

Dabei ist sizeof(x) die Größe von x in Byte zur Berechnung der Seitenanzahl eines bestimmten Eingabe- oder Ausgabewerts.

Beachten Sie außerdem, welche Auswirkung dieser Parameter auf die Anzahl und die mögliche Größe von Blockcursorn hat. Große Zeilenblöcke führen zu einer erhöhten Leistung, wenn die Anzahl oder die Größe der Zeilen, die übertragen werden, groß ist (z. B., wenn die Datenmenge größer als 4 096 Byte ist). Dies hat jedoch den Nachteil, dass größere Satzblöcke die Größe des für jede Verbindung benötigten Arbeitsspeichers erhöhen.

Größere Satzblöcke können außerdem mehr Abrufanforderungen verursachen, als tatsächlich für die Anwendung erforderlich wären. Sie können die Anzahl der Abrufanforderungen mit der Klausel OPTIMIZE FOR in der Anweisung SELECT in Ihrer Anwendung steuern. Weitere Informationen zur Klausel OPTIMIZE FOR finden Sie in „Klausel OPTIMIZE FOR n ROWS" auf Seite 87.

3 Kommastellen bei Dezimaldivision (min_dec_div_3)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	No [Yes, No]

Der zusätzliche Datenbankkonfigurationsparameter *min_dec_div_3* bietet die Möglichkeit, die Berechnung der Kommastellen bei der Dezimaldivision in SQL schnell zu ändern. Mögliche Werte für *min_dec_div_3* sind "Yes" und "No". Der Standardwert für *min_dec_div_3* ist "No".

Der Datenbankkonfigurationsparameter *min_dec_div_3* ändert die Anzahl der Kommastellen im Ergebnis einer Dezimalrechenoperation, die eine Division umfasst. Wenn der Wert des Parameters "No" ist, wird die Anzahl der Kommastellen als 31-p+s-s' berechnet. Weitere Informationen finden Sie in Kapitel 3 des Handbuchs *SQL Reference*, das sich mit der Dezimalrechnung in SQL beschäftigt. Wenn der Wert "Yes" ist, wird die Anzahl der Kommastellen als MAX(3, 31-p+s-s') berechnet. Dies führt dazu, dass das Ergebnis einer Dezimaldivision immer mindestens 3 Kommastellen hat. Die Genauigkeit ist immer 31.

Eine Änderung dieses Datenbankkonfigurationsparameters kann Änderungen bei Anwendungen für vorhandene Datenbanken bedingen. Dazu kann es kommen, wenn die Anzahl der Kommastellen bei Ergebnissen von Dezimaldivisionen von der Änderung dieses Datenbankkonfigurationsparameters

betroffen wäre. Nachfolgend werden einige mögliche Szenarios beschrieben, die Auswirkungen auf Anwendungen haben können. Betrachten Sie diese Szenarios, bevor Sie den Parameter *min_dec_div_3* auf einem Datenbankserver mit vorhandenen Datenbanken ändern.

- Wenn die Anzahl der Kommastellen im Ergebnis für eine der Sichtspalten geändert wird, könnte eine Sicht, die in einer Umgebung mit einer Einstellung definiert ist, mit SQLCODE -344 fehlschlagen, wenn nach Änderung des Datenbankkonfigurationsparameters auf diese Sicht verwiesen wird. Die Nachricht SQL0344N verweist auf rekursive allgemeine Tabellenausdrücke, wenn der Objektname (erstes Token) jedoch eine Sicht ist, müssen Sie die Sicht löschen und sie erneut erstellen, um diesen Fehler zu vermeiden.
- Das Verhalten eines statischen Pakets ändert sich erst, wenn das Paket implizit oder explizit erneut gebunden wird. Zum Beispiel werden nach einer Änderung des Werts von NO in YES die zusätzlichen Kommastellen in den Ergebnissen u. U. erst berücksichtigt, nachdem das Paket erneut gebunden wurde. Für jedes geänderte statische Paket kann mit einem expliziten REBIND-Befehl eine erneute Bindeoperation erzwungen werden.
- Eine Prüfung auf Integritätsbedingung schließt u. U. einige Werte aus, die zuvor akzeptiert wurden. Solche Zeilen verletzen jetzt die Integritätsbedingung, werden jedoch erst entdeckt, wenn diejenige der Spalten, die von der auf Integritätsbedingung geprüften Zeile berührt wird, aktualisiert wird oder die Anweisung SET INTEGRITY mit der Option IMMEDIATE CHECKED verarbeitet wird. Sie können die Prüfung auf eine solche Integritätsbedingung erzwingen, indem Sie eine ALTER TABLE-Anweisung ausführen, um die Integritätsbedingung zu löschen und die Integritätsbedingung dann mit einer ALTER TABLE-Anweisung wieder hinzuzufügen.

Anmerkung: Für DB2 Version 7 gelten außerdem die folgenden Einschränkungen:

1. Der Befehl `GET DB CFG FOR DBNAME` zeigt die Einstellung *min_dec_div_3* nicht an. Die aktuelle Einstellung lässt sich am besten dadurch ermitteln, dass Sie beobachten, welche zusätzlichen Auswirkungen das Ergebnis einer Dezimaldivision hat. Betrachten Sie z. B. die folgende Anweisung:

   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```

   Wenn diese Anweisung den SQLCODE-Wert SQL0419N zurückgibt, unterstützt die Datenbank *min_dec_div_3* nicht oder der Parameter ist auf "No" gesetzt. Wenn die Anweisung `1,000` zurückgibt, ist *min_dec_div_3* auf "Yes" gesetzt.
2. *min_dec_div_3* erscheint nicht in der Liste der Konfigurationsschlüsselwörter, wenn Sie den folgenden Befehl ausführen: `? UPDATE DB CFG`

E/A-Blockgröße für Clients (rqrioblk)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 32 767 [4 096 – 65 535]

Maßeinheit Byte

Zuordnung

- Wenn eine ferne Client-Anwendung eine Verbindungsanforderung für eine Server-Datenbank absetzt
- Wenn ein Blockcursor geöffnet wird, werden zusätzliche Blöcke auf dem Client geöffnet

Freigabe

- Wenn die ferne Anwendung die Verbindung zur Server-Datenbank trennt
- Wenn der Blockcursor geschlossen wird

Zugehörige Parameter „E/A-Blockgröße für DOS-Requester (dos_rqrioblk)" auf Seite 441

Mit diesem Parameter wird die Größe des Puffers für die Kommunikation zwischen fernen Anwendungen und ihren Datenbankagenten auf dem Datenbank-Server festgelegt. Wenn ein Datenbank-Client die Verbindung zu einer fernen Datenbank anfordert, wird dieser Kommunikationspuffer auf dem Client angelegt. Auf dem Datenbank-Server wird zu Anfang ein Kommunikationspuffer von 32 767 Byte zugeordnet, bis eine Verbindung hergestellt ist und der Server den Wert des Parameters *rqrioblk* auf dem Client ermitteln kann. Wenn der Server diesen Wert ermittelt hat, wird der Kommunikationspuffer auf dem Server neu zugeordnet, sofern der Client-Puffer nicht 32 767 Byte groß ist.

Außer für diesen Kommunikationspuffer wird dieser Parameter auch dazu verwendet, die Größe des E/A-Blocks auf dem Datenbank-Client festzulegen, wenn ein Blockcursor geöffnet wird. Dieser Speicher für Blockcursor wird aus

dem privaten Adressraum der Anwendung zugeordnet. Sie sollten daher die optimale Größe des privaten Speichers, der jedem Anwendungsprogramm zugeordnet werden soll, ermitteln. Wenn der Datenbank-Client keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.

Empfehlung: Bei Cursorn ohne Blockung könnte ein Grund zur Erhöhung des Werts für diesen Parameter darin bestehen, dass die Daten (z. B. LOB-Daten), die durch eine einzige SQL-Anweisung übertragen werden sollen, so umfangreich sind, dass der Standardwert nicht ausreicht.

Beachten Sie außerdem, welche Auswirkung dieser Parameter auf die Anzahl und die mögliche Größe von Blockcursorn hat. Große Zeilenblöcke führen zu einer erhöhten Leistung, wenn die Anzahl oder die Größe der Zeilen, die übertragen werden, groß ist (z. B., wenn die Datenmenge größer als 4 096 Byte ist). Dies hat jedoch den Nachteil, dass größere Satzblöcke die Größe des für jede Verbindung benötigten Arbeitsspeichers erhöhen.

Größere Satzblöcke können außerdem mehr Abrufanforderungen verursachen, als tatsächlich für die Anwendung erforderlich wären. Sie können die Anzahl der Abrufanforderungen mit der Klausel OPTIMIZE FOR in der Anweisung SELECT in Ihrer Anwendung steuern. Weitere Informationen zur Klausel OPTIMIZE FOR finden Sie im Abschnitt „Klausel OPTIMIZE FOR n ROWS" auf Seite 87.

E/A-Blockgröße für DOS-Requester (dos_rqrioblk)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 4 096 [4 096 – 65 535]

Maßeinheit Byte

Zuordnung

- Wenn ein ferner DOS- oder Windows 3.1-Client eine Verbindungsanforderung an eine Server-Datenbank absetzt
- Wenn ein Blockcursor geöffnet wird, werden zusätzliche Blöcke auf dem Client geöffnet

Freigabe

- Wenn die ferne Anwendung die Verbindung zur Datenbank trennt
- Wenn ein Blockcursor geschlossen wird

Zugehörige Parameter

„E/A-Blockgröße für Clients (rqrioblk)" auf Seite 440

Mit diesem Parameter wird die Größe des Kommunikationspuffers zwischen DOS/Windows 3.1-Anwendungen und deren Datenbankagenten auf dem Datenbank-Server festgelegt. Dieser Parameter ähnelt dem Parameter *rqrioblk*. Mit ihm kann jedoch ein anderer Wert für mit DOS/Windows 3.1-Clients verwendete Blöcke festgelegt werden. In einer DB2-Konfigurationsdatei können Sie den Parameter *rqrioblk* (für Windows 32-Bit-, OS/2- und UNIX-Clients verwendet) und den Parameter *dos_rqrioblk* (für DOS- und Windows 3.1-Clients verwendet) festlegen.

Außer für diesen Kommunikationspuffer wird dieser Parameter auch dazu verwendet, die Größe des E/A-Blocks auf dem Datenbank-Client festzulegen, wenn ein Blockcursor geöffnet wird. Dieser Speicher für Blockcursor wird aus dem privaten Adressraum der Anwendung zugeordnet. Sie sollten daher die optimale Größe des privaten Speichers, der jedem Anwendungsprogramm zugeordnet werden soll, ermitteln. Wenn der Datenbank-Client keinen Bereich für einen Blockcursor aus dem privaten Speicher der Anwendung zuordnen kann, wird ein Cursor ohne Blockung geöffnet.

Empfehlung: Bei Cursorn ohne Blockung könnte ein Grund zur Erhöhung des Werts für diesen Parameter darin bestehen, dass die Daten (z. B. LOB-Daten), die durch eine einzige SQL-Anweisung übertragen werden sollen, so umfangreich sind, dass der Standardwert nicht ausreicht.

Beachten Sie außerdem, welche Auswirkung dieser Parameter auf die Anzahl und die mögliche Größe von Blockcursorn hat. Große Zeilenblöcke führen zu einer erhöhten Leistung, wenn die Anzahl oder die Größe der Zeilen, die übertragen werden, groß ist (z. B., wenn die Datenmenge größer als 4 096 Byte ist). Dies hat jedoch den Nachteil, dass größere Satzblöcke die Größe des für jede Verbindung benötigten Arbeitsspeichers erhöhen.

Größere Satzblöcke können außerdem mehr Abrufanforderungen verursachen, als tatsächlich für die Anwendung erforderlich wären. Sie können die Anzahl der Abrufanforderungen mit der Klausel OPTIMIZE FOR in der Anweisung SELECT in Ihrer Anwendung steuern. Weitere Informationen zur Klausel OPTIMIZE FOR finden Sie im Abschnitt „Klausel OPTIMIZE FOR n ROWS" auf Seite 87.

Exemplarspeicher des Datenbankmanagers

Die folgenden Parameter wirken sich auf den Speicher aus, der auf Exemplarebene zugeordnet und verwendet wird:

- „Zwischenspeicher für den Datenbanksystemmonitor (mon_heap_sz)"
- „Verzeichnis-Cache-Unterstützung (dir_cache)" auf Seite 444
- „Prüfpuffergröße (audit_buf_sz)" auf Seite 446
- „Maximaler Zwischenspeicher für Java-Interpreter (java_heap_sz)" auf Seite 447

Zwischenspeicher für den Datenbanksystemmonitor (mon_heap_sz)

Konfigurationsart
Datenbankmanager

Gilt für
- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart
Konfigurierbar

Standardwert [Bereich]

UNIX 56 [0 – 60 000]

OS/2 und Windows NT Datenbank-Server mit lokalen und fernen Clients und Satellitendatenbank-Server mit lokalen Clients 32 [0 – 60 000]

OS/2 und Windows NT Datenbank-Server mit lokalen Clients
12 [0 – 60 000]

Maßeinheit
Seiten (4 KB)

Zuordnung
Wenn der Datenbankmanager mit dem Befehl *db2start* gestartet wird

Freigabe Wenn der Datenbankmanager mit dem Befehl *db2stop* gestoppt wird

Zugehörige Parameter „Standardschalter für den Datenbanksystemmonitor (dft_monswitches)" auf Seite 547

Mit diesem Parameter wird der Speicherbereich in Seiten festgelegt, der für Daten des Datenbanksystemmonitors zugeordnet werden soll. Der Speicher wird aus dem Monitorzwischenspeicher zugeordnet, wenn Sie eine Datenbankmonitorfunktion ausführen, wie z. B. Erstellen einer Momentaufnahme (Snapshot), Aktivieren eines Monitorschalters oder eines Ereignismonitors, Zurücksetzen eines Monitors o. ä.

Erhält der Parameter den Wert 0, kann der Datenbankmanager keine Daten des Datenbanksystemmonitors sammeln.

Empfehlung: Der für die Monitorfunktionen erforderliche Speicherbereich ist von der Anzahl der Überwachungsanwendungen (Anwendungen, die Momentaufnahmen machen, oder Ereignismonitoren), den aktivierten Schaltern und der Datenbankaktivität abhängig.

Mit Hilfe der folgenden Formel kann ein Näherungswert für die Anzahl der für den Monitorfreispeicher erforderlichen Seiten berechnet werden:

```
( Anzahl der Überwachungsanwendungen + 1 ) *
( Anzahl der Datenbanken *
  (800 + ( Anzahl der Tabellen im Zugriff * 20 )
   + ( ( Anzahl der verbundenen Anwendungen + 1) *
       (200 + (Anzahl der Tabellenbereiche * 100) ) ) ) )
/ 4096
```

Wenn der verfügbare Speicher in diesem Freispeicher erschöpft ist, wird eine der folgenden Maßnahmen ausgeführt:

- Wenn die erste Anwendung eine Verbindung zu der Datenbank herstellt, für die dieser Ereignismonitor definiert ist, wird eine Fehlernachricht der Ebene 2 in die Dateien `db2alert.log` und `db2diag.log` geschrieben.
- Wenn ein Ereignismonitor, der mit Hilfe der Anweisung SET EVENT MONITOR dynamisch gestartet wird, fehlschlägt, wird ein Fehlercode an die Anwendung zurückgegeben.
- Wenn ein Monitorbefehl oder eine API-Unterroutine fehlschlägt, wird ein Fehlercode an die Anwendung zurückgegeben.

Verzeichnis-Cache-Unterstützung (dir_cache)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] Yes [Yes; No]

Zuordnung

- Wenn eine Anwendung die erste Verbindungsanforderung absetzt, wird der private Cache zugeordnet.
- Wenn ein Datenbankmanagerexemplar gestartet (`db2start`) wird, wird der gemeinsam benutzte Cache zugeordnet.

Freigabe

- Wenn ein Anwendungsprozess beendet wird, wird der private Cache freigegeben.
- Wenn ein Datenbankmanagerexemplar gestoppt (`db2stop`) wird, wird der gemeinsam benutzte Cache freigegeben.

Wenn der Parameter *dir_cache* auf "Yes" gesetzt wird, werden die Datenbank-, Knoten- und DCS-Verzeichnisdateien im Cache zwischengespeichert. Durch die Verwendung des Verzeichnis-Cache wird der Aufwand für die Verbindung verringert, indem die E/A-Operationen für Verzeichnisdateien vermieden und die Suchoperationen in Verzeichnissen zum Abruf von Verzeichnisdaten minimiert werden. Es gibt zwei Arten von Verzeichnis-Caches:

- Ein privater Cache, der für jeden Anwendungsprozess auf dem System, auf dem die Anwendung ausgeführt wird, zugeordnet und verwendet wird.
- Ein gemeinsam benutzter Cache, der für einige der internen Datenbankmanagerprozesse zugeordnet und verwendet wird.

Anmerkung: Für die unterstützten Windows-Umgebungen ist nur der private Cache gültig.

Ein privater Cache ist ein privater Speicherbereich für eine Anwendung, der zur Zwischenspeicherung der Daten verwendet wird, die in allen Verzeichnisdateien gelesen werden, wenn die Anwendung die erste Verbindungsanforderung absetzt. Der Cache wird vom Anwendungsprozess auch für nachfolgende Verbindungsanforderungen verwendet und während der Dauer des

Anwendungsprozesses beibehalten. Wenn eine Datenbank nicht im privaten Cache gefunden wird, werden die Verzeichnisdateien nach den Daten durchsucht, aber der Cache wird nicht aktualisiert. Wenn die Anwendung einen Verzeichniseintrag ändert, wird durch die nächste Verbindungsanforderung innerhalb dieser Anwendung eine Aktualisierung des Caches für diese Anwendung bewirkt. Der private Cache für andere Anwendungen wird nicht aktualisiert. Wenn der Anwendungsprozess beendet ist, wird der Cache freigegeben. (Zur Aktualisierung des Verzeichnis-Cache, der von einer Sitzung des Befehlszeilenprozessors verwendet wird, geben Sie den Befehl `db2 terminate` ein.)

Ein gemeinsam benutzter Cache ist gemeinsam benutzter Speicher, der zur Zwischenspeicherung von Daten verwendet wird, die in allen Verzeichnisdateien gelesen werden, wenn ein Datenbankmanagerexemplar gestartet wird (`db2start`). Dieser Cache wird von einigen der Datenbankmanagerprozesse verwendet und bleibt bestehen, bis das Exemplar gestoppt wird (`db2stop`). Wenn ein Verzeichniseintrag in diesem Cache nicht gefunden wird, werden die Verzeichnisdateien nach den Daten durchsucht. Dieser gemeinsam benutzte Cache wird während der Ausführung des Exemplars zu keinem Zeitpunkt aktualisiert.

Empfehlung: Verwenden Sie einen Verzeichnis-Cache, wenn Ihre Verzeichnisdateien nicht häufig geändert werden und die Leistung von entscheidender Bedeutung ist.

Auf fernen Clients kann darüber hinaus der Verzeichnis-Cache von Vorteil sein, wenn Ihre Anwendungen mehrere verschiedene Verbindungsanforderungen absetzen. In diesem Fall verringert das Zwischenspeichern die Häufigkeit, mit der eine einzige Anwendung die Verzeichnisdateien lesen muss.

Der Verzeichnis-Cache kann auch die Leistung bei der Erstellung von Momentaufnahmen des Datenbanksystemmonitors erhöhen. Außerdem sollten Sie beim Aufruf der Momentaufnahme den Datenbanknamen explizit angeben und nicht den Aliasnamen für die Datenbank verwenden.

Anmerkung: Bei der Erstellung von Momentaufnahmen können Fehler auftreten, wenn der Verzeichnis-Cache aktiviert wurde und Datenbanken katalogisiert, entkatalogisiert, erstellt oder gelöscht werden, nachdem der Datenbankmanager gestartet wurde.

Prüfpuffergröße (audit_buf_sz)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 0 [0 – 65 000]

Maßeinheit Seiten (4 KB)

Zuordnung Beim Starten von DB2

Freigabe Beim Stoppen von DB2

Mit diesem Parameter wird die Größe des Puffers für die Prüfung der Datenbank angegeben. Weitere Informationen zur Prüffunktion finden Sie im Abschnitt zum Prüfen von DB2-Aktivitäten im Handbuch *Systemverwaltung: Implementierung*.

Der Standardwert für diesen Konfigurationsparameter ist Null (0). Wenn der Wert Null (0) ist, wird der Prüfpuffer nicht verwendet. Wenn der Wert größer als Null (0) ist, wird dem Prüfpuffer Speicherbereich zugeordnet, in den die von der Prüffunktion generierten Prüfsätze gestellt werden. Der Wert multipliziert mit 4-KB-Seiten ergibt die für den Prüfpuffer zugeordnete Speichermenge. Der Prüfpuffer kann nicht dynamisch zugeordnet werden; DB2 muss gestoppt und anschließend erneut gestartet werden, bevor der neue Wert für diesen Parameter in Kraft tritt.

Wenn Sie den Standardwert für diesen Parameter in einen Wert ändern, der größer als Null (0) ist, schreibt die Prüffunktion Datensätze asynchron im Vergleich zur Ausführung der Anweisungen, die die Prüfsätze generieren, auf Platte. Durch das Erhöhen des Standardparameterwerts Null (0) wird die Leistung von DB2 verbessert. Der Wert Null (0) bedeutet, dass die Prüffunktion Datensätze synchron zur (d. h. zur gleichen Zeit wie die) Ausführung der Anweisungen, die die Prüfsätze generieren, auf Platte schreibt. Die synchrone Verarbeitung während der Prüfung verringert die Leistung von unter DB2 ausgeführten Anwendungen.

Maximaler Zwischenspeicher für Java-Interpreter (java_heap_sz)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients

- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart	Konfigurierbar
Standardwert [Bereich]	512 [0 - 4 096]
Maßeinheit	Seiten (4 KB)
Zuordnung	Beim Starten einer Java-Anwendung
Freigabe	Beim Beenden einer Java-Anwendung
Zugehörige Parameter	„Java Development Kit 1.1: Installationspfad (jdk11_path)" auf Seite 556

Dieser Parameter legt die maximale Größe des Zwischenspeichers fest, der vom Java-Interpreter verwendet wird.

Es gibt einen Zwischenspeicher für jeden DB2-Prozess (einen für jeden Agenten oder Subagenten auf UNIX-gestützten Plattformen und einen für jedes Exemplar auf anderen Plattformen); außerdem gibt es einen Zwischenspeicher für jede abgeschirmte benutzerdefinierte Funktion und für jede abgeschirmte gespeicherte Prozedur. Nur die Agenten oder Prozesse, die in Java geschriebene benutzerdefinierte Funktionen oder gespeicherte Prozeduren ausführen, ordnen diesen Speicher zu. Bei partitionierten Datenbanksystemen wird in jeder Partition derselbe Wert verwendet.

Sperren

Die folgenden Parameter beeinflussen die Sperrenverwaltung in Ihrer Umgebung:

- „Intervall für Prüfung gegenseitiger Sperren (dlchktime)"
- „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450
- „Zeitlimit für Sperren (locktimeout)" auf Seite 451

Siehe auch „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415.

„Sperren" auf Seite 57 enthält einen allgemeinen Überblick über die Verwendung von Sperren durch den Datenbankmanager zur Erhaltung der Datenintegrität.

Intervall für Prüfung gegenseitiger Sperren (dlchktime)

Konfigurationsart	Datenbank

Parameterart	Konfigurierbar
Standardwert [Bereich]	10 000 (10 Sekunden) [1 000 – 600 000]
Maßeinheit	Millisekunden
Zugehörige Parameter	• „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415 • „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450

Bei Auftreten gegenseitiger Sperren müssen zwei oder mehr Anwendungen, die auf dieselbe Datenbank zugreifen, unendlich lange auf eine Ressource warten. Die Anwendungen warten unendlich lange, da jede Anwendung eine Ressource gesperrt hat, die von der anderen Anwendung zur Fortsetzung der Verarbeitung benötigt wird.

Der Parameter für das Intervall zur Prüfung auf gegenseitige Sperren definiert die Häufigkeit, mit der vom Datenbankmanager alle mit der Datenbank verbundenen Anwendungen auf gegenseitige Sperren überprüft werden.

Anmerkungen:

1. In einer Umgebung mit partitionierten Datenbanken gilt dieser Parameter nur für den Katalogknoten.
2. In einer Umgebung mit partitionierten Datenbanken wird eine gegenseitige Sperre erst nach der zweiten Iteration markiert.

Empfehlung: Durch Erhöhung des Werts für diesen Parameter wird die Häufigkeit der Prüfung auf gegenseitige Sperren verringert, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung gegenseitiger Sperren warten müssen, erhöht wird.

Durch Angabe eines niedrigeren Werts für diesen Parameter wird die Häufigkeit der Prüfung auf gegenseitige Sperren erhöht, wodurch die Zeit, die Anwendungsprogramme auf die Aufhebung gegenseitiger Sperren warten müssen, verkürzt, die Zeit, die der Datenbankmanager auf die Prüfung gegenseitiger Sperren verwendet, jedoch verlängert wird. Wenn das Prüfintervall zu klein ist, kann dies zu einer Verschlechterung der Laufzeitleistung führen, weil der Datenbankmanager häufig nach gegenseitigen Sperren sucht. Wenn dieser Parameter mit einem niedrigeren Wert versehen wird, um den gemeinsamen Zugriff zu verbessern, sollten Sie darauf achten, dass die Parameter *maxlocks* und *locklist* entsprechend eingestellt sind, um unnötige Sperreneskalationen zu vermeiden, die zu Zugriffskonflikten und weiteren gegenseitigen Sperren führen können.

Maximale Anzahl Sperren pro Anwendung (maxlocks)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 10 [1 – 100]

OS/2 und Windows NT
22 [1 – 100]

Maßeinheit Prozent

Zugehörige Parameter

- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
- „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461

Eine Sperreneskalation ist der Prozess, durch den jeweils mehrere Zeilensperren für eine Tabelle durch eine Tabellensperre ersetzt werden, um die Anzahl der Sperren in der Liste zu verringern. Mit diesem Parameter wird definiert, wie viel Prozent der Sperrenliste eine Anwendung belegen muss, damit der Datenbankmanager eine Eskalation ausführt. Wenn die Anzahl der Sperren, die von einer Anwendung aktiviert wurden, diesen Prozentwert der Gesamtgröße der Sperrenliste erreicht, wird für die Sperren dieser Anwendung eine Sperreneskalation ausgeführt. Eine Sperreneskalation wird auch dann ausgeführt, wenn in der Sperrenliste kein weiterer Platz mehr verfügbar ist.

Der Datenbankmanager ermittelt die zu eskalierenden Sperren, indem er die Sperrenliste für die Anwendung nach der Tabelle mit den meisten gesperrten Zeilen durchsucht. Wenn nach der Ersetzung dieser Sperren durch eine einzige Tabellensperre der Wert für *maxlocks* nicht mehr überschritten wird, wird die Sperreneskalation beendet. Andernfalls wird die Eskalation fortgesetzt, bis der von dieser Anwendung belegte Prozentsatz der Sperrenliste unter den Wert von *maxlocks* gesunken ist. Das Produkt aus dem Wert des Parameters *maxlocks* und dem Wert des Parameters *maxappls* darf nicht kleiner als 100 sein.

Empfehlung: Mit der folgenden Formel können Sie *maxlocks* so einstellen, dass eine Anwendung das Zweifache der durchschnittlichen Anzahl von Sperren enthalten kann:

```
maxlocks = 2 * 100 / maxappls
```

Dabei wird `2` verwendet, um das Zweifache der durchschnittlichen Anzahl zu erzielen, und `100` ist der höchste zulässige Prozentwert. Wenn Sie nur wenige

Anwendungen haben, die gleichzeitig aktiv sind, können Sie als Alternative zu der ersten Formel auch die folgende Formel verwenden:

```
maxlocks = 2 * 100 / (durchschnittliche Anzahl der gleichzeitig aktiven
Anwendungen)
```

Bei der Festlegung von *maxlocks* können Sie die gleichzeitige Verwendung von *locklist*, der Größe der Sperrenliste, in Erwägung ziehen. Die maximale Anzahl der Sperren, die eine Anwendung enthalten kann, bevor eine Sperreneskalation erfolgt, wird wie folgt berechnet:

```
maxlocks * locklist * 4 096 / (100 * 36)
```

Dabei ist `4 096` die Anzahl der Bytes auf einer Seite, `100` ist der höchste zulässige Prozentwert für *maxlocks* und `36` ist die Anzahl der Byte pro Sperre. Wenn Sie wissen, dass eine Ihrer Anwendungen 1 000 Sperren benötigt, und Sie keine Sperreneskalation wünschen, sollten Sie die Werte für *maxlocks* und *locklist* in dieser Formel so wählen, dass das Ergebnis größer als 1 000 ist. (Wenn Sie für *maxlocks* 10 und für *locklist* 100 verwenden, ist das Ergebnis der Formel höher als die erforderlichen 1 000 Sperren.)

Wenn Sie für *maxlocks* einen zu niedrigen Wert wählen, tritt eine Sperreneskalation auf, obwohl für andere gleichzeitig ausgeführte Anwendungen noch genügend Speicher für Sperren verfügbar ist Wenn ein zu hoher Wert für *maxlocks* gewählt wird, belegen einige wenige Anwendungen u. U. fast den gesamten Speicher für Sperren, während andere Anwendungen eine Sperreneskalation durchführen müssen. Die Notwendigkeit einer Sperreneskalation geht in diesem Fall zulasten des gleichzeitigen Zugriffs.

Mit Hilfe des Datenbanksystemmonitors können Sie diesen Konfigurationsparameter verfolgen und seinen Wert optimieren.

Zeitlimit für Sperren (locktimeout)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] -1 [-1; 0 – 30 000]

Maßeinheit Sekunden

Zugehörige Parameter

- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
- „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450

Mit diesem Parameter wird die Anzahl der Sekunden angegeben, die eine Anwendung auf eine Sperre wartet. Dies trägt zur Vermeidung globaler gegenseitiger Sperren von Anwendungen bei.

Wenn dieser Parameter auf `0` gesetzt wird, wird nicht auf Sperren gewartet. In diesem Fall erhält die Anwendung, wenn zum Zeitpunkt der Anforderung keine Sperre verfügbar ist, sofort die Rückmeldung -911.

Wenn dieser Parameter auf den Wert `-1` gesetzt wird, wird die Überwachung des Zeitlimits für Sperren inaktiviert. In diesem Fall wird auf eine Sperre gewartet (wenn zum Zeitpunkt der Anforderung keine verfügbar ist), bis eine der folgenden Situationen eintritt:
- Die Sperre wird erteilt
- Eine gegenseitige Sperre tritt ein

Empfehlung: In einer Umgebung, in der Transaktionen verarbeitet werden (OLTP-Umgebung), können Sie einen Anfangswert von 30 Sekunden verwenden. In einer Umgebung, in der nur Abfragen durchgeführt werden, können Sie mit einem höheren Wert beginnen. In beiden Fällen sollten Sie diesen Parameter jedoch mit Hilfe von Vergleichstests (Benchmark-Tests) optimieren.

Wenn Sie bei der Arbeit mit Data Links File Manager (DLFM) im Protokoll db2diag.log des Exemplars von Data Links File Manager Überschreitungen der Sperrzeit feststellen, erhöhen Sie den Wert für den Parameter *locktimeout*. Erhöhen Sie gegebenenfalls auch den Wert für *locklist*.

Der Wert sollte so eingestellt werden, dass schnell erkannt wird, wenn Wartezeiten aufgrund außergewöhnlicher Situationen, wie einer blockierten Transaktion (z. B. weil ein Benutzer seine Workstation verlassen hat), auftreten. Sie sollten den Wert so hoch einstellen, dass gültige Sperranforderungen das Zeitlimit nicht aufgrund von Spitzenbelastungen überschreiten, da bei hoher Systemauslastung länger auf Sperren gewartet werden muss.

Mit dem Datenbanksystemmonitor können Sie die Häufigkeit, mit der eine Anwendung (eine Verbindung) das Zeitlimit für Sperren überschreitet, ermitteln oder erkennen, dass eine Datenbank eine Zeitlimitüberschreitung für alle verbundenen Anwendungen festgestellt hat. Weitere Informationen finden Sie in der Beschreibung des Monitorelements *locks_timeouts* (Anzahl der Zeitlimitüberschreitungen für Sperren) im Handbuch *System Monitor Guide and Reference*.

Hohe Werte für das Monitorelement *lock_timeout* können folgende Ursachen haben:
- Ein zu niedriger Wert für diesen Konfigurationsparameter

- Eine Anwendung (Transaktion), die Sperren über einen längeren Zeitraum aufrechterhält. Mit Hilfe des Datenbanksystemmonitors können Sie solche Anwendungen näher untersuchen.
- Ein Problem in Bezug auf den gleichzeitigen Zugriff, verursacht durch Sperreneskalationen (von Sperren auf Zeilenebene zu Sperren auf Tabellenebene). Weitere Informationen finden Sie in „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450 und in „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415.

Weitere Informationen zur Verwendung dieses Parameters finden Sie in „Warten auf Sperren und Zeitlimits" auf Seite 66.

Ein-/Ausgabe und Speicher

Die folgenden Parameter beeinflussen den Ein-/Ausgabeaufwand und den Speicherbedarf, die mit dem Betrieb der Datenbank verbunden sind:

- „Schwellenwert für geänderte Seiten (chngpgs_thresh)"
- „Anzahl asynchroner Seitenlöschfunktionen (num_iocleaners)" auf Seite 454
- „Anzahl von E/A-Servern (num_ioservers)" auf Seite 456
- „Markierung für Indexsortierung (indexsort)" auf Seite 457
- „Markierung für Sequenzerkennung (seqdetect)" auf Seite 457
- „Standardwert für PREFETCHSIZE (dft_prefetch_sz)" auf Seite 458
- „Standardanzahl von SMS-Behältern (numsegs)" auf Seite 459
- „Standardwert für EXTENTSIZE bei Tabellenbereichen (dft_extent_sz)" auf Seite 459
- „Segmentgröße für erweiterten Speicher (estore_seg_sz)" auf Seite 460
- „Segmentanzahl für erweiterten Speicher (num_estore_segs)" auf Seite 460

Schwellenwert für geänderte Seiten (chngpgs_thresh)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	60 [5 – 99]
Maßeinheit	Prozent
Zugehörige Parameter	„Anzahl asynchroner Seitenlöschfunktionen (num_iocleaners)" auf Seite 454

Asynchrone Seitenlöschfunktionen schreiben geänderte Seiten aus dem Pufferpool (oder den Pufferpools) auf Platte, bevor der Bereich im Pufferpool von einem Datenbankagenten angefordert wird. Daher müssen Datenbankagenten in der Regel nicht die Auslagerung geänderter Seiten abwarten, bevor sie den Speicherbereich im Pufferpool nutzen können. Dadurch wird die Gesamtleistung der Datenbankanwendungen verbessert.

Mit diesem Parameter können Sie den Prozentsatz der geänderten Seiten angeben, bei dem die asynchronen Seitenlöschfunktionen gestartet werden, wenn sie zum gegebenen Zeitpunkt nicht aktiv sind. Wenn die Seitenlöschfunktionen gestartet werden, erstellen sie eine Liste der Seiten, die auf Platte zu schreiben sind. Wenn das Schreiben dieser Seiten auf Platte abgeschlossen ist, werden die Seitenlöschfunktionen wieder inaktiv und warten auf den nächsten Startauslöser.

In einer Umgebung mit reinem Lesezugriff (in der z. B. nur Abfragen ausgeführt werden) werden diese Seitenlöschfunktionen nicht verwendet.

Empfehlung: Bei Datenbanken mit hohem Aufkommen an aktualisierenden Transaktionen können Sie allgemein sicherstellen, dass genügend freie Seiten im Pufferpool verfügbar sind, indem Sie diesen Parameter auf einen Wert setzen, der gleich groß wie oder kleiner als der Standardwert ist. Ein Prozentsatz über dem Standardwert kann sich positiv auf die Leistung auswirken, wenn in Ihrer Datenbank eine kleine Anzahl sehr großer Tabellen gespeichert ist.

Anzahl asynchroner Seitenlöschfunktionen (num_iocleaners)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	1 [0 – 255]
Maßeinheit	Zähler
Zugehörige Parameter	• „Größe des Pufferpools (buffpage)" auf Seite 404 • „Schwellenwert für geänderte Seiten (chngpgs_thresh)" auf Seite 453

Mit diesem Parameter können Sie die Anzahl asynchroner Seitenlöschfunktionen für eine Datenbank angeben. Diese Seitenlöschfunktionen schreiben geänderte Seiten aus dem Pufferpool auf Platte, bevor der Bereich im Pufferpool von einem Datenbankagenten angefordert wird. Daher müssen Datenbankagenten in der Regel nicht die Auslagerung geänderter Seiten abwarten, bevor sie den Speicherbereich im Pufferpool nutzen können. Dadurch wird die Gesamtleistung der Datenbankanwendungen verbessert.

Wenn der Parameter auf den Wert 0 gesetzt wird, werden keine Seitenlöschfunktionen gestartet, und infolgedessen schreiben die Agenten alle geänderten Seiten aus dem Pufferpool auf Platte. Dieser Parameter kann erhebliche Auswirkungen auf die Leistung einer Datenbank haben, die auf mehrere physische Speichereinheiten verteilt ist, weil in diesem Fall die Wahrscheinlichkeit größer ist, dass auf einer dieser Einheiten momentan keine Operationen aus-

geführt werden. Wenn keine Seitenlöschfunktionen konfiguriert sind, können Ihre Anwendungen in regelmäßigen Abständen auf volle Protokolle stoßen.

Wenn die Anwendungen für eine Datenbank im Wesentlichen aus Transaktionen bestehen, mit denen die Daten aktualisiert werden, führt eine Erhöhung der Anzahl der Seitenlöschfunktionen zu einer Leistungsverbesserung. Durch die Erhöhung der Anzahl der Seitenlöschfunktionen wird außerdem die Zeit für Wiederherstellungen nach Systemausfällen, zum Beispiel aufgrund von Netzausfall, verringert, weil der Inhalt der Datenbank auf der Platte zu einem gegebenen Zeitpunkt aktueller ist.

Empfehlung: Bei der Einstellung dieses Parameters müssen folgende Faktoren beachtet werden:

- Anwendungsart
  - Wenn es sich um eine reine Abfragedatenbank handelt, die nicht aktualisiert wird, setzen Sie diesen Parameter auf den Wert Null (0). Eine Ausnahme hiervon wäre, wenn die Abfrageauslastung dazu führt, dass viele TEMP-Tabellen erstellt werden. (Verwenden Sie das Dienstprogramm EXPLAIN, um dies festzustellen.)
  - Wenn Transaktionen für die Datenbank ausgeführt werden, setzen Sie diesen Parameter auf einen Wert zwischen 1 und der Anzahl der physischen Speichereinheiten, die für diese Datenbank verwendet werden.
- Auslastung

  Umgebungen mit hohem Aufkommen an aktualisierenden Transaktionen machen eventuell die Konfiguration weiterer Seitenlöschfunktionen erforderlich.
- Pufferpoolgrößen (*buffpage*)

  Umgebungen mit großen Pufferpools machen eventuell auch die Konfiguration weiterer Seitenlöschfunktionen erforderlich.

Mit Hilfe des Datenbanksystemmonitors können Sie diesen Konfigurationsparameter optimieren, indem Sie die Informationen des Ereignismonitors zu Schreibaktivitäten aus einem Pufferpool heranziehen:

- Der Wert des Parameters kann verringert werden, wenn die beiden folgenden Bedingungen erfüllt sind:
  - *pool_data_writes* ist ungefähr gleich *pool_async_data_writes*
  - *pool_index_writes* ist ungefähr gleich *pool_async_index_writes*
- Der Wert des Parameters sollte erhöht werden, wenn eine der folgenden Bedingungen erfüllt ist:
  - *pool_data_writes* ist viel größer als *pool_async_data_writes*.
  - *pool_index_writes* ist viel größer als *pool_async_index_writes*.

Weitere Informationen finden Sie in den Beschreibungen zu folgenden Monitorelementen im Handbuch *System Monitor Guide and Reference*:

- *pool_data_writes* (Schreibvorgänge von Pufferpooldaten)

- *pool_index_writes* (Schreibvorgänge von Pufferpoolindizes)
- *pool_async_data_writes* (Asynchrone Schreibvorgänge von Pufferpooldaten)
- *pool_async_index_writes* (Asynchrone Schreibvorgänge von Pufferpool-indizes)

Anzahl von E/A-Servern (num_ioservers)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	3 [1 – 255] 1 [1 – 255] auf dem Satellitendatenbank-Server mit lokalen Clients
Maßeinheit	Zähler
Zuordnung	Wenn eine Anwendung die Verbindung zu einer Datenbank herstellt
Freigabe	Wenn eine Anwendung die Verbindung zu einer Datenbank trennt
Zugehörige Parameter	• „Standardwert für PREFETCHSIZE (dft_prefetch_sz)" auf Seite 458 • „Markierung für Sequenzerkennung (seqdetect)" auf Seite 457

E/A-Server werden für Datenbankagenten verwendet, um Vorablese-E/A-Operationen und asynchrone E/A-Operationen von Dienstprogrammen wie BACKUP (Sicherung) und RESTORE (Wiederherstellung) auszuführen. Mit diesem Parameter wird die Anzahl der E/A-Server für eine Datenbank definiert. Zu jedem beliebigen Zeitpunkt kann nur diese Anzahl von E/A-Servern zum Vorablesen und für Dienstprogramme für eine Datenbank aktiv sein. Ein E/A-Server wartet, während eine vom ihm eingeleitete E/A-Operation ausgeführt wird. Nicht vorabgelesene Ein-/Ausgaben werden direkt von den Datenbankagenten terminiert, so dass diese Ein-/Ausgaben nicht der Begrenzung durch den Parameter *num_ioservers* unterliegen.

Empfehlung: Um alle E/A-Einheiten des Systems voll auszunutzen, empfiehlt sich im Allgemeinen ein Wert, der um 1 oder 2 höher ist als die Anzahl der physischen Einheiten, auf denen sich die Datenbank befindet. Es ist besser, zusätzliche E/A-Server zu konfigurieren, da mit jedem E/A-Server nur geringfügiger Systemaufwand verbunden ist und alle nicht benötigten E/A-Server inaktiv bleiben.

Weitere Informationen finden Sie in „Vorablesen von Daten in den Pufferpool" auf Seite 301 und „Konfigurieren von E/A-Servern für Vorablesezugriff und parallele E/A" auf Seite 304.

Markierung für Indexsortierung (indexsort)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	Yes [Yes; No]

Mit diesem Parameter wird angegeben, ob bei der Indexerstellung eine Sortierung der Indexschlüssel stattfindet. Die Leistung der Indexerstellung wird verbessert, wenn die Sortierung zuerst stattfindet, besonders bei Indizes mit niedrigen Clusterverhältnissen oder Clusterfaktoren. Auch die Leistung von Abfragen kann verbessert werden, wenn Indizes bei der Erstellung sortiert werden. Durch diese Leistungsverbesserung kommt es jedoch zu einem erhöhten Bedarf an Plattenspeicherplatz für die Sortierung, die das Doppelte des Speicherbereichs einer Indexerstellung ohne Anfangssortierung erforderlich machen könnte.

Empfehlung: Verwenden Sie die Standardeinstellung (Yes), sofern Sie über genügend Plattenspeicherplatz verfügen. Beachten Sie, dass der für die Sortierung erforderliche Plattenspeicherplatz ungefähr so groß ist wie der Platz, der für eine SELECT-Anweisung mit einer Klausel ORDER BY auf die indizierten Spalten benötigt wird, mit der die Spalten des Index aus der Tabelle ausgewählt werden.

Wenn Sie in einer symmetrischen Mehrprozessorumgebung (SMP-Umgebung) diesen Parameter auf den Wert "No" setzen, wird der Mehrprozessor, der in einer SMP-Umgebung eingesetzt werden kann, während der Indexerstellung nicht verwendet.

Markierung für Sequenzerkennung (seqdetect)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	Yes [Yes; No]
Zugehörige Parameter	„Standardwert für PREFETCHSIZE (dft_prefetch_sz)" auf Seite 458

Der Datenbankmanager kann die E/A-Operationen überwachen und, wenn Seiten sequenziell gelesen werden, das E/A-Vorablesen (I/O Prefetching) aktivieren. Diese Art des sequenziellen Vorablesens ist die *Sequenzerkennung*. Mit dem Konfigurationsparameter *seqdetect* können Sie steuern, ob der Datenbankmanager die Sequenzerkennung ausführen soll.

Wenn für diesen Parameter der Wert ″No″ angegeben wird, findet das Vorablesen nur dann statt, wenn der Datenbankmanager erkennt, dass es nützlich ist, z. B. bei Tabellensortierungen, Tabellensuchen oder einem Vorablesezugriff über Listen.

Empfehlung: In den meisten Fällen sollte der Standardwert für diesen Parameter verwendet werden. Inaktivieren Sie die Sequenzerkennung nur dann, wenn andere Optimierungsversuche bisher nicht zur Behebung schwerwiegender Leistungsprobleme bei Abfragen geführt haben.

Standardwert für PREFETCHSIZE (dft_prefetch_sz)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 32 [0 — 32 767]

OS/2 und Windows NT
16 [0 — 32 767]

Maßeinheit Seiten

Zugehörige Parameter

- „Standardwert für EXTENTSIZE bei Tabellenbereichen (dft_extent_sz)” auf Seite 459
- „Anzahl von E/A-Servern (num_ioservers)” auf Seite 456

Bei der Erstellung eines Tabellenbereichs kann wahlfrei `PREFETCHSIZE n` angegeben werden, wobei `n` die Anzahl der Seiten ist, die der Datenbankmanager liest, wenn ein Vorablesezugriff erfolgt. Wird `PREFETCHSIZE` in der Anweisung CREATE TABLESPACE nicht angegeben, verwendet der Datenbankmanager den Wert, der durch diesen Parameter definiert wird.

Weitere Informationen finden Sie in „Vorablesen von Daten in den Pufferpool” auf Seite 301.

Empfehlung: Mit Tools zur Systemüberwachung können Sie feststellen, ob Ihre CPU ausgelastet ist, während das System auf Ein-/Ausgaben wartet. Die Erhöhung dieses Parameterwerts kann nützlich sein, wenn für die Tabellenbereiche, die verwendet werden, kein Wert für PREFETCHSIZE definiert ist.

Dieser Parameter enthält den Standardwert für die gesamte Datenbank und ist eventuell nicht für alle Tabellenbereiche innerhalb der Datenbank geeignet. Beispielsweise kann der Wert 32 für einen Tabellenbereich mit dem Wert 32 Seiten für EXTENTSIZE geeignet sein, aber nicht für einen Tabellenbereich mit

dem Wert 25 Seiten für EXTENTSIZE. Im Idealfall sollten Sie den Wert für PREFETCHSIZE für jeden Tabellenbereich explizit angegeben.

Sie sollten als Wert für diesen Parameter einen Faktor oder ein ganzzahliges Vielfaches des Wertes des Parameters *dft_extent_sz* angeben, um die E/A-Operationen für Tabellenbereiche zu minimieren, die mit dem Standardwert für EXTENTSIZE (Parameter *dft_extent_sz*) definiert sind. Wenn z. B. für den Parameter *dft_extent_sz* der Wert 32 angegeben ist, könnten Sie den Wert von *dft_prefetch_sz* auf 16 (einen Bruchteil von 32) oder auf 64 (ein ganzzahliges Vielfaches von 32) setzen. Wenn für PREFETCHSIZE ein Vielfaches des Werts für EXTENTSIZE angegeben ist, kann der Datenbankmanager parallele E/A-Operationen ausführen, wenn die folgenden Bedingungen erfüllt sind:

- Die Bereiche, die vorab gelesen werden, befinden sich auf verschiedenen physischen Einheiten.
- Es sind mehrere E/A-Server konfiguriert (*num_ioservers*).

Standardanzahl von SMS-Behältern (numsegs)

Konfigurationsart	Datenbank
Parameterart	Informativ
Maßeinheit	Zähler

Dieser Parameter, der sich ausschließlich auf SMS-Tabellenbereiche bezieht, gibt die Anzahl der Behälter an, die innerhalb der Standardtabellenbereiche erstellt werden. Dieser Parameter zeigt den Wert an, der bei der Erstellung der Datenbank verwendet wurde, unabhängig davon, ob er in dem Befehl CREATE DATABASE explizit oder implizit angegeben wurde. Von der Anweisung CREATE TABLESPACE wird dieser Parameter **nicht** verwendet.

Weitere Informationen finden Sie im Abschnitt zu physischen Datenbankverzeichnissen im Handbuch *Systemverwaltung: Konzept*.

Standardwert für EXTENTSIZE bei Tabellenbereichen (dft_extent_sz)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	32 [2 – 256]
Maßeinheit	Seiten
Zugehörige Parameter	„Standardwert für PREFETCHSIZE (dft_prefetch_sz)" auf Seite 458

Bei der Erstellung eines Tabellenbereichs kann wahlfrei `EXTENTSIZE n` angegeben werden, wobei n die Speicherbereichsgröße ist. Wenn Sie `EXTENTSIZE n` in

der Anweisung CREATE TABLESPACE nicht angeben, verwendet der Datenbankmanager den Wert, der durch diesen Parameter definiert wird.

Weitere Informationen finden Sie im Abschnitt zum Entwerfen und Auswählen von Tabellenbereichen im Handbuch *Systemverwaltung: Konzept*.

Empfehlung: In vielen Fällen geben Sie die Speicherbereichsgröße bei der Erstellung eines Tabellenbereichs explizit an. Bevor Sie einen Wert für diesen Parameter wählen, sollten Sie verstehen, wie die Speicherbereichsgröße in der Anweisung CREATE TABLESPACE explizit gewählt wird. Weitere Informationen finden Sie in „Auswirkung des Tabellenbereichs auf die Abfrageoptimierung" auf Seite 107.

Segmentgröße für erweiterten Speicher (estore_seg_sz)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	16 000 [0 – 1 048 575]
Maßeinheit	Seiten
Zugehörige Parameter	„Segmentanzahl für erweiterten Speicher (num_estore_segs)"

Dieser Parameter gibt die Anzahl der Seiten in jedem Segment des erweiterten Speichers in der Datenbank an. Dieser Parameter wird nur verwendet, wenn Ihre Maschine über mehr adressierbaren Realspeicher als die maximal adressierbare virtuelle Speichermenge verfügt.

Empfehlung: Dieser Parameter ist nur wirksam, wenn erweiterter Speicher verfügbar ist, und er wird wie für den Parameter *num_estore_segs* dargestellt verwendet. Beim Angeben der in jedem Segment des erweiterten Speichers zu verwendenden Anzahl von Seiten sollten Sie auch die Segmentanzahl im erweiterten Speicher berücksichtigen, indem Sie den Parameter *num_estore_segs* überprüfen und ggf. ändern. Weitere Informationen zum erweiterten Speicher finden Sie in „Erweitern von Speicher" auf Seite 330.

Segmentanzahl für erweiterten Speicher (num_estore_segs)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	0 [0 – 214 7483 647]
Zugehörige Parameter	„Segmentgröße für erweiterten Speicher (estore_seg_sz)"

Dieser Parameter gibt die Anzahl der Segmente im erweiterten Speicher an, die von der Datenbank verwendet werden können.

Die Standardeinstellung ist 0, d. h. für die Datenbank werden keine Segmente im erweiterten Speicher bereitgestellt.

Empfehlung: Verwenden Sie diesen Parameter nur, um die Verwendung von Segmenten im erweiterten Speicher einzurichten, wenn der Speicherplatz Ihrer Systemumgebung den maximalen Adressraum übersteigt und Sie diesen Speicher nutzen möchten. Beim Angeben der Segmentanzahl sollten Sie auch die Größe der einzelnen Segmente berücksichtigen, indem Sie den Parameter *estore_seg_sz* überprüfen und ggf. ändern.

Wenn die Konfigurationsparameter *num_estore_segs* und *estore_seg_sz* gesetzt sind, sollten Sie mit Hilfe der Anweisungen CREATE/ALTER BUFFERPOOL angeben, welche Pufferpools den erweiterten Speicher verwenden werden. Weitere Informationen zum erweiterten Speicher finden Sie in „Erweitern von Speicher" auf Seite 330.

Agenten

Die folgenden Parameter können die Anzahl der Anwendungen beeinflussen, die gleichzeitig ausgeführt werden können, um eine optimale Leistung zu erzielen:

- „Maximale Anzahl aktiver Anwendungen (maxappls)"
- „Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)" auf Seite 463
- „Maximale Anzahl offener Datenbankdateien pro Anwendung (maxfilop)" auf Seite 464
- „Maximale Anzahl offener Dateien (maxtotfilop)" auf Seite 465
- „Agentenpriorität (agentpri)" auf Seite 466
- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468
- „Maximale Anzahl gleichzeitig aktiver Agenten (maxcagents)" auf Seite 469
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470
- „Maximale Anzahl logischer Agenten (max_logicagents)" auf Seite 471
- „Größe des Agentenpools (num_poolagents)" auf Seite 472
- „Anfangswert für die Anzahl Agenten im Pool (num_initagents)" auf Seite 473

Maximale Anzahl aktiver Anwendungen (maxappls)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 40 [1 – 60 000]

OS/2 und Windows NT Datenbank-Server mit lokalen und fernen Clients
20 [1 – 60 000]

OS/2 und Windows NT Datenbank-Server mit lokalen Clients
10 [1 – 60 000]

Maßeinheit Zähler

Zugehörige Parameter

- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470
- „Maximale Anzahl Sperren pro Anwendung (maxlocks)" auf Seite 450
- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
- „Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)" auf Seite 463

Mit diesem Parameter wird die maximale Anzahl der (sowohl lokalen als auch fernen) Anwendungen angegeben, die gleichzeitig auf eine Datenbank zugreifen können. Da jede Anwendung, die die Verbindung zu einer Datenbank herstellt, die Zuordnung privaten Speichers erforderlich macht, entsteht durch Erhöhung dieses Parameters möglicherweise mehr Speicherbedarf.

Der Wert dieses Parameters muss größer oder gleich der Summe der verbundenen Anwendungen plus der Anzahl solcher Anwendungen sein, die gleichzeitig im Begriff sind, eine zweiphasige Festschreibung oder ROLLBACK-Operation zu beenden. Addieren Sie zu dieser Summe die Anzahl unbestätigter Transaktionen, die zu einem gegebenen Zeitpunkt vorhanden sein können. Weitere Informationen zu unbestätigten Transaktionen finden Sie im Abschnitt zum Wiederherstellen bei Problemen während zweiphasiger Festschreibung im Handbuch *Systemverwaltung: Konzept*.

Wenn eine Anwendung versucht, die Verbindung zu einer Datenbank herzustellen, jedoch der Wert des Parameters *maxappls* bereits erreicht wurde, wird an die Anwendung ein Fehler zurückgegeben, dass die maximale Anzahl von Anwendungen bereits mit der Datenbank verbunden ist.

Da mehrere Anwendungen Data Links Manager verwenden, muss der Wert für *maxappls* erhöht werden. Berechnen Sie den erforderlichen Wert anhand folgender Formel:

```
<maxappls> = 5 * (Anzahl Knoten) + (Höchstanzahl aktiver Anwendungen,
                  die Data Links Manager verwenden)
```

Der unterstützte Maximalwert für Data Links Manager ist 2 000.

In einer Umgebung mit partitionierten Datenbanken ist dies die maximale Anzahl von Anwendungen, die gleichzeitig auf einer Datenbankpartition aktiv sein kann. Dieser Parameter beschränkt die Anzahl aktiver Anwendungen für eine Datenbankpartition auf einem Datenbankpartitions-Server unabhängig davon, ob der Server der Koordinatorknoten für die Anwendung ist oder nicht. Für den Katalogknoten in einer Umgebung mit partitionierten Datenbanken ist ein höherer Wert für *maxappls* erforderlich als für andere Umgebungsarten, weil in der Umgebung mit partitionierten Datenbanken jede Anwendung eine Verbindung zum Katalogknoten benötigt.

Empfehlung: Wenn der Wert dieses Parameters erhöht wird, ohne dass der Wert des Parameters *maxlocks* verringert oder der Wert des Parameters *locklist* erhöht wird, könnte die maximale Anzahl Sperren (Parameter *locklist*) eher erreicht werden als die maximale Anzahl der Anwendungen, was zu allgemeinen Problemen durch Sperreneskalation führen kann.

Bis zu einem gewissen Grad wird die maximale Anzahl von Anwendungen auch vom Paramater *maxagents* bestimmt. Eine Anwendung kann nur dann eine Verbindung zur Datenbank herstellen, wenn eine freie Verbindung (*maxappls*) sowie ein freier Agent (*maxagents*) verfügbar ist. Darüber hinaus wird die maximale Anzahl von Anwendungen auch vom Konfigurationsparameter *max_coordagents* gesteuert, weil keine neuen Anwendungen (d. h. Koordinationsagenten) gestartet werden können, wenn der Wert von *max_coordagents* erreicht ist.

Durchschnittliche Anzahl aktiver Anwendungen (avg_appls)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	1 [1 – maxappls]
Maßeinheit	Zähler
Zugehörige Parameter	• „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461

Mit Hilfe dieses Parameters versucht das SQL-Optimierungsprogramm zu ermitteln, wie viel Pufferpool zur Laufzeit für den ausgewählten Zugriffsplan verfügbar ist.

Empfehlung: Wenn DB2 in einer Mehrplatzsystemumgebung ausgeführt wird, ist es vorteilhaft, wenn das SQL-Optimierungsprogramm weiß, dass mehrere Abfragebenutzer das System verwenden (besonders bei komplexen Abfragen und einem großen Pufferpool), so dass das Optimierungsprogramm in seinen Annahmen zur Verfügbarkeit von Pufferpool weniger großzügig ist.

Bei der Einstellung dieses Parameters sollten Sie die Anzahl der Anwendungen mit komplexen Abfragen berechnen, die die Datenbank durchschnittlich verwenden. Aus dieser Berechnung sollten alle einfachen OLTP-Anwendungen (Online-Transaktionsprogramme) ausgeklammert werden. Wenn Sie bei der Bestimmung dieser Anzahl Probleme haben, multiplizieren Sie folgende Werte miteinander:

- Eine Durchschnittsanzahl aller Anwendungen, die für Ihre Datenbank ausgeführt werden. Der Datenbanksystemmonitor kann Informationen zur Anzahl der Anwendungen zu einem gegebenen Zeitpunkt liefern, so dass Sie anhand mehrerer Probewerte die Durchschnittsanzahl über einen gewissen Zeitraum hinweg berechnen können. Die Informationen des Datenbanksystemmonitors umfassen sowohl OLTP- als auch Nicht-OLTP-Anwendungen.
- Den von Ihnen geschätzten Prozentsatz an Anwendungen mit komplexen Abfragen.

Wie bei der Anpassung anderer Konfigurationsparameter, die das Optimierungsprogramm beeinflussen, sollten Sie auch den Wert dieses Parameters in kleinen Schritten ändern. Dadurch können Sie die Differenzen bei der Pfadauswahl minimieren.

Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

Maximale Anzahl offener Datenbankdateien pro Anwendung (maxfilop)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 64 [2 – 1950]

OS/2 und Windows NT
64 [2 – 32 768]

Maßeinheit Zähler

Zugehörige Parameter

- „Maximale Anzahl offener Dateien (maxtotfilop)" auf Seite 465

- „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461

Mit diesem Parameter wird die maximale Anzahl der Dateikennungen angegeben, die für jeden einzelnen Datenbankagenten geöffnet sein können. Wenn durch das Öffnen einer Datei dieser Wert überschritten wird, werden einige von diesem Agenten verwendete Dateien geschlossen. Wenn der Wert für den Parameter *maxfilop* zu klein ist, kann der Systemaufwand für das Öffnen und Schließen von Dateien, um diesen Grenzwert nicht zu überschreiten, übermäßig anwachsen und die Leistung beeinträchtigen.

Sowohl SMS-Tabellenbereiche als auch DMS-Tabellenbereichsdateicontainer werden bei der Interaktion des Datenbankmanagers mit dem Betriebssystem als Dateien behandelt, so dass Dateikennungen für sie erforderlich sind. In der Regel werden von SMS-Tabellenbereichen vergleichsweise mehr Dateien verwendet, als von DMS-Dateitabellenbereichen Behälter verwendet werden. Daher wird für diesen Parameter ein höherer Wert benötigt, wenn SMS-Tabellenbereiche verwendet werden, als für DMS-Dateitabellenbereiche erforderlich wäre.

Mit Hilfe dieses Parameters kann außerdem sichergestellt werden, dass die Gesamtzahl der Dateikennungen, die vom Datenbankmanager verwendet werden, den Grenzwert des Betriebssystems nicht überschreitet, indem die Anzahl der Dateikennungen pro Agent auf einen bestimmten Wert gesetzt wird. Die tatsächliche Anzahl ist je nach Anzahl der gleichzeitig aktiven Agenten unterschiedlich.

Maximale Anzahl offener Dateien (maxtotfilop)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 16 000 [100 – 32 768]

Maßeinheit Zähler

Zugehörige Parameter „Maximale Anzahl offener Datenbankdateien pro Anwendung (maxfilop)" auf Seite 464

Mit diesem Parameter wird die maximale Anzahl von Dateien definiert, die von allen Agenten und anderen Threads, die in einem Datenbankmanagerexemplar ausgeführt werden, geöffnet werden können. Wenn durch das Öffnen einer Datei der Wert dieses Parameters überschritten wird, wird ein Fehler an die Anwendung zurückgegeben.

Anmerkung: Dieser Parameter gilt nicht für auf UNIX basierende Plattformen.

Empfehlung: Bei der Einstellung dieses Parameters sollte die Anzahl der Dateikennungen beachtet werden, die für jede Datenbank in dem Datenbankmanagerexemplar verwendet werden könnten. Ermitteln Sie auf folgende Weise einen oberen Grenzwert für diesen Parameter:

1. Berechnen Sie anhand der folgenden Formel die maximale Anzahl von Dateikennungen, die für jede Datenbank in dem Exemplar geöffnet werden könnten:

   ```
   maxappls * maxfilop
   ```

2. Berechnen Sie die Summe der obigen Ergebnisse, und stellen Sie sicher, dass sie den Maximalwert des Parameters nicht überschreitet.

Wenn eine neue Datenbank erstellt wird, sollten Sie den Wert für diesen Parameter neu bestimmen.

Agentenpriorität (agentpri)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

AIX -1 [41 - 125]

Andere UNIX-Plattformen
-1 [41 - 128]

Windows NT
-1 [0 - 6]

OS/2 -1 [200 - 231; 300 - 331; 400 - 431]

Mit diesem Parameter wird die Priorität gesteuert, die sowohl allen Agenten als auch anderen Prozessen und Threads des Datenbankmanagerexemplars vom Scheduler des Betriebssystems zugewiesen wird. In einer Umgebung mit partitionierten Datenbanken gehören dazu auch koordinierende Agenten und Subagenten, parallele Systemsteuerprogramme und die FCM-Dämonen. Durch diese Priorität wird festgelegt, wie den DB2-Prozessen, -Agenten und -Threads im Vergleich zu den anderen Prozessen und Threads, die auf dem System ausgeführt werden, CPU-Zeit zugewiesen wird. Wenn der Parameter auf den Wert -1 gesetzt ist, wird keine besondere Aktion ausgeführt, und der Datenbankmanager erhält seine CPU-Zeit in der normalen Weise, in der das Betriebssystem allen Prozessen und Threads Prozessorzeit zuweist. Wenn der Parameter auf einen anderen Wert als -1 gesetzt wird, erstellt der Datenbankmanager seine Prozesse und Threads mit einer statischen Priorität, die dem Wert des Parameters entspricht. Dadurch können Sie mit diesem Parameter die Priorität steuern, mit der die Prozesse und Threads des Datenbankmanagers auf Ihrem System ausgeführt werden.

Mit diesem Parameter kann der Durchsatz des Datenbankmanagers erhöht werden. Die Werte für diesen Parameters sind von dem Betriebssystem abhängig, auf dem der Datenbankmanager ausgeführt wird. Beispielsweise ergeben in einer auf UNIX basierenden Umgebung niedrige numerische Werte hohe Prioritäten. Wenn der Parameter auf einen Wert zwischen 41 und 125 gesetzt wird, erstellt der Datenbankmanager seine Agenten mit einer statischen UNIX-Priorität, die dem Wert dieses Parameters entspricht. Dies ist in auf UNIX basierenden Umgebungen von Bedeutung, weil numerisch niedrige Werte hohe Prioritäten für den Datenbankmanager ergeben. Bei anderen Prozessen (einschließlich Anwendungen und Benutzern) können jedoch Verzögerungen auftreten, da sie nicht genügend CPU-Zeit erhalten. Sie sollten den Wert für diesen Parameter mit den anderen Aktivitäten, die Sie auf der Maschine erwarten, abstimmen.

In einer OS/2-Umgebung ergeben höhere numerische Werte höhere Prioritäten.

Empfehlung: Zu Anfang sollte der Standardwert verwendet werden. Dieser Wert stellt einen guten Kompromiss zwischen den Antwortzeiten für andere Benutzer bzw. Anwendungen und dem Durchsatz des Datenbankmanagers dar.

Wenn die Datenbankleistung von Bedeutung ist, können Sie durch Vergleichstests (Benchmark-Tests) die optimale Einstellung für diesen Parameter bestimmen. Eine Erhöhung der Priorität des Datenbankmanagers sollte nur mit großer Vorsicht vorgenommen werden, da die Leistung anderer Benutzerprozesse erheblich beeinträchtigt werden kann, besonders dann, wenn die CPU-Auslas-

tung sehr hoch ist. Durch Erhöhen der Priorität der Datenbankmanagerprozesse und -Threads können bedeutende Leistungssteigerungen erzielt werden.

Anmerkung: Wenn Sie auf UNIX-Plattformen für diesen Parameter einen anderen als den Standardwert verwenden, können Sie Governor nicht verwenden, um Agentenprioritäten zu ändern.

Maximale Anzahl von Agenten (maxagents)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 200 [1 – 64 000]

400 [1 – 64 000] auf einem partitionierten Datenbank-Server mit lokalen und fernen Clients

10 [1 – 64 000] auf dem Satellitendatenbank-Server mit lokalen Clients

Maßeinheit Zähler

Zugehörige Parameter

- „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461
- „Maximale Anzahl gleichzeitig aktiver Agenten (maxcagents)" auf Seite 469
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470
- „Maximale Anzahl von DARI-Prozessen (maxdari)" auf Seite 475
- „Minimaler reservierter privater Speicher (min_priv_mem)" auf Seite 433
- „Größe des Agentenpools (num_poolagents)" auf Seite 472

Mit diesem Parameter wird die maximale Anzahl von Datenbankmanageragenten (Koordinationsagenten und Subagenten) angegeben, die zu einem

gegebenen Zeitpunkt zur Verfügung stehen, um Anwendungsanforderungen zu empfangen. Wenn Sie die Anzahl koordinierender Agenten begrenzen möchten, verwenden Sie den Parameter *max_coordagents*.

Dieser Parameter kann in Umgebungen mit eingeschränkten Speicherkapazitäten nützlich sein, um den Gesamtspeicherbedarf des Datenbankmanagers zu begrenzen, da jeder zusätzliche Agent zusätzlichen Speicher benötigt.

Empfehlung: Der Wert des Parameters *maxagents* sollte mindestens der Summe der Werte für *maxappls* aller Datenbanken, auf die gleichzeitig zugegriffen werden kann, entsprechen. Wenn die Anzahl der Datenbanken größer als der Wert des Parameters *numdb* ist, dann besteht die sicherste Methode zur Angabe dieses Werts darin, das Produkt von *numdb* und dem größten Wert für *maxappls* zu bilden.

Für die Initialisierung und Verwaltung jedes weiteren Agenten ist zusätzlicher Speicher erforderlich, der beim Starten des Datenbankmanagers zugeordnet wird.

Maximale Anzahl gleichzeitig aktiver Agenten (maxcagents)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] -1 (*max_coordagents*) [-1; 1 – *max_coordagents*]

Maßeinheit Zähler

Zugehörige Parameter

- „Maximale Anzahl aktiver Anwendungen (maxappls)" auf Seite 461
- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470

Mit diesem Parameter wird die maximale Anzahl von Datenbankmanageragenten festgelegt, die gleichzeitig eine Datenbankmanagertransaktion ausführen können. Dieser Parameter wird zur Steuerung der Systembelastung in

Phasen hoher gleichzeitiger Anwendungsaktivität verwendet. Sie können beispielsweise ein System haben, das eine große Anzahl von Verbindungen anfordert, jedoch nur über eine begrenzte Speicherkapazität zur Verarbeitung dieser Verbindungen verfügt. In diesem Fall kann die Anpassung dieses Parameters sehr nützlich sein, da es hier aufgrund hoher gleichzeitiger Aktivität zu übermäßigem Seitenwechseln durch das Betriebssystem kommen kann.

Dieser Parameter schränkt nicht die Anzahl der Anwendungen ein, die mit einer Datenbank verbunden sein können. Er begrenzt nur die Anzahl der Datenbankmanageragenten, die gleichzeitig vom Datenbankmanager verarbeitet werden können, wodurch der Bedarf an Systemressourcen in Phasen sehr starker Belastung eingeschränkt wird.

Ein Wert von –1 gibt an, dass der Grenzwert gleich dem Wert des über den Parameter *max_coordagents* festgelegten Grenzwerts ist.

Empfehlung: In den meisten Fällen kann der Standardwert für diesen Parameter übernommen werden. In Fällen, wo es durch den umfassenden gleichzeitigen Zugriff von Anwendungen zu Problemen kommt, können Sie durch Vergleichstests (Benchmark-Tests) die beste Einstellung für diesen Parameter ermitteln, um die Leistung der Datenbank zu optimieren.

Maximale Anzahl koordinierender Agenten (max_coordagents)

Konfigurationsart
Datenbankmanager

Gilt für
- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart
Konfigurierbar

Standardwert [Bereich]
-1 (*maxagents* – *num_initagents*)

[-1, 0–*maxagents*]

In Umgebungen mit partitionierten Datenbanken und Umgebungen, in denen *intra_parallel* auf "Yes" gesetzt ist, ist der Standardwert *maxagents* minus *num_initagents*; andernfalls ist der Standardwert *maxagents*. Dadurch wird sichergestellt, dass in Datenbankumgebungen ohne partitionierte Datenbanken *max_coorda-*

gents immer gleich *maxagents* ist, außer wenn das System für partitionsinterne Parallelität konfiguriert ist.

Wenn Sie nicht mit einer Umgebung mit partitionierten Datenbanken arbeiten und den Parameter *intra_parallel* nicht aktiviert haben, muss *max_coordagents* gleich *maxagents* sein.

Zugehörige Parameter

- „Anfangswert für die Anzahl Agenten im Pool (num_initagents)" auf Seite 473
- „Größe des Agentenpools (num_poolagents)" auf Seite 472
- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468
- „Partitionsinterne Parallelität aktivieren (intra_parallel)" auf Seite 543

Dieser Parameter legt die maximale Anzahl koordinierender Agenten fest, die in einer Umgebung mit partitionierten oder nicht partitionierten Datenbanken gleichzeitig auf einem Server vorhanden sein können.

Für jede lokale oder ferne Anwendung, die eine Verbindung zu einer Datenbank oder einem Exemplar herstellt, wird ein koordinierender Agent aufgerufen. Anforderungen, für die eine Exemplarverbindung erforderlich ist, sind z. B. CREATE DATABASE, DROP DATABASE und Befehle des Datenbanksystemmonitors.

Maximale Anzahl logischer Agenten (max_logicagents)

Konfigurationsart	Datenbankmanager
Parameterart	Konfigurierbar
Standardwert [Bereich]	-1 (*max_coordagents*) [-1; *max_coordagents* — 64 000]

Dieser Parameter steuert die maximale Anzahl der Anwendungen, die mit dem Exemplar verbunden sein können. In der Regel wird jede Anwendung einem Koordinationsagenten zugeordnet. Ein Agent erleichtert die Operationen zwischen der Anwendung und der Datenbank. Die Konzentratorfunktion wird nicht aktiviert, wenn der Standardwert für diesen Parameter verwendet wird. Daher arbeitet jeder Agent mit seinem eigenen privaten Speicher und benutzt Ressourcen des Datenbankmanagers und globale Datenbankressourcen, wie z. B. den Pufferpool, gemeinsam mit anderen Agenten. Die Konzentratorfunktion wird hingegen aktiviert, wenn der Parameter auf einen Wert gesetzt wird, der den Standardwert übersteigt. Es ist die Aufgabe des

Konzentrators, die Anzahl der Server-Ressourcen pro Client-Anwendung so weit zu verringern, dass ein DB2 Connect-Gateway mehr als 10 000 Client-Verbindungen verarbeiten kann.

Weitere Informationen und Beispiele zur Verwendung von DB2 Connect als XA-Transaktionsunterstützungskonzentrator finden Sie im Handbuch *DB2 Connect Benutzerhandbuch*.

Ein Wert von –1 gibt an, dass der Grenzwert gleich dem Wert des über den Parameter *max_coordagents* festgelegten Grenzwerts ist.

Größe des Agentenpools (num_poolagents)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] -1 [-1, 0 — *maxagents*]

Der Wert für einen Server mit einer nicht partitionierten Datenbank und lokalen Clients ist bei Verwendung des Standardwerts der größere der beiden Werte *maxagents*/50 und *max_querydegree*.

Der Wert für einen Server mit einer nicht partitionierten Datenbank und lokalen und fernen Clients ist bei Verwendung des Standardwerts der größere der beiden folgenden Werte: *maxagents*/50 x *max_querydegree* oder *maxagents - max_coordagents*.

Der Wert für einen Datenbankpartitions-Server ist bei Verwendung des Standardwerts der größere der beiden folgenden Werte: *maxagents*/10 x *max_querydegree* oder *maxagents - max_coordagents*.

Zugehörige Parameter

- „Anfangswert für die Anzahl Agenten im Pool (num_initagents)” auf Seite 473

- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468
- „Maximaler Grad der Parallelität bei Abfragen (max_querydegree)" auf Seite 541
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470

Dieser Parameter stellt eine Richtlinie für die maximale Größe des Agentenpools dar (er ersetzt den Parameter *max_idleagents*, der in DB2 Version 2 verwendet wurde).

Der Agentenpool enthält Subagenten und freie Agenten. Freie Agenten können als parallele Subagenten oder als Koordinationsagenten verwendet werden. Wurden mehr Agenten erstellt, als der Wert dieses Parameters angibt, werden diese nach Ausführung ihrer aktuellen Anforderung nicht in den Pool zurückgestellt sondern beendet.

Ist der Wert für diesen Parameter 0, werden nach Bedarf Agenten erstellt und möglicherweise beendet, sobald sie ihre aktuelle Anforderung ausgeführt haben. Ist der Wert *maxagents* und ist der Pool gefüllt mit zugeordneten Subagenten, kann der Server nicht als Koordinatorknoten verwendet werden, weil keine neuen Koordinationsagenten erstellt werden können.

Empfehlung: Wenn Sie eine Entscheidungshilfeumgebung verwenden, in der nur wenige Anwendungen gleichzeitig mit der Datenbank verbunden sind, setzen Sie *num_poolagents* auf einen kleinen Wert, um zu verhindern, dass ein Agentenpool mit freien Agenten angefüllt wird.

Wenn Sie eine Transaktionsverarbeitungsumgebung verwenden, in der viele Anwendungen gleichzeitig mit der Datenbank verbunden sind, erhöhen Sie den Wert für *num_poolagents*, um den Aufwand für die häufige Erstellung und Beendigung von Agenten zu vermeiden.

Anfangswert für die Anzahl Agenten im Pool (num_initagents)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 0 [0 — *num_poolagents*]

Zugehörige Parameter

- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468
- „Größe des Agentenpools (num_poolagents)" auf Seite 472
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470

Dieser Parameter gibt an, wie viele freie Agenten beim Ausführen von DB2START im Agentenpool erstellt werden.

Gespeicherte Prozeduren (DARI)

Die folgenden Parameter können die DARI-Anwendungen beeinflussen:

- „DARI-Prozess beibehalten (keepdari)"
- „Maximale Anzahl von DARI-Prozessen (maxdari)" auf Seite 475
- „DARI-Prozess mit JVM initialisieren (initdari_jvm)" auf Seite 477
- „Anfangsanzahl der abgeschirmten DARI-Prozesse im Pool (num_initdaris)" auf Seite 477

Anmerkung: Der Begriff DARI bezieht sich auf gespeicherte Prozeduren.

DARI-Prozess beibehalten (keepdari)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] Yes [Yes; No]

Zugehörige Parameter „Maximale Anzahl von DARI-Prozessen (maxdari)" auf Seite 475

Mit diesem Parameter wird angegeben, ob ein DARI-Prozess nach Abschluss eines DARI-Aufrufs beibehalten wird oder nicht. DARI-Prozesse werden als getrennte Definitionseinheiten des Systems erstellt, um vom Benutzer geschriebenen DARI-Code vom Agentenprozess des Datenbankmanagers zu isolieren. Dieser Parameter gilt nur für Datenbank-Server.

Wenn der Parameter *keepdari* auf *no* gesetzt ist, wird für jeden DARI-Aufruf ein neuer DARI-Prozess erstellt und wieder gelöscht. Wenn der Parameter *keepdari* auf *yes* gesetzt ist, wird ein DARI-Prozess für nachfolgende DARI-Aufrufe wieder verwendet. Wird der Datenbankmanager gestoppt, werden alle noch nicht beendeten DARI-Prozesse beendet.

Wenn dieser Parameter auf *yes* gesetzt wird, werden vom Datenbankmanager zusätzliche Systemressourcen für jeden DARI-Prozess in Anspruch genommen, der aktiviert wird, solange die durch den Parameter *maxdari* definierte Anzahl noch nicht erreicht ist. Dies gilt nur, wenn kein bereits aktiver DARI-Prozess zur Verarbeitung eines nachfolgenden DARI-Aufrufs verfügbar ist. Dieser Parameter wird ignoriert, wenn der Parameter *maxdari* den Wert 0 hat.

Empfehlung: In einer Umgebung, in der die Anzahl der DARI-Anforderungen im Vergleich zu den übrigen Anforderungen groß ist und Systemressourcen nicht begrenzt sind, kann dieser Parameter auf *yes* gesetzt werden. Dadurch wird die DARI-Leistung erhöht, weil der zusätzliche Systemaufwand zur Ersterstellung eines DARI-Prozesses vermieden wird, da ein bereits vorhandener DARI-Prozess zur Verarbeitung des Aufrufs verwendet wird.

Zum Beispiel könnte bei einer OLTP-Bankanwendung (OLTP - Online-Transaktionsprogramm) für Soll- und Haben-Transaktionen der Code, mit dem jede Transaktion erfasst wird, in einer gespeicherten Prozedur ausgeführt werden, die in einem DARI-Prozess arbeitet. In dieser Anwendung wird die Hauptarbeit aus DARI-Prozessen heraus geleistet. Wenn dieser Parameter auf *no* gesetzt wird, entsteht für jede Transaktion der Systemaufwand zur Erstellung eines neuen DARI-Prozesses, wodurch die Leistung erheblich beeinträchtigt wird. Wenn dieser Parameter jedoch auf *yes* gesetzt wird, versucht jede Transaktion, einen vorhandenen DARI-Prozess zu verwenden, wodurch der zusätzliche Systemaufwand vermieden wird.

Maximale Anzahl von DARI-Prozessen (maxdari)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] -1 (*max_coordagents*) [-1; 0 – *max_coordagents*]

Maßeinheit Zähler

Zugehörige Parameter

- „Maximale Anzahl von Agenten (maxagents)" auf Seite 468
- „DARI-Prozess beibehalten (keepdari)" auf Seite 474
- „Anfangsanzahl der abgeschirmten DARI-Prozesse im Pool (num_initdaris)" auf Seite 477
- „Maximale Anzahl koordinierender Agenten (max_coordagents)" auf Seite 470

Mit diesem Parameter wird die maximale Anzahl von DARI-Prozessen angegeben, die auf dem Datenbank-Server aktiv sein können. Wenn dieser Grenzwert erreicht wird, können keine neuen DARI-Anforderungen mehr aufgerufen werden. Dieser Parameter gilt nur für Datenbank-Server.

Da für einen koordinierenden Agenten nicht mehr als ein DARI-Prozess aktiv sein kann, wird die maximale Anzahl von DARI-Prozessen auch von der maximalen Anzahl koordinierender Agenten (*max_coordagents*) beschränkt.

Empfehlung: Wenn die Umgebung die DARI-Funktion innerhalb des Datenbankmanagers verwendet, dann kann mit Hilfe dieses Parameters sichergestellt werden, dass eine geeignete Anzahl von DARI-Prozessen zur Verarbeitung von DARI-Aufrufen, die innerhalb des Datenbankmanagers gleichzeitig erfolgen, verfügbar ist.

Wenn der Wert dieses Parameters auf –1 gesetzt wird, ist die maximale Anzahl von DARI-Prozessen gleich dem für den Parameter *max_coordagents* definierten Wert.

Wenn sich herausstellt, dass der Standardwert für Ihre Umgebung nicht geeignet ist, weil eine unangemessen große Menge Systemressourcen für DARI-Prozesse verwendet wird und es deshalb zu Leistungseinbußen des Datenbankmanagers kommt, kann anhand der folgenden Informationen eine erste Optimierung für die Einstellung dieses Parameters vorgenommen werden:

```
maxdari = Anzahl Anwendungen, die gleichzeitig DARI-Aufrufe absetzen dürfen
```

Wenn der Parameter *keepdari* auf den Wert *yes* gesetzt ist, bleibt jeder erstellte DARI-Prozess bestehen und verbraucht auch dann noch Systemressourcen, wenn die Verarbeitung des DARI-Aufrufs beendet und die Steuerung an den Agenten zurückgegeben wurde.

Wenn in Ihrer Umgebung Systemressourcen so eingeschränkt sind, dass die Verarbeitungsressourcen für DARI nicht zur Verfügung stehen, können Sie DARI dadurch inaktivieren, dass Sie diesen Parameter auf den Wert 0 setzen.

DARI-Prozess mit JVM initialisieren (initdari_jvm)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] No [Yes; No]

Zugehörige Parameter

- „Maximale Anzahl von DARI-Prozessen (maxdari)" auf Seite 475
- „Anfangsanzahl der abgeschirmten DARI-Prozesse im Pool (num_initdaris)"
- „DARI-Prozess beibehalten (keepdari)" auf Seite 474

Dieser Parameter gibt an, ob jeder abgeschirmte DARI-Prozess beim Start die Java Virtual Machine (JVM) lädt. Dieser Parameter verringert die einleitende Startzeit für abgeschirmte gespeicherte Java-Prozeduren, insbesondere wenn er zusammen mit dem Parameter *num_initdaris* verwendet wird. Dieser Parameter kann eventuell die einleitende Ladezeit für abgeschirmte gespeicherte Nicht-Java-Prozeduren erhöhen, weil für sie die JVM nicht erforderlich ist.

Anfangsanzahl der abgeschirmten DARI-Prozesse im Pool (num_initdaris)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 0 [0 — *maxdari*]

Zugehörige Parameter

- „Maximale Anzahl von DARI-Prozessen (maxdari)" auf Seite 475
- „DARI-Prozess mit JVM initialisieren (initdari_jvm)" auf Seite 477
- „DARI-Prozess beibehalten (keepdari)" auf Seite 474

Dieser Parameter gibt die Anfangsanzahl der inaktiven abgeschirmten DARI-Prozesse an, die bei der Ausführung von DB2START im DARI-Pool erstellt werden. Wenn Sie diesen Parameter entsprechend einstellen, wird die einleitende Startzeit für abgeschirmte gespeicherte Prozeduren verringert. Dieser Parameter wird ignoriert, wenn *keepdari* nicht angegeben wird.

Protokollieren und Wiederherstellung

Die Wiederherstellung der Umgebung kann eine wichtige Maßnahme sein, um den Verlust kritischer Daten zu verhindern. Es steht eine Reihe von Parametern zur Verfügung, die Ihnen bei der Verwaltung der Umgebung helfen, um sicherstellen zu können, dass eine geeignete Wiederherstellung der Daten bzw. Transaktionen durchgeführt werden kann. Diese Parameter werden in folgende Kategorien unterteilt:

- „Datenbankprotokolldateien"
- „Datenbankprotokollierung" auf Seite 486
- „Wiederherstellung" auf Seite 492
- „Wiederherstellung der verteilten Arbeitseinheit" auf Seite 499.

Datenbankprotokolldateien

Die folgenden Parameter enthalten Informationen zu Anzahl, Größe und Status der Dateien, die zur Datenbankprotokollierung verwendet werden:

- „Protokolldateigröße (logfilsiz)"
- „Anzahl primärer Protokolldateien (logprimary)" auf Seite 480
- „Anzahl sekundärer Protokolldateien (logsecond)" auf Seite 482
- „Datenbankprotokollpfad ändern (newlogpath)" auf Seite 483
- „Pfad zu Protokolldateien (logpath)" auf Seite 485
- „Erste aktive Protokolldatei (loghead)" auf Seite 485

Protokolldateigröße (logfilsiz)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 1000 [4 — 65 535]

	Windows NT	250 [4 — 65 535]
	OS/2	250 [4 — 65 535]
Maßeinheit	Seiten (4 KB)	
Zugehörige Parameter		• „Anzahl primärer Protokolldateien (logprimary)" auf Seite 480 • „Anzahl sekundärer Protokolldateien (logsecond)" auf Seite 482 • „Vor dem bedingten Prüfpunkt zu schreibende Protokollsätze (softmax)" auf Seite 487

Mit diesem Parameter wird die Größe jeder primären und sekundären Protokolldatei definiert. Die Größe dieser Protokolldateien beschränkt die Anzahl der Protokollsätze, die in die Dateien geschrieben werden können, bevor sie voll sind und eine neue Protokolldatei erforderlich wird.

Die Verwendung primärer und sekundärer Protokolldateien sowie die Maßnahmen, die ausgeführt werden, wenn eine Protokolldatei voll ist, sind von der Art der ausgeführten Protokollierung abhängig:

- Umlaufprotokollierung

  Eine primäre Protokolldatei kann wieder verwendet werden, wenn die in ihr aufgezeichneten Änderungen festgeschrieben wurden. Wenn die Größe der Protokolldatei beschränkt ist und Anwendungen eine große Anzahl von Änderungen an der Datenbank vorgenommen haben, ohne die Änderungen festzuschreiben, kann eine primäre Protokolldatei schnell voll werden. Wenn alle primären Protokolldateien voll sind, ordnet der Datenbankmanager sekundäre Protokolldateien für die neuen Protokollsätze zu.
- Protokollierung mit Protokollspeicherung

  Wenn eine primäre Protokolldatei voll ist, wird das Protokoll archiviert und eine neue primäre Protokolldatei zugeordnet.

Empfehlung: Die Größe der Protokolldateien muss mit der Anzahl der primären Protokolldateien abgestimmt werden:

- Der Wert des Parameters *logfilsiz* sollte erhöht werden, wenn an der Datenbank eine große Anzahl von Aktualisierungs-, Lösch- und/oder Einfügetransaktionen ausgeführt wird, durch die die Protokolldatei sehr schnell voll wird.

  Anmerkung: Die Protokolldateien dürfen insgesamt maximal 32 GB groß sein. Das heißt, dass die Anzahl der Protokolldateien (*logprimary* + *logsecond*) multipliziert mit der Größe der einzelnen Protokolldateien in Byte (*logfilsiz* * 4096) kleiner als 32 GB sein muss.

Eine zu kleine Protokolldatei kann sich auf die Systemleistung auswirken, da das Archivieren alter Protokolldateien, das Zuordnen neuer Protokolldateien sowie das Warten auf verwendbare Protokolldateien einen erhöhten Systemaufwand mit sich bringen.

- Der Wert des Parameters *logfilsiz* sollte verringert werden, wenn Plattenspeicherplatz knapp ist, da primäre Protokolldateien im Voraus mit dieser Größe zugeordnet werden.

  Eine zu große Protokolldatei kann Ihre Flexibilität bei der Verwaltung archivierter Protokolldateien bzw. beim Kopieren der Protokolldateien einschränken, da möglicherweise auf einigen Platten keine vollständige Protokolldatei gespeichert werden kann.

Wenn Sie die Protokollspeicherung verwenden, wird die aktuelle aktive Protokolldatei geschlossen und abgeschnitten, wenn die letzte Anwendung die Verbindung zu einer Datenbank trennt. Für die nächste zur Datenbank hergestellte Verbindung wird die nächste Protokolldatei verwendet. Wenn Sie also die Protokollanforderungen Ihrer gleichzeitig ausgeführten Anwendungen kennen, können Sie möglicherweise eine Protokolldateigröße festlegen, durch die nicht zu viel Speicher umsonst zugeordnet wird.

Weitere Informationen zu diesem Parameter finden Sie im Abschnitt zu den Konfigurationsparametern für Datenbankprotokollierung im Handbuch *Systemverwaltung: Implementierung*.

Anzahl primärer Protokolldateien (logprimary)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	3 [2 – 128]
Maßeinheit	Zähler
Zuordnung	• Wenn die Datenbank erstellt wird. • Wenn ein Protokoll an eine andere Speicherposition versetzt wird (dies geschieht, wenn der Parameter *logpath* aktualisiert wird). • Wenn nach der Erhöhung des Werts für den Parameter *logprimary* die nächste Verbindung zur Datenbank hergestellt wird, nachdem alle Benutzer die Verbindung getrennt hatten. • Wenn nach der Archivierung einer Protokolldatei eine neue Protokolldatei

zugeordnet wird. (Der Parameter *logretain* oder *userexit* muss aktiviert sein.)
- Wenn der Parameter *logfilsiz* geändert wurde, wird für die aktiven Protokolldateien die neue Größe aktiv, wenn die nächste Verbindung zur Datenbank hergestellt wird, nachdem alle Benutzer die Verbindung zur Datenbank getrennt hatten.

Freigabe

Eine Freigabe erfolgt, wenn der Wert für diesen Parameter herabgesetzt wird. Wenn der Wert herabgesetzt wird, werden nicht mehr benötigte Protokolldateien bei der nächsten Verbindung zur Datenbank gelöscht.

Zugehörige Parameter

- „Protokolldateigröße (logfilsiz)" auf Seite 478
- „Anzahl sekundärer Protokolldateien (logsecond)" auf Seite 482
- „Protokollspeicherung für Wiederherstellung (logretain)" auf Seite 490
- „Benutzerausgang für Protokollierung aktivieren (userexit)" auf Seite 491

Die primären Protokolldateien reservieren eine feste Menge Speicher, der den Wiederherstellungsprotokollen zugeordnet wird. Mit diesem Parameter kann die Anzahl der im Voraus zugeordneten primären Protokolldateien festgelegt werden.

Bei der Umlaufprotokollierung werden die primären Protokolldateien nacheinander wiederholt verwendet. Das heißt, wenn ein Protokoll voll ist, wird die nächste primäre Protokolldatei in der Reihenfolge verwendet, wenn sie verfügbar ist. Ein Protokoll wird als verfügbar angesehen, wenn alle Arbeitseinheiten mit Protokollsätzen in diesem Protokoll festgeschrieben oder rückgängig gemacht wurden. Wenn die nächste Protokolldatei in der Reihenfolge nicht verfügbar ist, wird eine sekundäre Protokolldatei zugeordnet und verwendet. Es werden solange weitere sekundäre Protokolldateien zugeordnet und verwendet, bis die nächste primäre Protokolldatei in der Reihenfolge verfügbar wird oder der durch den Parameter *logsecond* festgelegte Grenzwert erreicht wird. Sobald diese sekundären Protokolldateien vom Datenbankmanager nicht mehr benötigt werden, wird der für sie verwendete Speicher wieder freigegeben.

Für die Anzahl der primären und sekundären Protokolldateien muss folgende Gleichung gelten:
- (*logprimary* + *logsecond*) <= 128

Empfehlung: Der Wert, der für diesen Parameter gewählt wird, ist von einer Reihe von Faktoren abhängig, zu denen auch die Art der Protokollierung, die Größe der Protokolldateien und die Art der Verarbeitungsumgebung (z.B. die Länge der Transaktionen und die Häufigkeit des Festschreibens) gehören.

Durch die Erhöhung dieses Werts wird mehr Plattenspeicher für die Protokolle erforderlich, weil die primären Protokolldateien bereits bei der ersten Verbindung zur Datenbank im Voraus zugeordnet werden.

Wenn sich herausstellt, dass häufig sekundäre Protokolldateien zugeordnet werden, kann die Systemleistung möglicherweise durch eine Vergrößerung der Protokolldateien (*logfilsiz*) oder durch eine Erhöhung der Anzahl primärer Protokolldateien verbessert werden.

Bei Datenbanken, auf die nicht oft zugegriffen wird, sollte zur Einsparung von Plattenspeicherplatz dieser Parameter auf den Wert 2 gesetzt werden. Bei Datenbanken, für die die aktualisierende Wiederherstellung aktiviert ist, sollte dieser Parameter auf einen größeren Wert gesetzt werden, um den erhöhten Systemaufwand aufgrund der beinahe sofort erforderlichen Zuordnung neuer Protokolle zu vermeiden.

Sie können den Datenbanksystemmonitor zur Ermittlung der geeigneten Größe für die primären Protokolldateien verwenden.

Weitere Informationen finden Sie in den Beschreibungen zu folgenden Monitorelementen im Handbuch *System Monitor Guide and Reference*:
- *sec_log_used_top (Max. verwendeter sekundärer Protokollbereich)*
- *tot_log_used_top (Max. verwendeter Gesamtbereich für Protokolle)*
- *sec_logs_allocated (Momentan zugeordnete sekundäre Protokolle)*

Die Beobachtung dieser Monitorwerte über einen gewissen Zeitraum hinweg kann sich für die geeignete Einstellung der Parameter als nützlich erweisen, da Durchschnittswerte Ihre tatsächlichen Anforderungen wahrscheinlich besser widerspiegeln.

Anzahl sekundärer Protokolldateien (logsecond)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	2 [0 – 126]
Maßeinheit	Zähler
Zuordnung	Nach Bedarf, wenn *logprimary* nicht ausreichend ist (siehe unten).

Freigabe Wenn der Datenbankmanager feststellt, dass sie nicht mehr erforderlich sind.

Zugehörige Parameter

- „Protokolldateigröße (logfilsiz)" auf Seite 478
- „Anzahl primärer Protokolldateien (logprimary)" auf Seite 480
- „Protokollspeicherung für Wiederherstellung (logretain)" auf Seite 490
- „Benutzerausgang für Protokollierung aktivieren (userexit)" auf Seite 491

Dieser Parameter gibt die Anzahl der sekundären Protokolldateien an, die (nach Bedarf) erstellt und für Wiederherstellungsprotokolle verwendet werden. Wenn die primären Protokolldateien voll sind, werden die sekundären Protokolldateien (in der Größe *logfilsiz*) nacheinander so zugeordnet, wie sie benötigt werden, und zwar höchstens so viele, wie durch diesen Parameter festgelegt wird. Wenn mehr sekundäre Protokolldateien erforderlich sind, als von diesem Parameter zugelassen werden, wird ein Fehlercode an die Anwendung zurückgegeben, und die Datenbank wird beendet.

Unter „Anzahl primärer Protokolldateien (logprimary)" auf Seite 480 finden Sie weitere Informationen zur Verwendung sekundärer Protokolldateien.

Empfehlung: Verwenden Sie sekundäre Protokolldateien für Datenbanken, die in periodischen Zeitabständen immer wieder große Mengen an Protokollspeicher benötigen. Sekundäre Protokolldateien sollten beispielsweise eingesetzt werden, wenn eine Anwendung, die einmal im Monat ausgeführt wird, mehr Protokollspeicher benötigt, als durch die primären Protokolldateien zur Verfügung gestellt wird. Da sekundäre Protokolldateien keinen permanenten Dateispeicherplatz erforderlich machen, sind sie für solche Zwecke gut geeignet.

Datenbankprotokollpfad ändern (newlogpath)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] Null [gültiger Pfad oder gültige Einheit]

Zugehörige Parameter

- „Pfad zu Protokolldateien (logpath)" auf Seite 485
- „Datenbank ist konsistent (database_consistent)" auf Seite 511

Mit diesem Parameter können Sie eine Zeichenfolge mit bis zu 242 Byte angeben, um die Speicherposition der Protokolldateien zu ändern. Die Zeichenfolge kann auf einen Pfadnamen oder auf eine unformatierte Einheit verweisen. Verweist die Zeichenfolge auf einen Pfadnamen, muss es sich um einen vollständig qualifizierten Pfad handeln und nicht um einen relativen Pfad.

Anmerkung: In einer Umgebung mit partitionierten Datenbanken wird die Knotennummer automatisch an den Pfad angefügt. Dadurch wird die Eindeutigkeit des Pfads in Konfigurationen mit mehreren logischen Knoten sichergestellt.

Geben Sie zum Angeben einer Einheit eine Zeichenfolge an, die vom Betriebssystem als eine Einheit erkannt wird. Beispiel:

- Unter Windows NT, `\\.\d:` oder `\\.\PhysicalDisk5`

  Anmerkung: Damit Protokolle in eine Einheit geschrieben werden können, muss Windows NT Version 4.0 mit Service Pack 3 oder höher installiert sein.
- Auf UNIX-Plattformen `/dev/rdblog8`

Anmerkung: Eine Einheit können Sie nur auf AIX-, Windows 2000-, Windows NT-, Solaris-, HP-UX-, NUMA-Q- und Linux-Plattformen angeben.

Diese Einstellung wird nur dann zum Wert des Parameters *logpath* , wenn die beiden folgenden Bedingungen zutreffen:

- Die Datenbank ist in einem konsistenten Zustand, wie durch den Parameter *database_consistent* angegeben.
- Alle Benutzer sind von der Datenbank getrennt.

Wenn die erste neue Verbindung zur Datenbank hergestellt wird, versetzt der Datenbankmanager die Protokolle an die neue, von *logpath* angegebene Speicherposition.

Im alten Protokollpfad befinden sich eventuell noch Protokolldateien. Diese Protokolldateien wurden eventuell nicht archiviert. Sie müssen sie möglicherweise manuell archivieren. Wenn Sie zudem für diese Datenbank Replikation ausführen, benötigt Replikation eventuell weiterhin die vor der Protokollpfadänderung vorhandenen Protokolldateien. Wenn der Datenbankkonfigurationsparameter für Benutzer-Exit (*userexit*) für die Datenbank auf "Yes" gesetzt ist und wenn alle Protokolldateien entweder automatisch durch DB2 oder von Ihnen selbst manuell archiviert wurden, dann kann DB2 die Protokolldateien zum Beenden des Replikationsprozesses abrufen. Andernfalls können Sie die Dateien aus dem alten Protokollpfad in den neuen Protokollpfad kopieren.

Empfehlung: Es empfiehlt sich, die Protokolldateien auf einer physischen Platte zu speichern, auf der **nicht** häufig E/A-Operationen auftreten. Sie sollten beispielsweise die Protokolldateien nicht auf derselben Platte wie das Betriebssystem oder umfangreiche Datenbanken speichern. Dadurch werden die Protokolliervorgänge effizient und verursachen nur ein Minimum an Systemaufwand, wie z. B. Warten auf E/A-Operationen.

Mit dem Datenbanksystemmonitor können Sie die Anzahl der E/A-Operationen für die Datenbankprotokollierung verfolgen.

Weitere Informationen finden Sie in den Beschreibungen der folgenden Monitorelemente im Handbuch *System Monitor Guide and Reference*:
- *log_reads* (Anzahl gelesener Protokollseiten)
- *log_writes* (Anzahl geschriebener Protokollseiten)

Die oben genannten Datenelemente geben das Volumen der E/A-Aktivitäten für die Datenbankprotokollierung zurück. Sie können ein Tool zur Betriebssystemüberwachung verwenden, um Daten zu anderen Platten-E/A-Aktivitäten zu sammeln. Anschließend können Sie beide Arten von E/A-Aktivitäten vergleichen.

Pfad zu Protokolldateien (logpath)

Konfigurationsart	Datenbank
Parameterart	Informativ
Zugehörige Parameter	„Datenbankprotokollpfad ändern (newlogpath)" auf Seite 483

Dieser Parameter gibt den aktuellen Pfad an, der für die Protokolldateien verwendet wird. Dieser Parameter kann nicht direkt geändert werden, da der Wert vom Datenbankmanager festgelegt wird, wenn eine Änderung am Parameter *newlogpath* wirksam wird.

Bei der Erstellung einer Datenbank wird die zugehörige Datei des Wiederherstellungsprotokolls in einem Unterverzeichnis des Verzeichnisses erstellt, in dem sich die Datenbank befindet. Standardmäßig wird dieses Unterverzeichnis mit dem Namen SQLOGDIR in dem Verzeichnis angelegt, das für die Datenbank erstellt wurde.

Erste aktive Protokolldatei (loghead)

Konfigurationsart	Datenbank
Parameterart	Informativ

Dieser Parameter gibt den Namen der zurzeit aktiven Protokolldatei an.

Datenbankprotokollierung

Die folgenden Parameter können den Typ und die Leistung der Datenbankprotokollierung beeinflussen:

- „Anzahl der Gruppenfestschreibungen (mincommit)"
- „Vor dem bedingten Prüfpunkt zu schreibende Protokollsätze (softmax)" auf Seite 487
- „Protokollspeicherung für Wiederherstellung (logretain)" auf Seite 490
- „Benutzerausgang für Protokollierung aktivieren (userexit)" auf Seite 491

Anzahl der Gruppenfestschreibungen (mincommit)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	1 [1 – 25]
Maßeinheit	Zähler

Mit diesem Parameter können Sie das Schreiben von Protokollsätzen auf die Festplatte verzögern, bis eine Mindestanzahl von Festschreibungen (COMMIT-Operationen) ausgeführt wurde. Diese Verzögerung kann dazu beitragen, den Aufwand des Datenbankmanager für das Schreiben von Protokollsätzen zu reduzieren. Dies führt zu einer Leistungssteigerung, wenn mehrere Anwendungen für eine Datenbank ausgeführt und von den Anwendungen viele Festschreibungen innerhalb kurzer Zeit angefordert werden.

Diese Gruppierung von Festschreibungen wird nur dann ausgeführt, wenn der Wert dieses Parameters größer als eins und die Anzahl der Anwendungen, die mit der Datenbank verbunden sind, größer oder gleich dem Wert dieses Parameters ist. Wenn die Gruppierung von Festschreibungen ausgeführt wird, werden Festschreibungsanforderungen von Anwendungen solange zurückgehalten, bis entweder eine Sekunde vergangen oder die Anzahl der Festschreibungsanforderungen gleich dem Wert dieses Parameters ist.

Am Wert dieses Parameters vorgenommene Änderungen werden sofort wirksam. Sie brauchen nicht abzuwarten, bis alle Anwendungen von der Datenbank getrennt wurden.

Empfehlung: Erhöhen Sie den Wert dieses Parameters über seinen Standardwert hinaus, wenn mehrere Schreib-/Leseanwendungen regelmäßig gleichzeitig Datenbankfestschreibungen anfordern. Dadurch wird eine höhere Effizienz bei E/A-Operationen für Protokolldateien erzielt, da diese Operationen weniger häufig auftreten und bei jeder Operation mehr Protokollsätze geschrieben werden.

Sie könnten auch die Anzahl der Transaktionen pro Sekunde ermitteln und diesen Parameter so anpassen, dass die Höchstanzahl von Transaktionen pro Sekunde (oder ein großer Prozentsatz davon) von diesem Parameter berücksichtigt wird. Durch die Berücksichtigung der Spitzenaktivitäten würde der Systemaufwand für das Schreiben von Protokollsätzen in Phasen mit hohem Transaktionsaufkommen minimiert.

Wenn der Wert des Parameters *mincommit* erhöht wird, kann es auch notwendig werden, den Wert des Parameters *logbufsz* zu erhöhen, um zu vermeiden, dass ein voller Protokollpuffer eine Schreiboperation während dieser Phasen mit hohem Transaktionsaufkommen erzwingt. In diesem Fall sollte der Wert für den Parameter *logbufsz* nach folgender Formel festgelegt werden:

```
mincommit * (durchschnittlicher Protokollbereich für eine Transaktion)
```

Sie können den Datenbanksystemmonitor folgendermaßen zur Optimierung des Werts dieses Parameters verwenden:

- Berechnen der Höchstanzahl von Transaktionen pro Sekunde:

  Durch das Ziehen von Stichproben über einen typischen Arbeitstag hinweg können Sie die Phasen mit hohem Transaktionsaufkommen ermitteln. Sie können die Gesamtanzahl der Transaktionen durch Addieren der Werte für folgende Monitorelemente berechnen:
  - *commit_sql_stmts (versuchte COMMIT-Anweisungen)*
  - *rollback_sql_stmts (versuchte ROLLBACK-Anweisungen)*

  Anhand dieser Stichproben und der verfügbaren Zeitmarken können Sie die Anzahl der Transaktionen pro Sekunde berechnen.
- Berechnen des pro Transaktion verwendeten Protokollspeicherbereichs:

  Durch das Ziehen von Stichproben über einen gewissen Zeitraum hinweg und für eine Anzahl von Transaktionen können Sie den durchschnittlich verwendeten Protokollspeicherbereich mit dem folgenden Monitorelement berechnen:
  - *log_space_used (Protokollbereich für Arbeitseinheit)*

  Weitere Informationen zum Datenbanksystemmonitor finden Sie in *System Monitor Guide and Reference*.

Vor dem bedingten Prüfpunkt zu schreibende Protokollsätze (softmax)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	100 [1 – 100 * *logprimary*]
Maßeinheit	Prozentsatz der Größe einer primären Protokolldatei

Zugehörige Parameter

- „Protokolldateigröße (logfilsiz)" auf Seite 478
- „Anzahl primärer Protokolldateien (logprimary)" auf Seite 480

Dieser Parameter hat folgende Funktionen:

- Beeinflussen der Anzahl von Protokolldateien, die für die Wiederherstellung nach einem Systemabsturz (z. B. nach einem Stromausfall) erforderlich sind. Wenn beispielsweise der Standardwert verwendet wird, versucht der Datenbankmanager, die Anzahl der wiederherzustellenden Protokolldateien auf 1 zu halten. Wenn Sie 300 als Wert für diesen Parameter angeben, versucht der Datenbankmanager, die Anzahl der wiederherzustellenden Protokolldateien auf 3 zu halten.

  Beim Steuern der Anzahl der Protokolldateien, die für die Wiederherstellung nach einem Systemabsturz erforderlich sind, verwendet der Datenbankmanager diesen Parameter zum Starten der Seitenlöschfunktionen. Dadurch wird sichergestellt, dass Seiten, die älter sind als das angegebene Wiederherstellungsfenster, bereits auf Platte geschrieben sind. („Wiederherstellungsfenster" bezeichnet hierbei den Zeitraum, der für die Verarbeitung der Protokolldateien bei der Wiederherstellung nach einem Systemabsturz erforderlich ist.)
- Festlegen der Frequenz der bedingten Prüfpunkte.

Zum Zeitpunkt einer Datenbankstörung, die zum Beispiel durch einen Stromausfall verursacht wird, kann für in der Datenbank ausgeführte Änderungen Folgendes gelten:

- Die Änderungen wurden nicht festgeschrieben, jedoch wurden die Daten im Pufferpool aktualisiert.
- Die Änderungen wurden festgeschrieben, jedoch noch nicht vom Pufferpool auf die Festplatte geschrieben.
- Die Änderungen wurden festgeschrieben und vom Pufferpool auf die Festplatte geschrieben.

Wenn eine Datenbank erneut gestartet wird, werden die Protokolldateien verwendet, um eine Wiederherstellung der Datenbank nach dem Systemabsturz auszuführen, die sicherstellt, dass die Datenbank in einem konsistenten Zustand verbleibt (d. h., alle festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen, und keine der nicht festgeschriebenen Transaktionen werden in der Datenbank nachvollzogen).

Der Datenbankmanager verwendet eine Protokollsteuerdatei, um festzustellen, welche Datensätze aus der Protokolldatei in der Datenbank nachvollzogen werden müssen. Diese Protokollsteuerdatei wird in regelmäßigen Abständen auf die Festplatte geschrieben, und der Datenbankmanager kann abhängig von der Frequenz dieses Ereignisses Protokollsätze festgeschriebener Transaktionen oder Protokollsätze zu Änderungen, die bereits aus dem Pufferpool auf

Platte geschrieben wurden, nachvollziehen. Diese Protokollsätze haben keine Auswirkung auf die Datenbank, das Nachvollziehen dieser Protokollsätze führt jedoch zu einem gewissen erhöhten Systemaufwand während des Neustarts der Datenbank.

Die Protokollsteuerdatei wird immer dann auf die Festplatte geschrieben, wenn eine Protokolldatei voll ist, und außerdem bei bedingten Prüfpunkten. Sie können diesen Konfigurationsparameter dazu verwenden, zusätzliche bedingte Prüfpunkte auszulösen.

Die Ablaufsteuerung für bedingte Prüfpunkte wird mit Hilfe der Differenz zwischen dem „aktuellen Stand" und dem „aufgezeichneten Stand" festgelegt. Diese Differenz wird als Prozentsatz vom Wert des Parameters *logfilsiz* angegeben. Der „aufgezeichnete Stand" wird anhand des ältesten gültigen Protokollsatzes ermittelt, der von der Protokollsteuerdatei auf der Festplatte angegeben wird, während der „aktuelle Stand" anhand der Protokollsteuerinformationen im Hauptspeicher ermittelt wird. (Der älteste gültige Protokollsatz ist der erste Protokollsatz, der bei einem Wiederherstellungsprozess gelesen würde.) Der bedingte Prüfpunkt wird ausgelöst, wenn der nach der folgenden Formel berechnete Wert größer oder gleich dem Wert dieses Parameters ist:

```
( (Bereich zw. aufgezeichnetem u. aktuellem Stand) / logfilsiz ) * 100
```

Empfehlung: Sie können den Wert dieses Parameters erhöhen oder verringern, je nachdem, ob Ihr Wiederherstellungsfenster größer oder kleiner als eine Protokolldatei sein soll. Wenn Sie den Wert dieses Parameters herabsetzen, wird der Datenbankmanager veranlasst, die Seitenlöschfunktionen häufiger auszulösen und häufiger bedingte Prüfpunkte anzusetzen. Diese Maßnahmen können die Anzahl der Protokollsätze, die verarbeitet werden müssen, und die Anzahl der überschüssigen Protokollsätze, die während der Wiederherstellung verarbeitet werden, verringern.

Beachten Sie jedoch, dass mehr Auslöser von Seitenlöschfunktionen und häufigere bedingte Prüfpunkte den Systemaufwand erhöhen, der mit der Protokollierung der Datenbank verbunden ist, was sich negativ auf die Leistung des Datenbankmanagers auswirken kann. Daneben können auch folgende Umstände dazu führen, dass häufigere bedingte Prüfpunkte die für den Neustart einer Datenbank benötigte Zeit nicht verkürzen:

- Es werden sehr lange Transaktionen mit wenigen COMMIT-Punkten ausgeführt.
- Der Pufferpool ist sehr groß, und die Seiten mit den festgeschriebenen Transaktionen werden nicht sehr oft auf die Platte zurückgeschrieben. (Beachten Sie, dass durch die Verwendung asynchroner Seitenlöschfunktionen solche Situationen vermieden werden können. Siehe „Anzahl asynchroner Seitenlöschfunktionen (num_iocleaners)" auf Seite 454.)

In beiden Fällen ändern sich die Protokollsteuerdaten im Hauptspeicher nur selten, und es ist nur dann sinnvoll, die Protokollsteuerdaten auf die Festplatte zu schreiben, wenn sie sich geändert haben.

Protokollspeicherung für Wiederherstellung (logretain)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] Nein [Recovery; Capture; Nein]

Zugehörige Parameter

- „Benutzerausgang für Protokollierung aktivieren (userexit)" auf Seite 491
- „Status der Protokollspeicherung für Wiederherstellung (log_retain_status)" auf Seite 512
- „Sicherung anstehend (backup_pending" auf Seite 511

Es gibt folgende Werte:

- "No" gibt an, dass die Protokolle nicht gespeichert werden.
- "Recovery" gibt an, dass die Protokolle gespeichert werden und zur Vorwärtswiederherstellung verwendet werden können. Wenn Sie Datenreplikation einsetzen, kann das Capture-Programm zudem die in den Protokollen aufgezeichneten Aktualisierungen in die Änderungstabelle schreiben.
- "Capture" gibt an, dass die Protokolle nur gespeichert werden, damit das Capture-Programm die Aktualisierungen in die Änderungstabelle schreiben kann. Diese Protokolle werden eventuell für aktualisierende Wiederherstellung verwendet, sofern sie nicht nach ihrer Verwendung durch das Capture-Programm bei der Datenreplikation entfernt wurden.

Ist *logretain* auf "Recovery" oder *userexit* auf "Yes" gesetzt, werden die aktiven Protokolldateien gespeichert und als Online-Archivprotokolldateien in einer aktualisierenden Wiederherstellung eingesetzt. Dies wird Protokollierung mit Protokollspeicherung genannt.

Nach dem Setzen von *logretain* auf "Recovery" und/oder von *userexit* auf "Yes" müssen Sie eine Gesamtsicherung der Datenbank vornehmen. Dieser Status wird durch den Markierungsparameter *backup_pending* angezeigt.

Ist *logretain* auf "No" und *userexit* auf "No" gesetzt, ist die aktualisierende Wiederherstellung für die Datenbank nicht verfügbar.

Wenn *logretain* auf "Capture" gesetzt ist, ruft das Capture-Programm im Rahmen der Datenreplikation den Befehl PRUNE LOGFILE zum Löschen der

Protokolldateien auf, sobald das Capture-Programm die Benutzung der Protokolldateien beendet hat. Setzen Sie *logretain* nicht auf "Capture", wenn Sie die Datenbank aktualisierend wiederherstellen wollen.

Ist *logretain* auf "No" und *userexit* auf "No" gesetzt, werden die Protokolle nicht gespeichert. In diesem Fall löscht der Datenbankmanager alle Protokolldateien im Verzeichnis *logpath* (einschließlich der Online-Archivprotokolldateien), ordnet er neue aktive Protokolldateien zu und aktiviert er erneut Umlaufprotokollierung.

Benutzerausgang für Protokollierung aktivieren (userexit)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] No [Yes; No]

Zugehörige Parameter

- „Protokollspeicherung für Wiederherstellung (logretain)" auf Seite 490
- „Status des Benutzerausgangs für Protokollierung (user_exit_status)" auf Seite 512
- „Sicherung anstehend (backup_pending" auf Seite 511

Wenn dieser Parameter aktiviert ist, wird unabhängig von der Einstellung des Parameters *logretain* eine Protokollierung mit Protokollspeicherung ausgeführt. Mit diesem Parameter wird außerdem angegeben, dass ein Benutzerausgangsprogramm verwendet werden soll, um Protokolldateien zu archivieren und wieder abzurufen. Protokolldateien werden archiviert, wenn sie vom Datenbankmanager geschlossen werden. Die Protokolldateien werden wieder abgerufen, wenn das Dienstprogramm ROLLFORWARD sie zur Wiederherstellung einer Datenbank benötigt.

Nach der Aktivierung des Parameters *logretain* und/oder *userexit* muss eine Gesamtsicherung der Datenbank ausgeführt werden. Dieser Status wird durch den Markierungsparameter *backup_pending* angezeigt.

Wenn beide Parameter inaktiviert werden, ist die aktualisierende Wiederherstellung der Datenbank nicht mehr möglich, da keine Protokolldateien mehr gespeichert werden. In diesem Fall löscht der Datenbankmanager alle Protokolldateien im Verzeichnis des Parameters *logpath* (einschließlich der Online-Archivprotokolldateien), ordnet neue aktive Protokolldateien zu und reaktiviert die Umlaufprotokollierung.

Weitere Informationen zum Benutzer-Exit-Programm finden Sie im Abschnitt zum Benutzer-Exit für Datenbankwiederherstellung im Handbuch *Systemverwaltung: Implementierung*.

Wiederherstellung

Die folgenden Parameter wirken sich auf die verschiedenen Aspekte der Datenbankwiederherstellung aus:

- „Automatischer Neustart aktiviert (autorestart)"
- „Zeitpunkt für Indexneuerstellung (indexrec)" auf Seite 493
- „Standardanzahl von Sitzungen für Wiederherstellung (dft_loadrec_ses)" auf Seite 494
- „Anzahl der Datenbanksicherungen (num_db_backups)" auf Seite 495
- „Aufbewahrungszeitraum für Wiederherstellungsprotokoll (rec_his_retentn)" auf Seite 496
- „Geänderte Seiten protokollieren (trackmod)" auf Seite 497

Siehe auch „Wiederherstellung der verteilten Arbeitseinheit" auf Seite 499.

Die folgenden Parameter werden bei Verwendung von Tivoli Storage Manager (TSM) verwendet:

- „Tivoli Storage Manager-Verwaltungsklasse (tsm_mgmtclass)" auf Seite 497
- „Tivoli Storage Manager-Kennwort (tsm_password)" auf Seite 497
- „Tivoli Storage Manager-Knotenname (tsm_nodename)" auf Seite 498
- „Tivoli Storage Manager-Eignername (tsm_owner)" auf Seite 498

Automatischer Neustart aktiviert (autorestart)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	On [On; Off]

Wenn dieser Parameter auf ON gesetzt wird, ruft der Datenbankmanager bei Bedarf automatisch das Dienstprogramm zum Neustarten der Datenbank auf, wenn eine Anwendung die Verbindung zur Datenbank herstellt. Das Programm zum Neustarten der Datenbank wird verwendet, um eine *Wiederherstellung nach Systemabsturz* auszuführen. Es wird ausgeführt, wenn die Datenbank abnormal beendet wurde, während Anwendungen mit ihr verbunden waren. Eine abnormale Beendigung der Datenbank könnte z. B. durch einen Stromausfall oder einen Fehler der Systemsoftware verursacht werden. Bei der Wiederherstellung werden festgeschriebene Transaktionen, die sich im Pufferpool befanden, jedoch zum Zeitpunkt des Fehlers nicht auf die Festplatte

geschrieben waren, in der Datenbank nachvollzogen. Außerdem werden alle nicht festgeschriebenen Transaktionen, die auf Platte geschrieben wurden, wieder zurückgesetzt.

Wenn der Parameter *autorestart* nicht aktiviert ist, empfängt eine Anwendung den Fehler SQL1015N, wenn sie versucht, die Verbindung zu einer Datenbank herzustellen, für die eine Wiederherstellung nach einem Systemabsturz ausgeführt werden muss (für die die Datenbank neu gestartet werden muss). In diesem Fall kann die Anwendung das Dienstprogramm zum Neustarten der Datenbank aufrufen, oder Sie können die Datenbank neu starten, indem Sie die Funktion zum Neustarten im Wiederherstellungsprogramm auswählen.

Zeitpunkt für Indexneuerstellung (indexrec)

Konfigurationsart Datenbank und Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX Datenbankmanager
restart [restart; access]

OS/2 und Windows NT Datenbankmanager
access [restart; access]

Datenbank system (Systemwert verwenden) [system; restart; access]

Zugehörige Parameter „Automatischer Neustart aktiviert (autorestart)" auf Seite 492

Dieser Parameter gibt an, wann der Datenbankmanager versucht, ungültige Indizes neu zu erstellen. Für diesen Parameter sind drei Einstellungen möglich:

SYSTEM *Systemeinstellung verwenden* veranlasst, dass ungültige Indizes zu dem Zeitpunkt neu erstellt werden, der in der Konfigurationsdatei des Datenbankmanagers angegeben ist. (Anmerkung: Diese Einstellung ist nur für Datenbankkonfigurationen gültig.)

ACCESS *Bei Indexzugriff* veranlasst, dass ungültige Indizes neu erstellt werden, wenn zum ersten Mal auf den Index zugegriffen wird.

RESTART *Bei Datenbankneustart* veranlasst, dass ungültige Indizes neu erstellt werden, wenn der Befehl RESTART DATABASE explizit oder implizit abgesetzt wird. Beachten Sie, dass der Befehl RESTART DATABASE implizit abgesetzt wird, wenn der Parameter *autorestart* aktiviert ist.

Informationen zu numerischen Äquivalenten und API-Konstanten für diese Werte finden Sie im Handbuch *Administrative API Reference*.

Indizes können ungültig werden, wenn nicht behebbare Plattenfehler auftreten. Wenn dabei die Daten selbst beschädigt werden, können die Daten verloren gehen. Wird jedoch ein Index beschädigt, kann der Index durch Neuerstellung wiederhergestellt werden. Wird ein Index neu erstellt, während Benutzer mit der Datenbank verbunden sind, können zwei Probleme auftreten:

- Während der Erstellung der Indexdatei kann es zu einer unerwarteten Verschlechterung der Antwortzeit kommen. Benutzer, die auf die Tabelle zugreifen und diesen bestimmten Index verwenden, müssen auf die Neuerstellung des Index warten.
- Nach der Indexneuerstellung können unerwartete, aktive Sperren auftreten, besonders dann, wenn die Benutzertransaktion, die die Indexneuerstellung ausgelöst hat, keine COMMIT- oder ROLLBACK-Operation ausgeführt hat.

Empfehlung: Die beste Option für diesen Parameter auf einem Server mit vielen Benutzern, wenn die Zeit für den Neustart keine kritische Rolle spielt, ist die Indexneuerstellung beim Neustart der Datenbank (DATABASE RESTART) als Teil des Vorgangs, mit dem die Datenbank nach einem Systemabsturz wiederhergestellt und online verfügbar gemacht wird.

Bei der Einstellung „ACCESS" für diesen Parameter verschlechtert sich während der Indexneuerstellung die Leistung des Datenbankmanagers. Jeder Benutzer, der auf den Index oder die Tabelle zugreift, der bzw. die gerade neu erstellt wird, muss zunächst auf die Beendigung der Indexneuerstellung warten.

Wenn dieser Parameter auf „RESTART" gesetzt wird, dauert der Neustart der Datenbank wegen der Indexneuerstellung länger, jedoch wird die normale Verarbeitung nicht mehr beeinträchtigt, sobald die Datenbank wieder online verfügbar ist.

Standardanzahl von Sitzungen für Wiederherstellung (dft_loadrec_ses)

Konfigurationsart Datenbank

Parameterart	Konfigurierbar
Standardwert [Bereich]	1 [1 – 30 000]
Maßeinheit	Zähler

Mit diesem Parameter wird die Standardanzahl von Sitzungen angegeben, die während des Wiederabrufens einer Tabellenladekopie verwendet werden. Der Wert sollte auf eine optimale Anzahl von E/A-Sitzungen gesetzt werden, die zum Wiederabrufen einer Ladekopie verwendet werden sollen. Das Wiederabrufen einer Ladekopie ist eine ähnliche Operation wie die Wiederherstellung. Sie können diesen Parameter durch Einträge überschreiben, die sich in der Datei mit den Angaben zur Speicherposition der exportierten Daten befinden. Diese Datei wird von der Umgebungsvariablen DB2LOADREC angegeben.

Die Standardanzahl der Puffer, die für den Abruf der Ladekopien verwendet werden, ist um zwei größer als der Wert dieses Parameters. Die Anzahl der Puffer kann ebenfalls in der Datei mit den Angaben zur Speicherposition der exportierten Daten überschrieben werden.

Dieser Parameter ist nur gültig, wenn die aktualisierende Wiederherstellung aktiviert ist.

Weitere Informationen zur Wiederherstellung finden Sie im Handbuch *Versetzen von Daten Dienstprogramme und Referenz*.

Anzahl der Datenbanksicherungen (num_db_backups)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	12 [1 — 32 768]
Zugehörige Parameter	„Aufbewahrungszeitraum für Wiederherstellungsprotokoll (rec_his_retentn)" auf Seite 496

Dieser Parameter gibt die Anzahl der Datenbanksicherungen an, die für eine Datenbank beibehalten werden sollen. Wird die angegebene Anzahl Sicherungen erreicht, werden alte Sicherungen in der Datei des Wiederherstellungsprotokolls als abgelaufen markiert. Die Einträge in der Datei des Wiederherstellungsprotokolls für Tabellenbereichssicherungen und Ladekopiesicherungen, die mit der abgelaufenen Datenbanksicherung verbunden sind, werden ebenfalls als abgelaufen markiert. Wird eine Sicherung als abgelaufen markiert, können die physischen Sicherungen von ihrem Speicherort (z. B. Platte, Band, ADSTAR Distributed Storage Manager) gelöscht werden. Die nächste Datenbanksicherung entfernt die abgelaufenen Einträge aus der Datei des Wiederherstellungsprotokolls.

Wird eine Datenbanksicherung in der Protokolldatei als abgelaufen markiert, werden entsprechende über einen DB2 Data Links Manager verbundene Dateisicherungen aus dem Archivierungs-Server gelöscht.

Der Konfigurationsparameter *rec_his_retentn* muss auf einen Wert gesetzt werden, der mit dem Wert von *num_db_backups* kompatibel ist. Wenn *num_db_backup* z. B. auf einen hohen Wert gesetzt ist, muss der Wert für *rec_his_retentn* hoch genug sein, um diese Anzahl der Sicherungen unterstützen zu können.

Aufbewahrungszeitraum für Wiederherstellungsprotokoll (rec_his_retentn)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	366 [-1; 0 — 30 000]
Maßeinheit	Tage
Zugehörige Parameter	„Anzahl der Datenbanksicherungen (num_db_backups)" auf Seite 495

Mit diesem Parameter wird angegeben, wie viele Tage die Protokolldaten zu Sicherungen aufbewahrt werden sollen. Wenn die Datei des Wiederherstellungsprotokolls nicht benötigt wird, um Sicherungen, Wiederherstellungen und Ladevorgänge festzuhalten, kann dieser Parameter auf einen kleineren Wert gesetzt werden.

Wenn für diesen Parameter der Wert -1 angegeben wird, kann die Datei des Wiederherstellungsprotokolls nur explizit mit Hilfe der verfügbaren Befehle oder APIs entfernt werden. Wenn der Wert nicht -1 ist, wird die Datei des Wiederherstellungsprotokolls nach jeder vollständigen Sicherung der Datenbank entfernt.

Der Wert dieses Parameters überschreibt den Wert des Parameters *num_db_backups*, *rec_his_retentn* und *num_db_backups* müssen jedoch zusammenpassen. Wenn der Wert für *num_db_backups* groß ist, muss der Wert für *rec_his_retentn* groß genug sein, um die angegebene Anzahl von Sicherungen unterstützen zu können.

Unabhängig davon, wie kurz der Aufbewahrungszeitraum ist, werden die aktuellste Datenbanksicherung und die zugehörige Wiederherstellungsgruppe immer zurückbehalten, sofern Sie nicht das Dienstprogramm PRUNE mit der Angabe WITH FORCE OPTION verwenden. Weitere Informationen zu diesem Dienstprogramm finden Sie in *Command Reference*.

Geänderte Seiten protokollieren (trackmod)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] No [Yes, No]

Wenn dieser Parameter auf "Yes" gesetzt wird, verfolgt der Datenbankmanager Datenbankänderungen, damit das Sicherungsprogramm feststellen kann, welche Untergruppen der Datenbankseiten bei einer Teilsicherung zu berücksichtigen und eventuell in das Sicherungsabbild aufzunehmen sind. Wenn Sie diesen Parameter auf "Yes" setzen, müssen Sie zunächst eine vollständige Sicherung der Datenbank durchführen, damit Sie über eine Ausgangsbasis für Teilsicherungen verfügen. Wenn dieser Parameter aktiviert ist und ein Tabellenbereich erstellt wird, müssen Sie außerdem eine Sicherung erstellen, die den betreffenden Tabellenbereich umfasst. Dabei kann es sich entweder um eine Datenbanksicherung oder eine Tabellenbereichssicherung handeln. Nach erfolgter Sicherung dürfen Teilsicherungen diesen Tabellenbereich umfassen.

Tivoli Storage Manager-Verwaltungsklasse (tsm_mgmtclass)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [beliebige Zeichenfolge]

Die Tivoli Storage Manager-Verwaltungsklasse (TSM) gibt an, wie der TSM-Server die Sicherungsversionen der Objekte verwalten soll, die gesichert werden.

Standardmäßig ist keine TSM-Verwaltungsklasse festgelegt.

Die Verwaltungsklasse wird vom Tivoli Storage Manager-Administrator zugeordnet. Sobald die Zuordnung erfolgt ist, müssen Sie diesen Parameter auf den Namen der Verwaltungsklasse setzen. Bei der Durchführung einer TSM-Sicherung verwendet der Datenbankmanager diesen Parameter, um die Verwaltungsklasse an TSM zu übergeben.

Weitere Informationen zu Tivoli Storage Manager finden Sie im entsprechenden Abschnitt im Handbuch *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*.

Tivoli Storage Manager-Kennwort (tsm_password)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [beliebige Zeichenfolge]

Mit diesem Parameter kann die Standardeinstellung für das Kennwort, das dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden. Dieses Kennwort ist erforderlich, um eine Datenbank wiederherstellen zu können, die in TSM von einem anderen Knoten aus gesichert wurde.

Anmerkung: Wenn der Parameter *tsm_nodename* bei einer Sicherung mit DB2 (z. B. mit dem Befehl BACKUP DATABASE) überschrieben wurde, muss möglicherweise auch der Parameter *tsm_password* festgelegt werden.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus die Sicherung ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter *tsm_nodename* bei einer Sicherung mit DB2 überschrieben wird.

Weitere Informationen zu Tivoli Storage Manager finden Sie im entsprechenden Abschnitt im Handbuch *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*.

Tivoli Storage Manager-Knotenname (tsm_nodename)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [beliebige Zeichenfolge]

Mit diesem Parameter kann die Standardeinstellung für den Knotennamen, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden. Der Knotenname ist erforderlich, um eine Datenbank wiederherstellen zu können, die von einem anderen Knoten aus in TSM gesichert wurde.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus die Sicherung ausgeführt wurde. Es ist möglich, dass der Wert für den Parameter *tsm_nodename* bei einer Sicherung mit DB2 (z. B. mit dem Befehl BACKUP DATABASE) überschrieben wird.

Weitere Informationen zu Tivoli Storage Manager finden Sie im entsprechenden Abschnitt im Handbuch *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*.

Tivoli Storage Manager-Eignername (tsm_owner)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [beliebige Zeichenfolge]

Mit diesem Parameter kann die Standardeinstellung für den Eigner, der dem Produkt Tivoli Storage Manager (TSM) zugeordnet ist, überschrieben werden. Der Eignername ist erforderlich, um eine Datenbank wiederherstellen zu können, die von einem anderen Knoten aus in ADSM gesichert wurde. Es ist möglich, dass der Wert für den Parameter *tsm_owner* bei einer Sicherung mit DB2 (z. B. mit dem Befehl BACKUP DATABASE) überschrieben wird.

Anmerkung: Beim Eignernamen muss die Groß-/Kleinschreibung beachtet werden.

Standardmäßig können Sie eine Datenbank in TSM nur auf dem Knoten wiederherstellen, von dem aus die Sicherung ausgeführt wurde.

Weitere Informationen zu Tivoli Storage Manager finden Sie im entsprechenden Abschnitt im Handbuch *Datenwiederherstellung und hohe Verfügbarkeit Handbuch und Referenz*.

Wiederherstellung der verteilten Arbeitseinheit

Die folgenden Parameter wirken sich auf die Wiederherstellung von DUOW-Transaktionen (Distributed Unit of Work - verteilte Arbeitseinheit) aus:

- „Name für Transaktionsmanagerdatenbank (tm_database)"
- „Intervall für Transaktionsresynchronisation (resync_interval)" auf Seite 500
- „Protokolldateipfad für SPM (spm_log_path)" auf Seite 501
- „Name des Synchronisationspunktmanagers (spm_name)" auf Seite 502
- „Protokolldateigröße für SPM (spm_log_file_sz)" auf Seite 502
- „SPM-Maximum für Resynchronisationsagenten (spm_max_resync)" auf Seite 503

Name für Transaktionsmanagerdatenbank (tm_database)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 1ST_CONN [beliebiger gültiger Datenbankname]

Mit diesem Parameter wird der Name der Transaktionsmanagerdatenbank (TMD) für jedes DB2-Exemplar angegeben. Eine TMD kann eine der folgenden Datenbanken sein:

- Eine lokale Datenbank von DB2 Universal Database
- Eine ferne Datenbank von DB2 Universal Database, die sich nicht auf einem Host oder System IBM AS/400 befindet
- Eine Datenbank von DB2 für OS/390 Version 5, auf die über TCP/IP und den Synchronisationspunktmanager (SPM) zugegriffen wird, wird nicht verwendet.

Es handelt sich dabei um eine Datenbank, die zu Protokoll- und Koordinierungsfunktionen verwendet wird und zur Wiederherstellung unbestätigter Transaktionen dient.

Dieser Parameter kann auf den Wert **1ST_CONN** gesetzt werden, wodurch festgelegt wird, dass die Transaktionsmanagerdatenbank die erste Datenbank ist, zu der ein Benutzer eine Verbindung herstellt.

Weitere Informationen zur verteilten Arbeitseinheit finden Sie im entsprechenden Abschnitt im Handbuch *Systemverwaltung: Konzept*.

Empfehlung: Zur Vereinfachung der Verwaltung und des Betriebs können Sie einige exemplarübergreifende Datenbanken erstellen und diese Datenbanken ausschließlich als Transaktionsmanagerdatenbanken verwenden.

Intervall für Transaktionsresynchronisation (resync_interval)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 180 [1 – 60 000]

Maßeinheit Sekunden

Mit diesem Parameter wird das Zeitintervall in Sekunden angegeben, während dessen ein Transaktionsmanager (TM), ein Ressourcenmanager (RM) oder ein Synchronisationspunktmanager (SPM) versuchen soll, jede ausstehende unbestätigte Transaktion, die in dem TM, RM oder SPM gefunden wird, wiederherzustellen. Dieser Parameter ist gültig, wenn Sie Transaktionen in einer Umgebung mit verteilten Arbeitseinheiten (DUOW) ausführen.

Weitere Informationen zur verteilten Arbeitseinheit finden Sie im Abschnitt zu verteilten Datenbanken im Handbuch *Systemverwaltung: Konzept*.

Empfehlung: Wenn in Ihrer Umgebung unbestätigte Transaktionen keine Störungen anderer Transaktionen für Ihre Datenbank verursachen, können Sie den Wert dieses Parameters auch erhöhen. Wenn Sie ein DB2 Connect-Gateway zum Zugriff auf DRDA2-Anwendungsserver verwenden, sollten Sie die Auswirkungen bedenken, die unbestätigte Transaktionen auf Anwendungsservern haben können, auch wenn lokal keine Störungen beim Datenzugriff zu erwarten sind. Gibt es keine unbestätigten Transaktionen, sind die Auswirkungen auf die Leistung minimal.

Protokolldateipfad für SPM (spm_log_path)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert sqllib/spmlog [ein beliebiger gültiger Pfad oder eine beliebige gültige Einheit]

Dieser Parameter gibt an, in welches Verzeichnis die SPM-Protokolle geschrieben werden. Standardmäßig werden die Protokolle in das Verzeichnis `sqllib/spmlog` geschrieben; dies kann in Umgebungen mit hohem Transaktionsaufkommen zu E/A-Engpässen führen. Verwenden Sie diesen Parameter, damit SPM-Protokolldateien auf eine Platte mit kürzeren Zugriffszeiten als das aktuelle Verzeichnis `sqllib/spmlog` geschrieben werden. Dadurch wird der gemeinsame Zugriff der SPM-Agenten optimiert.

Weitere Informationen zum Synchronisationspunktmanager finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

Weitere Informationen zur Wiederherstellung unbestätigter DRDA-Transaktionen finden Sie im entsprechenden Abschnitt des Handbuchs *Systemverwaltung: Konzept*.

Name des Synchronisationspunktmanagers (spm_name)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert Vom TCP/IP-Host-Namen abgeleitet

Mit diesem Parameter wird der Name des Exemplars des Synchronisationspunktmanagers (SPM) gegenüber dem Datenbankmanager identifiziert.

Weitere Informationen zum Synchronisationspunktmanager finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

Weitere Informationen zur Wiederherstellung unbestätigter DRDA-Transaktionen finden Sie im entsprechenden Abschnitt des Handbuchs *Systemverwaltung: Konzept*.

Protokolldateigröße für SPM (spm_log_file_sz)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 256 [4 — 1 000]

Maßeinheit Seiten

Mit diesem Parameter wird die Größe der Protokolldatei für den Synchronisationspunktmanager (SPM) in 4-KB-Seiten angegeben. Die Protokolldatei befindet sich im Unterverzeichnis `spmlog` im Verzeichnis `sqllib` und wird erstellt, wenn der Synchronisationspunktmanager zum ersten Mal gestartet wird.

Weitere Informationen zum Synchronisationspunktmanager finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

Weitere Informationen zur Wiederherstellung unbestätigter DRDA-Transaktionen finden Sie im entsprechenden Abschnitt des Handbuchs *Systemverwaltung: Konzept*.

Empfehlung: Die Protokolldatei des Synchronisationspunktmanagers sollte einerseits groß genug sein, um die Leistung zu gewährleisten, andererseits aber auch klein genug, um die Verschwendung von Speicherbereich zu vermeiden. Die erforderliche Größe hängt von der Anzahl der Transaktionen ab, die geschützte Dialoge verwenden, und von der Häufigkeit, mit der COMMIT- oder ROLLBACK-Operationen ausgeführt werden.

Gehen Sie wie folgt vor, um die Größe der SPM-Protokolldatei zu ändern:
1. Verwenden Sie den Befehl LIST DRDA INDOUBT TRANSACTIONS, um festzustellen, ob es unbestätigte Transaktionen gibt.
2. Wenn es keine unbestätigten Transaktionen gibt, stoppen Sie den Datenbankmanager.
3. Aktualisieren Sie die Konfiguration des Datenbankmanagers mit dem neuen Wert für die Größe der SPM-Protokolldatei.
4. Wechseln Sie in das Verzeichnis `$HOME/sqllib`, und löschen Sie die aktuelle SPM-Protokolldatei mit dem Befehl `rm -fr spmlog`. (Anmerkung: Dies ist der AIX-Befehl. Auf anderen Systemen sind wahrscheinlich andere Löschbefehle erforderlich.)
5. Starten Sie den Datenbankmanager. Beim Start des Datenbankmanagers wird eine neue SPM-Protokolldatei in der angegebenen Größe erstellt.

SPM-Maximum für Resynchronisationsagenten (spm_max_resync)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 20 [10 — 256]

Mit diesem Parameter wird die Anzahl der Agenten angegeben, die gleichzeitig Resynchronisationsoperationen ausführen können.

Weitere Informationen zur Wiederherstellung unbestätigter DRDA-Transaktionen finden Sie im entsprechenden Abschnitt des Handbuchs *Systemverwaltung: Konzept*.

Weitere Informationen zum Synchronisationspunktmanager finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

Datenbankverwaltung

Es ist eine Reihe von Parametern verfügbar, die Informationen zur Datenbank bereitstellen oder die Verwaltung der Datenbank beeinflussen. Diese werden in folgende Kategorien unterteilt:

- „Query Enabler"
- „Attribute" auf Seite 505
- „DB2 Data Links Manager" auf Seite 508
- „Status" auf Seite 511
- „Compilereinstellungen" auf Seite 513.

Query Enabler

Die folgenden Parameter stellen Informationen zur Steuerung von Query Enabler bereit:

- „Dynamische SQL-Abfrageverwaltung (dyn_query_mgmt)"

Dynamische SQL-Abfrageverwaltung (dyn_query_mgmt)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] 0 (DISABLE) [1(ENABLE), 0 (DISABLE)]

Dieser Parameter ist dort relevant, wo DB2 Query Patroller installiert ist. Wird der Datenbankkonfigurationsparameter *dyn_query_mgmt* auf „ENABLE" gesetzt, und übersteigt der Aufwand der dynamischen Abfrage den Wert des Trap-Schwellenwerts (*trap_threshold*) für den Benutzer bzw. die Gruppe (wie in der Benutzerprofiltabelle von DB2 Query Patroller angegeben), wird diese Abfrage von DB2 Query Patroller abgefangen. Der Trap-Schwellenwert (*trap_threshold*) ist ein aufwandorientierter Auslöser für das Abfangen von Abfragen, der vom Benutzer in DB2 Query Patroller eingerichtet wird. Wird eine dynamische Abfrage abgefangen, wird für den Benutzer ein Dialog aufgerufen, in dem er Laufzeitparameter angeben kann.

Es werden keine Abfragen abgefangen, wenn *dyn_query_mgmt* auf „DISABLE" gesetzt wird.

Attribute

Die folgenden Parameter stellen allgemeine Informationen zur Datenbank bereit:

- „Release-Stand der Datenbankkonfiguration (release)"
- „Release-Stand der Datenbank (database_level)"
- „Datenbankgebiet (territory)" auf Seite 506
- „Landescode der Datenbank (country)" auf Seite 506
- „Codierter Zeichensatz für die Datenbank (codeset)" auf Seite 506
- „Codepage für die Datenbank (codepage)" auf Seite 507
- „Informationen zur Sortierfolge (collate_info)" auf Seite 507
- „Kopierschutz aktiviert (copyprotect)" auf Seite 508

Mit Ausnahme des Parameters *copyprotect* dienen diese Parameter lediglich der Information.

Release-Stand der Datenbankkonfiguration (release)

Konfigurationsart Datenbankmanager, Datenbank

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Informativ

Zugehörige Parameter „Release-Stand der Datenbank (database_level)"

Dieser Parameter gibt den Release-Stand der Konfigurationsdatei an.

Release-Stand der Datenbank (database_level)

Konfigurationsart Datenbank

Parameterart Informativ

Zugehörige Parameter „Release-Stand der Datenbankkonfiguration (release)"

Dieser Parameter gibt den Release-Stand des Datenbankmanagers an, der die Datenbank verwenden kann. Im Fall einer nicht abgeschlossenen oder fehlgeschlagenen Migration (Umstellung) spiegelt dieser Parameter den Release-Stand der nicht umgestellten Datenbank wider und kann daher vom Parameter *release* (Release-Stand der Datenbankkonfiguration) abweichen. Ansonsten ist der Wert des Parameters *database_level* mit dem Wert des Parameters *release* identisch.

Datenbankgebiet (territory)

Konfigurationsart Datenbank

Parameterart Informativ

Zugehörige Parameter „Landescode der Datenbank (country)"

Dieser Parameter zeigt das Gebiet an, mit dem die Datenbank erstellt wurde. Der Wert für das Gebiet wird vom Datenbankmanager zur Bestimmung der Werte für den Parameter *country* verwendet. Weitere Informationen zur Verwendung des Parameters *territory* durch den Datenbankmanager finden Sie im Anhang zur Unterstützung von Landessprachen im Handbuch *Systemverwaltung: Konzept*.

Landescode der Datenbank (country)

Konfigurationsart Datenbank

Parameterart Informativ

Zugehörige Parameter „Datenbankgebiet (territory)"

Dieser Parameter zeigt den Landescode an, der bei der Erstellung der Datenbank verwendet wurde. Der Wert des Parameters *country* wird mit Hilfe des Werts für den Parameter *territory* abgeleitet. Weitere Informationen zur Verwendung des Landescodes durch den Datenbankmanager finden Sie im Anhang zur Unterstützung von Landessprachen im Handbuch *Systemverwaltung: Konzept*.

Codierter Zeichensatz für die Datenbank (codeset)

Konfigurationsart Datenbank

Parameterart Informativ

Zugehörige Parameter „Codepage für die Datenbank (codepage)" auf Seite 507

Dieser Parameter zeigt den codierten Zeichensatz an, der zur Erstellung der Datenbank verwendet wurde. Der codierte Zeichensatz wird vom Datenbankmanager zur Bestimmung des Parameters *codepage* verwendet. Weitere Infor-

mationen zur Verwendung des codierten Zeichensatzes durch den Datenbankmanager finden Sie im Anhang zur Unterstützung von Landessprachen im Handbuch *Systemverwaltung: Konzept*.

Codepage für die Datenbank (codepage)

Konfigurationsart Datenbank

Parameterart Informativ

Zugehörige Parameter „Codierter Zeichensatz für die Datenbank (codeset)" auf Seite 506

Dieser Parameter zeigt die Codepage an, die zur Erstellung der Datenbank verwendet wurde. Der Wert des Parameters *codepage* wird mit Hilfe des Werts für den Parameter *codeset* abgeleitet. Weitere Informationen zur Verwendung der Codepage durch den Datenbankmanager finden Sie im Anhang zur Unterstützung von Landessprachen im Handbuch *Systemverwaltung: Konzept*.

Informationen zur Sortierfolge (collate_info)

Dieser Parameter kann nur mit der API GET DATABASE CONFIGURATION angezeigt werden. Er kann **nicht** mit Hilfe des Befehlszeilenprozessors oder der Steuerzentrale angezeigt werden.

Konfigurationsart Datenbank

Parameterart Informativ

Dieser Parameter enthält 260 Byte mit Informationen zur Sortierfolge der Datenbank. Die ersten 256 Byte geben die Sortierfolge der Datenbank an, wobei Byte „n" die Sortierwertigkeit des Codepunkts enthält, dessen zugrundeliegende dezimale Darstellung „n" in der Codepage der Datenbank ist.

Die letzten 4 Byte enthalten interne Informationen zur Art der Sortierfolge. Sie können diese Byte wie einen Wert des Typs INTEGER (ganze Zahl) für die Plattform der Datenbank behandeln. Es gibt drei Werte:

- **0** – Die Sortierfolge enthält nicht eindeutige Wertigkeiten.
- **1** – Die Sortierfolge enthält ausschließlich eindeutige Wertigkeiten.
- **2** – Die Sortierfolge ist die Identitätssortierfolge, anhand der Zeichenfolgen Byte für Byte verglichen werden.

Wenn Sie diese internen Informationen zur Art der Sortierfolge verwenden, müssen Sie eine Bytefolgeumkehrung in Betracht ziehen, wenn Informationen zu einer Datenbank auf einer anderen Plattform abgerufen werden.

Sie können die Sortierfolge bei der Erstellung der Datenbank angeben.

Kopierschutz aktiviert (copyprotect)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	No [Yes; No]

Mit diesem Parameter wird das Kopierschutzattribut aktiviert, das standardmäßig inaktiviert ist. In Versionen vor Version 2 des Datenbankmanagers wurde das Kopierschutzattribut standardmäßig aktiviert.

Dieser Parameter gilt nicht für auf UNIX basierende Umgebungen.

Auf die Dienstprogramme zur Sicherung und Wiederherstellung von Datenbanken hat der Parameter *copyprotect* keine Wirkung. Eine kopiergeschützte Datenbank kann gesichert und auf einer anderen Workstation wiederhergestellt werden. Nachdem sie katalogisiert wurde, kann dort auf sie zugegriffen werden.

Achtung: Entfernen Sie den Kopierschutz von allen Datenbanken, bevor Sie entweder den Datenbankmanager oder das Betriebssystem erneut installieren. Wenn Sie den Kopierschutz nicht entfernen, erhalten Sie eine Fehlernachricht, wenn Sie auf die Datenbank zugreifen möchten. Nach der erneuten Installation können Sie den Kopierschutz wieder aktivieren.

DB2 Data Links Manager

Die folgenden Parameter beziehen sich auf den DB2 Data Links Manager:

- „Ablaufintervall für Data Link-Zugriffs-Token (dl_expint)"
- „Anzahl Data Link-Kopien (dl_num_copies)" auf Seite 509
- „Data Links-Zeit nach DROP (dl_time_drop)" auf Seite 509
- „Data Links-Token-Algorithmus (dl_token)" auf Seite 510
- „Data Links-Token in Großbuchstaben (dl_upper)" auf Seite 510
- „Unterstützung der Aktivierung von Data Links (datalinks)" auf Seite 510

Ablaufintervall für Data Link-Zugriffs-Token (dl_expint)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	60 [-1, 1 — 31 536 000]
Maßeinheit	Sekunden

Dieser Parameter gibt das Zeitintervall (in Sekunden) an, in dem das generierte Dateizugriffssteuerungs-Token gültig ist. Der Gültigkeitszeitraum beginnt mit dem Zeitpunkt der Generierung des Tokens. Der Data Links Filesystem Filter prüft die Gültigkeit des Tokens gegen dieses Gültigkeitsintervall.

Informationen zu Dateizugriffssteuerungs-Token finden Sie im Handbuch *DB2 Data Links Manager Quick Beginnings*.

Der Standardwert für diesen Parameter ist sechzig (60) Sekunden. Wenn dieser Parameter auf ″-1″ gesetzt wird, läuft die Gültigkeit des Zugriffssteuerungs-Tokens ab. Dies kann umgangen werden, indem dieser Parameter auf seinen Maximalwert 31536000 (Sekunden) gesetzt wird. Dies entspricht einer Verfallszeit von einem Jahr, ein Wert, der wahrscheinlich für alle Anwendungen geeignet ist.

Dieser Parameter gilt für DATALINK-Spalten, in denen „READ PERMISSION DB″ angegeben ist.

Anzahl Data Link-Kopien (dl_num_copies)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] 0 [0 – 15]

Dieser Parameter gibt an, wie viele zusätzliche Kopien im Archivierungsserver (z. B. in einem TSM-Server) von einer Datei erstellt werden, wenn die Datei mit der Datenbank verbunden ist.

Der Standardwert für diesen Konfigurationsparameter ist Null (0).

Dieser Parameter gilt für DATALINK-Spalten, in denen „Recovery=Yes″ angegeben ist.

Data Links-Zeit nach DROP (dl_time_drop)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] 1 [0 — 365]

Maßeinheit Tage

Dieser Parameter gibt den Zeitraum (in Tagen) an, in dem Dateien auf einem Archivierungsserver (z. B. auf einem TSM-Server) aufbewahrt werden, nachdem DROP DATABASE abgesetzt wurde.

Der Standardwert für diesen Konfigurationsparameter ist ein (1) Tag. Der Wert Null (0) bedeutet, dass die Dateien sofort nach dem Absetzen eines DROP-Befehls vom Archivierungsserver gelöscht werden. (Die tatsächliche Datei wird nur gelöscht, wenn der Parameter ON UNLINK DELETE für die DATALINK-Spalte angegeben wurde.)

Dieser Parameter gilt für DATALINK-Spalten, in denen „Recovery=Yes" angegeben ist.

Data Links-Token-Algorithmus (dl_token)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] MAC0 [MAC0; MAC1]

Dieser Parameter gibt den Algorithmus an, der bei der Generierung von DATALINK-Token für Dateizugriffssteuerung verwendet wird. Der Wert von MAC1 (Nachrichtenauthentifizierungscode) generiert einen sichereren Nachrichtenauthentifizierungscode als MAC0, der Leistungsaufwand ist jedoch höher.

Informationen zu Dateizugriffssteuerungs-Token finden Sie im Handbuch *DB2 Data Links Manager Quick Beginnings*.

Dieser Parameter gilt für DATALINK-Spalten, in denen „READ PERMISSION DB" angegeben ist.

Data Links-Token in Großbuchstaben (dl_upper)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] NO [YES; NO]

Der Parameter gibt an, ob die Token für Dateizugriffssteuerung Großbuchstaben verwenden. Der Wert „YES" gibt an, dass alle Zeichen in einem Token für Zugriffssteuerung Großbuchstaben sind. Der Wert „NO" gibt an, dass der Token Groß- und Kleinbuchstaben enthalten kann.

Informationen zu Dateizugriffssteuerungs-Token finden Sie im Handbuch *DB2 Data Links Manager Quick Beginnings*.

Dieser Parameter gilt für DATALINK-Spalten, in denen „READ PERMISSION DB" angegeben ist.

Unterstützung der Aktivierung von Data Links (datalinks)

Konfigurationsart Datenbankmanager

Parameterart Konfigurierbar

Standardwert [Bereich] NO [YES; NO]

Dieser Parameter gibt an, ob die Unterstützung für Data Links aktiviert ist. Der Wert „YES" gibt an, dass die Unterstützung für Data Links über den Data Links Manager aktiviert ist, der in Basisdateisystemen (z. B. JFS unter AIX) gespeicherte Dateien verbindet. Der Wert „NO" gibt an, dass die Unterstützung für Data Links nicht aktiviert ist.

Status

Die folgenden Parameter stellen Informationen zum Status der Datenbank bereit:

- „Sicherung anstehend (backup_pending"
- „Datenbank ist konsistent (database_consistent)"
- „Aktualisierende Wiederherstellung anstehend (rollfwd_pending)" auf Seite 512
- „Status der Protokollspeicherung für Wiederherstellung (log_retain_status)" auf Seite 512
- „Status des Benutzerausgangs für Protokollierung (user_exit_status)" auf Seite 512
- „Wiederherstellung anstehend (restore_pending)" auf Seite 513
- „Zuordnung aus mehreren Seiten bestehender Datei aktiv (multipage_alloc)" auf Seite 513

Sicherung anstehend (backup_pending

Konfigurationsart Datenbank

Parameterart Informativ

Wenn dieser Parameter aktiviert ist, bedeutet dies, dass Sie eine Gesamtsicherung der Datenbank ausführen müssen, bevor Sie auf sie zugreifen. Dieser Parameter ist nur aktiviert, wenn die Datenbankkonfiguration geändert wurde, so dass die Datenbank jetzt nicht mehr nicht wiederherstellbar sondern wiederherstellbar ist (das bedeutet, die Parameter *logretain* und *userexit* waren zunächst auf NO gesetzt, später wird jedoch einer der Parameter (oder beide) auf YES gesetzt und die Aktualisierung der Datenbankkonfiguration wird akzeptiert).

Datenbank ist konsistent (database_consistent)

Konfigurationsart Datenbank

Parameterart Informativ

Dieser Parameter gibt an, ob die Datenbank in einem konsistenten Zustand ist.

YES gibt an, dass alle Transaktionen festgeschrieben oder rückgängig gemacht wurden, so dass die Daten konsistent sind. Wenn es zu einem Systemabsturz

kommt, während die Datenbank konsistent ist, sind keinerlei Maßnahmen erforderlich, um die Datenbank wieder verwendbar zu machen.

NO gibt an, dass noch eine Transaktion ansteht oder eine andere Funktion in der Datenbank noch nicht abgeschlossen wurde, so dass die Daten zu diesem Zeitpunkt nicht konsistent sind. Wenn es zu einem Systemabsturz kommt, während die Datenbank nicht konsistent ist, müssen Sie die Datenbank mit dem Befehl RESTART DATABASE erneut starten, um sie wieder verwendbar zu machen. Weitere Informationen zum Befehl RESTART DATABASE finden Sie im Handbuch *Command Reference*.

Aktualisierende Wiederherstellung anstehend (rollfwd_pending)

Konfigurationsart Datenbank

Parameterart Informativ

Dieser Parameter kann einen der folgenden Status anzeigen:
- **DATABASE**, d. h. eine aktualisierende Wiederherstellung ist für diese Datenbank erforderlich
- **TABLESPACE**, d. h. mindestens ein Tabellenbereich muss aktualisierend wiederhergestellt werden
- **NO**, d. h. die Datenbank ist verwendbar und keine aktualisierende Wiederherstellung erforderlich

Die Wiederherstellung (mit ROLLFORWARD DATABASE) muss erfolgreich beendet sein, bevor auf die Datenbank bzw. den Tabellenbereich zugegriffen werden kann. Weitere Informationen zu ROLLFORWARD DATABASE finden Sie im Handbuch *Command Reference*.

Status der Protokollspeicherung für Wiederherstellung (log_retain_status)

Konfigurationsart Datenbank

Parameterart Informativ

Zugehörige Parameter „Protokollspeicherung für Wiederherstellung (logretain)" auf Seite 490

Wenn dieser Parameter gesetzt ist, zeigt er an, dass Protokolldateien für die aktualisierende Wiederherstellung gespeichert werden.

Dieser Parameter wird gesetzt, wenn der Parameter *logretain* auf Recovery gesetzt ist.

Status des Benutzerausgangs für Protokollierung (user_exit_status)

Konfigurationsart Datenbank

Parameterart Informativ

Zugehörige Parameter „Benutzerausgang für Protokollierung aktivieren (userexit)" auf Seite 491

Wenn der Wert dieses Parameters "Yes" ist, heißt dies, dass der Datenbankmanager für die aktualisierende Wiederherstellung aktiviert ist und dass das Benutzerausgangsprogramm verwendet wird, um Protokolldateien zu archivieren und wieder abzurufen, wenn es vom Datenbankmanager aufgerufen wird.

Wiederherstellung anstehend (restore_pending)

Konfigurationsart Datenbank

Parameterart Informativ

Dieser Parameter gibt an, ob sich die Datenbank im Status *Wiederherstellung anstehend* befindet.

Zuordnung aus mehreren Seiten bestehender Datei aktiv (multipage_alloc)

Konfigurationsart Datenbank

Parameterart Informativ

Die Zuordnung aus mehreren Seiten bestehender Dateien erhöht die Leistung beim Einfügen. Sie gilt nur für SMS-Tabellenbereiche. Wenn dieser Parameter aktiviert ist, sind alle SMS-Tabellenbereiche davon betroffen; es ist keine Auswahl für einzelne SMS-Tabellenbereiche möglich.

Der Standardwert für den Parameter ist "No", d. h. die Zuordnung aus mehreren Seiten bestehender Dateien ist inaktiviert.

Nach der Datenbankerstellung kann der Parameter auf "Yes" gesetzt werden, um die Zuordnung aus mehreren Seiten bestehender Dateien zu ermöglichen. Dies kann mit dem Tool *db2empfa* ausgeführt werden. Wenn der Parameter auf "Yes" gesetzt ist, kann er nicht mehr auf "No" zurückgesetzt werden.

Weitere Informationen zu diesem Tool finden Sie im Handbuch *Command Reference*.

Compilereinstellungen

Die folgenden Parameter stellen Informationen zur Verfügung, die den Compiler beeinflussen:

- „Bei arithmetischen Ausnahmebedingungen fortsetzen (dft_sqlmathwarn)" auf Seite 515
- „Grad der Parallelität (dft_degree)" auf Seite 516
- „Standardabfrageoptimierungsklasse (dft_queryopt)" auf Seite 517
- „Standardaktualisierungsalter (dft_refresh_age)" auf Seite 518

- „Anzahl der häufigsten Werte (num_freqvalues)” auf Seite 518
- „Anzahl der Quantile für Spalten (num_quantiles)” auf Seite 519

Bei arithmetischen Ausnahmebedingungen fortsetzen (dft_sqlmathwarn)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	No [No, Yes]

Mit diesem Parameter wird der Standardwert festgelegt, der bestimmt, ob arithmetische Fehler und Fehler bei Abfrageumwandlungen beim Kompilieren von SQL-Anweisungen als Fehler oder als Warnungen behandelt werden. Bei statischen SQL-Anweisungen wird der Wert dieses Parameters dem Paket während des Bindens zugeordnet. Bei dynamischen SQL-DML-Anweisungen wird der Wert dieses Parameters verwendet, wenn die Anweisung vorbereitet wird (PREPARE).

Achtung: Wenn Sie den Wert des Parameters *dft_sqlmathwarn* für eine Datenbank ändern, kann sich die Funktionsweise von Prüfungen auf Integritätsbedingung, Auslösern und Sichten, die arithmetische Ausdrücke enthalten, ändern. Dies wiederum kann sich auf die Datenintegrität der Datenbank auswirken. Sie sollten die Einstellung des Parameters *dft_sqlmathwarn* für eine Datenbank nur nach einer gründlichen Analyse der Auswirkungen der neuen Behandlung arithmetischer Ausnahmebedingungen auf Integritätsprüfungen, Auslöser und Sichten ändern. Auch alle weiteren Änderungen sind mit derselben Sorgfalt vorzunehmen.

Betrachten Sie beispielsweise die folgende Prüfung auf Integritätsbedingung, die eine arithmetische Divisionsoperation enthält:

```
A/B > 0
```

Wenn *dft_sqlmathwarn* gleich „No" ist und eine Einfügeoperation (INSERT) mit B=0 versucht wird, wird die Division durch 0 als arithmetischer Fehler verarbeitet. Die Einfügeoperation schlägt fehl, weil DB2 die Integritätsbedingung nicht prüfen kann. Wenn der Parameter *dft_sqlmathwarn* auf „Yes" gesetzt wird, wird die Division durch 0 als arithmetische Warnung mit dem Ergebnis NULL verarbeitet. Das Ergebnis NULL führt dazu, dass das Vergleichselement „>" mit dem Wert UNKNOWN ausgewertet wird und die Einfügeoperation erfolgreich ist. Wenn *dft_sqlmathwarn* wieder auf „No" gesetzt wird, schlägt der Versuch, dieselbe Zeile einzufügen, fehl, weil der Fehler der Division durch 0 DB2 daran hindert, die Integritätsbedingung auszuwerten. Die Zeile, die mit B=0 eingefügt wurde, als der Parameter *dft_sqlmathwarn* den Wert „Yes" hatte, bleibt in der Tabelle und kann ausgewählt werden. Aktualisierungen der Zeile, für die die Integritätsbedingung ausgewertet wird, schlagen fehl, während Aktualisierungen, die keine erneute Prüfung der Integritätsbedingung erfordern, erfolgreich ausgeführt werden.

Bevor Sie den Parameter *dft_sqlmathwarn* von „No" in „Yes" ändern, sollten Sie in Betracht ziehen, die Integritätsbedingung so umzuschreiben, dass sie Nullwerte aus arithmetischen Ausdrücken gesondert behandelt. Zum Beispiel:

```
( A/B > 0 ) AND ( CASE
                  WHEN A IS NULL THEN 1
                  WHEN B IS NULL THEN 1
                  WHEN A/B IS NULL THEN 0
                  ELSE 1
                  END
                = 1 )
```

Diese Bedingung kann verwendet werden, wenn A und B Nullwerte haben können, d. h. die Dateneingabe nicht erforderlich ist. Wenn A oder B keinen Nullwert haben kann, kann die entsprechende Klausel WHEN...IS NULL entfernt werden.

Bevor Sie den Parameter *dft_sqlmathwarn* von „Yes" in „No" ändern, sollten Sie zunächst prüfen, welche Daten inkonsistent werden könnten, indem Sie zum Beispiel Vergleichselemente wie die folgenden verwenden:

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

Wenn inkonsistente Zeilen isoliert sind, können Sie geeignete Maßnahmen zur Korrektur der Inkonsistenz ergreifen, bevor Sie den Parameter *dft_sqlmathwarn* ändern. Sie können Integritätsbedingungen mit arithmetischen Ausdrücken nach der Änderung auch manuell gegenprüfen. Dazu setzen Sie die betroffenen Tabellen in den Status *Überprüfung anstehend* (mit der Klausel OFF der Anweisung SET CONSTRAINTS) und fordern anschließend eine Prüfung an (mit der Klausel IMMEDIATE CHECKED der Anweisung SET CONSTRAINTS). Inkonsistente Daten werden durch einen arithmetischen Fehler angezeigt, der verhindert, dass die Integritätsbedingung ausgewertet wird.

Empfehlung: Verwenden Sie die Standardeinstellung "No", sofern Sie keine Abfragen benötigen, deren Verarbeitung arithmetische Ausnahmebedingungen einschließt. In diesem Fall geben Sie den Wert "Yes" an. Dieser Fall kann eintreten, wenn Sie SQL-Anweisungen verarbeiten, die auf anderen Datenbankmanagern unabhängig von möglichen arithmetischen Ausnahmebedingungen Ergebnisse liefern.

Grad der Parallelität (dft_degree)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	1 [-1, 1 – 32 767]
Zugehörige Parameter	„Maximaler Grad der Parallelität bei Abfragen (max_querydegree)" auf Seite 541

Mit diesem Parameter wird der Standardwert für das Sonderregister CURRENT DEGREE und die Bindeoption DEGREE angegeben.

Der Standardwert ist 1.

Bei Angabe des Werts 1 wird keine partitionsinterne Parallelität verwendet. Der Wert -1 bedeutet, dass das Optimierungsprogramm den Grad der partitionsinternen Parallelität je nach Anzahl der Prozessoren und Art der Abfrage festlegt.

Der Grad der partitionsinternen Parallelität für eine SQL-Anweisung wird während der Kompilierung der Anweisung mit Hilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption DEGREE angegeben. Der maximale Grad der partitionsinternen Parallelität zur Laufzeit für eine aktive Anwendung wird mit dem Befehl SET RUNTIME DEGREE angegeben. Der Konfigurationsparameter Maximaler Grad der Parallelität bei Abfragen (*max_querydegree*) gibt den maximalen Grad der partitionsinternen Parallelität für SQL-Abfragen an.

Der tatsächlich zur Laufzeit verwendete Grad ist der niedrigste der folgenden Werte:
- Konfigurationsparameter *max_querydegree*
- Grad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

Standardabfrageoptimierungsklasse (dft_queryopt)

Konfigurationsart	Datenbank
Parameterart	Konfigurierbar
Standardwert [Bereich]	5 [0 — 9]
Maßeinheit	Abfrageoptimierungsklasse (siehe unten)

Mit der Abfrageoptimierungsklasse können Sie das Optimierungsprogramm anweisen, beim Kompilieren von SQL-Abfragen verschiedene Grade der Optimierung zu verwenden. Dieser Parameter bietet zusätzliche Flexibilität, indem die Standardabfrageoptimierungsklasse für den Fall festgelegt wird, dass weder die Anweisung SET CURRENT QUERY OPTIMIZATION noch der Bindebefehl QUERYOPT verwendet werden.

Derzeit sind folgende Abfrageoptimierungsklassen definiert:
0 - Minimale Abfrageoptimierung
1 - Abfrageoptimierung in etwa vergleichbar mit DB2 Version 1
2 - Geringe Optimierung
3 - Mittlere Abfrageoptimierung

5 - Starke Abfrageoptimierung, die über heuristische Methoden den Aufwand bei der Auswahl eines Zugriffsplans reduziert. Dies ist der Standardwert.
7 - Starke Abfrageoptimierung
9 - Maximale Abfrageoptimierung

Empfehlung: Weitere Informationen und eine Anleitung zur Auswahl einer geeigneten Abfrageoptimierungsklasse finden Sie im Abschnitt „Anpassen der Optimierungsklasse" auf Seite 76.

Weitere Informationen, wie ein Programm die Konfigurationsparameter der Datenbank abrufen und ändern kann, finden Sie im Handbuch *Administrative API Reference*.

Standardaktualisierungsalter (dft_refresh_age)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] 0 [0, 99999999999999 (ANY)]

Für diesen Parameter wird der für REFRESH AGE verwendete Standardwert eingesetzt, wenn das Sonderregister CURRENT REFRESH AGE nicht angegeben wird. Dieser Parameter gibt einen Zeitmarkendifferenzwert mit dem Datentyp DECIMAL(20,6) an. Diese Zeitdifferenz stellt die maximale Dauer seit der Verarbeitung einer Anweisung REFRESH TABLE für eine spezifische Übersichtstabelle der Art REFRESH DEFERRED dar, während der diese Übersichtstabelle zum Optimieren der Verarbeitung einer Abfrage verwendet werden kann. Wenn für CURRENT REFRESH AGE der Wert 99999999999999 (ANY) gilt und die Abfrageoptimierungsklasse (QUERY OPTIMIZATION) fünf oder mehr ist, werden Übersichtstabellen der Art REFRESH DEFERRED zum Optimieren der Verarbeitung einer dynamischen SQL-Abfrage herangezogen.

Anzahl der häufigsten Werte (num_freqvalues)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] 10 [0 — 32 767]

Maßeinheit Zähler

Zugehörige Parameter

- „Anzahl der Quantile für Spalten (num_quantiles)" auf Seite 519
- „Größe des Statistikzwischenspeichers (stat_heap_sz)" auf Seite 427

Mit diesem Parameter können Sie die Anzahl der „häufigsten Werte" angeben, die gesammelt werden, wenn die Option WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird. Wenn der Wert dieses Parameters erhöht wird, erhöht sich auch die Menge an Statistikzwischenspeicher (*stat_heap_sz*), die zum Sammeln statistischer Daten benötigt wird.

Die Statistik der „häufigsten Werte" unterstützt das Optimierungsprogramm beim Feststellen der Verteilung der Datenwerte innerhalb einer Spalte. Wenn der Wert erhöht wird, stehen dem SQL-Optimierungsprogramm mehr Daten zur Verfügung. Andererseits ist für den Katalog mehr Speicher erforderlich. Wenn 0 angegeben wird, wird keine Statistik über häufige Werte erhoben, auch wenn Sie anfordern, dass statistische Informationen zur Datenverteilung gesammelt werden.

Durch Aktualisieren dieses Parameters kann das Optimierungsprogramm bessere Schätzwerte für die Auswahl von nicht gleichmäßig verteilten Daten mit Hilfe einiger Vergleichselemente (=, <, >, IS NULL, IS NOT NULL) erzielen. Exaktere Berechnungen der Auswahlmöglichkeiten ermöglichen die Wahl effizienterer Zugriffspläne.

Nachdem Sie den Wert dieses Parameters verändert haben, müssen Sie wie folgt vorgehen:
- Führen Sie den Befehl RUNSTATS aus, nachdem alle Benutzer die Verbindung zur Datenbank getrennt und Sie die Verbindung zur Datenbank wiederhergestellt haben.
- Binden Sie alle Pakete mit statischen SQL-Anweisungen erneut.

Weitere Informationen finden Sie in „Erfassen und Verwenden von Verteilungsstatistiken" auf Seite 140.

Empfehlung: Zur Aktualisierung dieses Parameters sollten Sie den Grad der Ungleichmäßigkeit der Daten in den wichtigsten Spalten (in den wichtigsten Tabellen) feststellen, für die in der Regel Auswahlvergleichselemente angegeben werden. Dies kann mit Hilfe einer SQL-Anweisung SELECT geschehen, die eine Rangfolge des Vorkommens jedes Werts in einer Spalte liefert. Dabei dürfen Sie einheitlich verteilte, eindeutige, lange oder LOB-Spalten nicht berücksichtigen. Ein geeigneter praktischer Wert für diesen Parameter liegt im Bereich zwischen 10 und 100.

Beachten Sie, dass das Sammeln statistischer Daten über die häufigsten Werte erhebliche CPU- und Speicherressourcen (*stat_heap_sz*) erfordert.

Anzahl der Quantile für Spalten (num_quantiles)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] 20 [0 – 32 767]

Maßeinheit Zähler

Zugehörige Parameter

- „Anzahl der häufigsten Werte (num_freqvalues)" auf Seite 518
- „Größe des Statistikzwischenspeichers (stat_heap_sz)" auf Seite 427

Mit diesem Parameter wird die Anzahl der Quantile gesteuert, die gesammelt werden, wenn die Option WITH DISTRIBUTION im Befehl RUNSTATS angegeben wird. Wenn der Wert dieses Parameters erhöht wird, erhöht sich auch die Menge an Statistikzwischenspeicher (*stat_heap_sz*), die zum Sammeln statistischer Daten benötigt wird.

Die Statistik der „Quantile" unterstützt das Optimierungsprogramm beim Feststellen der Verteilung der Datenwerte innerhalb einer Spalte. Wenn der Wert erhöht wird, stehen dem SQL-Optimierungsprogramm mehr Daten zur Verfügung. Andererseits ist für den Katalog mehr Speicher erforderlich. Wenn die Werte 0 oder 1 angegeben werden, werden keine Quantil-Statistikdaten erhoben, auch wenn Sie anfordern, dass Verteilungsstatistikdaten gesammelt werden.

Durch Aktualisieren dieses Parameters können bessere Schätzwerte für die Auswahl von nicht gleichmäßig verteilten Daten mit Hilfe von Bereichsvergleichselementen erzielt werden. Unter anderem entscheidet das Optimierungsprogramm mit Hilfe dieser Informationen, ob eine Indexsuche oder eine Tabellensuche gewählt wird. (Beim Zugriff auf einen Bereich von Werten, die häufig vorkommen, ist eine Tabellensuche effizienter, während bei einem Bereich von Werten, die nicht häufig vorkommen, eine Indexsuche effizienter ist.)

Nachdem Sie den Wert dieses Parameters verändert haben, müssen Sie wie folgt vorgehen:

- Führen Sie den Befehl RUNSTATS aus, nachdem alle Benutzer die Verbindung zur Datenbank getrennt und Sie die Verbindung zur Datenbank wiederhergestellt haben.
- Binden Sie alle Pakete mit statischen SQL-Anweisungen erneut.

Weitere Informationen finden Sie in „Erfassen und Verwenden von Verteilungsstatistiken" auf Seite 140.

Empfehlung: Der Standardwert für diesen Parameter garantiert einen maximalen Schätzfehler von ungefähr 2,5 % für alle einseitigen Bereichsvergleichs-

elemente (>, >=, < oder <=) und einen maximalen Fehler von 5 % für jedes BETWEEN-Vergleichselement. Nachfolgend eine einfache Möglichkeit, um die Anzahl der Quantile einzugrenzen:

- Bestimmen Sie den maximalen Fehler, der bei der Schätzung der Anzahl von Zeilen jeder Bereichsabfrage tolerierbar ist, als Prozentwert P.
- Die Anzahl der Quantile sollte ca. 100/P sein, wenn die meisten Ihrer Vergleichselemente BETWEEN-Vergleichselemente sind, und 50/P, wenn die meisten Ihrer Vergleichselemente andere Arten von Bereichsvergleichselementen (<, <=, > oder >=) sind.

Zum Beispiel ergeben 25 Quantile einen maximalen Schätzfehler von 4% bei BETWEEN-Vergleichselementen und 2% bei Vergleichselementen mit ″>″. Ein geeigneter praktischer Wert für diesen Parameter liegt im Bereich zwischen 10 und 50.

Übertragung

Die folgenden Gruppen von Parametern stellen Informationen zur Verwendung von DB2 in einer Client-/Server-Umgebung bereit:

- „Konfiguration der Übertragungsprotokolle″
- „Verteilte Services″ auf Seite 526
- „DB2 Discovery″ auf Seite 531

Konfiguration der Übertragungsprotokolle

Die folgenden Parameter dienen zur Konfiguration von Datenbank-Clients und Datenbank-Servern:

- „NetBIOS-Knotenname (nname)″
- „TCP/IP-Servicename (svcename)″ auf Seite 522
- „APPC-Transaktionsprogrammname (tpname)″ auf Seite 523
- „Name des IPX/SPX-Datei-Servers (fileserver)″ auf Seite 524
- „Objektname für IPX/SPX-DB2-Server (objectname)″ auf Seite 525
- „IPX/SPX-Socket-Nummer (ipx_socket)″ auf Seite 525

NetBIOS-Knotenname (nname)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert Null

Mit diesem Parameter können Sie dem Datenbankexemplar auf einer Workstation in der NetBIOS-LAN-Umgebung einen eindeutigen Namen zuordnen. Der Wert von *nname* ist die Basis für die tatsächlichen NetBIOS-Namen, die in NetBIOS für eine Workstation registriert werden.

Da das NetBIOS-Protokoll seine Verbindungen mit Hilfe dieser NetBIOS-Namen erstellt, muss der Parameter *nname* sowohl für den Client als auch für den Server festgelegt werden.

Client-Anwendungen müssen den *nname*-Wert des Servers mit der Datenbank kennen, auf die zugegriffen werden soll. Der *nname*-Wert des Servers muss im Client-Knotenverzeichnis als Parameter „server-nname" mit dem Befehl CATALOG NETBIOS NODE katalogisiert werden.

Wenn der Knotenname *nname* auf dem Server-Knoten in einen neuen Namen geändert wird, müssen alle Clients, die auf Datenbanken dieses Servers zugreifen, diesen neuen Namen für den Server katalogisieren.

TCP/IP-Servicename (svcename)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert Null

Dieser Parameter enthält den Namen des TCP/IP-Anschlusses, an dem ein Datenbank-Server auf Datenübertragungen von fernen Client-Knoten wartet. Dieser Name muss der erste zweier aufeinander folgender Anschlüsse sein, die für die Verwendung durch den Datenbankmanager reserviert sind. Der zweite Anschluss wird zur Verarbeitung von Unterbrechungsanforderungen von Clients mit einer früheren Programmversion verwendet.

Damit Verbindungsanforderungen von einem Datenbank-Client mit Hilfe von TCP/IP empfangen werden können, muss der Datenbank-Server an einem Anschluss, der für diesen Server reserviert ist, empfangsbereit sein. Der

Systemadministrator für den Datenbank-Server muss einen Anschluss (Nummer *n*) reservieren und den zugeordneten TCP/IP-Servicenamen in der Datei *services* auf dem Server definieren. Wenn der Datenbank-Server Anforderungen von Clients mit einer früheren Programmversion unterstützen soll, muss ein zweiter Anschluss (Nummer *n+1* für Unterbrechungsanforderungen) in der Datei *services* auf dem Server definiert werden.

Der Anschluss (Nummer *n*) des Datenbank-Servers und sein TCP/IP-Servicename müssen in der Datei *services* auf dem Datenbank-Client definiert werden. Bei Clients mit einer früheren Programmversion (vor Version 2) muss außerdem der Unterbrechungsanschluss (Nummer *n+1*) in der Datei *services* des Clients definiert werden.

Die Speicherposition der Datei *services* hängt von der Betriebsumgebung ab. Zum Beispiel:
- Unter UNIX — /etc/services
- Unter OS/2 — \tcpip\etc\services
- Unter OS/2 Warp — \mptn\etc\services

Der Parameter *svcename* sollte auf den Servicenamen gesetzt werden, der dem Hauptverbindungsanschluss zugeordnet ist, so dass der Datenbank-Server nach dem Start weiß, an welchem Anschluss er für eingehende Verbindungsanforderungen empfangsbereit sein muss. Wenn Sie einen Client mit einer früheren Programmversion unterstützen oder verwenden, wird der Servicename für den Unterbrechungsanschluss in der Konfigurationsdatei nicht gespeichert. Die Nummer des Unterbrechungsanschlusses kann anhand der Nummer des Hauptverbindungsanschlusses ermittelt werden (*Nummer des Unterbrechungsanschlusses = Nummer des Hauptverbindungsanschlusses* + 1).

Weitere Informationen zur Einrichtung von TCP/IP für Datenbank-Server finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

APPC-Transaktionsprogrammname (tpname)

Konfigurationsart Datenbankmanager

Gilt für
- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert Null

Mit diesem Parameter wird der Name eines fernen Transaktionsprogramms definiert, das der Datenbank-Client verwenden muss, wenn er unter Verwendung des APPC-Übertragungsprotokolls eine Zuordnungsanforderung an den Datenbank-Server absetzt. Dieser Parameter muss in der Konfigurationsdatei auf dem Datenbank-Server eingestellt werden.

Der Wert dieses Parameters muss mit dem Transaktionsprogrammnamen übereinstimmen, der in der SNA-Transaktionsprogrammdefinition konfiguriert ist. Weitere Informationen zum Einrichten von APPC für Ihr DB2-Produkt finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

Empfehlung: Nur folgende Zeichen sind zur Verwendung in diesem Namen zulässig:
- Buchstaben (A - Z oder a - z)
- Numerische Zeichen (0 - 9)
- Dollarzeichen ($), Nummernzeichen (#), kommerzielles A (@) und Punkt (.)

Name des IPX/SPX-Datei-Servers (fileserver)

Konfigurationsart Datenbankmanager

Gilt für
- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert Null

Zugehörige Parameter
- „Objektname für IPX/SPX-DB2-Server (objectname)" auf Seite 525
- „IPX/SPX-Socket-Nummer (ipx_socket)" auf Seite 525

Mit diesem Parameter wird der Name des NetWare-Datei-Servers angegeben, auf dem die Internet-Adresse des Datenbankmanagers registriert ist. Die Internet-Adresse des Datenbankmanagers wird in der Registrierdatenbank des NetWare-Datei-Servers gespeichert. Wenn der registrierte Name des Datei-Servers geändert wird, müssen auf allen Clients, die auf das Server-Exemplar zugreifen, folgende Schritte ausgeführt werden:
- Entkatalogisieren des Server-Knotens (UNCATALOG)
- Katalogisieren des Server-Knotens mit dem neuen Namen für den Datei-Server (CATALOG)

Weitere Informationen finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

Objektname für IPX/SPX-DB2-Server (objectname)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert Null

Zugehörige Parameter

- „Name des IPX/SPX-Datei-Servers (fileserver)" auf Seite 524
- „IPX/SPX-Socket-Nummer (ipx_socket)"

Mit diesem Parameter wird der Name des Datenbankmanagerexemplars in einem IPX/SPX-Netzwerk angegeben. Jedes Server-Exemplar, das auf einem NetWare-Datei-Server registriert ist, muss einen eindeutigen Namen haben. Wenn dieser Name auf dem Datenbank-Server geändert wird, muss auf allen Clients, die auf den Server zugreifen, der Server-Knoten entkatalogisiert und mit dem neuen Objektnamen erneut katalogisiert werden.

Weitere Informationen zum Einrichten von IPX/SPX für Ihr DB2-Produkt finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

IPX/SPX-Socket-Nummer (ipx_socket)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 879E [879E – 87A2] Zur Vermeidung von Kon-

flikten sind fünf Socket-Nummern (879E - 87A2) bei Novell eindeutig zur Verwendung durch DB2 registriert.

Zugehörige Parameter

- „Name des IPX/SPX-Datei-Servers (fileserver)" auf Seite 524
- „Objektname für IPX/SPX-DB2-Server (objectname)" auf Seite 525

Dieser Parameter gibt eine „bekannte" Socket-Nummer an und stellt den Verbindungsendpunkt in der Internet-Adresse eines DB2-Servers dar. Die Socket-Nummer muss für jedes DB2-Server-Exemplar auf einem System und für alle Novell-IPX/SPX-Anwendungen, die auf derselben Maschine ausgeführt werden, eindeutig sein. Dadurch wird sichergestellt, dass der DB2-Server für eingehende IPX/SPX-Verbindungsanforderungen über diese Socket-Nummer empfangsbereit ist.

Weitere Informationen zum Einrichten von IPX/SPX für Ihr DB2-Produkt finden Sie im Handbuch *Installation und Konfiguration Ergänzung*.

Verteilte Services

Die folgenden Parameter dienen zur Konfiguration der Datenbank-Clients und Datenbank-Server unter Verwendung der DCE-Verzeichnisservices:

- „Verzeichnisserviceart (dir_type)"
- „Verzeichnispfadname (dir_path_name)" auf Seite 527
- „Verzeichnisobjektname (dir_obj_name)" auf Seite 528
- „Name des Leitweginformationsobjekts (route_obj_name)" auf Seite 529
- „Standard-Client-Übertragungsprotokoll (dft_client_comm)" auf Seite 530
- „Standard-Client-Adapternummer (dft_client_adpt)" auf Seite 531

Informationen zur Verwendung von DCE-Verzeichnissen durch DB2 finden Sie im Abschnitt zur Verwendung der DCE-Verzeichnisservices im Handbuch *Systemverwaltung: Implementierung*.

Verzeichnisserviceart (dir_type)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- UNIX- und OS/2-Client
- UNIX- und OS/2-Datenbank-Server mit lokalen Clients

- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] NONE [NONE; DCE]

Zugehörige Parameter

- „Verzeichnisobjektname (dir_obj_name)" auf Seite 528
- „Verzeichnispfadname (dir_path_name)"
- „Name des Leitweginformationsobjekts (route_obj_name)" auf Seite 529
- „Standard-Client-Übertragungsprotokoll (dft_client_comm)" auf Seite 530
- „Standard-Client-Adapternummer (dft_client_adpt)" auf Seite 531

Mit diesem Parameter wird angegeben, ob die DCE-Verzeichnisservices verwendet werden oder nicht.

Wenn dieser Parameter auf NONE gesetzt wird, werden nur Dateien in lokalen Verzeichnissen nach dem Ziel der CONNECT- oder ATTACH-Anforderungen durchsucht. Sie können aber weiterhin die Parameter *dir_path_name* und *dir_obj_name* verwenden, um die Namen Ihres Datenbankexemplars und Ihrer Datenbanken in den DCE-Namensbereich einzutragen.

Wenn dieser Parameter auf DCE gesetzt wird, wird das DCE-Verzeichnis durchsucht, wenn eine Anwendung, die in diesem Datenbankmanager-Exemplar ausgeführt wird, das Ziel ihrer CONNECT- oder ATTACH-Anforderungen nicht finden kann.

Informationen zu numerischen Äquivalenten und API-Konstanten für diese Werte finden Sie im Handbuch *Administrative API Reference*.

Verzeichnispfadname (dir_path_name)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- UNIX- und OS/2-Client
- UNIX- und OS/2-Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert /.:/subsys/database/

Zugehörige Parameter

- „Verzeichnisobjektname (dir_obj_name)"
- „Verzeichnisserviceart (dir_type)" auf Seite 526
- „Name des Leitweginformationsobjekts (route_obj_name)" auf Seite 529

Der eindeutige Name des Datenbankmanagerexemplars im globalen Namensbereich wird aus diesem Wert und dem Wert des Parameters *dir_obj_name* zusammengesetzt.

Alle Client-Anwendungen, die innerhalb dieses Exemplars ausgeführt werden, verwenden diesen Parameter auch als Standardpfadnamen für ihre CONNECT- oder ATTACH-Anforderungen, sofern er nicht vom Wert der Umgebungsvariablen DB2DIRPATHNAME überschrieben wird.

Empfehlung: Verwenden Sie den Namen, den Sie von Ihrem DCE-Administrator erhalten haben.

Verzeichnisobjektname (dir_obj_name)

Konfigurationsart Datenbankmanager, Datenbank

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- UNIX- und OS/2-Client
- UNIX- und OS/2-Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert Null

Zugehörige Parameter

- „Verzeichnisserviceart (dir_type)" auf Seite 526
- „Verzeichnispfadname (dir_path_name)" auf Seite 527

Mit diesem Parameter wird der Objektname angegeben, der Ihr Datenbankmanagerexemplar (oder Ihre Datenbank) im Verzeichnis darstellt. Die Verkettung dieses Werts mit dem Wert für *dir_path_name* ergibt einen Globalnamen, der das Datenbankmanagerexemplar oder die Datenbank in dem Namens-

bereich eindeutig identifiziert, der von den durch den Parameter *dir_type* definierten Verzeichnisservices verwaltet wird.

Dieser Parameter hat nur dann eine Funktion, wenn der Parameter *dir_path_name* definiert ist.

Die Gesamtlänge der Werte für die Konfigurationsparameter *dir_path_name* und *dir_obj_name* muss unter 255 Zeichen liegen.

Weitere Informationen finden Sie im Abschnitt zur Verwendung der Verzeichnisservices für die Umgebung für verteilte Datenverarbeitung im Handbuch *Systemverwaltung: Implementierung*.

Name des Leitweginformationsobjekts (route_obj_name)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert Null

Zugehörige Parameter

- „Verzeichnispfadname (dir_path_name)" auf Seite 527
- „Verzeichnisserviceart (dir_type)" auf Seite 526

Mit diesem Parameter wird der Name des Standardeintrags für das Leitweginformationsobjekt festgelegt, der von allen Client-Anwendungen verwendet wird, die versuchen, auf einen DRDA-Server zuzugreifen. Dieser Parameter gilt nur für OS/2-Umgebungen und für auf UNIX basierende Umgebungen.

Wenn der Wert dieses Parameters mit /.:/ oder /.../ beginnt, wird er so verwendet, wie er ist. Andernfalls wird er an den Wert des Parameters *dir_path_name* (oder der Umgebungsvariablen DB2DIRPATHNAME) angehängt, um den vollständigen Namen des Leitweginformationsobjekts zu bilden.

Sie können die Umgebungsvariable DB2ROUTE verwenden, um diesen Standardwert zu überschreiben.

Dieser Parameter hat nur dann eine Funktion, wenn der Parameter *dir_type* auf den Wert DCE gesetzt ist.

Weitere Informationen finden Sie im Abschnitt zur Verwendung der Verzeichnisservices für die Umgebung für verteilte Datenverarbeitung im Handbuch *Systemverwaltung: Implementierung*.

Standard-Client-Übertragungsprotokoll (dft_client_comm)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [NULL; TCPIP; APPC; IPXSPX (nur OS/2); NETBIOS (nur OS/2)]

Zugehörige Parameter „Verzeichnisserviceart (dir_type)" auf Seite 526

Mit diesem Parameter werden die Übertragungsprotokolle angegeben, die von Client-Anwendungen in diesem Exemplar für ferne Verbindungen verwendet werden können. Der Wert ist eine Zeichenfolge, die aus einem oder mehreren Token besteht. Wenn mehrere Token angegeben werden, müssen diese durch Komma getrennt werden. Die Reihenfolge der Token wird bei der Verarbeitung beachtet.

Dieser Parameter kann nur mit DCE verwendet werden und ist nur in OS/2-Umgebungen und auf UNIX basierenden Umgebungen gültig.

Sie können den Wert dieses Parameters temporär überschreiben, indem Sie die Umgebungsvariable DB2CLIENTCOMM definieren.

Wenn der Wert dieses Parameters NULL ist und die Umgebungsvariable nicht definiert wurde, wird das erste Protokoll verwendet, das im Globalverzeichnisobjekt des Servers angegeben ist.

Dieser Parameter wird ignoriert, wenn der Parameter *dir_type* den Wert NONE hat.

Empfehlung: Das am häufigsten verwendete Protokoll sollte zuerst angegeben werden.

Standard-Client-Adapternummer (dft_client_adpt)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 0 [0–15]

Zugehörige Parameter

- „Standard-Client-Übertragungsprotokoll (dft_client_comm)" auf Seite 530.
- „Verzeichnisserviceart (dir_type)" auf Seite 526. (wenn *dir_type* auf DCE gesetzt ist)

Dieser Parameter definiert die Standard-Client-Adapternummer für das NetBIOS-Protokoll, dessen Server-nname aus den DCE-Zellenverzeichnisdiensten abgerufen wird. Dieser Parameter gilt nur für die OS/2-Umgebung.

Dieser Parameter kann nur mit DCE verwendet werden.

Sie können den Wert dieses Parameters temporär überschreiben, indem Sie die Umgebungsvariable DB2CLIENTADPT definieren. Wenn diese Umgebungsvariable eine nichtnumerische oder eine außerhalb des gültigen Bereichs liegende Nummer enthält, wird die Adapternummer 0 (Null) verwendet.

DB2 Discovery

Die folgenden Parameter dienen zur Einrichtung von DB2 Discovery:

- „Discovery-Unterstützung für diese Datenbank (discover_db)"
- „Discovery-Modus (discover)" auf Seite 532
- „Discovery-Kommunikationsprotokoll (discover_comm)" auf Seite 533
- „Discover-Server-Exemplar (discover_inst)" auf Seite 534

Discovery-Unterstützung für diese Datenbank (discover_db)

Konfigurationsart Datenbank

Parameterart Konfigurierbar

Standardwert [Bereich] Enable [Disable, Enable]

Mit diesem Parameter kann verhindert werden, dass Informationen zu einer Datenbank an einen Client zurückgegeben werden, wenn eine Discovery-Anforderung auf dem Server empfangen wird.

Standardmäßig ist dieser Parameter so eingestellt, dass Discovery-Unterstützung für diese Datenbank aktiviert ist.

Durch Ändern dieses Parameterwerts in „Disable" können Datenbanken, die sensible Daten enthalten, vor dem Discovery-Prozess verdeckt werden. Diese Maßnahme kann zusätzlich zu anderen Datenbanksicherheitsfunktionen für die Datenbank ergriffen werden.

Informationen zu numerischen Äquivalenten und API-Konstanten für diese Werte finden Sie im Handbuch *Administrative API Reference*.

Discovery-Modus (discover)

Konfigurationsart	Datenbankmanager
Gilt für	• Datenbank-Server mit lokalen und fernen Clients • Client • Datenbank-Server mit lokalen Clients • Partitionierter Datenbank-Server mit lokalen und fernen Clients • Satellitendatenbank-Server mit lokalen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	SEARCH [DISABLE, KNOWN, Search]
Zugehörige Parameter	„Discovery-Kommunikationsprotokoll (discover_comm)" auf Seite 533

Aus der Perspektive eines Verwaltungs-Servers bestimmt dieser Konfigurationsparameter die Art von Discovery-Modus, die beim Start von DB2ADMIN gestartet wird.

- Wenn beim Start von DB2ADMIN für *discover* SEARCH angegeben ist, dann werden die entsprechenden Verbindungsmanager für Discovery-Operationen für alle in *discover_comm* angegebenen Protokolle gestartet. Zudem werden Verbindungsmanager für alle in der Registrierungsvariablen DB2COMM angegebenen Protokolle gestartet. Dadurch kann der Verwaltungs-Server die entsprechenden Discovery-Anforderungen von Clients bearbeiten. SEARCH stellt eine Implikation der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität zur Verfügung. Wenn für *discover* SEARCH angegeben ist, bearbeitet der Verwaltungs-Server sowohl Discovery-Anforderungen mit dem Parameter SEARCH als auch Discovery-Anforderungen mit dem Parameter KNOWN von Clients.

- Wenn beim Start von DB2ADMIN für *discover* KNOWN angegeben ist, dann werden nur die in der Registrierungsvariablen DB2COMM angegebenen Verbindungsmanager gestartet. Diese Verbindungsmanager bearbeiten Discovery-Anforderungen mit dem Parameter KNOWN.
- Wenn beim Start von DB2ADMIN für *discover* DISABLE angegeben ist, dann bearbeitet der Verwaltungs-Server keine Art von Discovery-Anforderung.

Aus der Perspektive eines Server-Exemplars bedeutet die Angabe DISABLE für *discover*, dass die Informationen für dieses Server-Exemplar im Wesentlichen vor Clients verdeckt sind. Der Verwaltungs-Server packt keine Informationen zu diesem Exemplar, wenn von einem Client eine Discovery-Anforderung mit dem Parameter KNOWN für dieses System abgesetzt wird.

Aus der Perspektive eines Clients tritt nur Folgendes ein:

- Wenn für *discover* SEARCH angegeben ist, kann der Client Discovery-Anforderungen mit dem Parameter SEARCH zum Suchen von DB2-Server-Systemen im Netzwerk absetzen. Discovery mit dem Parameter SEARCH stellt eine Implikation der durch Discovery mit dem Parameter KNOWN bereitgestellten Funktionalität zur Verfügung. Wenn für *discover* SEARCH angegeben ist, können vom Client sowohl Discovery-Anforderungen mit dem Parameter SEARCH als auch Discovery-Anforderungen mit dem Parameter KNOWN abgesetzt werden.
- Wenn für *discover* KNOWN angegeben ist, können vom Client nur Discovery-Anforderungen mit dem Parameter KNOWN abgesetzt werden. Durch Angabe einiger Verbindungsinformationen für den Verwaltungs-Server auf einem bestimmten System werden alle Exemplar- und Datenbankinformationen auf dem DB2-System an den Client zurückgegeben.
- Wenn für *discover* DISABLE angegeben ist, ist Discovery auf dem Client inaktiviert.

Der Standard-Discovery-Modus ist SEARCH.

Informationen zu numerischen Äquivalenten und feAPI-Konstanten finden Sie im Handbuch *Administrative API Reference*.

Weitere Informationen zu DB2 Discovery finden Sie im Handbuch *Einstieg* für Ihre Plattform.

Discovery-Kommunikationsprotokoll (discover_comm)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients

- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] Keiner [Beliebige Kombination aus NetBIOS und TCP/IP]

Zugehörige Parameter „Discovery-Modus (discover)" auf Seite 532

Aus der Perspektive des Verwaltungs-Servers definiert dieser Parameter Discovery Manager für Suchvorgänge, die beim Start von DB2ADMIN gestartet werden. Diese Manager verwalten Discovery-Suchanforderungen von Clients.

Anmerkung: Die in *discover_comm* definierten Protokolle müssen auch in der Registrierungsvariablen DB2COMM angegeben werden.

Aus der Perspektive eines Clients definiert dieser Parameter die Protokolle, die Clients zum Absetzen von Discovery-Suchanforderungen verwenden.

Es können mehrere Protokolle, durch Kommas getrennt, angegeben werden; es sind jedoch keine Angaben für diesen Parameter erforderlich.

Der Standardwert für diesen Parameter ist "Keiner". Dies bedeutet, dass es keine Discovery-Kommunikationsprotokolle für Suchvorgänge gibt.

Discover-Server-Exemplar (discover_inst)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] ENABLE [ENABLE, DISABLE]

Dieser Parameter gibt an, ob dieses Exemplar von DB2 Discovery erkannt werden kann. Der Standardwert, "enable", gibt an, dass das Exemplar erkannt werden kann, der Wert "disable" dagegen verhindert, dass das Exemplar erkannt wird.

Informationen zu numerischen Äquivalenten und API-Konstanten für diese Werte finden Sie im Handbuch *Administrative API Reference*.

Weitere Informationen zu DB2 Discovery finden Sie im Handbuch *Einstieg* für Ihre Plattform.

Partitionierte Datenbank

Die folgenden Gruppen von Parametern stellen Informationen zum Parallelbetrieb und zu Umgebungen mit partitionierten Datenbanken bereit:

- „Übertragung"
- „Parallelverarbeitung" auf Seite 541.

Übertragung

Die folgenden Parameter stellen Informationen zur Datenfernverarbeitung in der Umgebung mit partitionierten Datenbanken bereit:

- „Antwortzeit für Knotenverbindung (conn_elapse)"
- „Anzahl der FCM-Nachrichtenanker (fcm_num_anchors)" auf Seite 536
- „Anzahl der FCM-Puffer (fcm_num_buffers)" auf Seite 536
- „Anzahl der FCM-Verbindungseinträge (fcm_num_connect)" auf Seite 538
- „Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb)" auf Seite 539
- „Maximale Anzahl Wiederholungen für Knotenverbindungen (max_connretries)" auf Seite 540
- „Maximale Zeitdifferenz zwischen Knoten (max_time_diff)" auf Seite 540
- „db2start/db2stop-Zeitlimit (start_stop_time)" auf Seite 541.

Antwortzeit für Knotenverbindung (conn_elapse)

Konfigurationsart	Datenbankmanager
Gilt für	Partitionierter Datenbank-Server mit lokalen und fernen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	10 [0 – 100]
Maßeinheit	Sekunden
Zugehörige Parameter	„Maximale Anzahl Wiederholungen für Knotenverbindungen (max_connretries)" auf Seite 540

Dieser Parameter gibt an, innerhalb wie viel Sekunden eine TCP/IP-Verbindung zwischen zwei Datenbankpartitions-Servern aufgebaut werden muss. Wird der Verbindungsaufbau innerhalb der angegebenen Zeit erfolgreich durchgeführt, wird der Datenaustausch freigegeben. Wird die Verbindung nicht rechtzeitig aufgebaut, wird ein weiterer Versuch zum Verbindungsaufbau durchgeführt. Kommt innerhalb der im Parameter *max_connretries* angegebenen Anzahl von Neuversuchen keine Verbindung zustande, wird eine Fehlernachricht ausgegeben.

Anzahl der FCM-Nachrichtenanker (fcm_num_anchors)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] -1 [-1, 128 – *fcm_num_rqb*]

Bei nicht partitionierten Datenbanksystemen muss der Parameter *intra_parallel* aktiv sein, damit dieser Parameter verwendet werden kann.

Zugehörige Parameter

- „Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb)" auf Seite 539
- „Partitionsinterne Parallelität aktivieren (intra_parallel)" auf Seite 543

Dieser Parameter gibt die Anzahl der FCM-*Nachrichtenanker* an. Agenten verwenden die Nachrichtenanker, um untereinander Nachrichten weiterzugeben. Der Standardwert (-1) bedeutet 75 Prozent des für *fcm_num_rqb* angegebenen Werts.

Anzahl der FCM-Puffer (fcm_num_buffers)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients

- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart

Konfigurierbar

Standardwert [Bereich]

512, 1 024 oder 4 096 [128 — *fcm_num_rqb*]

- Datenbank-Server mit lokalen und fernen Clients: Der Standardwert ist 1 024
- Datenbank-Server mit lokalen Clients: Der Standardwert ist 512
- Partitionierter Datenbank-Server mit lokalen und fernen Clients: Der Standardwert ist 4 096.

Bei Datenbanksystemen mit einer Partition muss der Parameter *intra_parallel* aktiv sein, damit dieser Parameter verwendet werden kann.

Dieser Parameter gibt die Anzahl von 4-KB-Puffern an, die sowohl zwischen den als auch innerhalb der Datenbankserver für interne Kommunikation (Nachrichten) verwendet werden.

Weitere Informationen zu FCM finden Sie im Abschnitt zum Aktivieren der FCM-Kommunikation im Handbuch *Systemverwaltung: Implementierung*.

Wenn Sie über mehrere logische Knoten auf einem Prozessor verfügen, kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen. Außerdem kann es erforderlich sein, den Wert für diesen Parameter zu erhöhen, wenn wegen der Anzahl von Benutzern im System, wegen der Anzahl von Datenbankpartitions-Servern im System oder wegen der Komplexität der Anwendungen die verfügbaren Nachrichtenpuffer nicht ausreichen.

Wenn Sie auf Nicht-AIX-Systemen mehrere logische Knoten verwenden, wird ein Pool mit der im Parameter *fcm_num_buffers* angegebenen Anzahl von Puffern von allen logischen Knoten auf derselben Maschine gemeinsam benutzt. Unter AIX gilt dagegen folgendes:

- Wenn in dem vom Datenbankmanager verwendeten allgemeinen Speicher genug Speicherplatz vorhanden ist, wird der FCM-Pufferzwischenspeicher von dort zugeordnet. In diesem Fall erhält jeder Datenbankpartitions-Server die im Parameter *fcm_num_buffers* angegebene Anzahl Puffer; d. h. es gibt

keinen von allen Datenbankpartitions-Servern gemeinsam benutzten Pool mit FCM-Puffern (dies war neu in DB2 Version 5).

- Ist in dem vom Datenbankmanager verwendeten allgemeinen Speicher nicht genug Speicherplatz verfügbar, wird der FCM-Pufferzwischenspeicher aus einem separaten Speicherbereich (gemeinsamer AIX-Speicher) zugeordnet, der von allen logischen Knoten auf derselben Maschine gemeinsam benutzt wird. Ein Pool mit der im Parameter *fcm_num_buffers* angegebenen Anzahl von Puffern wird von allen logischen Knoten auf derselben Maschine gemeinsam benutzt. Dies gilt gleichermaßen für Nicht-AIX-Systeme und für DB2 Parallel Edition Version 1.2 unter AIX.

Empfehlung für Anwender von Parallel Edition unter AIX: Wenn Sie mit mehreren logischen Knoten arbeiten, kann der in Parallel Edition Version 1.2 verwendete Wert für *fcm_num_buffers* dazu führen, dass pro Maschine nun bedeutend mehr Speicherplatz belegt wird. Beispiel: Eine Konfiguration mit vier logischen Knoten kann jetzt vier Mal so viele FCM-Puffer wie bisher belegen.

Überprüfen Sie daher den verwendeten Wert. Überlegen Sie, wie viele FCM-Puffer auf der Maschine (oder den Maschinen) mit mehreren logischen Knoten insgesamt zugeordnet werden. Ändern Sie gegebenenfalls den Wert für *fcm_num_buffers*, um das System zu optimieren.

Anzahl der FCM-Verbindungseinträge (fcm_num_connect)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] -1 [-1, 128 — *fcm_num_rqb*]

Bei nicht partitionierten Datenbanksystemen muss der Parameter *intra_parallel* aktiv sein, damit dieser Parameter verwendet werden kann.

Zugehörige Parameter „Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb)" auf Seite 539

Dieser Parameter gibt die Anzahl der FCM-*Verbindungseinträge* an. Agenten verwenden Verbindungseinträge, um Daten untereinander weiterzuleiten. Der Standardwert (-1) bedeutet 75 Prozent des für *fcm_num_rqb* angegebenen Werts.

Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX-Plattformen (32 Bit)
256, 512, 2 048 [128 — 120 000]

UNIX-Plattformen (64 Bit)
256, 512, 2 048 [128 — 524 288]

OS/2 und Windows NT
10 000 [250 — 2 097 152]

- Datenbank-Server mit lokalen und fernen Clients: Der Standardwert ist 512
- Datenbank-Server mit lokalen Clients: Der Standardwert ist 256
- Partitionierter Datenbank-Server mit lokalen und fernen Clients: Der Standardwert ist 2 048.

Bei nicht partitionierten Datenbanksystemen muss der Parameter *intra_parallel* aktiv sein, damit dieser Parameter verwendet werden kann.

Dieser Parameter gibt die Anzahl der FCM-*Anforderungsblöcke* an. Anforderungsblöcke bilden die Methode zur Übermittlung von Informationen zwischen einem FCM-Dämon und einem Agenten oder zwischen Agenten.

Die erforderliche Anzahl von Anforderungsblöcken variiert entsprechend der Anzahl der Benutzer im System, der Anzahl der Datenbankpartitions-Server im System und der Komplexität der ausgeführten Abfragen. Beginnen Sie zunächst mit der Standardanzahl, und verwenden Sie die Ergebnisse des Datenbanksystemmonitors zur Feinabstimmung dieses Parameters.

Maximale Anzahl Wiederholungen für Knotenverbindungen (max_connretries)

Konfigurationsart	Datenbankmanager
Gilt für	Partitionierter Datenbank-Server mit lokalen und fernen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	5 [0 – 100]
Zugehörige Parameter	„Antwortzeit für Knotenverbindung (conn_elapse)" auf Seite 535

Wenn der Verbindungsaufbau zwischen zwei Datenbankpartitions-Servern fehlschlägt (z. B. bei Erreichen des im Parameter *conn_elapse* angegebenen Werts), gibt *max_connretries* an, wie viele Wiederholungen durchgeführt werden dürfen, um die Verbindung zu einem Datenbankpartitions-Server herzustellen. Bei Überschreitung des für diesen Parameter angegebenen Werts wird eine Fehlernachricht zurückgegeben.

Maximale Zeitdifferenz zwischen Knoten (max_time_diff)

Konfigurationsart	Datenbankmanager
Gilt für	Partitionierter Datenbank-Server mit lokalen und fernen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	60 [1–1 440]
Maßeinheit	Minuten

Jeder Datenbankpartitions-Server verfügt über eine eigene Systemuhr. Dieser Parameter gibt die maximale Zeitdifferenz (in Minuten) an, die zwischen den in der Knotenkonfigurationsdatei aufgelisteten Datenbankpartitions-Servern zulässig ist.

Sind einer Transaktion zwei oder mehr Datenbankpartitions-Server zugeordnet, deren Systemzeiten sich um mehr als diese Zeitdifferenz unterscheiden, wird die Transaktion zurückgewiesen, und in der Datei `db2diag.log` wird eine Warnung oder eine Fehlernachricht protokolliert. (Die Transaktion wird nur zurückgewiesen, wenn eine Datenänderung mit ihr verbunden ist.)

DB2 Universal Database Enterprise - Extended Edition verwendet die *Weltzeit*, damit unterschiedliche Zeitzonen beim Einstellen dieses Parameters nicht ins Gewicht fallen. Die Weltzeit ist identisch mit der Westeuropäischen Zeit (Greenwich Mean Time).

db2start/db2stop-Zeitlimit (start_stop_time)

Konfigurationsart	Datenbankmanager
Gilt für	Partitionierter Datenbank-Server mit lokalen und fernen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	10 [1 — 1 440]
Maßeinheit	Minuten

Dieser Parameter gilt nur in einer Umgebung mit partitionierten Datenbanken. Er gibt die Zeit (in Minuten) an, innerhalb der alle Datenbankpartitions-Server auf den Befehl DB2START oder DB2STOP reagieren müssen. Außerdem wird er als Zeitlimit für ADDNODE-Operationen verwendet.

Datenbankpartitions-Server, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2START reagieren, senden eine Nachricht an das Fehlerprotokoll `db2start`, das sich im Unterverzeichnis `log` des Unterverzeichnisses `sqllib` im Benutzerverzeichnis für das Exemplar befindet. Vor dem Neustart dieser Knoten sollten Sie auf diesen Knoten einen Befehl DB2STOP absetzen.

Datenbankpartitions-Server, die nicht innerhalb der angegebenen Zeit auf einen Befehl DB2STOP reagieren, senden eine Nachricht an das Fehlerprotokoll `db2stop`, das sich im Unterverzeichnis `log` des Unterverzeichnisses `sqllib` im Benutzerverzeichnis für das Exemplar befindet. Sie können DB2STOP entweder für jeden oder einmal für alle nicht reagierenden Datenbankpartitions-Server absetzen. (Die bereits gestoppten Server melden, dass sie bereits gestoppt sind.)

Parallelverarbeitung

Die folgenden Parameter stellen Informationen zur Parallelverarbeitung bereit:

- „Maximaler Grad der Parallelität bei Abfragen (max_querydegree)"
- „Partitionsinterne Parallelität aktivieren (intra_parallel)" auf Seite 543.

Maximaler Grad der Parallelität bei Abfragen (max_querydegree)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients

- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] -1 (ANY) [ANY, 1 — 32 767] (ANY bedeutet vom System ermittelt)

Zugehörige Parameter

- „Grad der Parallelität (dft_degree)" auf Seite 516
- „Partitionsinterne Parallelität aktivieren (intra_parallel)" auf Seite 543

Dieser Parameter gibt den maximalen Grad partitionsinterner Parallelität an, der für SQL-Anweisungen verwendet wird, die auf diesem Exemplar des Datenbankmanagers ausgeführt werden. Eine SQL-Anweisung verwendet bei ihrer Ausführung innerhalb einer Partition nicht mehr als diese Anzahl paralleler Operationen. Der Konfigurationsparameter *intra_parallel* muss aktiviert sein, damit die Datenbankpartition die partitionsinterne Parallelität verwenden kann.

Der Standardwert für diesen Konfigurationsparameter lautet -1. Dieser Wert bedeutet, dass das System den vom Optimierungsprogramm festgelegten Parallelitätsgrad verwendet. Andernfalls wird der vom Benutzer angegebene Wert verwendet.

Anmerkung: Der Grad der Parallelität für eine SQL-Anweisung kann bei der Kompilierung der Anweisung mit Hilfe des Sonderregisters CURRENT DEGREE oder der Bindeoption DEGREE angegeben werden.

Der maximale Grad der Parallelität für eine aktive Anwendung bei Abfragen kann mit dem Befehl SET RUNTIME DEGREE geändert werden. Der zur Laufzeit tatsächlich verwendete Parallelitätsgrad ist der niedrigste der folgenden Werte:
- Konfigurationsparameter *max_querydegree*
- Parallelitätsgrad der Anwendung zur Laufzeit
- Parallelitätsgrad bei der Kompilierung der SQL-Anweisung

Eine Ausnahmesituation bezüglich der Festlegung des tatsächlichen Parallelitätsgrads bei Abfragen tritt beim Erstellen eines Index auf. Ist in diesem Fall die Parallelität *intra_parallel* aktiviert und die Tabelle so groß, dass die Verwendung mehrerer Prozessoren vorteilhaft wäre, wird beim Erstellen

eines Index die Anzahl der Online-Prozessoren (maximal jedoch 6) plus 1 Prozessor verwendet. Die Angabe des anderen oben genannten Parameters, der Bindeoption oder des Sonderregisters hat keine Auswirkung.

Partitionsinterne Parallelität aktivieren (intra_parallel)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] NO (0) [SYSTEM (-1), NO (0), YES (1)]

Der Wert -1 bewirkt, dass der Parameter auf „YES" oder „NO" gesetzt wird, abhängig von der Hardware, auf welcher der Datenbankmanager ausgeführt wird.

Zugehörige Parameter „Maximaler Grad der Parallelität bei Abfragen (max_querydegree)" auf Seite 541

Dieser Parameter gibt an, ob der Datenbankmanager partitionsinterne Parallelität verwenden kann.

Ist dieser Parameter auf "YES" gesetzt, können die durch die Parallelverarbeitung erreichten Leistungssteigerungen unter anderem bei Datenbankabfragen und der Indexerstellung genutzt werden.

Anmerkung: Wenn Sie diesen Parameterwert ändern, werden Pakete möglicherweise erneut an die Datenbank gebunden. In diesem Fall kann es beim erneuten Binden zu einer Verschlechterung der Leistung kommen.

Exemplarverwaltung

Eine Reihe von Parametern steht zur Verwaltung der Datenbankmanager-exemplare zur Verfügung. Diese Parameter werden in folgende Kategorien unterteilt:

- „Diagnose"
- „Parameter des Datenbanksystemmonitors" auf Seite 547
- „Systemverwaltung" auf Seite 548
- „Exemplarverwaltung" auf Seite 557

Diagnose

Mit den folgenden Parametern können die Diagnoseinformationen, die vom Datenbankmanager verfügbar gemacht werden, gesteuert werden:

- „Aufzeichnungsebene bei Fehlerdiagnose (diaglevel)"
- „Verzeichnispfad für Diagnosedaten (diagpath)" auf Seite 545
- „Aufzeichnungsebene (notifylevel)" auf Seite 546.

Aufzeichnungsebene bei Fehlerdiagnose (diaglevel)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] 3 [0 — 4]

Zugehörige Parameter „Verzeichnispfad für Diagnosedaten (diagpath)" auf Seite 545

Durch diesen Parameter wird die Art der Diagnosefehler festgelegt, die in der Datei `db2diag.log` aufgezeichnet werden. Gültige Werte:

0 – Keine Aufzeichnung von Diagnosedaten

1 – Nur schwerwiegende Fehler

2 – Alle Fehler

3 – Alle Fehler und Warnungen

4 – Alle Fehler, Warnungen und Informationsnachrichten

Der Konfigurationsparameter *diagpath* wird zur Angabe des Verzeichnisses verwendet, in dem sich die Fehlerdatei, die Ereignisprotokolldatei (nur unter Windows NT), die Alert-Protokolldatei und alle Speicherauszugsdateien befinden, die je nach Wert des Parameters *diaglevel* generiert werden.

Empfehlung: Sie können den Wert dieses Parameters erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können.

Verzeichnispfad für Diagnosedaten (diagpath)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [beliebiger gültiger Pfadname]

Zugehörige Parameter „Aufzeichnungsebene bei Fehlerdiagnose (diaglevel)" auf Seite 544

Mit diesem Parameter können Sie einen vollständig qualifizierten Pfad für DB2-Diagnosedaten angeben. Im angegebenen Verzeichnis können abhängig von Ihrer Plattform Speicherauszugsdateien, Alarmnachrichtendateien (Trap Files), eine Fehlerprotokolldatei und eine Alert-Protokolldatei gespeichert werden.

Wenn dieser Parameter den Wert "Null" hat, werden die Diagnoseinformationen in Dateien eines der folgenden Verzeichnisse (bzw. Ordner) geschrieben:

- Für OS/2- und unterstützte Windows-Umgebungen:
  - Wenn die Umgebungsvariable bzw. das Schlüsselwort DB2INSTPROF **nicht** festgelegt ist, werden die Informationen in `x:\SQLLIB\DB2INSTANCE` geschrieben. Dabei ist `x:\SQLLIB` die Laufwerkangabe und das Verzeichnis, die in der Umgebungsvariablen bzw. dem Schlüsselwort DB2PATH angegeben sind und `DB2INSTANCE` ist der Name des Exemplareigners.

    Anmerkung: Das Verzeichnis muss nicht SQLLIB heißen.

  - Wenn die Umgebungsvariable bzw. das Schlüsselwort DB2INSTPROF festgelegt ist, werden die Informationen in `x:\DB2INSTPROF\DB2INSTANCE` geschrieben. Dabei ist DB2INSTPROF der Name des Exemplarprofilverzeichnisses und DB2INSTANCE der Name des Exemplareigners.
- In auf UNIX basierenden Umgebungen: `INSTHOME/sqllib/db2dump`, wobei `INSTHOME` das Ausgangsverzeichnis (Home) des Exemplareigners ist.
- In Macintosh Umgebungen: DB2-Ordner

Empfehlung: Verwenden Sie den Standardwert, oder geben Sie eine zentrale Speicherposition für die Diagnosedaten mehrerer Exemplare an.

In einer Umgebung mit partitionierten Datenbanken muss der angegebene Pfad auf einem gemeinsam benutzten Dateisystem angelegt sein.

Aufzeichnungsebene (notifylevel)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients unter Windows NT
- Client unter Windows NT
- Datenbank-Server mit lokalen Clients unter Windows NT
- Partitionierter Datenbank-Server mit lokalen und fernen Clients unter Windows NT
- Satellitendatenbank-Server mit lokalen Clients unter Windows 95, Windows 98 und Windows NT

Parameterart Konfigurierbar

Standardwert [Bereich] 2 [0 — 4]

Dieser Parameter gibt die Art von Fehlernachrichten mit Verwaltungshinweisen an, die in eine Datei geschrieben werden. Bei einem Server der Satellitenknotenart werden Fehler in die Aufzeichnungsdatei *instance*`.nfy` geschrieben. Bei allen anderen Knotenarten ist dieser Parameter nur auf der Windows NT-Plattform verfügbar, und Fehler werden in das Windows NT-Ereignisprotokoll geschrieben. Die Fehler können von DB2, den Programmen Capture und Apply sowie Benutzeranwendungen geschrieben werden.

Gültige Werte für diesen Parameter sind:

0 — Keine Aufzeichnung von Diagnosedaten

1 — Nur schwerwiegende Fehler

2 — Alle Fehler

3 — Alle Fehler und Warnungen

4 — Alle Fehler, Warnungen und Informationsnachrichten

Eine Benutzeranwendung muss die API db2AdminMsgWrite aufrufen, um in die Aufzeichnungsdatei oder das Windows NT-Ereignisprotokoll schreiben zu können. Weitere Informationen zu dieser API finden Sie im Handbuch *Administrative API Reference*.

Empfehlung: Sie können den Wert dieses Parameters erhöhen, um zusätzliche Fehlerbestimmungsdaten zu sammeln, die zur Lösung eines Problems beitragen können.

Parameter des Datenbanksystemmonitors

Mit den folgenden Parametern können verschiedene Funktionen des Datenbanksystemmonitors gesteuert werden:

- „Standardschalter für den Datenbanksystemmonitor (dft_monswitches)"

Standardschalter für den Datenbanksystemmonitor (dft_monswitches)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert Alle Schalter inaktiviert

Dieser Parameter ist insofern außergewöhnlich, als Sie mit ihm eine Reihe von Schaltern einstellen können, die intern jeweils als ein Bit des Parameters dargestellt werden. Je nachdem, welche Schnittstelle Sie verwenden, um die Konfiguration des Datenbankmanagers zu aktualisieren, können Sie diesen Parameter eventuell direkt ändern. Sie können auch jeden dieser Schalter unabhängig aktualisieren, indem Sie die folgenden Parameter einstellen:

dft_mon_uow Standardwert des UOW-Schalters (Unit of Work - Arbeitseinheit) von Snapshot Monitor

dft_mon_stmt	Standardwert des Schalters für SQL-Anweisungen von Snapshot Monitor
dft_mon_table	Standardwert des Schalters für Tabellen von Snapshot Monitor
dft_mon_bufpool	Standardwert des Schalters für Pufferpool von Snapshot Monitor
dft_mon_lock	Standardwert des Schalters für Sperren von Snapshot Monitor
dft_mon_sort	Standardwert des Schalters für Sortiervorgänge von Snapshot Monitor

Änderungen an diesen Schaltern von Datenbanksystemmonitor werden sofort wirksam, d. h. Sie brauchen den Datenbankmanager nicht zu stoppen und erneut zu starten.

Anmerkung: Eine vorhandene Monitoranwendung verwendet nicht automatisch den neuen Standardwert eines Schalters. Damit der neue Wert (oder die neuen Werte) verwendet werden, muss die Anwendung beendet und erneut mit dem Exemplar verbunden werden.

Weitere Informationen zu Snapshot Monitor und zur Verwendung der Monitorschalter finden Sie im Handbuch *System Monitor Guide and Reference*.

Empfehlung: Jeder auf ON gesetzte (aktivierte) Schalter weist den Datenbankmanager an, die zu diesem Schalter gehörenden Monitordaten zu sammeln. Das Einholen zusätzlicher Monitordaten erhöht den Systemaufwand des Datenbankmanagers, wodurch die Systemleistung beeinträchtigt werden kann.

Alle Überwachungsanwendungen erhalten diese Standardeinstellungen für die Schalter, wenn die Anwendung die erste Überwachungsanforderung absetzt (z. B. einen Schalter einstellt, den Ereignismonitor aktiviert, eine Momentaufnahme macht). In der Konfigurationsdatei sollten Sie einen Schalter nur dann aktivieren, wenn Sie Daten sammeln möchten, sobald der Datenbankmanager gestartet wird. (Andernfalls kann jede Überwachungsanwendung ihre eigenen Schalter einstellen, und die gesammelten Daten beziehen sich dann auf den Zeitpunkt, zu dem die Schalter eingestellt wurden.)

Systemverwaltung

Die folgenden Parameter beziehen sich auf die Systemverwaltung:

- „Übertragungsbandbreite (comm_bandwidth)" auf Seite 549
- „CPU-Geschwindigkeit (cpuspeed)" auf Seite 549
- „Maximale Anzahl gleichzeitig aktiver Datenbanken (numdb)" auf Seite 550

- „Name des TP-Monitors (tp_mon_name)" auf Seite 552
- „Knotenart des Systems (nodetype)" auf Seite 554
- „Standardzeichenfolge für Abrechnung (dft_account_str)" auf Seite 555
- „Java Development Kit 1.1: Installationspfad (jdk11_path)" auf Seite 556
- „Unterstützung für Systeme mit zusammengeschlossenen Datenbanken (federated)" auf Seite 556.

Übertragungsbandbreite (comm_bandwidth)

Konfigurationsart	Datenbankmanager
Gilt für	Partitionierter Datenbank-Server mit lokalen und fernen Clients
Parameterart	Konfigurierbar
Standardwert [Bereich]	-1 [.1 – 100 000] Durch den Wert -1 wird der Parameter auf den Standardwert zurückgesetzt. Der Standardwert wird abhängig davon berechnet, ob ein Hochgeschwindigkeitsschalter verwendet wird.
Maßeinheit	Megabyte pro Sekunde

Der Wert, der (in MB pro Sekunde) für die Übertragungsbandbreite berechnet wird, wird vom SQL-Optimierungsprogramm zur Abschätzung des Aufwands für bestimmte Operationen zwischen den Datenbankpartitions-Servern eines partitionierten Datenbanksystems herangezogen. Das Optimierungsprogramm modelliert nicht die Kosten der Datenfernverarbeitung zwischen einem Client und einem Server. Dieser Parameter sollte daher nur die nominale Bandbreite zwischen den Datenbankpartitions-Servern darstellen.

Sie können diesen Wert explizit festlegen, um ein Modell einer Produktionsumgebung auf Ihrem Testsystem zu erstellen oder die Auswirkungen einer Hardwareaufrüstung zu bewerten.

Empfehlung: Sie sollten diesen Parameter nur anpassen, wenn Sie ein Modell einer anderen Umgebung erstellen wollen.

Die Übertragungsbandbreite wird vom Optimierungsprogramm bei der Bestimmung der Zugriffspfade verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

CPU-Geschwindigkeit (cpuspeed)

Konfigurationsart	Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] -1 [1^{-10} — 1] Der Wert -1 gibt an, dass der Wert dieses Parameters gemäß den Ergebnissen eines Messprogramms neu eingestellt wird.

Maßeinheit Sekunden

Die CPU-Geschwindigkeit (in Millisekunden pro Instruktion) wird vom SQL-Optimierungsprogramm verwendet, um den Aufwand für die Ausführung bestimmter Operationen zu berechnen. Der Wert dieses Parameters wird automatisch bei der Installation des Datenbankmanagers auf der Grundlage der Ausgabe eines Programms zum Messen der CPU-Geschwindigkeit festgelegt. Dieses Programm wird ausgeführt, wenn Benchmark-Ergebnisse aus einem der folgenden Gründe nicht verfügbar sind:

- Auf der Plattform wird die Datei `db2spec.dat` nicht unterstützt.
- Die Datei `db2spec.dat` wird **nicht** gefunden.
- Die Daten für IBM RISC System/6000, Modell 530H werden in der Datei nicht gefunden.
- Die Daten für Ihr System werden in der Datei nicht gefunden.

Sie können diesen Wert explizit festlegen, um ein Modell einer Produktionsumgebung auf Ihrem Testsystem zu erstellen oder die Auswirkungen einer Hardwareaufrüstung zu bewerten. Wenn der Wert auf -1 gesetzt wird, wird *cpuspeed* erneut berechnet.

Empfehlung: Sie sollten diesen Parameter nur anpassen, wenn Sie ein Modell einer anderen Umgebung erstellen wollen.

Der Wert für die CPU-Geschwindigkeit wird vom Optimierungsprogramm bei der Bestimmung von Zugriffspfaden verwendet. Wenn Sie diesen Parameter geändert haben, sollten Sie Anwendungen eventuell erneut binden (mit dem Befehl REBIND PACKAGE).

Maximale Anzahl gleichzeitig aktiver Datenbanken (numdb)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX 8 [1 — 256]

OS/2 und Windows NT Datenbank-Server mit lokalen und fernen Clients
8 [1 — 256]

OS/2 und Windows NT Datenbank-Server mit lokalen Clients und Satellitendatenbank-Server mit lokalen Clients
3 [1 — 256]

Maßeinheit Zähler

Mit diesem Parameter wird die Anzahl der lokalen Datenbanken angegeben, die gleichzeitig aktiv sein können (d. h. auf die von Anwendungen zugegriffen werden kann). In einer Umgebung mit partitionierten Datenbanken beschränkt dieser Parameter die Anzahl aktiver Datenbankpartitionen auf einem Datenbankpartitions-Server unabhängig davon, ob dieser Server der Koordinatorknoten für die Anwendung ist oder nicht.

Da jede Datenbank Speicher belegt und eine aktive Datenbank ein neues Segment des gemeinsam benutzten Speichers verwendet, können Sie den Bedarf an Systemressourcen verringern, indem Sie die Anzahl der getrennten Datenbanken auf Ihrem System begrenzen. Jedoch ist es nicht sinnvoll, die Anzahl der Datenbanken ohne weitere Überlegungen einfach zu reduzieren. Das heißt, alle Daten ohne Rücksicht auf ihre Zusammengehörigkeit in einer Datenbank zusammenzufassen, senkt zwar den Plattenspeicherbedarf, ist letztendlich jedoch nicht unbedingt zweckmäßig. Im allgemeinen ist es sinnvoll, nur funktionell zusammengehörige Daten in derselben Datenbank zu speichern.

Empfehlung: In der Regel ist es am besten, diesen Wert auf die tatsächliche Anzahl der bereits für den Datenbankmanager definierten Datenbanken zu setzen und um einen angemessenen Wert zu erhöhen, um für zukünftige Datenbanken, die kurzfristig (z. B. innerhalb von 6 - 12 Monaten) hinzugefügt werden, noch Spielraum zu haben. Der tatsächliche, über der eigentlichen

Anzahl der Datenbanken liegende Wert sollte nicht allzu groß sein, aber er sollte das Hinzufügen neuer Datenbanken erlauben, ohne dass dieser Parameter häufig aktualisiert werden muss.

Eine Änderung des Parameters *numdb* kann Auswirkungen auf die Gesamtmenge an Speicher haben, der zugeordnet wird. Daher ist es nicht empfehlenswert, diesen Parameter häufig zu aktualisieren. Bei der Aktualisierung dieses Parameters sollten Sie die anderen Konfigurationsparameter beachten, die die Speicherzuordnung für eine Datenbank oder für eine mit der Datenbank verbundene Anwendung beeinflussen:

- „Größe des Pufferpools (buffpage)" auf Seite 404
- „Maximaler Speicher für Sperrenliste (locklist)" auf Seite 415
- „Zwischenspeicher für Anwendungen (applheapsz)" auf Seite 426
- „Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz)" auf Seite 420
- „Zwischenspeicher für Sortierlisten (sortheap)" auf Seite 422
- „SQL-Anweisungszwischenspeicher (stmtheap)" auf Seite 425
- „Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz)" auf Seite 436
- „Zwischenspeicher für Datenbank (dbheap)" auf Seite 408
- „Zwischenspeicher für den Datenbanksystemmonitor (mon_heap_sz)" auf Seite 443
- „Größe des Statistikzwischenspeichers (stat_heap_sz)" auf Seite 427

Name des TP-Monitors (tp_mon_name)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert Kein Standardwert

Gültige Werte

- CICS
- MQ
- ENCINA
- CB

- SF
- TUXEDO
- TOPEND
- Kein Eintrag oder ein weiterer Wert (unter UNIX, OS/2 und Windows NT; keine weiteren möglichen Werte unter Solaris oder SINIX)

Mit diesem Parameter wird der Name des verwendeten Monitors für das Transaktionsprogramm (TP) angegeben.

- Sind Anwendungen in einer WebSphere Enterprise Edition CICS-Umgebung aktiv, muss für den Parameter „CICS" angegeben werden.
- Sind Anwendungen in einer WebSphere Enterprise Edition Encina-Umgebung aktiv, muss für den Parameter „ENCINA" angegeben werden.
- Sind Anwendungen in einer WebSphere Enterprise Edition Component Broker-Umgebung aktiv, muss für den Parameter „CB" angegeben werden.
- Sind Anwendungen in einer IBM MQSeries-Umgebung aktiv, muss für den Parameter „MQ" angegeben werden.
- Sind Anwendungen in einer BEA Tuxedo-Umgebung aktiv, muss für den Parameter „TUXEDO" angegeben werden.
- Sind Anwendungen in einer IBM San Francisco-Umgebung aktiv, muss für den Parameter „SF" angegeben werden.

Benutzer von **IBM WebSphere EJB** und **Microsoft Transaction Server** müssen für diesen Parameter keinen Wert konfigurieren.

Wenn keines der oben genannten Produkte verwendet wird, sollte dieser Parameter nicht konfiguriert, sondern ohne Angabe eines Werts belassen werden.

In vorangehenden Versionen von DB2 Universal Database in den Umgebungen OS/2 und Windows NT enthielt dieser Parameter den Pfad und den Namen der DLL, in der die Funktionen *ax_reg* und *ax_unreg* des XA-Transaktionsmanagers gespeichert waren. Dieses Format wird noch immer unterstützt. Wenn der Wert für diesen Parameter mit keinem der oben genannten TP-Monitornamen übereinstimmt, wird angenommen, dass der angegebene Wert der Name einer Bibliothek ist, die die Funktionen *ax_reg* und *ax_unreg* enthält. Dies gilt für die Umgebungen UNIX, OS/2 und Windows NT.

Benutzer von TXSeries CICS und Encina: In vorangegangenen Versionen dieses Produkts unter OS/2 und Windows NT war es erforderlich, diesen Parameter als „libEncServer:C" oder „libEncServer:E" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „CICS" oder „ENCINA" konfiguriert wird, ist dies ausreichend.

Benutzer von MQSeries: In vorangegangenen Versionen dieses Produkts unter OS/2 und Windows NT war es erforderlich, diesen Parameter als „mqmax" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „MQ" konfiguriert wird, ist dies ausreichend.

Benutzer von Component Broker: In vorangegangenen Versionen dieses Produkts unter OS/2 und Windows NT war es erforderlich, diesen Parameter als „somtrx1i" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „CB" konfiguriert wird, ist dies ausreichend.

Benutzer von San Francisco: In vorangegangenen Versionen dieses Produkts unter OS/2 und Windows NT war es erforderlich, diesen Parameter als „ibmsfDB2" zu konfigurieren. Dies wird noch immer unterstützt, ist aber nicht mehr erforderlich. Wenn der Parameter als „SF" konfiguriert wird, ist dies ausreichend.

Die maximale Länge der Zeichenfolge, die für diesen Parameter angegeben werden kann, beträgt 19 Zeichen.

Diese Informationen können auch in der Zeichenfolge XA OPEN von DB2 Universal Database konfiguriert werden. Wenn mehrere TP-Monitore in einem einzigen DB2-Exemplar verwendet werden, muss diese Funktion verwendet werden. Zusätzliche Informationen zur Verwendung der Zeichenfolge XA OPEN finden Sie im Handbuch *Systemverwaltung: Konzept*.

Knotenart des Systems (nodetype)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Informativ

Dieser Parameter liefert Informationen zu den DB2-Produkten, die auf Ihrem System installiert sind, und daher auch zur Art Ihrer Datenbankmanagerkonfiguration. Die folgenden Werte können von diesem Parameter zurückgegeben werden, und dieser Knotenart können die folgenden Produkte zugeordnet sein:

- **Datenbank-Server mit lokalen und fernen Clients** – Ein DB2-Server-Produkt, das lokale und ferne Datenbank-Clients unterstützt und auf andere ferne Datenbank-Server zugreifen kann.
- **Client** – Ein Datenbank-Client, mit dem auf ferne Datenbank-Server zugegriffen werden kann.
- **Datenbank-Server mit lokalen Clients** – Ein Verwaltungssystem für relationale DB2-Datenbanken, das lokale Datenbank-Clients unterstützt und auf andere ferne Datenbank-Server zugreifen kann.
- **Partitionierter Datenbank-Server mit lokalen und fernen Clients** – Ein DB2-Server-Produkt, das lokale und ferne Datenbank-Clients unterstützt, auf andere ferne Datenbank-Server zugreifen kann und Partitionsparallelität unterstützt.
- **Satellitendatenbank-Server mit lokalen Clients** Ein Verwaltungssystem für relationale DB2-Datenbanken, das lokale Datenbank-Clients unterstützt und auf andere ferne Datenbank-Server zugreifen kann.

Informationen zu numerischen Äquivalenten und API-Konstanten für diese Werte finden Sie im Handbuch *Administrative API Reference*.

Standardzeichenfolge für Abrechnung (dft_account_str)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [beliebige gültige Zeichenfolge]

Bei jeder Verbindungsanforderung durch eine Anwendung wird eine Abrechnungszeichenfolge vom Anwendungs-Requester an einen DRDA-Anwendungs-Server gesendet, die aus einem von DB2 Connect erstellten Präfix und einem vom Benutzer eingegebenen Suffix besteht. Anhand dieser Abrechnungsdaten kann ein Systemadministrator die Ressourcennutzung für jeden Benutzerzugriff bestimmen.

Anmerkung: Dieser Parameter ist nur für DB2 Connect gültig.

Das Suffix wird vom Anwendungsprogramm, das die API `sqlesact()` aufruft, oder dem Benutzer, der die Umgebungsvariable DB2ACCOUNT definiert, übergeben. Wenn von der API oder der Umgebungsvariablen kein Suffix bereitgestellt wird, verwendet DB2 Connect den Wert dieses Parameters als Standardsuffix. Dieser Parameter ist insbesondere für Datenbank-Clients früherer Versionen (alle vor Version 2) nützlich, die nicht über Funktionen zum Senden einer Abrechnungszeichenfolge an DB2 Connect verfügen.

Empfehlung: Verwenden Sie in der Abrechnungszeichenfolge folgende Zeichen:
- Alphabetische Zeichen (A - Z)
- Numerische Zeichen (0 - 9)
- Unterstreichungszeichen (_).

Java Development Kit 1.1: Installationspfad (jdk11_path)

Konfigurationsart Datenbankmanager

Gilt für
- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] NULL [gültiger Pfad]

Zugehörige Parameter
- „Maximaler Zwischenspeicher für Java-Interpreter (java_heap_sz)" auf Seite 447

Dieser Parameter gibt das Verzeichnis an, in dem Java Development Kit 1.1 installiert ist. CLASSPATH und andere vom Java-Interpreter verwendete Umgebungsvariablen werden aus dem Wert dieses Parameters errechnet.

Da es für diesen Parameter keinen Standardwert gibt, sollten Sie beim Installieren des Java Development Kit einen Wert für diesen Parameter angeben.

Unterstützung für Systeme mit zusammengeschlossenen Datenbanken (federated)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] No [Yes; No]

Dieser Parameter aktiviert bzw. inaktiviert Unterstützung für Anwendungen, die verteilte Anforderungen für von Datenquellen (wie die DB2-Produktfamilie und Oracle) verwaltete Daten übergeben.

Exemplarverwaltung

Die folgenden Parameter beziehen sich auf die Sicherheit und Verwaltung des Datenbankmanagerexemplars:

- „SYSADM-Gruppenname (sysadm_group)"
- „SYSCTRL-Gruppenname (sysctrl_group)" auf Seite 559
- „SYSMAINT-Gruppenname (sysmaint_group)" auf Seite 560
- „Authentifizierungsart (authentication)" auf Seite 561
- „Katalogisieren ohne Berechtigung zulässig (catalog_noauth)" auf Seite 563
- „Standarddatenbankpfad (dftdbpath)" auf Seite 563
- „LOGON für DB2START/DB2STOP erforderlich (ss_logon)" auf Seite 565
- „Alle Clients akzeptieren (trust_allclnts)" auf Seite 565
- „Identifikationsüberprüfung für gesicherte Clients (trust_clntauth)" auf Seite 566

SYSADM-Gruppenname (sysadm_group)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert Null

Zugehörige Parameter

- „SYSCTRL-Gruppenname (sysctrl_group)" auf Seite 559
- „SYSMAINT-Gruppenname (sysmaint_group)" auf Seite 560

Die Berechtigung SYSADM (Systemadministrator) stellt die höchste Ebene der Berechtigungen innerhalb des Datenbankmanagers dar, von der aus sämtliche Datenbankobjekte gesteuert werden. Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSADM für das Datenbankmanagerexemplar besitzt.

Die Berechtigung SYSADM wird von den Sicherheitseinrichtungen festgelegt, die in einer bestimmten Betriebsumgebung verwendet werden.

- Unter Windows 95 oder Windows 98 muss für die SYSADM-Gruppe der Wert "NULL" angegeben werden.

  Dieser Parameter muss für Clients unter Windows 95 oder Windows 98 bei Verwendung von Systemschutz den Wert „NULL" haben, weil unter dem Betriebssystem Windows 95 oder Windows 98 keine Gruppeninformationen gespeichert werden und somit keine Möglichkeit zur Überprüfung besteht, ob ein Benutzer einer bestimmten SYSADM-Gruppe angehört. Bei Angabe eines Gruppennamens kann keiner der Benutzer dieser Gruppe angehören.
- Unter den Betriebssystemen Windows NT und Windows 2000 kann dieser Parameter auf jede lokale Gruppe gesetzt werden, deren Name aus 8 oder weniger Zeichen besteht und in der Sicherheitsdatenbank von Windows NT und Windows 2000 definiert ist. Wenn für diesen Parameter der Wert „NULL" definiert wird, haben alle Mitglieder der Administratorengruppe die Berechtigung SYSADM.
- Wenn bei auf UNIX basierenden Systemen der Wert „NULL" für diesen Parameter angegeben wird, gilt die Primärgruppe des Exemplareigners standardmäßig als SYSADM-Gruppe.

  Ist der Wert nicht „NULL", kann als SYSADM-Gruppe jeder gültige UNIX-Gruppenname definiert werden.
- Unter OS/2 haben die in der Benutzerprofilverwaltung (User Profile Management, UPM) als Administratoren definierten Benutzer die Berechtigung SYSADM, wenn für diesen Parameter der Wert „NULL" angegeben wird.

  Wenn für diesen Parameter ein Gruppenname angegeben wird, haben nur Benutzer, die dieser Gruppe angehören, die Berechtigung SYSADM. Es kann jede beliebige Gruppe der UPM-Gruppen (UPM - User Profile Management, Benutzerprofilverwaltung) angegeben werden.

Wenn bei Verwendung der DCE-Sicherheit der Wert „NULL" für den Parameter *sysadm_group* definiert ist, wird der DCE-Standardgruppenname

DB2ADMIN verwendet. Es muss bereits ein gültiger DCE-Principal mit der Berechtigung DB2ADMIN vorhanden sein. Sie können einen anderen Gruppennamen angeben.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSADM_GROUP NULL aus. Das Schlüsselwort „NULL" muss in Großbuchstaben angegeben werden. Sie können auch das Notizbuch **Exemplar konfigurieren** über die DB2-Steuerzentrale aufrufen und den Parameter über das Notizbuch definieren.

SYSCTRL-Gruppenname (sysctrl_group)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert Null

Zugehörige Parameter

- „SYSADM-Gruppenname (sysadm_group)" auf Seite 557
- „SYSMAINT-Gruppenname (sysmaint_group)" auf Seite 560

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSCTRL (Systemsteuerung) besitzt. Die Berechtigung SYSCTRL umfasst Zugriffsrechte, die sich auf Systemressourcen auswirkende Operationen ermöglichen, aber keinen direkten Zugriff auf Daten zulassen.

Achtung: Dieser Parameter muss für Clients unter Windows 95 und Windows 98 den Wert NULL besitzen, wenn Systemschutz verwendet wird (d. h., die Authentifizierung ist CLIENT, SERVER, DCS oder eine andere gültige Authentifizierung). Der Grund hierfür ist, dass die Betriebssysteme Windows 95 und Windows 98 keine Gruppeninformationen speichern und somit keine Möglichkeit bieten, festzustellen, ob ein Benutzer ein Mitglied einer bestimmten SYSCTRL-Gruppe ist. Bei Angabe eines Gruppennamens kann keiner der Benutzer

dieser Gruppe angehören. Dies gilt nicht, wenn die Identifikationsüberprüfung DCE verwendet wird. In diesem Fall dürfen Gruppennamen angegeben werden.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSCTRL_GROUP NULL aus. Das Schlüsselwort „NULL" muss in Großbuchstaben angegeben werden. Sie können auch das Notizbuch **Exemplar konfigurieren** über die DB2-Steuerzentrale aufrufen und den Parameter über das Notizbuch definieren.

SYSMAINT-Gruppenname (sysmaint_group)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert Null

Zugehörige Parameter

- „SYSADM-Gruppenname (sysadm_group)" auf Seite 557
- „SYSCTRL-Gruppenname (sysctrl_group)" auf Seite 559

Mit diesem Parameter wird der Gruppenname definiert, der die Berechtigung SYSMAINT (Systempflege) besitzt. Mit der Berechtigung SYSMAINT können Operationen zur Pflege aller Datenbanken, die zu einem Exemplar gehören, durchgeführt werden, jedoch ohne direkten Zugriff auf Daten.

Achtung: Dieser Parameter muss für Clients unter Windows 95 und Windows 98 den Wert NULL besitzen, wenn Systemschutz verwendet wird (d. h., die Authentifizierung ist CLIENT, SERVER, DCS oder eine andere gültige Authentifizierung). Der Grund hierfür ist, dass die Betriebssysteme Windows 95 und Windows 98 keine Gruppeninformationen speichern und somit keine Möglichkeit bieten, festzustellen, ob ein Benutzer ein Mitglied einer bestimmten SYSMAINT-Gruppe ist. Bei Angabe eines Gruppennamens kann keiner der Benut-

zer dieser Gruppe angehören. Dies gilt nicht, wenn die Identifikationsüberprüfung DCE verwendet wird. In diesem Fall dürfen Gruppennamen angegeben werden.

Wenn Sie den Parameter auf seinen Standardwert (NULL) zurücksetzen wollen, führen Sie den Befehl UPDATE DBM CFG USING SYSMAINT_GROUP NULL aus. Das Schlüsselwort „NULL" muss in Großbuchstaben angegeben werden. Sie können auch das Notizbuch **Exemplar konfigurieren** über die DB2-Steuerzentrale aufrufen und den Parameter über das Notizbuch definieren.

Authentifizierungsart (authentication)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DCS; DCS_ENCRYPT; DCE; DCE_SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT]

Mit diesem Parameter wird festgelegt, wie und wo die Authentifizierung eines Benutzers stattfindet.

Wird als Wert SERVER angegeben, werden die Benutzer-ID und das Kennwort vom Client an den Server gesendet, damit die Authentifizierung auf dem Server ausgeführt werden kann. Der Wert SERVER_ENCRYPT unterscheidet sich vom Wert SERVER nur darin, dass über das Netzwerk gesendete Kennwörter verschlüsselt werden.

Der Wert CLIENT gibt an, dass alle Authentifizierungen auf dem Client stattfinden, so dass auf dem Server keine Authentifizierung mehr ausgeführt werden muss.

Der Wert DCS gibt an, dass die Authentifizierung auf dem Host oder System IBM AS/400 stattfindet. Der Wert DCS_ENCRYPT unterscheidet sich vom

Wert DCS nur darin, dass über das Netzwerk gesendete Kennwörter verschlüsselt werden. Wenn Sie APPC und ein Kommunikationsprogramm verwenden, das das Kennwort des Clients nicht an den DB2-Server weitergibt, können Sie DCS angeben, um Folgendes zu erzielen:
- Authentifizierung der Art SERVER für Nicht-DRDA-Clients
- Authentifizierung der Art CLIENT für DRDA-Clients

Der Wert DCE bedeutet, dass die Authentifizierung auf dem DCE-Server mit Hilfe der DCE-Sicherheitsservices durchgeführt wird. Der Wert DCE_SERVER_ENCRYPT unterscheidet sich vom Wert DCE nur darin, dass über das Netzwerk gesendete Kennwörter verschlüsselt werden. Der Wert DCE_SERVER_ENCRYPT gilt nur für Verwendung auf einem Server. Dieser Wert gibt an, dass der Server Authentifizierung der Art DCE oder SERVER_ENCRYPT akzeptiert.

Der Wert KERBEROS bedeutet, dass die Authentifizierung auf einem Kerberos-Server mit Hilfe des Kerberos-Sicherheitsprotokolls für Authentifizierung ausgeführt wird. Bei Verwendung der Authentifizierungsart KRB_SERVER_ENCRYPT auf dem Server und Unterstützung des Kerberos-Sicherheitssystems durch die Clients ist die tatsächliche Systemauthentifizierungsart KERBEROS. Unterstützen die Clients das Kerberos-Sicherheitssystem nicht, entspricht die tatsächliche Systemauthentifizierungsart SERVER_ENCRYPT.

Anmerkung: Die Kerberos-Authentifizierungsarten werden nur auf Servern unterstützt, die unter Windows 2000 ausgeführt werden.

Folgende Authentifizierungswerte unterstützen Kennwortverschlüsselung: SERVER_ENCRYPT, DCS_ENCRYPT, DCE_SERVER_ENCRYPT und KRB_SERVER_ENCRYPT. Diese Werte sind hinsichtlich der Authentifizierungsposition mit SERVER, DCS, DCE und KERBEROS funktionsgleich. Der einzige Unterschied ist, dass abgesetzte Kennwörter an der Quelle verschlüsselt und am Ziel entschlüsselt werden müssen (wie von der an der Quelle katalogisierten Authentifizierungsart angegeben). Verschlüsselte und nicht verschlüsselte Werte mit übereinstimmenden Authentifizierungspositionen können dann verwendet werden, um verschiedene Verschlüsselungskombinationen aus Client und Gateway oder Gateway und Server zu wählen. Dies beeinflusst nicht, wo die Authentifizierung stattfindet.

Informationen zu numerischen Äquivalenten und API-Konstanten finden Sie im Handbuch *Administrative API Reference*.

Weitere Informationen dazu, wann und warum DCE oder DCS zu verwenden ist, und zu Authentifizierungsaspekten bei zusammengeschlossenen Datenbanken finden Sie im Kapitel zum Steuern des Datenbankzugriffs im Handbuch *Systemverwaltung: Implementierung*.

Empfehlung: In der Regel ist der Standardwert (SERVER) geeignet. Wenn Ihre eingehenden Anforderungen entweder von Kerberos, DB2 Connect oder DCE verarbeitet werden, lesen Sie die Informationen im Kapitel zum Steuern des Datenbankzugriffs im Handbuch *Systemverwaltung: Implementierung*.

Katalogisieren ohne Berechtigung zulässig (catalog_noauth)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Client
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients
- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

Datenbank-Server mit lokalen und fernen Clients; Datenbank-Server mit lokalen und fernen Clients

NO [NO (0) — YES (1)]

Client; Datenbank-Server mit lokalen Clients; Satellitendatenbank-Server mit lokalen Clients

YES [NO (0) — YES (1)]

Dieser Parameter gibt an, ob Benutzer Datenbanken und Knoten oder DCS- und ODBC-Verzeichnisse ohne SYSADM-Berechtigung katalogisieren und aus dem Katalog entfernen können. Der Standardwert (0) für diesen Parameter gibt an, dass SYSADM-Berechtigung erforderlich ist. Wenn dieser Parameter auf 1 gesetzt ist, ist keine SYSADM-Berechtigung erforderlich.

Standarddatenbankpfad (dftdbpath)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

- Satellitendatenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich]

UNIX Benutzerverzeichnis des Exemplareigners [beliebiger vorhandener Pfad]

OS/2 und Windows NT Laufwerk, auf dem DB2 installiert ist [beliebiger vorhandener Pfad]

Dieser Parameter enthält den Standarddateipfad, der zur Erstellung von Datenbanken unter dem Datenbankmanager verwendet wird. Wird bei der Erstellung einer Datenbank kein Pfad angegeben, wird die Datenbank in dem Pfad erstellt, der durch den Parameter *dftdbpath* angegeben wird.

In einer Umgebung mit partitionierten Datenbanken müssen Sie sicherstellen, dass der Pfad, in dem die Datenbank erstellt wird, kein an das Dateisystem NFS angehängter Pfad ist (auf UNIX-Plattformen) bzw. kein Netzwerkpfad ist (in der Windows NT-Umgebung). Der angegebene Pfad muss physisch auf jedem Datenbankpartitions-Server vorhanden sein. Um Verwirrung zu vermeiden, ist es am besten, einen Pfad anzugeben, der auf jedem Datenbankpartitions-Server lokal angehängt ist. Die Länge des Pfads darf maximal 205 Zeichen betragen. Das System hängt den Knotennamen am Ende des Pfads an.

Weil Datenbanken auf beträchtliche Größen anwachsen und möglicherweise viele Benutzer Datenbanken erstellen können (je nach Umgebung und Zielsetzung), ist es häufig sehr praktisch, alle Datenbanken an einer einzigen definierten Position erstellen und speichern zu lassen. Außerdem ist es von Vorteil, Datenbanken von anderen Anwendungen und Daten sowohl aus Integritätsgründen als auch zur leichteren Sicherung und Wiederherstellung zu trennen.

In UNIX-Umgebungen darf die Länge des im Parameter *dftdbpath* definierten Namens 215 Zeichen nicht überschreiten, und es muss ein gültiger, absoluter Pfadname sein. Unter OS/2 und Windows NT kann der im Parameter *dftdbpath* angegebene Name ein Laufwerkbuchstabe sein, der wahlfrei mit einem Doppelpunkt versehen werden kann.

Empfehlung: Wenn die Möglichkeit besteht, legen Sie umfangreiche Datenbanken auf einer anderen Platte an als andere Daten, auf die häufig zugegriffen wird, wie z. B. Dateien des Betriebssystems oder die Datenbankprotokolldateien.

LOGON für DB2START/DB2STOP erforderlich (ss_logon)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] YES [NO (0), YES (1)]

Dieser Parameter gilt nur für die OS/2-Umgebung. Wird für diesen Parameter der Standardwert akzeptiert, ist eine LOGON-Benutzer-ID und ein LOGON-Kennwort erforderlich, damit DB2START oder DB2STOP abgesetzt werden kann.

Alle Clients akzeptieren (trust_allclnts)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] YES [NO, YES, DRDAONLY]

Zugehörige Parameter

- „Authentifizierungsart (authentication)" auf Seite 561
- „Identifikationsüberprüfung für gesicherte Clients (trust_clntauth)" auf Seite 566

Dieser Parameter ist nur aktiviert, wenn der Parameter *authentication* auf CLIENT gesetzt ist.

Dieser Parameter und der Parameter *trust_clntauth* werden verwendet, um festzustellen, wo Benutzerberechtigungen für die Datenbankumgebung überprüft werden.

Durch Übernehmen des Standardwerts „YES" für diesen Parameter werden alle Clients als gesicherte Clients akzeptiert. Dies bedeutet, der Server geht davon aus, dass auf jedem Client eine Sicherheitsstufe vorhanden ist und auf jedem Client eine Gültigkeitsprüfung für Benutzer ausgeführt werden kann.

Dieser Parameter kann nur auf „NO" gesetzt werden, wenn der Parameter *authentication* auf CLIENT gesetzt ist. Ist dieser Parameter auf „NO" gesetzt, müssen die ungesicherten Clients beim Herstellen einer Verbindung zum Server eine Benutzer-ID mit Kennwort angeben. Ungesicherte Clients sind Betriebssystemplattformen, die nicht über ein Sicherheitssubsystem zur Gültigkeitsprüfung für Benutzer verfügen.

Das Setzen dieses Parameters auf „DRDAONLY" schützt vor allen Clients mit Ausnahme von DRDA-Clients von DB2 für MVS und OS/390, DB2 für VM und VSE und DB2 für OS/400. Nur diese Clients dürfen die Authentifizierung auf der Client-Seite ausführen. Alle anderen Clients müssen eine Benutzer-ID und ein Kennwort bereitstellen, deren Gültigkeit vom Server überprüft wird.

Wenn *trust_allclnts* auf „DRDAONLY" gesetzt ist, wird mit dem Parameter *trust_clntauth* ermittelt, wo die Authentifizierung der Clients erfolgt. Wenn *trust_clntauth* auf „CLIENT" gesetzt ist, findet die Authentifizierung auf dem Client statt. Wenn *trust_clntauth* auf „SERVER" gesetzt ist, erfolgt die Authentifizierung auf dem Client, sofern kein Kennwort angegeben ist, und auf dem Server, sofern ein Kennwort angegeben ist.

Weitere Informationen zu gesicherten Clients finden Sie im Abschnitt zur Auswahl einer Authentifizierungsmethode für Ihren Server im Handbuch *Systemverwaltung: Implementierung*.

Identifikationsüberprüfung für gesicherte Clients (trust_clntauth)

Konfigurationsart Datenbankmanager

Gilt für

- Datenbank-Server mit lokalen und fernen Clients
- Datenbank-Server mit lokalen Clients
- Partitionierter Datenbank-Server mit lokalen und fernen Clients

Parameterart Konfigurierbar

Standardwert [Bereich] CLIENT [CLIENT, SERVER]

Zugehörige Parameter

- „Authentifizierungsart (authentication)" auf Seite 561
- „Alle Clients akzeptieren (trust_allclnts)" auf Seite 565

Dieser Parameter gibt an, ob eine Identifikationsüberprüfung gesicherter Clients auf dem Server oder auf dem Client erfolgt, wenn der Client eine Benutzer-ID mit Kennwort für eine Verbindung angibt. Dieser Parameter (und *trust_allclnts*) ist nur aktiviert, wenn der Parameter *authentication* auf CLIENT gesetzt ist. Wird keine Benutzer-ID mit Kennwort angegeben, geht das System davon aus, dass die Identifikationsüberprüfung des Benutzers vom Client ausgeführt wurde, d. h. auf dem Server erfolgt keine weitere Identifikationsüberprüfung.

Ist dieser Parameter auf CLIENT (die Standardeinstellung) gesetzt, kann der gesicherte Client eine Verbindung herstellen, ohne eine Benutzer-ID mit Kennwort anzugeben, und es wird angenommen, dass die Identifikationsüberprüfung des Benutzers bereits vom Betriebssystem vorgenommen wurde. Ist der Parameter auf SERVER gesetzt, werden Benutzer-ID und Kennwort auf dem Server überprüft.

Der numerische Wert für CLIENT ist 0. Der numerische Wert für SERVER ist 1.

Weitere Informationen zu gesicherten Clients finden Sie im Abschnitt zur Auswahl einer Authentifizierungsmethode für Ihren Server im Handbuch *Systemverwaltung: Implementierung*.

Teil 4. Anhänge und Schlussteil

Anhang A. DB2-Registrierungsvariablen und DB2-Umgebungsvariablen

Die folgende Liste enthält die DB2-Registrierungsvariablen und die DB2-Umgebungsvariablen, mit denen Sie zur Einrichtung Ihres Systems u. U. vertraut sein müssen. Jedem Parameter wurde eine kurze Beschreibung beigefügt. Außerdem ist es möglich, dass einige Parameter für Ihre Umgebung nicht relevant sind.

Sie können mit folgendem Befehl eine Liste aller unterstützten Registervariablen anzeigen:

```
db2set -lr
```

Sie können mit folgendem Befehl den Wert einer Variablen im aktuellen oder Standardexemplar ändern:

```
db2set registrierungs_variablen_name=neuer_wert
```

Zur Aktualisierung von Umgebungsvariablen muss der Befehl `set` verwendet und anschließend das System erneut gestartet werden.

Die Werte für die geänderten Registrierungsvariablen müssen festgelegt werden, bevor der Befehl **db2start** abgesetzt wird. Weitere Informationen zum Ändern und Verwenden von Registrierungsvariablen finden Sie im Band *Systemverwaltung: Implementierung*.

Die in den Beschreibungen der binären Registrierungsvariablen verwendeten Werte weisen äquivalente Werte auf. Die Werte YES, 1 und ON sind alle äquivalent. Ebenso sind auch die Werte NO, 0 und OFF alle äquivalent. Die äquivalenten Werte sind austauschbar, wenn sie verwendet werden.

Tabelle 21. Allgemeine Registrierungsvariablen

Variablenname	**Betriebssystem**	**Werte**
Beschreibung		
DB2ACCOUNT	Alle	Standardwert = Null
Die Abrechnungszeichenfolge für den Benutzer, die an den fernen Host gesendet wird. Einzelheiten siehe *DB2 Connect Benutzerhandbuch*.		
DB2BIDI	Alle	Standardwert = NO Werte: YES oder NO

Tabelle 21. Allgemeine Registrierungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Diese Variable aktiviert die bidirektionale Unterstützung von Sprachen, und die Variable DB2CODEPAGE dient zur Deklaration der zu verwendenden Codepage. Im Band *Systemverwaltung: Konzept*, im Anhang zur Unterstützung von Landessprachen, finden Sie weitere Informationen zur bidirektionalen Unterstützung von Sprachen.		
DB2_BLOCK_ON_LOG_DISK_FULL	ALL	Standardwert = No Werte: Yes oder No
Diese DB2-Registrierungsvariable kann so eingestellt werden, dass keine Fehler aufgrund erschöpfter Festplattenkapazität generiert werden, wenn DB2 keine neue Protokolldatei im aktiven Protokollpfad erstellen kann. Stattdessen versucht DB2 alle fünf Minuten, die Protokolldatei zu erstellen, bis die Erstellung gelingt. Nach jedem Versuch schreibt DB2 eine Nachricht in die Datei `db2diag.log`. Die Überwachung der Datei `db2diag.log` stellt die einzige Möglichkeit dar, mit Gewissheit festzustellen, dass eine Anwendung blockiert ist. Solange die Protokolldatei noch nicht erstellt werden konnte, kann keine der Benutzeranwendungen, die versucht, Tabellendaten zu aktualisieren, Transaktionen festschreiben. Auf Lesezugriff beschränkte Abfragen sind u. U. nicht direkt betroffen; wenn jedoch eine Abfrage auf Daten zugreifen muss, die aufgrund einer Aktualisierungsanforderung gesperrt sind, oder auf eine Datenseite, die im Pufferpool von der aktualisierenden Anwendung korrigiert wird, erscheinen auch auf Lesezugriff beschränkte Abfragen blockiert.		
DB2CODEPAGE	Alle	Standardwert: abgeleitet aus dem vom Betriebssystem definierten Landescode
Gibt die Codepage der Daten an, die an DB2 für Datenbank-Client-Anwendungen übergeben werden. Der Benutzer sollte DB2CODEPAGE nicht definieren, sofern er nicht in der DB2-Dokumentation oder vom DB2-Service dazu angeleitet wird. Wird DB2CODEPAGE auf einen Wert gesetzt, der vom Betriebssystem nicht unterstützt wird, können unerwartete Ergebnisse auftreten. Im Normalfall müssen Sie DB2CODEPAGE nicht definieren, weil DB2 die Daten zur Codepage automatisch vom Betriebssystem abruft.		
DB2COUNTRY	Alle	Standardwert: abgeleitet aus dem vom Betriebssystem definierten Landescode
Gibt den Landes-, Regions- oder Gebietscode der Client-Anwendung an, was sich auf die Formate für Datum und Uhrzeit auswirkt.		
DB2DBDFT	Alle	Standardwert = Null
Gibt den Aliasnamen der Datenbank an, die für implizite Verbindungen zu verwenden ist. Wenn eine Anwendung keine Datenbankverbindung hat, aber SQL-Anweisungen abgesetzt werden, wird eine implizite Verbindung hergestellt, wenn die Umgebungsvariable DB2DBDFT mit einer Standarddatenbank definiert wurde.		

Tabelle 21. Allgemeine Registrierungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2DBMSADDR	32-Bit-Windows-Betriebssysteme	Standardwert = 0x20000000 für Windows NT, 0x90000000 für Windows 95 Wert: 0x20000000 bis 0xB0000000 in Inkrementen von 0x10000
Gibt die Standardadresse des Datenbankmanagers für gemeinsam benutzten Speicher im hexadezimalen Format an. Wenn der Befehl *db2start* aufgrund einer Adresskollision beim gemeinsam benutzen Speicher fehlschlägt, kann diese Registrierungsvariable geändert werden, um das Datenbankmanagerexemplar zu zwingen, den gemeinsam benutzen Speicher an einer anderen Adresse anzulegen.		
DB2_DISABLE_FLUSH_LOG	Alle	Standardwert = OFF Wert: ON oder OFF
Gibt an, ob das Schließen der aktiven Protokolldatei nach Abschluss der Onlinesicherung inaktiviert werden soll. Wenn eine Onlinesicherung vollständig abgeschlossen ist, wird die letzte aktive Protokolldatei abgeschnitten, geschlossen und für die Archivierung bereitgestellt. Dadurch wird sichergestellt, dass Ihre Onlinesicherung über einen vollständigen Satz archivierter Protokolle für eine eventuelle Wiederherstellung verfügt. Sie können das Schließen der letzten aktiven Protokolldatei inaktivieren, wenn Sie Bedenken haben, dass Teile des Speicherbereichs für die Protokollfolgenummer (Log Sequence Number, LSN) verschwendet werden. Jedesmal wenn eine aktive Protokolldatei abgeschnitten wird, wird die LSN um einen Betrag erhöht, der proportional zum abgeschnittenen Speicherbereich ist. Wenn Sie jeden Tag eine große Zahl an Onlinesicherungen ausführen, können Sie das Schließen der letzten aktiven Protokolldatei inaktivieren. Sie können auch das Schließen der letzten aktiven Protokolldatei inaktivieren, wenn Sie kurz nach dem vollständigen Abschluss der Onlinesicherung Nachrichten mit dem Inhalt empfangen, dass die Protokolldatei voll ist. Wenn eine Protokolldatei abgeschnitten wird, wird der reservierte Speicherbereich für die aktive Protokolldatei durch einen Betrag erhöht, der proportional zur Größe des abgeschnittenen Protokolls ist. Der Speicherbereich für die aktive Protokolldatei wird freigegeben, wenn die abgeschnittene Protokolldatei wiederhergestellt wird. Die Wiederherstellung tritt auf, kurz nachdem die Protokolldatei inaktiv wird. Nur während des kurzen Intervalls dazwischen können Sie Nachrichten mit dem Inhalt empfangen, dass das Protokoll voll ist.		
DB2DISCOVERYTIME	OS/2- und 32-Bit-Windows-Betriebssysteme	Standardwert = 40 Sek., Minimum = 20 Sek.
Gibt die Zeitdauer an, die der Discovery-Prozess (SEARCH) nach DB2-Systemen sucht.		
DB2INCLUDE	Alle	Standardwert = aktuelles Verzeichnis

Tabelle 21. Allgemeine Registrierungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt einen Pfad an, der bei der Verarbeitung der SQL-Anweisung INCLUDE textdatei während der Ausführung des DB2-Befehls PREP zu verwenden ist. Er gibt eine Liste von Verzeichnissen an, in denen sich die Kopfdatei für INCLUDE befinden könnte. Im Handbuch *Application Development Guide* finden Sie eine Beschreibung, wie DB2INCLUDE in den verschiedenen vorkompilierten Sprachen verwendet wird.		
DB2INSTDEF	OS/2- und 32-Bit-Windows-Betriebssysteme	Standardwert = DB2
Definiert den Wert, der zu verwenden ist, wenn DB2INSTANCE nicht definiert ist.		
DB2INSTOWNER	Windows NT	Standardwert = Null
Die Registrierungsvariable, die bei der ersten Erstellung des Exemplars in der DB2-Profilregistrierdatenbank erstellt wird. Diese Variable wird auf den Namen der Exemplareignermaschine gesetzt.		
DB2_LIC_STAT_SIZE	Alle	Standardwert = Null Bereich: 0 bis 32 767
Die Registrierungsvariable wird zur Festlegung der Maximalgröße (in MB) der Datei mit den Lizenzstatistikdaten für das System verwendet. Der Wert 0 schaltet das Sammeln von Lizenzstatistikdaten aus. Wenn die Variable nicht erkannt wird oder nicht definiert ist, wird standardmäßig ein uneingeschränkter Wert angenommen. Die Statistikdaten werden über die Lizenzzentrale angezeigt.		
DB2NBDISCOVERRCVBUFS	Alle	Standardwert = 16 Puffer, Minimum = 16 Puffer
Diese Variable wird für den NetBIOS Discovery-Prozess (SEARCH) verwendet. Die Variable gibt die Anzahl der gleichzeitigen Discovery-Antworten an, die von einem Client empfangen werden können. Wenn der Client mehr gleichzeitige Antworten empfängt, als von dieser Variable festgelegt, werden die überzähligen Antworten von der NetBIOS-Schicht gelöscht. Der Standardwert ist 16 NetBIOS-Empfangspuffer. Wenn ein kleinerer Wert als der Standardwert gewählt wird, wird der Standardwert verwendet.		
DB2OPTIONS	Alle außer Windows 3.1 und Macintosh	Standardwert = Null
Legt Optionen für den Befehlszeilenprozessor fest.		
DB2SLOGON	Windows 3.x	Standardwert = Null, Werte: YES oder NO
Ermöglicht eine sichere Anmeldung in DB2 für Windows 3.x. Wenn DB2SLOGON=YES definiert ist, schreibt DB2 Benutzer-IDs und Kennwörter nicht in eine Datei, sondern verwendet ein Hauptspeichersegment, um sie zu verwalten. Wenn DB2SLOGON aktiviert ist, muss der Benutzer bei jedem Start von Windows 3.x eine Anmeldung ausführen.		

Tabelle 21. Allgemeine Registrierungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2TIMEOUT	Windows 3.x und Macintosh	Standardwert = (nicht definiert)
Dient zur Steuerung des Zeitlimits für Clients unter Windows 3.x und Macintosh während langer SQL-Abfragen. Nach Überschreitung des Zeitlimits wird der Benutzer über ein Dialogfenster gefragt, ob die Abfrage unterbrochen oder fortgesetzt werden soll. Der Mindestwert für diese Variable ist 30 Sekunden. Wenn DB2TIMEOUT auf einen Wert zwischen 1 und 30 gesetzt wird, wird der Mindestwert verwendet. Wenn DB2TIMEOUT auf den Wert 0 oder einen negativen Wert gesetzt wird, wird die Zeitsperre inaktiviert. Standardmäßig ist diese Einrichtung inaktiviert.		
DB2TRACENAME	Windows 3.x und Macintosh	Standardwert = DB2WIN.TRC (auf Windows 3.x), DB2MAC.TRC (auf Macintosh)
Definiert unter Windows 3.x und Macintosh den Namen der Datei, in der Trace-Informationen gespeichert werden. Der Standardwert ist *db2tracename*=`DB2WIN.TRC`. Die Datei wird im aktuellen Exemplarverzeichnis (z. B. `\sqllib\db2`) gespeichert. Es wird ausdrücklich empfohlen, bei der Benennung der Trace-Datei einen vollständigen Pfadnamen anzugeben.		
DB2TRACEON	Windows 3.x und Macintosh	Standardwert = NO Werte: YES oder NO
Aktiviert unter Windows 3.x und Macintosh den Trace, um Informationen für IBM im Falle eines Problems bereitzustellen. (Es wird empfohlen, den Trace nur dann zu aktivieren, wenn Sie auf ein Problem stoßen, dass Sie nicht lösen können.) Im Handbuch *Troubleshooting Guide* finden Sie Informationen zur Verwendung der Trace-Einrichtung mit Clients.		
DB2TRCFLUSH	Windows 3.x und Macintosh	Standardwert = NO Werte: YES oder NO
Unter Windows 3.x und Macintosh kann DB2TRACEFLUSH in Verbindung mit DB2TRACEON=`YES` verwendet werden. Das Definieren von DB2TRACEFLUSH=`YES` sorgt dafür, dass jeder Trace-Satz unverzüglich in die Trace-Datei geschrieben wird. Dadurch wird das DB2-System erheblich verlangsamt, so dass die Standardeinstellung DB2TRACEFLUSH=`NO` ist. Diese Einstellung ist nützlich, wenn eine Anwendung das System blockiert und einen Neustart des Systems erforderlich macht. Durch Definieren dieses Schlüsselworts wird sichergestellt, dass die Trace-Datei und die Trace-Einträge durch den Neustart nicht verloren gehen.		
DB2TRCSYSERR	Windows 3.x und Macintosh	Standardwert = 1 Werte: 1-32 767
Gibt die Anzahl der Systemfehler an, die vom Trace zu erfassen sind, bevor der Client den Trace inaktiviert. Durch den Standardwert wird ein Systemfehler erfasst, nach dem der Trace inaktiviert wird.		
DB2YIELD	Windows 3.x	Standardwert = NO Werte: YES oder NO

Tabelle 21. Allgemeine Registrierungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Definiert das Verhalten des Clients unter Windows 3.x bei der Kommunikation mit einem fernen Server. Wenn NO definiert ist, überlässt der Client die CPU anderen Anwendungen unter Windows 3.x nicht, so dass die Windows-Umgebung angehalten wird, während die Client-Anwendung mit dem fernen Server kommuniziert. In diesem Fall müssen Sie auf die Beendigung der Kommunikation warten, bevor Sie andere Operationen wieder aufnehmen können. Wenn YES definiert ist, funktioniert Ihr System normal. Es wird empfohlen, die Ausführung der Anwendung mit DB2YIELD=YES zu versuchen. Wenn es zu einem Systemabsturz kommt, müssen Sie DB2YIELD=NO definieren. Bei der Anwendungsentwicklung ist sicherzustellen, dass die Anwendung so geschrieben wird, dass sie Windows-Nachrichten (Messages) akzeptiert und verarbeitet, während sie auf die Beendigung einer Fernübertragungsoperation wartet		

Tabelle 22. Systemumgebungsvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2CONNECT_IN_APP_PROCESS	Alle	Standardwert = YES Werte: YES oder NO
Wenn diese Variable auf den Wert NO gesetzt wird, werden lokale DB2 Connect-Clients auf einer Maschine mit DB2 Connect Enterprise Edition gezwungen, innerhalb eines Agenten aktiv zu sein. Einige Vorteile der Ausführung innerhalb eines Agenten bestehen darin, dass lokale Clients auf diese Weise überwacht werden und die SYSPLEX-Unterstützung nutzen können.		
DB2DOMAINLIST	Nur Windows NT-Server	Standardwert = Null Werte: Eine Liste mit Windows NT-Domänennamen, die durch Kommas („,“) getrennt werden.
Definiert eine oder mehrere Windows NT-Domänen. Nur Verbindungs- und Anschlussanforderungen von Benutzern, die zu diesen Domänen gehören, werden akzeptiert. Diese Registrierungsvariable sollte nur in einer reinen Windows NT-Domänenumgebung mit DB2-Servern und -Clients mit Version 7.1 (oder höher) verwendet werden.		
DB2ENVLIST	UNIX	Standardwert: Null
Listet spezifische Variablennamen entweder für gespeicherte Prozeduren oder für benutzerdefinierte Funktionen auf. In der Standardeinstellung werden mit dem Befehl **db2start** alle Benutzerumgebungsvariablen mit Ausnahme derer, die das Präfix **DB2** oder **db2** haben, gefiltert. Wenn spezifische Registrierungsvariablen entweder an gespeicherte Prozeduren oder an benutzerdefinierte Funktionen übergeben werden müssen, können Sie die Variablennamen in der Registrierungsvariablen DB2ENVLIST auflisten. Trennen Sie die einzelnen Variablennamen durch ein oder mehrere Leerzeichen. DB2 erstellt eigene Variablen PATH und LIBPATH, wenn also PATH oder LIBPATH in DB2ENVLIST angegeben wird, wird der eigentliche Wert des Variablennamens am Ende des von DB2 erstellten Werts angehängt.		

Tabelle 22. Systemumgebungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2INSTANCE	Alle	Standardwert = DB2INSTDEF auf OS/2- und 32-Bit-Windows-Betriebssystemen.
Die Umgebungsvariable, die zur Definition des Exemplars dient, das standardmäßig aktiv ist. Unter UNIX müssen Benutzer einen Wert für DB2INSTANCE festlegen.		
DB2INSTPROF	OS/2-, Windows 3.x- und 32-Bit-Windows-Betriebssysteme	Standardwert: Null
Die Umgebungsvariable, die zur Angabe der Position des Exemplarverzeichnisses auf den Betriebssystemen OS/2, Windows 3,x und 32-Bit-Windows dient, wenn sie von DB2PATH abweicht.		
DB2LIBPATH	UNIX	Standardwert: Null
Gibt den Wert von LIBPATH in der Registrierungsvariablen DB2LIBPATH an. Der Wert von LIBPATH kann nicht von Elter- in Kindprozesse übernommen werden, wenn sich die Benutzer-ID geändert hat. Da Root der Eigner des Programms **db2start** ist, kann DB2 die LIBPATH-Einstellungen der Endbenutzer nicht übernehmen. Wenn Sie den Variablennamen, LIBPATH, in der Registrierungsvariable DB2ENVLIST auflisten, müssen Sie auch den Wert von LIBPATH in der Registrierungsvariablen DB2LIBPATH angeben. Das Programm **db2start** liest dann den Wert von DB2LIBPATH und fügt diesen Wert am Ende des von DB2 erstellten LIBPATH an.		
DB2NODE	Alle	Standardwert: Null Werte: 1 bis 999
Dient zur Angabe des logischen Zielknotens eines Datenbankpartitions-Servers von DB2 Enterprise - Extended Edition, zu dem die Verbindung hergestellt bzw. für den ein Anschluss ausgeführt werden soll. Wenn diese Variable nicht definiert wird, wird als logischer Zielknoten standardmäßig der logische Knoten angenommen, der auf der Maschine mit Anschluss `0` definiert ist.		
DB2_PARALLEL_IO	Alle	Standardwert: Null Werte: * (d. h. alle Tabellenbereiche) oder eine durch Kommas getrennte Liste mit mehr als einem definierten Tabellenbereich
Wenn Daten aus Tabellenbereichsbehältern gelesen oder in diese geschrieben werden, kann DB2 die parallele Ein-/Ausgabe verwenden, falls die Anzahl der Behälter in der Datenbank größer als eins ist. Verwenden Sie diese Registrierungsvariable, um die parallele Ein-/Ausgabe für einen einzelnen Behälter zu erzwingen. Geben Sie nach der Festlegung der Registrierungsvariablen einen Befehl DB2STOP aus, und geben Sie anschließend DB2START ein, damit die Änderungen wirksam werden.		
DB2PATH	OS/2-, Windows 3.x- und 32-Bit-Windows-Betriebssysteme	Standardwert: (je nach Betriebssystem unterschiedlich)

Tabelle 22. Systemumgebungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Die Umgebungsvariable, die zur Angabe des Verzeichnisses dient, in dem das Produkt auf den Betriebssystemen OS/2, Windows 3.x und 32-Bit-Windows installiert ist.		
DB2_STRIPED_CONTAINERS	Alle	Standardwert: Null Werte: ON, Null
Wenn RAID-Einheiten für Tabellenbereichsbehälter verwendet werden, wird empfohlen, dass der Tabellenbereich mit einem Wert für EXTENTSIZE erstellt wird, der der Größe des einheitenübergreifend gespeicherten RAID-Datenblocks (Stripe Size) oder einem Vielfachen dieser Größe entspricht. Jedoch können die Speicherbereiche aufgrund der Behälterkennung, die eine Seite lang ist, nicht mit den einheitenübergreifend gespeicherten RAID-Datenblöcke ausgerichtet werden, und es kann während einer E/A-Anforderung notwendig werden, auf mehr physische Platten zuzugreifen als optimal wäre. Wenn DMS-Tabellenbereichsbehälter verwendet werden, wird dieses Problem durch Zuordnen eines eigenen (vollständigen) Speicherbereichs für den Befehl vermieden. Dadurch wird das Problem umgangen, es ist aber ein zusätzlicher Speicherbereich für den Systemaufwand innerhalb des Behälters erforderlich. Geben Sie nach der Festlegung dieser Registrierungsvariablen einen Befehl DB2STOP aus, und geben Sie anschließend DB2START ein, damit die Änderungen wirksam werden.		

Tabelle 23. Kommunikationsvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2CHECKCLIENTINTERVAL	AIX, nur Server	Standardwert = 0 Werte: ein numerischer Wert größer als Null
Hiermit wird der Status der APPC-Client-Verbindungen geprüft. Ermöglicht das frühe Erkennen einer Client-Beendigung, anstatt zu warten, bis die Abfrage abgeschlossen ist. Wird DB2CHECKCLIENTINTERVAL auf Null eingestellt, wird keine Überprüfung vorgenommen. Wird ein numerischer Wert größer als Null eingestellt, stellt der Wert interne DB2-Arbeitseinheiten dar. Die folgenden Angaben sind Richtwerte für die Überprüfungshäufigkeit: 300 bei gelegentlichen Überprüfungen; 100 bei regelmäßigen Überprüfungen; 50 bei zahlreichen Überprüfungen. Je häufiger Sie den Client-Status während der Ausführung einer Datenbankanforderung prüfen, desto länger dauert die Beendigung der Abfrage. Wenn die DB2-Auslastung hoch ist (d. h., wenn viele interne Anforderungen anliegen), dann hat das Einstellen von DB2CHECKCLIENTINTERVAL auf einen niedrigen Wert eine größere Auswirkung auf die Leistung als in einer Situation, in der die Auslastung niedrig ist und DB2 die meiste Zeit wartet.		

Tabelle 23. Kommunikationsvariablen (Forts.)

Variablenname Beschreibung	Betriebssystem	Werte
DB2COMM	Alle, nur Server	Standardwert = Null Werte: eine beliebige Kombination aus APPC, IPXSPX, NETBIOS, NPIPE, TCPIP
Definiert die Kommunikationsmanager, die gestartet werden, wenn der Datenbankmanager gestartet wird. Wenn dieser Parameter nicht definiert wird, werden auf dem Server keine DB2-Kommunikationsmanager gestartet.		
DB2_FORCE_NLS_CACHE	AIX, HP_UX, Solaris	Standardwert = FALSE Werte: TRUE oder FALSE
Dient zur Verhinderung der Möglichkeit von Zugriffskonflikten in Multi-Thread-Anwendungen. Wenn diese Registrierungsvariable „TRUE" ist, werden die Informationen zur Codepage und zum Landescode beim Erstzugriff eines Threads auf die Informationen gesichert. Von da an werden die zwischengespeicherten Informationen für einen anderen Thread verwendet, der diese Informationen anfordert. Dadurch wird der Zugriffskonflikt ausgeschaltet, was in bestimmten Situationen zu einem Leistungsvorteil führt. Diese Einstellung sollte nicht verwendet werden, wenn die Anwendung die länderspezifischen Angaben zwischen Verbindungen ändert. In solch einer Situation ist sie wahrscheinlich kaum erforderlich, da Multi-Thread-Anwendungen ihre länderspezifischen Angaben in der Regel nicht ändern, weil dies die Thread-Stabilität gefährden könnte.		
DB2NBADAPTERS	OS/2 und Windows NT	Standardwert = 0 Bereich: 0-15, Mehrere Werte müssen durch Kommas getrennt werden.
Dient zur Definition, welche lokalen Adapter für die DB2-NetBIOS-LAN-Übertragung zu verwenden sind. Jeder lokale Adapter wird mit Hilfe seiner logischen Adapternummer angegeben.		
DB2NBCHECKUPTIME	OS/2 und Windows NT, nur Server	Standardwert = 1 Minute Werte: 1-720
Definiert das Zeitintervall zwischen den Aufrufen der Prüfprozedur des NetBIOS-Protokolls. Die Prüfzeit wird in Minuten angegeben. Niedrigere Werte stellen sicher, dass die Prüfprozedur des NetBIOS-Protokolls häufiger aktiv wird und Speicher und andere Systemressourcen freigibt, die zurückbleiben, wenn Agenten oder Sitzungen unerwartet beendet werden.		
DB2NBINTRLISTENS	OS/2 und Windows NT, nur Server	Standardwert = 1 Werte: 1-10 Mehrere Werte müssen durch Kommas getrennt werden.

Tabelle 23. Kommunikationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Definiert die Anzahl der NetBIOS-Befehle listen send (NCBs - NetBIOS-Steuerblöcke), die asynchron abgesetzt werden, wobei die Empfangsbereitschaft für Unterbrechungen ferner Clients bestehen bleibt. Diese Flexibilität ist für "unterbrechungsaktive" Umgebungen gedacht, um sicherzustellen, dass Unterbrechungsaufrufe von fernen Clients eine Verbindung herstellen können, wenn die Server mit der Bedienung anderer ferner Unterbrechungen beschäftigt sind. Durch Definieren eines niedrigeren Werts für DB2NBINTRLISTENS werden NetBIOS-Sitzungen und NetBIOS-Steuerblöcke auf dem Server geschont. In einer Umgebung jedoch, in der Client-Unterbrechungen regelmäßig vorkommen, ist ein höherer Wert für DB2NBINTRLISTENS eventuell sinnvoll, um auf unterbrechende Clients flexibler reagieren zu können. **Anmerkung:** Die angegebenen Werte sind positionsabhängig. Sie beziehen sich jeweils auf die entsprechenden Wertpositionen für DB2NBADAPTERS.		
DB2NBRECVBUFFSIZE	OS/2 und Windows NT, nur Server	Standardwert = 4096 Byte Bereich: 4096-65536
Definiert die Größe der Empfangspuffer des DB2-NetBIOS-Protokolls. Diese Puffer werden den NetBIOS-Empfangssteuerblöcken zugeordnet (NCBs). Niedrigere Werte sparen Server-Speicher, während höhere Werte erforderlich werden können, wenn die Client-Datenübertragungen umfangreicher werden.		
DB2NBBRECVNCBS	OS/2 und Windows NT, nur Server	Standardwert =10 Bereich: 1-99
Definiert die Anzahl der NetBIOS-Befehle receive_any (NCBs), die der Server beim Betrieb absetzt und verwaltet. Dieser Wert kann angepasst werden, je nach der Anzahl ferner Clients, mit denen der Server verbunden ist. Niedrigere Werte sparen Server-Ressourcen. **Anmerkung:** Für jeden benutzten Adapter kann ein eigener eindeutiger Wert für Empfangs-NCBs im Parameter DB2NBBRECVNCBS angegeben werden. Die angegebenen Werte sind positionsabhängig. Sie beziehen sich jeweils auf die entsprechenden Wertpositionen für DB2NBADAPTERS.		
DB2NBRESOURCES	Nur OS/2- und Windows NT-Server	Standardwert = Null
Definiert die Anzahl der NetBIOS-Ressourcen, die zur Verwendung durch DB2 in einer Mehrkontextumgebung zuzuordnen sind. Diese Variable ist auf den Client-Betrieb in einer Mehrkontextumgebung beschränkt.		
DB2NBSENDNCBS	OS/2 und Windows NT, nur Server	Standardwert = 6 Bereich: 1-720
Definiert die Anzahl der NetBIOS-Befehle send (NCBs), die der Server zur Verwendung empfängt. Dieser Wert kann angepasst werden je nach der Anzahl ferner Clients, mit denen der Server verbunden ist. Durch Definieren eines niedrigeren Werts für DB2NBSENDNCBS werden Serverressourcen geschont. Jedoch kann ein höherer Wert erforderlich werden, um zu vermeiden, dass der Server mit einem send-Befehl an einen fernen Client warten muss, wenn alle anderen send-Befehle in Gebrauch sind.		

Tabelle 23. Kommunikationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2NBSESSIONS	OS/2 und Windows NT, nur Server	Standardwert = Null Bereich: 5-254
Definiert die Anzahl von Sitzungen, die DB2 zur Reservierung für DB2 anfordern soll. Der Wert von DB2NBSESSIONS kann definiert werden, um eine spezifische Sitzung für jeden in DB2NBADAPTERS angegebenen Adapter anzufordern. **Anmerkung:** Die angegebenen Werte sind positionsabhängig. Sie beziehen sich jeweils auf die entsprechenden Wertpositionen für DB2NBADAPTERS.		
DB2NBXTRANCBS	OS/2 und Windows NT, nur Server	Standardwert = 5 pro Adapter Bereich: 5-254
Definiert die NetBIOS-Befehle "extra" (NCBs), die der Server reservieren muss, wenn der Befehl **db2start** ausgeführt wird. Der Wert von DB2NBXTRANCBS kann definiert werden, um eine spezifische Sitzung für jeden in DB2NBADAPTERS angegebenen Adapter anzufordern.		
DB2NETREQ	Windows 3.x	Standardwert = 3 Bereich: 0-25
Definiert die Anzahl von NetBIOS-Anforderungen, die gleichzeitig von Clients unter Windows 3.x ausgeführt werden können. Je höher dieser Wert definiert wird, umso mehr Speicher unterhalb der 1-MB-Ebene wird verwendet. Wenn die Anzahl gleichzeitiger Anforderungen von NetBIOS-Services die definierte Anzahl erreicht, werden nachfolgend eingehende Anforderungen für NetBIOS-Services in eine Warteschlange gestellt und der Reihe nach aktiviert, wenn aktuelle Anforderungen beendet werden. Wenn Sie für DB2NETREQ den Wert 0 (Null) angeben, setzt der Windows-Datenbank-Client NetBIOS-Aufrufe im synchronen Modus mit der NetBIOS-Option für Warten ab. In diesem Modus lässt der Datenbank-Client nur die aktuelle NetBIOS-Anforderung als aktive zu und verarbeitet keine andere Anforderung, bevor die aktuelle Anforderung beendet ist. Dies kann sich auf andere Anwendungsprogramme auswirken. Der Wert 0 ist nur zur Wahrung der Abwärtskompatibilität vorgesehen. Es ist ratsam, den Wert 0 nicht zu verwenden.		
DB2RETRY	OS/2 und Windows NT	Standardwert = 0 Bereich: 0-20 000
Die Anzahl der DB2-Versuche, die APPC-Empfangsfunktion erneut zu starten. Wenn das SNA-Subsystem auf dem Server/Gateway abgestürzt ist, kann die APPC-Empfangsfunktion mit Hilfe dieser Profilvariablen zusammen mit DB2RETRYTIME automatisch erneut gestartet werden, ohne die Client-Datenfernverarbeitung unter Verwendung anderer Protokolle zu unterbrechen. Bei einem derartigen Szenario braucht DB2 nicht gestoppt und erneut gestartet zu werden, um die APPC-Client-Datenfernverarbeitung wiederherzustellen.		
DB2RETRYTIME	OS/2 und Windows NT	Standardwert = 1 Minute Bereich: 0-7 200 Minuten

Tabelle 23. Kommunikationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
In Inkrementen von einer Minute die Anzahl Minuten, die DB2 zwischen dem Ausführen aufeinander folgender Wiederholungen zum Starten der APPC-Empfangsfunktion zulässt. Wenn das SNA-Subsystem auf dem Server/Gateway abgestürzt ist, kann die APPC-Empfangsfunktion mit Hilfe dieser Profilvariablen zusammen mit DB2RETRY automatisch erneut gestartet werden, ohne die Client-Datenfernverarbeitung unter Verwendung anderer Protokolle zu unterbrechen. Bei einem derartigen Szenario braucht DB2 nicht gestoppt und erneut gestartet zu werden, um die APPC-Client-Datenfernverarbeitung wiederherzustellen.		
DB2SERVICETPINSTANCE	OS/2, Windows NT, AIX und Sun Solaris	Standardwert = Null
Dient zur Lösung des durch folgende Ursachen ausgelösten Problems: • Auf einer Maschine sind mehrere Exemplare aktiv • Ein Exemplar der Version 6 oder der Version 7 auf derselben Maschine versucht, dieselben TP-Namen zu registrieren Wenn der Befehl **db2start** aufgerufen wird, startet das angegebene Exemplar die APPC-Empfangsfunktionen (Listeners) für die folgenden TP-Namen: • DB2DRDA • x'07'6DB		
DB2SOSNDBUF	Windows 95 und Windows NT	Standardwert = 32767
Definiert den Wert von TCP/IP-Sendepuffern auf den Betriebssystemen Windows 95 und Windows NT.		
DB2SYSPLEX_SERVER	OS/2, Windows NT und UNIX	Standardwert = Null
Gibt an, ob beim Herstellen einer Verbindung zu DB2 für OS/390 die SYSPLEX-Ausnutzung aktiviert wird. Ist diese Registrierungsvariable nicht vorhanden (Standardeinstellung), oder wird sie auf einen anderen Wert als Null eingestellt, wird die Ausnutzung aktiviert. Wird diese Registrierungsvariable auf Null (0) eingestellt, wird die Ausnutzung inaktiviert. Bei dieser Einstellung wird die SYSPLEX-Ausnutzung für den Gateway inaktiviert, unabhängig davon, wie der DCS-Datenbankkatalogeintrag angegeben wurde. Weitere Informationen finden Sie im Handbuch *Command Reference* und in den Erklärungen zum Befehl **CATALOG DCS DATABASE**.		
DB2TCPCONNMGRS	Alle	Standardwert = 1 auf seriellen Maschinen; auf symmetrischen Mehrprozessormaschinen die Quadratwurzel der Zahl von Prozessoren, auf ein Maximum von acht Verbindungsmanagern aufgerundet. Werte: 1 bis 8

Tabelle 23. Kommunikationsvariablen (Forts.)

Variablenname	**Betriebssystem**	**Werte**
Beschreibung		
Wenn die Registrierungsvariable nicht festgelegt ist, wird die Standardzahl von Verbindungsmanagern erstellt. Wenn die Registrierungsvariable festgelegt ist, setzt der hier zugeordnete Wert den Standardwert außer Kraft. Die Zahl der angegebenen TCP/IP-Verbindungsmanager wird bis zu einem Maximum von 8 erstellt. Wenn weniger als einer angegeben wird, wird DB2TCPCONNMGRS auf einen Wert von eins festgelegt und eine Warnung protokolliert, dass der Wert nicht innerhalb des gültigen Bereichs ist. Wenn ein Wert größer als acht angegeben wird, wird DB2TCPCONNMGRS auf einen Wert von acht festgelegt und eine Warnung protokolliert, dass der Wert nicht innerhalb des gültigen Bereichs ist. Werte zwischen eins und acht werden ohne Änderung verwendet. Wenn mehr als ein Verbindungsmanager erstellt wird, sollte der Verbindungsdurchsatz sich verbessern, wenn mehrere Client-Verbindungen gleichzeitig empfangen werden. Wenn der Benutzer eine SMP-Maschine verwendet oder die Registrierungsvariable DB2TCPCONNMGRS geändert hat, können beim TCP/IP-Verbindungsmanager zusätzliche Prozesse (unter UNIX) bzw. Threads (unter OS/2- und Windows-Betriebssystemen) auftreten. Zusätzliche Prozesse oder Threads erfordern zusätzlichen Speicher. **Anmerkung:** Wenn die Anzahl der Verbindungsmanager auf eins gesetzt wird, kommt es bei Fernverbindungen in Systemen mit zahlreichen Benutzern und/oder häufigem Verbindungsauf- und -abbau zu einem Leistungsabfall.		
DB2_VI_ENABLE	Windows NT	Standardwert = OFF Werte: ON oder OFF
Gibt an, ob das Übertragungsprotokoll VI Architecture (VI, Virtual Interface) verwendet werden soll oder nicht. Ist diese Registrierungsvariable `ON`, dann verwendet FCM VI für die Übertragung zwischen Knoten. Ist diese Registrierungsvariable `OFF`, dann verwendet FCM TCP/IP für die Übertragung zwischen Knoten. **Anmerkung:** Der Wert dieser Registrierungsvariablen muss in allen Datenbankpartitionen des Exemplars gleich sein.		
DB2_VI_VIPL	Windows NT	Standardwert = `vipl.dll`
Gibt den Namen der VIPL (Virtual Interface Provider Library) an, die von DB2 verwendet wird. Der in dieser Registrierungsvariablen verwendete Bibliotheksname muss in der Benutzerumgebungsvariablen PATH angegeben werden, um die Bibliothek erfolgreich laden zu können. Die zurzeit unterstützten Implementierungen verwenden alle denselben Bibliotheksnamen.		
DB2_VI_DEVICE	Windows NT	Standardwert = Null Werte: `nic0` oder `VINIC`
Gibt den symbolischen Namen der Einheit oder des VIP-Exemplars (VIP - Virtual Interface Provider) an, die bzw. das der NIC (Network Interface Card) zugeordnet ist. Unabhängige Hardware-Lieferanten produzieren jeweils eigene NICs. Pro Windows NT-Maschine ist nur jeweils eine NIC zulässig. Mehrere logische Knoten auf derselben physischen Maschine benutzen die NIC gemeinsam. Der symbolische Name der Einheit „VINIC" muss in Großbuchstaben eingegeben werden und kann nur mit Synfinity Interconnect verwendet werden. Alle sonstigen derzeit unterstützen Implementierungen verwenden „nic0" als symbolischen Namen der Einheit.		

Tabelle 24. DCE-Verzeichnisvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2DIRPATHNAME	OS/2-, UNIX- und 32-Bit-Windows-Betriebssysteme	Standardwert = Null
Definiert einen temporären Wert, um den Wert des Parameters DIR_PATH_NAME in der Konfigurationsdatei des Datenbankmanagers außer Kraft zu setzen. Wenn ein Verzeichnis-Server verwendet wird und das Ziel einer Anweisung CONNECT oder ATTACH nicht explizit katalogisiert ist, wird der Zielname mit dem Wert von DB2DIRPATHNAME (wenn definiert) zur Bildung eines vollständig qualifizierten DCE-Namens verknüpft. **Anmerkung:** Der Wert von DB2DIRPATHNAME hat keine Auswirkung auf den Globalnamen des Exemplars, der stets durch die Konfigurationsparameter DIR_PATH_NAME und DIR_OBJ_NAME des Datenbankmanagers identifiziert wird.		
DB2CLIENTADPT	OS/2- und 32-Bit-Windows-Betriebssysteme	Standardwert = Null Bereich: 0-15
Definiert die Client-Adapternummer für das NETBIOS-Protokoll auf OS/2- und Windows-32-Bit-Betriebssystemen. Der Wert für DB2CLIENTADPT überschreibt den Wert des Parameters DFT_CLIENT_ADPT in der Konfigurationsdatei des Datenbankmanagers.		
DB2CLIENTCOMM	OS/2-, UNIX- und 32-Bit-Windows-Betriebssysteme	Standardwert = Null
Definiert einen temporären Wert, um den Wert des Parameters DFT_CLIENT_COMM in der Konfigurationsdatei des Datenbankmanagers außer Kraft zu setzen. Wenn weder DFT_CLIENT_COMM noch DB2CLIENTCOMM angegeben wird, wird das erste Protokoll verwendet, das im Objekt gefunden wird. Wenn einer von ihnen oder beide definiert sind, wird nur das erste übereinstimmende Protokoll verwendet. In beiden Fällen wird keine Wiederholung versucht, wenn die erste Verbindung fehlschlägt.		
DB2ROUTE	OS/2-, UNIX- und 32-Bit-Windows-Betriebssysteme	Standardwert = Null
Definiert den Namen des Objekts der Leitweginformationen, das der Client verwendet, wenn er die Verbindung zu einer Datenbank mit einem anderen Datenbankprotokoll herstellt. Der Wert für DB2ROUTE überschreibt den Wert des Parameters ROUTE_OBJ_NAME in der Konfigurationsdatei des Datenbankmanagers.		

Tabelle 25. Befehlszeilenvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2BQTIME	Alle	Standardwert = 1 Sekunde Maximalwert: 1 Sekunde

Tabelle 25. Befehlszeilenvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Definiert die Zeitdauer, die das Front-End des Befehlszeilenprozessors inaktiv bleibt, bevor es prüft, ob der Back-End-Prozess aktiv ist und eine Verbindung zum Front-End herstellt.		
DB2BQTRY	Alle	Standardwert = 60 Wiederholungen Minimalwert: 0 Wiederholungen
Definiert die Anzahl der Wiederholungen, die das Front-End des Befehlszeilenprozessors festzustellen versucht, ob der Back-End-Prozess bereits aktiv ist. Dieser Parameter arbeitet in Verbindung mit dem Parameter DB2BQTIME.		
DB2IQTIME	Alle	Standardwert = 5 Sekunden Minimalwert: 1 Sekunde
Definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors an der Eingabewarteschlange darauf wartet, dass der Front-End-Prozess Befehle übergibt.		
DB2RQTIME	Alle	Standardwert = 5 Sekunden Minimalwert: 1 Sekunde
Definiert die Zeitdauer, die der Back-End-Prozess des Befehlszeilenprozessors auf eine Anforderung vom Front-End-Prozess wartet.		

Tabelle 26. MPP-Konfigurationsvariablen

Variablenname	Betriebssystem	Werte
DB2ATLD_PORTS	DB2 UDB EEE auf AIX, Solaris und Windows NT	Standardwert = 6000:6063 Wert: num1:num2, wobei beide zwischen 1 und 65535 liegen und num1<=num2 gilt.
Gibt den Bereich der Anschlussnummern an, die für die interne TCPIP-Kommunikation des Dienstprogramms Autoloader (Programm für automatisches Laden) verwendet werden. Wenn kein Wert definiert ist, verwendet Autoloader den internen Standardanschlussbereich 6000:6063. Wenn andere Anwendungen den Standardanschlussbereich von Autoloader verwenden, kann diese Variable dazu verwendet werden, einen alternativen Anschlussbereich festzulegen.		
DB2ATLD_PWFILE	DB2 UDB EEE auf AIX, Solaris und Windows NT	Standardwert = Null Wert: ein Ausdruck für einen Dateipfad
Gibt einen Pfad zu einer Datei an, die ein Kennwort enthält, das bei der Autoloader-Identifikationsüberprüfung verwendet wird. Wenn kein Wert definiert ist, extrahiert Autoloader das Kennwort entweder aus der zugehörigen Konfigurationsdatei oder fordert Sie auf, es anzugeben. Die Verwendung dieser Variablen berücksichtigt Sicherheitsbelange und ermöglicht die Trennung von Autoloader-Konfigurationsdaten von Identifikationsüberprüfungsdaten.		

Tabelle 26. MPP-Konfigurationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
DB2CHGPWD_EEE	DB2 UDB EEE auf AIX und Windows NT	Standardwert = Null Werte: YES oder NO
Mit diesem Parameter können Sie zulassen, dass andere Benutzer Kennwörter auf EEE-Systemen unter AIX oder Windows NT ändern. Es muss sichergestellt werden, dass die Kennwörter für alle Partitionen oder Knoten mit Hilfe eines Windows NT-Domänen-Controllers unter Windows NT oder NIS unter AIX zentral verwaltet werden. Wenn Kennwörter nicht zentral verwaltet werden, bleiben sie möglicherweise nicht für alle Partitionen bzw. Knoten konsistent. Dies könnte dazu führen, dass ein Kennwort nur in der Datenbankpartition geändert wird, mit der der Benutzer zur Durchführung der Änderung verbunden ist. Um diese globale Registrierungsvariable zu ändern, müssen Sie sich im Stammverzeichnis und im DAS-Exemplar befinden.		
DB2_FORCE_FCM_BP	AIX	Standardwert = No Werte: Yes oder No
Diese Registrierungsvariable ist für DB2 UDB EEE für AIX gültig, wenn mehrere logische Partitionen verwendet werden. Nach dem Absetzen von DB2START ordnet DB2 die FCM-Puffer aus dem globalen Datenbankspeicher zu oder, falls dort nicht genügend Platz vorhanden ist, aus einem separaten, gemeinsam benutzten Speichersegment, das von allen FCM-Dämonen (für dieses Exemplar) auf derselben physischen Maschine verwendet wird. Welches Segment gewählt wird, hängt in erster Linie von der Anzahl der zu erstellenden FCM-Puffer ab (was wiederum durch den Konfigurationsparameter FCM_NUM_BUFFERS des Datenbankmanagers festgelegt wird). Hat diese Registrierungsvariable den Wert "Yes", werden die FCM-Puffer immer in einem separaten Speichersegment erstellt. Wenn die FCM-Puffer in einem separaten Speichersegment erstellt werden, erfolgt die Übertragung zwischen FCM-Dämonen verschiedener logischer Partitionen auf demselben physischen Knoten über den gemeinsam benutzten Speicher. Ansonsten läuft die Übertragung zwischen FCM-Dämonen auf demselben Knoten über UNIX-Sockets ab. Der Vorteil einer Übertragung über einen gemeinsam benutzten Speicher in dieser Weise liegt in der höheren Geschwindigkeit. Der Nachteil ist, dass für andere Verwendungszwecke, vor allem für Datenbankpufferpools, weniger gemeinsam benutzte Speichersegmente verfügbar sind. Hierdurch wird die maximale Größe der Datenbankpufferpools verringert.		
DB2_NUM_FAILOVER_NODES	Alle	Standardwert: 2 Werte: 0 bis zur Anzahl der logischen Knoten
Gibt die Nummer der Knoten an, die als als Funktionsübernahmeknoten in einer Umgebung mit hoher Verfügbarkeit verwendet werden können. Bei hoher Verfügbarkeit kann ein Knoten beim Auftreten eines Fehlers als zweiter logischer Knoten auf einem anderen Host erneut gestartet werden. Die für diese Variable verwendete Zahl legt fest, wie viel Speicher für FCM-Ressourcen für Funktionsübernahmeknoten reserviert ist. Beispielsweise verfügt Host A über zwei logische Knoten: 1 und 2; Host B verfügt über zwei logische Knoten: 3 und 4. Angenommen, die Variable DB2_NUM_FAILOVER_NODES ist auf den Wert 2 festgelegt. Während der Ausführung von DB2START reservieren Host A und auch Host B genügend Speicher für den FCM, damit bis zu vier logische Knoten verwaltet werden können. Wenn dann bei einem Host ein Fehler auftritt, können die logischen Knoten für den fehlerhaften Host auf dem anderen Host erneut gestartet werden.		

Tabelle 26. MPP-Konfigurationsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
DB2PORTRANGE	Windows NT	Werte: nnnn:nnnn
Dieser Wert wird auf den TCP/IP-Anschlussbereich gesetzt, der vom FCM verwendet wird, so dass alle zusätzlichen auf einer anderen Maschine erstellten Partitionen den gleichen Anschlussbereich haben.		
DB2_UPDATE_PART_KEY	ALL	Standardwert = YES Werte: Yes oder No
Der Standardwert für FixPak 3 und höher ist "Yes". Diese Registrierungsvariable bestimmt, ob die Aktualisierung des Partitionierungsschlüssels zulässig ist oder nicht.		

Tabelle 27. SQL-Compiler-Variablen

Variablenname Beschreibung	Betriebssystem	Werte
DB2_ANTIJOIN	Alle	Standardwert = NO in EEE-Umgebung Standardwert = YES in Nicht-EEE-Umgebung Werte: YES oder NO
In DB2 Universal Database EEE-Umgebungen: Bei Angabe von "Yes" sucht das Optimierungsprogramm nach Gelegenheiten, „NOT EXISTS"-Unterabfragen in Antiverknüpfungen (Antijoins) umzusetzen, die von DB2 effizienter verarbeitet werden können. In Nicht-EEE-Umgebungen: Bei Angabe von "No" schränkt das Optimierungsprogramm die Gelegenheiten für die Umsetzung von „NOT EXISTS"-Unterabfragen in Antiverknüpfungen (Antijoins) ein.		
DB2_CORRELATED_PREDICATES	Alle	Standardwert = YES Werte: Yes oder No
Der Standardwert für diese Variable ist "Yes". Wenn in einer Verknüpfung eindeutige Indizes in korrelierten Spalten vorhanden sind und diese Registrierungsvariable auf "Yes" eingestellt ist, versucht das Optimierungsprogramm, die Korrelation der Verknüpfungsvergleichselemente zu erkennen und auszugleichen. Ist diese Registrierungsvariable auf "Yes" eingestellt, stellt das Optimierungsprogramm mit Hilfe der KEYCARD-Informationen aus den Statistiken des eindeutigen Index die Fälle von Korrelation fest und passt die kombinierten Selektivitäten der korrelierten Vergleichselemente dynamisch an, wodurch eine genauere Schätzung der Größe und des Aufwands für die Verknüpfung erzielt wird. Auch für die Korrelation einfacher Gleichheitsvergleichselemente wie `WHERE C1=5 AND C2=10` erfolgt eine Anpassung, wenn ein Index für C1 und C2 existiert. Der Index muss nicht eindeutig sein, aber die Spalten der Gleichheitsvergleichselemente müssen alle Spalten des Index abdecken.		
DB2_HASH_JOIN	Alle	Standardwert =NO Werte: YES oder NO
Gibt die Hash-Verknüpfung als mögliche Verknüpfungsmethode bei der Kompilierung eines Zugriffsplans an.		

Tabelle 27. SQL-Compiler-Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_LIKE_VARCHAR	Alle	Standardwert = Y,N Werte: Y, N, S oder eine Gleitkommakonstante zwischen 0 und 6,2

Steuert die Sammlung und Verwendung von Statistikdaten zu Unterelementen. Dabei handelt es sich um Statistikdaten zum Inhalt von Spaltendaten, wenn diese eine Struktur in Form einer Folge von Unterfeldern oder Unterelementen aufweisen, die durch Leerzeichen getrennt sind.

Diese Registrierungsvariable bestimmt, wie das Optimierungsprogramm ein Vergleichselement der folgenden Form behandelt:

```
COLUMN LIKE '%xxxxxx%'
```

Dabei ist `xxxxxx` eine beliebige Zeichenfolge.

Die folgende Syntax zeigt, wie diese Registrierungsvariable verwendet wird:

```
db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]
```

Dabei gilt Folgendes:

- Der Term vor dem Komma bzw. der einzige Term rechts des Vergleichselements hat folgende Bedeutung, allerdings nur für Spalten, für die keine positiven Statistikdaten zu Unterelementen vorliegen:
  - S – Das Optimierungsprogramm schätzt die Länge der einzelnen Elemente einer Folge miteinander verknüpfter Elemente, die eine Spalte bilden, ausgehend von der Länge der Zeichenfolge, die von den %-Zeichen eingeschlossen wird.
  - Y – Der Standardwert. Standardwert 1,9 als Algorithmusparameter verwenden. Unterelement-Algorithmus mit variabler Länge als Algorithmusparameter verwenden.
  - N – Unterelement-Algorithmus mit fester Länge verwenden.
  - num1 – Wert von num1 als Algorithmusparameter für den Unterelement-Algorithmus mit variabler Länge verwenden.
- Der Term nach dem Komma hat folgende Bedeutung:
  - N – Der Standardwert. Keine Statistikdaten zu Unterelementen sammeln oder verwenden.
  - Y – Statistikdaten zu Unterelementen sammeln. Unterelement-Algorithmus mit variabler Länge, der die gesammelten Statistikdaten verwendet, zusammen mit dem Standardwert 1,9 als Algorithmusparameter verwenden, wenn Spalten mit positiven Statistikdaten zu Unterelementen vorliegen.
  - num2 – Statistikdaten zu Unterelementen sammeln. Unterelement-Algorithmus mit variabler Länge, der die gesammelten Statistikdaten verwendet, zusammen mit dem Wert von num2 als Algorithmusparameter verwenden, wenn Spalten mit positiven Statistikdaten zu Unterelementen vorliegen.

Tabelle 27. SQL-Compiler-Variablen (Forts.)

Variablenname	**Betriebssystem**	**Werte**
Beschreibung		
DB2_SELECTIVITY	ALL	Standardwert = No Werte: Yes oder No
Diese Registrierungsvariable bestimmt, ob die SELECTIVITY-Klausel verwendet werden kann. Detaillierte Informationen zur SELECTIVITY-Klausel finden Sie im Handbuch *SQL Reference* in den Abschnitten zu Sprachelementen und Suchbedingungen. Wenn diese Registrierungsvariable die Einstellung "Yes" aufweist, kann die SELECTIVITY-Klausel angegeben werden, wenn das Vergleichselement ein Basisvergleichselement ist und mindestens ein Ausdruck Host-Variablen enthält.		
DB2_NEW_CORR_SQ_FF	Alle	Standardwert = OFF Werte: ON oder OFF

Betrifft, wenn auf „ON " gesetzt, den Selektivitätswert, der vom SQL-Optimierungsprogramm für bestimmte Vergleichselemente von Unterabfragen ermittelt wird. Diese Variable dient zur Verbesserung der Genauigkeit des Selektivitätswerts von Gleichheitsvergleichselementen in Unterabfragen, die die Spaltenfunktion MIN oder MAX in der SELECT-Liste der Unterabfrage enthalten. Zum Beispiel:
`SELECT * FROM T WHERE`
`T.COL = (SELECT MIN(T.COL)`
`FROM T WHERE ...)`		

DB2_PRED_FACTORIZE	Alle	Standardwert =NO Werte: YES oder NO

Tabelle 27. SQL-Compiler-Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

Gibt an, ob das Optimierungsprogramm nach Gelegenheiten suchen soll, um zusätzliche Vergleichselemente aus Disjunktionen zu extrahieren. In manchen Fällen können die zusätzlichen Vergleichselemente die geschätzte Kardinalität der vorläufigen und endgültigen Ergebnismengen verändern. Mit der folgenden Abfrage:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`
kann das Optimierungsprogramm die folgenden zusätzlichen Vergleichselemente generieren:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE n1.lastname IN`
`('SMITH', 'JONES')`
`AND n2.lastname IN`
`('SMITH', 'JONES')`
`AND`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`		

Tabelle 28. Leistungsvariablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_AVOID_PREFETCH	Alle	Standardwert = OFF Werte: ON oder OFF
Definiert, ob ein Vorablesezugriff bei der Wiederherstellung nach einem Systemabsturz verwendet werden soll oder nicht. Wenn DB2_AVOID_PREFETCH= `ON` ist, wird der Vorablesezugriff nicht verwendet.		

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2_AWE	Windows 2000	Standardwert = Null Werte: <eintrag>[,<eintrag>,...] where <eintrag>=<pufferpool-ID>,<anzahl der physischen seiten>, <anzahl der adressfenster>
Ermöglicht DB2 UDB unter Windows 2000 die Zuordnung von Pufferpools, die bis zu 64 GB Speicher belegen. Die Unterstützung von AWE-Puffpools erfordert eine entsprechende Konfigurierung von Windows 2000. Dazu gehören die Zuordnung des Rechts „Seiten im Speicher sperren" für den Benutzer, die Zuordnung der physischen Seiten und der Adressfensterseiten sowie die Einstellung der Registrierungsvariablen. Zur Einstellung dieser Variablen benötigen Sie die Pufferpool-ID des Pufferpools, der für die AWE-Unterstützung verwendet werden soll. Die ID des Pufferpools können Sie der Spalte BUFFERPOOLID in der Systemkatalogsicht SYSCAT.BUFFERPOOLS entnehmen. **Anmerkung:** Wenn die AWE-Unterstützung aktiviert ist, kann für keinen der Pufferpools in der Datenbank erweiterter Speicher verwendet werden. Außerdem müssen Pufferpools, auf die diese Registrierungsvariable verweist, bereits in SYSCAT.SYSBUFFERPOOLS vorhanden sein.		
DB2_BINSORT	Alle	Standardwert = YES Werte: YES oder NO
Aktiviert einen neuen Sortieralgorithmus, der die CPU-Zeit und die abgelaufene Zeit für Sortierungen reduziert. Dieser neue Algorithmus weitet die extrem effiziente Integer-Sortiertechnik von DB2 UDB auf alle Sortierdatentypen wie BIGINT, CHAR, VARCHAR, FLOAT und DECIMAL sowie auf Kombinationen aus diesen Datentypen aus. Zur Aktivierung dieses neuen Algorithmus verwenden Sie den folgenden Befehl: `db2set DB2_BINSORT = yes`		
DB2BPVARS	Windows NT	Standardwert = Pfad

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

Gibt den Pfad zu einer Datei an, die Parameter zur Verwendung bei der Pufferpooloptimierung enthält. Zur Zeit werden folgende Parameter unterstützt: NT_SCATTER_DMSFILE, NT_SCATTER_DMSDEVICE und NT_SCATTER_SMS.
 	
Für jeden dieser Parameter ist der Standardwert 0 (bzw. OFF). Die möglichen Werte sind: 0 (oder OFF) und 1 (oder ON). Jeder Parameter dient zur Aktivierung des gestreuten Lesens (Scatter Read) für die jeweilige Art von Behältern. Jeder von ihnen kann nur aktiviert werden (ON), wenn DB2NTNOCACHE in der Registrierdatenbank auf den Wert ON gesetzt ist. Falls DB2NTNOCACHE den Wert OFF hat (bzw. nicht gesetzt ist), wird eine Warnung in die Protokolldatei db2diag.log geschrieben, und das gestreute Lesen bleibt inaktiv. Die Parameter werden für Systeme mit einem hohen Aufkommen an sequenziellen Vorableseoperationen für die jeweilige Art von Behälter empfohlen, und für die Sie bereits die Verwendung von DB2NTNOCACHE mit dem Wert ON beschlossen haben.
Anmerkung: Wenn Sie DB2NTNOCACHE auf ON setzen, inaktivieren Sie die Zwischenspeicherung von Dateien unter Windows NT.
 	
Das folgende Beispiel zeigt, wie der Pfad zu dieser Datei definiert wird:
`db2set DB2BPVARS =`
`f:\BPVARSFILE`
 	
Der Inhalt dieser Datei können beliebige dieser Parameter in folgender Form sein:
`parameter=wert`		

DB2CHKPTR	Alle	Standardwert = OFF Werte: ON oder OFF
Definiert, ob die Zeigerprüfung für Eingaben erforderlich ist oder nicht.		
DB2_ENABLE_BUFPD	Alle	Standardwert = OFF Werte: ON oder OFF
Gibt an, ob DB2 Pufferung zur Verbesserung der Abfrageleistung einsetzt. Die Pufferung führt möglicherweise nicht in allen Umgebungen zu einer Verbesserung der Abfrageleistung. Um einzelne Verbesserungen in der Abfrageleistung zu ermitteln, sollten Tests durchgeführt werden.		
DB2_EXTENDED_OPTIMIZATION	Alle	Standardwert = OFF Werte: ON oder OFF
Gibt an, ob das Abfrageoptimierungsprogramm die Optimierungserweiterungen zum Verbessern der Abfrageleistung verwendet. Die Erweiterungen verbessern die Abfrageleistung möglicherweise nicht in allen Umgebungen. Um einzelne Verbesserungen in der Abfrageleistung zu ermitteln, sollten Tests durchgeführt werden.		

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2MAXFSCRSEARCH	Alle	Standardwert = 5 Werte: -1, 1 bis 33 554
Gibt an, wie viele Steuersätze für freien Speicherbereich beim Hinzufügen eines Eintrags zu einer Tabelle durchsucht werden. Standardmäßig werden fünf Steuersätze für freien Speicherbereich durchsucht. Mit diesem Wert können Sie die Gewichtung zwischen Einfügegeschwindigkeit und Speicherwiederverwendung verändern. Mit hohen Werten optimieren Sie die Speicherwiederverwendung. Mit niedrigen Werten optimieren Sie die Einfügegeschwindigkeit. Der Wert -1 zwingt den Datenbankmanager, alle Steuersätze für freien Speicherbereich zu durchsuchen.		
DB2MEMDISCLAIM	AIX	Standardwert = YES Werte: YES oder NO
Unter AIX verfügt der von DB2-Prozessen verwendete Speicher u. U. über einen zugeordneten Paging-Bereich. Die Reservierung dieses Paging-Bereichs bleibt möglicherweise auch dann bestehen, wenn der zugehörige Speicher freigegeben wird. Ob dies der Fall ist, hängt von den (einstellbaren) Richtlinien für die Speicherzuordnung im Rahmen der Verwaltung des virtuellen Speichers auf dem AIX-System ab. Die Registrierungsvariable DB2MEMDISCLAIM steuert, ob DB2-Agenten AIX ausdrücklich auffordern, die Zuordnung des reservierten Paging-Bereichs zu dem freigegebenen Speicher aufzuheben. Wenn DB2MEMDISCLAIM auf YES gesetzt wird, sinkt der Bedarf an Paging-Bereich und möglicherweise auch die auf Seitenwechsel zurückzuführende Plattenaktivität. Wenn DB2MEMDISCLAIM auf NO gesetzt wird, steigt der Bedarf an Paging-Bereich und möglicherweise auch die auf Seitenwechsel zurückzuführende Plattenaktivität. In einigen Fällen, in denen reichlich Paging-Bereich und so viel Realspeicher verfügbar ist, dass keine Seitenwechsel auftreten, bedingt die Einstellung NO eine geringfügige Leistungssteigerung.		
DB2MEMMAXFREE	Alle	Standardwert = 8 388 608 Byte Werte: 0 bis 2^{32}-1 Byte
Gibt die maximale ungenutzte Speicherkapazität (in Byte) an, die für DB2-Prozesse reserviert wird.		
DB2_MMAP_READ	AIX	Standardwert =ON Werte: ON oder OFF
Wird in Verbindung mit DB2_MMAP_WRITE verwendet, damit DB2 mmap als alternative E/A-Methode verwenden kann. In den meisten Umgebungen sollte mmap verwendet werden, um Sperren des Betriebssystems zu vermeiden, wenn mehrere Prozesse parallel in verschiedene Abschnitte derselben Datei schreiben. Vielleicht haben Sie von Parallel Edition V1.2 umgestellt, wo der Standardwert OFF war, so dass AIX DB2-Daten im Cache zwischenspeichern konnte, die von JFS-Dateisystemen in den Speicher (außerhalb des Pufferpools) eingelesen wurden. Wenn Sie eine vergleichbare Leistung mit DB2 UDB erzielen möchten, können Sie entweder die Größe des Pufferpools erhöhen oder DB2_MMAP_READ und DB2_MMAP_WRITE in `OFF` ändern.		
DB2_MMAP_WRITE	AIX	Standardwert =ON Werte: ON oder OFF

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Wird in Verbindung mit DB2_MMAP_READ verwendet, damit DB2 mmap als alternative E/A-Methode verwenden kann. In den meisten Umgebungen sollte mmap verwendet werden, um Sperren des Betriebssystems zu vermeiden, wenn mehrere Prozesse parallel in verschiedene Abschnitte derselben Datei schreiben. Vielleicht haben Sie von Parallel Edition V1.2 umgestellt, wo der Standardwert OFF war, so dass AIX DB2-Daten im Cache zwischenspeichern konnte, die von JFS-Dateisystemen in den Speicher (außerhalb des Pufferpools) eingelesen wurden. Wenn Sie eine vergleichbare Leistung mit DB2 UDB erzielen möchten, können Sie entweder die Größe des Pufferpools erhöhen oder DB2_MMAP_READ und DB2_MMAP_WRITE in `OFF` ändern.		
DB2_NO_PKG_LOCK	Alle	Standardwert = OFF Werte: ON oder OFF
Erlaubt den Betrieb des globalen SQL-Cache ohne Verwendung von Paketsperren zum Schutz von Paketeinträgen, die im Cache zwischengespeichert werden. (Paketsperren sind interne Systemsperren.) Sie können nun zur Verbesserung der Leistung (das Einrichten und Freigeben von Sperren beansprucht Zeit) in einem Modus „ohne Paketsperren" arbeiten. In diesem Modus sind bestimmte Datenbankoperationen nicht zulässig. Zu diesen Operationen können zählen: Operationen, die Pakete ungültig machen, Operationen, die Pakete unbrauchbar machen, und Operationen, die Pakete direkt ändern.		
DB2NTMEMSIZE	Windows NT	Standardwert =(je nach Speichersegment unterschiedlich)
Windows NT erfordert, dass alle gemeinsam benutzten Speichersegmente während der DLL-Initialisierung reserviert werden, um übereinstimmende Adressen in allen Prozessen zu gewährleisten. Der neue Profilregistrierungswert DB2NTMEMSIZE wurde eingeführt, um Benutzern das Überschreiben der DB2-Standardwerte unter Windows NT (falls erforderlich) zu ermöglichen. In den meisten Situationen sollte der Standardwert ausreichen. Die Angaben für Speichersegmente, Standardgrößen und Überschreibungsoptionen lauten wie folgt: 1) Datenbank-Kernel: Standardgröße ist 16777216 (16 MB); Überschreibungsoption ist DBMS:<anzahl_byte>. 2) Parallele FCM-Puffer: Standardgröße ist 22020096 (21 MB); Überschreibungsoption ist FCM:<anzahl_byte>. 3) GUI für Datenbankverwaltung: Standardgröße ist 33554432 (32 MB); Überschreibungsoption ist DBAT:<anzahl_byte>. 4) Abgeschirmte gespeicherte Prozeduren: Standardgröße ist 16777216 (16 MB); Überschreibungsoption ist APLD:<anzahl_byte> Sie können mehrere Segmente überschreiben, indem Sie die Überschreibungsoptionen durch ein Semikolon (;) voneinander trennen. Wenn Sie zum Beispiel den Datenbank-Kernel auf ungefähr 256 KB und die FCM-Puffer auf ungefähr 64 MB begrenzen wollen, machen Sie folgende Angaben: `db2set DB2NTMEMSIZE=DBMS:256000;FCM:64000000`		
DB2NTNOCACHE	Windows NT	Standardwert = OFF Wert: ON oder OFF

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Definiert, ob DB2 Datenbankdateien mit einer Option NOCACHE öffnet oder nicht. Wenn DB2NTNOCACHE=ON definiert ist, wird das Zwischenspeichern im System-Cache inaktiviert. Wenn DB2NTNOCACHE=OFF definiert ist, speichert das Betriebssystem DB2-Dateien im Cache. Dies gilt für alle Daten außer für Dateien, die Langfelddaten oder LOB-Daten enthalten. Durch Inaktivieren der Cache-Funktion des Betriebssystems steht mehr Speicher für die Datenbank zur Verfügung, so dass der Pufferpool oder der Sortierspeicher vergrößert werden können.		
DB2NTPRICLASS	Windows NT	Standardwert = Null Wert: R, H, (beliebiger anderer Wert)
Stellt die Prioritätsklasse für das DB2-Exemplar ein (Programm DB2SYSCS.EXE). Es gibt drei Prioritätsklassen: • NORMAL_PRIORITY_CLASS (die Standardprioritätsklasse) • REALTIME_PRIORITY_CLASS (Wert „R") • HIGH_PRIORITY_CLASS (Wert „H") Diese Variable wird in Verbindung mit einzelnen Thread-Prioritäten (mit DB2PRIORITIES eingestellt) verwendet, um die absolute Priorität von DB2-Threads relativ zu anderen Threads im System zu bestimmen. **Anmerkung:** Bei Verwendung dieser Variablen ist Sorgfalt geboten. Eine falsche Verwendung kann sich negativ auf die Gesamtleistung des Systems auswirken. Weitere Informationen finden Sie bei der API *SetPriorityClass()* in der Win32-Dokumentation.		
DB2NTWORKSET	Windows NT	Standardwert =1,1
Dient zur Änderung der minimalen und der maximalen Größe des Arbeitsspeichers, der für DB2 verfügbar ist. Standardmäßig kann der Arbeitsspeicher eines Prozesses, wenn unter Windows NT keine Seitenwechsel auftreten, nach Bedarf wachsen. Wenn jedoch Seitenwechsel auftreten, liegt der maximale Arbeitsspeicher, den ein Prozess haben kann, bei annähernd 1 MB. Mit DB2NTWORKSET kann dieser Standardwert überschrieben werden. Geben Sie DB2NTWORKSET für DB2 in der Syntax DB2NTWORKSET=`min,max` an, wobei min und max in Megabyte angegeben werden.		
DB2_OVERRIDE_BPF	Alle	Standardwert = nicht gesetzt Werte: eine positive numerische Anzahl von Seiten

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt die Größe des Pufferpools in Seiten an, der bei der Aktivierung der Datenbank oder bei der Herstellung der ersten Verbindung zu erstellen ist. Diese Variable ist nützlich, wenn es während der Datenbankaktivierung oder der ersten Verbindung aufgrund begrenzter Speicherkapazitäten zu Fehlern kommt. Wenn selbst ein Pufferpool der Minimalgröße von 16 Seiten vom Datenbankmanager nicht bereitgestellt werden kann, hat der Benutzer die Möglichkeit, mit Hilfe dieser Umgebungsvariablen eine kleinere Anzahl von Seiten anzugeben und den Versuch zu wiederholen. Die Speicherknappheit kann wegen eines realen Speichermangels (selten) auftreten, oder durch den Versuch seitens des Datenbankmanagers verursacht werden, Speicher für große, nicht korrekt konfigurierte Pufferpools zuzuordnen. Wenn dieser Wert gesetzt wird, überschreibt er die aktuelle Pufferpoolgröße.		
DB2_PINNED_BP	AIX, HP-UX	Standardwert = NO Werte: YES oder NO

Mit dieser Variable kann der globale Datenbankspeicher (einschließlich Pufferpools), der der Datenbank zugeordnet ist, bei einigen AIX-Betriebssystemen im Hauptspeicher gehalten werden. Wenn dieser globale Datenbankspeicher im Hauptspeicher des Systems bleibt, ist eine konsistentere Datenbankleistung möglich.
 	
Eine Auslagerung des Pufferpools aus dem Systemhauptspeicher z. B. könnte zu einer verminderten Datenbankleistung führen. Die geringere Zahl von Lese- und Schreibvorgängen auf der Festplatte, wenn sich die Pufferpools im Systemspeicher befinden, verbessert die Datenbankleistung. Falls andere Anwendungen mehr Hauptspeicher benötigen, können Sie abhängig vom Systemhauptspeicherbedarf die Auslagerung des globalen Datenbankspeichers aus dem Hauptspeicher zulassen.
 	
Wenn Sie mit HP-UX in einer 64-Bit-Umgebung arbeiten, müssen Sie zusätzlich zur Änderung dieser Registrierungsvariablen der DB2-Exemplargruppe das Zugriffrecht MLOCK einräumen. Dazu müssen Sie als Benutzer mit Root-Berechtigung die folgenden Schritte ausführen:
1. Fügen Sie die DB2-Exemplargruppe der Datei `/etc/privgroup` hinzu. Wenn die DB-Exemplargruppe beispielsweise zur Gruppe `db2iadm1` gehört, müssen Sie der Datei `/etc/privgroup` die folgende Zeile hinzufügen:
`db2iadm1 MLOCK`
2. Setzen Sie den folgenden Befehl ab:
`setprivgrp -f /etc/privgroup`		

DB2PRIORITIES	Alle	Einstellung der Werte von der Plattform abhängig
Steuert die Prioritäten von DB2-Prozessen und Threads.		
DB2_RR_TO_RS	Alle	Standardwert =NO Werte: YES oder NO

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Next Key-Sperre garantiert die Isolationsstufe RR (Wiederholtes Lesen), indem automatisch der nächste Schlüssel für alle INSERT- und DELETE-Anweisungen und der nächsthöhere Schlüsselwert über der Ergebnisgruppe für SELECT-Anweisungen gesperrt wird. Für UPDATE-Anweisungen, die Schlüsselkomponenten eines Index ändern, wird der ursprüngliche Indexschlüssel gelöscht und der neue Schlüsselwert eingefügt. Die Sperrung des nächsten Schlüssels erfolgt sowohl beim Einfügen als auch beim Löschen von Schlüsseln. Die Sperrung des nächsten Schlüssels ist erforderlich, um die Isolationsstufe RR gemäß ANSI- und SQL92-Standard sicherzustellen und ist der DB2-Standardwert. Wenn es den Anschein hat, dass die Anwendung gestoppt oder blockiert wird, sollten Sie eine Momentaufnahme Ihrer Anwendung untersuchen. Wenn das Problem mit *Next Key-Sperre* im Zusammenhang zu stehen scheint, können Sie die Registrierungsvariable DB2_RR_TO_RS unter zwei Bedingungen aktivieren. Sie können DB2_RR_TO_RS aktivieren, wenn ein Verhalten gemäß der Isolationsstufe RR für keine Ihrer Anwendungen unabdingbar ist und wenn akzeptiert wird, dass Suchoperationen nicht festgeschriebene Löschoperationen überspringen. Das Überspringen bei Suchoperationen betrifft die Isolationsstufen RR, RS (Lesestabilität) und CS (Cursorstabilität). (Für die Isolationsstufe UR (Nicht festgeschriebener Lesevorgang) erfolgt keine Zeilensperre.) Wenn DB2_RR_TO_RS aktiviert ist, kann für die Suche in Benutzertabellen kein RR-Verhalten garantiert werden, da beim Einfügen und Löschen von Indexschlüsseln keine Sperrung des nächsten Schlüssels erfolgt. Katalogtabellen sind von dieser Option nicht betroffen. Die andere Verhaltensänderung bei Aktivierung von DB2_RR_TO_RS besteht darin, dass Suchoperationen gelöschte, aber nicht festgeschriebene Zeilen überspringen, auch wenn die Zeile die Suchbedingungen erfüllt.		
DB2_SORT_AFTER_TQ	Alle	Standardwert =NO Werte: YES oder NO
Gibt an, wie das Optimierungsprogramm mit übertragenen Tabellenwarteschlangen in einer partitionierten Datenbank arbeitet, wenn die Daten für den Empfänger sortiert sein müssen und die Anzahl der Empfängerknoten der Anzahl der Senderknoten entspricht. Wenn DB2_SORT_AFTER_TQ= `NO`, ist, sortiert das Optimierungsprogramm in der Regel auf der sendenden Seite und fügt die Zeilen auf der empfangenden Seite zusammen. Wenn DB2_SORT_AFTER_TQ= `YES` ist, überträgt das Optimierungsprogramm in der Regel die Zeilen unsortiert, fügt sie nicht auf der empfangenden Seite zusammen und sortiert die Zeilen auf der empfangenden Seite, nachdem alle Zeilen empfangen wurden.		
DB2_STPROC_LOOKUP_FIRST	Alle	Standardwert = OFF Werte: ON oder OFF

Tabelle 28. Leistungsvariablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Die frühere Variable DB2_DARI_LOOKUP_ALL gibt an, ob der UDB-Server eine Katalogsuche nach allen DARI-Prozeduren und gespeicherten Prozeduren durchführt, bevor er im Unterverzeichnis *function* des Unterverzeichnisses *sqllib* sowie im Unterverzeichnis *unfenced* des Unterverzeichnisses *function* des Unterverzeichnisses *sqllib* sucht. **Anmerkung:** Für gespeicherte Prozeduren mit PARAMETER TYPE DB2DARI, die sich in den oben erwähnten Verzeichnissen befinden, wird durch Setzen dieses Werts auf „ON" die Leistung herabgesetzt, da die Katalogsuche (möglicherweise auf einem anderen Knoten in einer EEE-Konfiguration) durchgeführt wird, bevor die Funktionsverzeichnisse durchsucht werden. Wenn Sie eine gespeicherte Prozedur aufrufen, durchsucht DB2 standardmäßig zunächst das Unterverzeichnis *function* des Unterverzeichnisses *sqllib* und das Unterverzeichnis *unfenced* des Unterverzeichnisses *function* des Unterverzeichnisses *sqllib* nach einer gemeinsam benutzten Bibliothek, die denselben Namen aufweist wie die gespeicherte Prozedur und sucht erst dann im Systemkatalog nach dem Namen der gemeinsam benutzten Bibliothek für die gespeicherten Prozeduren. Nur gespeicherte Prozeduren mit PARAMETER TYPE DB2DARI können denselben Namen aufweisen wie ihre gemeinsam benutzte Bibliothek, so dass nur gespeicherte DB2DARI-Prozeduren Nutzen aus dem Standardverhalten von DB2 ziehen. Wenn Sie gespeicherte Prozeduren einsetzen, die unter einem anderen PARAMETER TYPE katalogisiert sind, verschlechtert sich die Leistung dieser gespeicherten Prozeduren aufgrund der Zeit, die DB2 auf die Suche in den genannten Verzeichnissen verwendet. Zur Verbesserung der Leistung gespeicherter Prozeduren, die nicht als PARAMETER TYPE DB2DARI katalogisiert sind, setzen Sie die Registrierungsvariable DB2_STPROC_LOOKUP_FIRST auf ON. Diese Registrierungsvariable zwingt DB2, den Namen der gemeinsam benutzten Bibliothek der gespeicherten Prozedur zunächst im Systemkatalog zu suchen, bevor die genannten Verzeichnisse durchsucht werden.		

Tabelle 29. Data Links-Variablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DLFM_BACKUP_DIR_NAME	AIX, Windows NT	Standardwert: Null Werte: TSM bzw. ein beliebiger gültiger Pfad
Gibt die zu verwendende Sicherungseinheit an. Wenn Sie die Einstellung dieser Registrierungsvariablen zur Laufzeit von TSM in einen Pfad (oder umgekehrt) ändern, werden die archivierten Dateien nicht verschoben. Nur neue Sicherungen werden an der neuen Lokation platziert. Davor archivierte Dateien werden nicht verschoben.		
DLFM_BACKUP_LOCAL_MP	AIX, Windows NT	Standardwert: Null Werte: ein beliebiger gültiger Pfad zum lokalen Mount-Punkt im DFS-System

Tabelle 29. Data Links-Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt den vollständig qualifizierten Pfad zu einem Mount-Punkt im DFS-System an. Wenn ein Pfad angegeben wird, wird dieser anstatt dem in der Variablen DLFM_BACKUP_DIR_NAME angegebenen verwendet.		
DLFM_BACKUP_TARGET	AIX, Windows NT	Standardwert: Null Werte: LOCAL, TSM, XBSA
Gibt die verwendete Sicherungsart an.		
DLFM_BACKUP_TARGET_LIBRARY	AIX, Windows NT	Standardwert: Null Werte: ein beliebiger gültiger Pfad zum Namen der DLL oder der gemeinsam benutzten Bibliothek
Gibt den vollständig qualifizierten Pfad zur DLL oder zur gemeinsam benutzten Bibliothek an. Diese Bibliothek wird unter Verwendung der Bibliothek *libdfmxbsa.a* geladen.		
DLFM_ENABLE_STPROC	AIX, Windows NT	Standardwert: NO Werte: YES oder NO
Gibt an, ob eine gespeicherte Prozedur zum Verbinden von Gruppen von Dateien verwendet wird.		
DLFM_FS_ENVIRONMENT	AIX, Windows NT	Standardwert: NATIVE Werte: NATIVE oder DFS
Gibt die Umgebung an, in der Data Links-Server arbeiten. NATIVE zeigt an, dass der Data Links-Server auf einer einzigen Maschine ausgeführt wird, wobei der Server Dateien auf der eigenen Maschine übernehmen kann. DFS zeigt an, dass sich der Data Links-Server in einer DFS-Umgebung (DFS=Distributed Filesystem) befindet, wobei der Server Dateien aus dem gesamten Dateisystem übernehmen kann. Das Mischen von DFS-Dateigruppen und NATIVE-Dateisystemen ist nicht zulässig.		
DLFM_GC_MODE	AIX, Windows NT	Standardwert: PASSIVE Werte: SLEEP, PASSIVE, oder ACTIVE
Gibt die Steuerung der Bereinigung des Speichers für ungenutzte Dateien (Garbage File Collection) auf dem Data Links-Server an. Beim Wert SLEEP erfolgt keine Bereinigung. Wenn auf den Wert PASSIVE gesetzt, wird die Bereinigung nur dann ausgeführt, wenn keine anderen Transaktionen aktiv sind. Wenn auf ACTIVE gesetzt, wird die Speicherbereinigung durchgeführt, auch wenn andere Anwendungen aktiv sind.		
DLFM_INSTALL_PATH	AIX, Windows NT	Standardwert Unter AIX: /usr/lpp/ db2_06_00 /adm Unter NT: DB2PATH /bin Bereich: beliebiger gültiger Pfad

Tabelle 29. Data Links-Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt den Pfad an, in dem die ausführbaren Dateien für Data Links installiert sind.		
DLFM_LOG_LEVEL	AIX, Windows NT	Standardwert: LOG_INFO Werte: LOG_CRIT, LOG_DEBUG, LOG_ERR, LOG_INFO, LOG_NOTICE, LOG_WARNING
Gibt die Stufe der aufzuzeichnenden Diagnoseinformationen an.		
DLFM_PORT	Alle außer Windows 3.n	Standardwert: 50100 Werte: jede gültige Anschlussnummer
Gibt die Anschlussnummer an, die zur Kommunikation mit den Data Links-Servern verwendet wird, auf denen Data Links Manager für DB2 aktiv ist. Diese Umgebungsvariable wird nur verwendet, wenn eine Tabelle eine Spalte „DATALINKS" enthält.		
DLFM_TSM_MGMTCLASS	AIX, Windows NT, Solaris	Standardwert: die Standard-TSM-Verwaltungsklasse Werte: jede gültige TSM-Verwaltungsklasse
Gibt an, welche TSM-Verwaltungsklasse zum Archivieren und Abrufen von verknüpften Dateien verwendet wird. Wird für diese Variable kein Wert festgelegt, wird die TSM-Standardverwaltungsklasse verwendet.		

Tabelle 30. Verschiedene Variablen

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2ADMINSERVER	OS/2, Windows 95, Windows NT und UNIX	Standardwert = Null
Definiert, welches DB2-Exemplar als DB2-Verwaltungs-Server eingerichtet wird.		
DB2CLIINIPATH	Alle	Standardwert = Null
Dient zum Überschreiben des Standardpfads der DB2-CLI/ODBC-Konfigurationsdatei (db2cli.ini) und zum Angeben einer alternativen Speicherposition auf dem Client. Der definierte Wert muss ein gültiger Pfad auf dem Client-System sein.		
DB2DEFPREP	Alle	Standardwert = NO Werte: ALL, YES oder NO

Tabelle 30. Verschiedene Variablen (Forts.)

Variablenname	**Betriebssystem**	**Werte**
Beschreibung		
Simuliert das Laufzeitverhalten der Precompiler-Option DEFERRED_PREPARE für Anwendungen, die vor der Verfügbarkeit dieser Option vorkompiliert wurden. Wenn zum Beispiel eine Anwendung von DB2 Version 2.1.1 oder früher in einer Umgebung von DB2 Version 2.1.2 oder später ausgeführt würde, könnte der Parameter DB2DEFPREP verwendet werden, um das gewünschte Verhalten einer „verzögerten Vorbereitung" (Deferred Prepare) zu simulieren.		
DB2_DJ_COMM	Alle	Standardwert = Null Werte sind: libdrda.a, libsqlnet.a, libnet8.a, libdrda.dll, libsqlnet.dll, libnet8.dll usw.
Gibt die Oberflächenbibliotheken (Wrapper Libraries) an, die beim Start des Datenbankmanagers geladen werden. Durch Angeben dieser Variable wird der Echtzeitaufwand für das Laden häufig benutzter Oberflächenbibliotheken verringert. Andere Werte für andere Betriebssysteme werden unterstützt (die Erweiterung .dll ist für das Betriebssystem Windows NT, die Erweiterung .a für das Betriebssystem AIX). Die Bibliotheksnamen sind je nach Protokoll und Betriebssystem verschieden. Diese Variable steht nur zur Verfügung, wenn der Parameter *federated* des Datenbankmanagers auf den Wert YES gesetzt ist.		
DB2DMNBCKCTLR	Windows NT	Standardwert = Null Werte: ? oder ein Domänenname
Wenn Sie den Namen der Domäne kennen, für die der DB2-Server der Sicherungs-Domänen-Controller ist, stellen Sie DB2DMNBCKCTLR= `DOMAIN_NAME` ein. Die Angabe für DOMÄNENNAME muss in Großbuchstaben erfolgen. Sie können DB2 die Domäne ermitteln lassen, für die die lokale Maschine ein Sicherungs-Domänen-Controller ist, indem Sie DB2DMNBCKCTLR=? einstellen. Wenn die Profilvariable DB2DMNBCKCTLR nicht oder auf Null eingestellt ist, führt DB2 die Identifikationsüberprüfung auf dem primären Domänen-Controller aus. **Anmerkung:** DB2 verwendet standardmäßig keinen vorhandenen Sicherungs-Domänen-Controller, weil ein Sicherungs-Domänen-Controller die Synchronisation mit dem primären Domänen-Controller verlieren und damit ein Sicherheitsproblem verursachen kann. Es kann zum Verlust der Synchronisation kommen, wenn die Sicherheitsdatenbank des primären Domänen-Controllers aktualisiert wird, die Änderungen jedoch nicht an einen Sicherungs-Domänen-Controller weitergegeben werden. Der Grund hierfür könnten Netzwerklatenzzeiten oder ein nicht funktionierender Computer-Browser-Dienst sein.		
DB2_ENABLE_LDAP	Alle	Standardwert =NO Werte: YES oder NO
Gibt an, ob LDAP (Lightweight Directory Access Protocol) verwendet wird. LDAP ist eine Zugriffsmethode auf Verzeichnisservices.		
DB2_FALLBACK	Windows NT	Standardwert = OFF Werte: ON oder OFF

Tabelle 30. Verschiedene Variablen (Forts.)

Variablenname	**Betriebssystem**	**Werte**
Beschreibung		
Diese Variable erlaubt Ihnen, alle Datenbankverbindungen bei der FALLBACK-Verarbeitung zu trennen. Sie wird in Verbindung mit der Unterstützung für die Funktionsübernahme in der Windows NT-Umgebung mit Microsoft Cluster Server (MSCS) verwendet. Wenn DB2_FALLBACK nicht oder auf OFF gesetzt ist und eine Datenbankverbindung bei der FALLBACK-Operation besteht, kann die DB2-Ressource nicht offline genommen werden. Das bedeutet, dass die FALLBACK-Verarbeitung fehlschlägt.		
DB2_FORCE_TRUNCATION	Alle	Standardwert =NO Werte: YES oder NO
Wird bei der Neustartwiederherstellung verwendet. Wenn der Wert „NO" ist, wird die Neustartwiederherstellung angehalten, wenn ermittelt wird, dass eine defekte Seite die Neustartwiederherstellung zu früh stoppt (d. h. noch nicht alle aktiven Protokolle gelesen wurden). Dies wird meistens durch eine defekte Seite in einem der Protokolle verursacht. Der Benutzer kann diese Variable auf „YES" setzen, um die Neustartwiederherstellung anzuweisen, die Verarbeitung so fortzusetzen, als wäre das Ende der Protokolle erreicht worden. Nachdem die Variable auf „YES" gesetzt wurde, werden Protokolle, die bei der Neustartwiederherstellung nicht gelesen wurden, überschrieben, wenn die Datenbank wieder aktiv wird. Der Standardwert ist „NO", d. h., die Verarbeitung soll **nicht** fortgesetzt werden, wenn eine defekte Seite nicht gefunden wird. Verwenden Sie diese Variable nur, wenn Sie vom IBM Kundendienst dazu aufgefordert werden.		
DB2_GRP_LOOKUP	Windows NT	Standardwert = Null Werte: LOCAL, DOMAIN
Mit dieser Variable wird DB2 mitgeteilt, wo Benutzerkonten zu überprüfen und Gruppenmitgliedsuchen durchzuführen sind. Setzen Sie diese Variable auf LOCAL, um DB2 zu veranlassen, das Aufzählen von Gruppen und die Überprüfung von Benutzerkonten immer auf dem DB2-Server durchzuführen. Setzen Sie die Variable auf DOMAIN, um DB2 zu veranlassen, das Aufzählen von Gruppen und die Überprüfung von Benutzerkonten immer in der Windows NT-Domäne durchzuführen, zu der das Benutzerkonto gehört.		
DB2_INDEX_2BYTEVARLEN	Alle	Standardwert = NO Werte: YES oder NO
Mit dieser Registrierungsvariablen können Spalten mit einer Länge von mehr als 255 Bytes als Teil eines Indexschlüssels definiert werden. Für Indizes, die erstellt wurden, bevor diese Variable auf YES gesetzt wurde, gilt auch weiterhin die Beschränkung auf 255 Bytes. Indizes, die nach dem Setzen der Variablen auf "Yes" erstellt werden, verhalten sich wie ein Zwei-Byte-Index, und zwar auch dann, wenn die Einstellung der Registrierungsvariablen wieder in "No" geändert wird. Änderungen dieser Registrierungsvariablen wirken sich auf mehrere SQL-Anweisungen aus, darunter auch CREATE TABLE, CREATE INDEX und ALTER TABLE. Weitere Informationen über diese Anweisungen finden Sie in den für das Handbuch *SQL Reference* dokumentierten Änderungen.		

Tabelle 30. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
DB2LDAP_BASEDN	Alle	Standardwert = Null Werte: ein beliebiger Basisdomänenname.
Gibt den Basisdomänennamen für das LDAP-Verzeichnis an.		
DB2LDAPCACHE	Alle	Standardwert = YES Werte: YES oder NO

Gibt an, dass der LDAP-Cache aktiviert werden muss. Dieser Cache wird zum Katalogisieren der Datenbank-, Knoten- und DCS-Verzeichnisse auf der lokalen Maschine verwendet.
 	
Führen Sie die folgenden Befehle aus, um sicherzustellen, dass sich in Ihrem Cache die aktuellsten Einträge befinden:
`REFRESH LDAP DB DIR`
`REFRESH LDAP NODE DIR`
 	
Diese Befehle aktualisieren das Datenbank- und das Knotenverzeichnis und entfernen daraus falsche Einträge.		

DB2LDAP_CLIENT_PROVIDER	Nur Windows 95/98/NT/2000	Standardwert =Null (wenn verfügbar, wird Microsoft verwendet; andernfalls wird IBM verwendet.) Werte: IBM oder Microsoft
Bei der Ausführung in einer Windows-Umgebung unterstützt DB2 die Verwendung von Microsoft LDAP-Clients oder von IBM LDAP-Clients zum Zugriff auf das LDAP-Verzeichnis. Diese Registrierungsvariable wird dazu verwendet, den von DB2 zu verwendenden LDAP-Client explizit auszuwählen. **Anmerkung:** Verwenden Sie zum Anzeigen des aktuellen Werts dieser Registrierungsvariablen den Befehl `db2set`: `db2set DB2LDAP_CLIENT_PROVIDER`		
DB2LDAPHOST	Alle	Standardwert = Null Werte: ein beliebiger gültiger Host-Name.
Gibt den Host-Namen der Lokation mit dem LDAP-Verzeichnis an.		
DB2LDAP_SEARCH_SCOPE	Alle	Standardwert = DOMAIN Werte: LOCAL, DOMAIN, GLOBAL

Tabelle 30. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Gibt den Bereich für die Suche nach Informationen an, die in Partitionen oder Domänen im Lightweight Directory Access Protocol (LDAP) gefunden werden. Der Wert „LOCAL" inaktiviert das Suchen im LDAP-Verzeichnis. Bei dem Wert „DOMAIN" wird das LDAP für die aktuelle Verzeichnispartition durchsucht. Bei dem Wert „GLOBAL" wird das LDAP in allen Verzeichnispartitionen durchsucht, bis das Objekt gefunden wird.		
DB2LOADREC	Alle	Standardwert = Null
Dient zum Überschreiben der Speicherposition der Ladekopie bei einer aktualisierenden Wiederherstellung (ROLLFORWARD). Wenn der Benutzer die physische Speicherposition der Ladekopie geändert hat, muss der Parameter DB2LOADREC vor dem Ausführen der aktualisierenden Wiederherstellung definiert werden.		
DB2LOCK_TO_RB	Alle	Standardwert = Null Werte: Statement
Definiert, ob durch Sperrenzeitlimits die gesamte Transaktion oder nur die aktuelle Anweisung rückgängig gemacht werden soll. Wenn DB2LOCK_TO_RB den Wert `STATEMENT` hat, wird aufgrund eines Zeitlimits für Sperren nur die aktuelle Anweisung rückgängig gemacht. Jeder andere Wert für diesen Parameter sorgt dafür, dass die gesamte Transaktion rückgängig gemacht wird.		
DB2_NEWLOGPATH2	UNIX	Standardwert = 0 Werte: 0 oder 1
Dieser Parameter gibt Ihnen die Möglichkeit anzugeben, ob über einen zweiten Pfad eine doppelte Protokollierung eingerichtet werden soll. Der verwendete Pfad wird generiert, indem eine „2" an den aktuellen Wert des Datenbankkonfigurationsparameters *logpath* angefügt wird.		
DB2NOEXITLIST	Alle	Standardwert = OFF Werte: ON oder OFF
Wenn diese Variable definiert ist, weist sie DB2 an, keine Ausgangslistenroutine in Anwendungen zu installieren und keine COMMIT-Operation auszuführen. Normalerweise installiert DB2 eine Prozessausgangslistenroutine in Anwendungen, und die Ausgangslistenroutine führt eine COMMIT-Operation aus, wenn die Anwendung normal beendet wird. Bei Anwendungen, die die DB2-Bibliothek dynamisch laden und aus dem Speicher entfernen, bevor die Anwendung beendet ist, schlägt der Aufruf der Ausgangslistenroutine fehl, weil die Routine nicht mehr in der Anwendung geladen ist. Wenn Ihre Anwendung so arbeitet, müssen Sie die Variable DB2NOEXITLIST angeben und sicherstellen, dass Ihre Anwendung alle erforderlichen COMMIT-Operationen explizit aufruft.		
DB2REMOTEPREG	Windows 95 und Windows NT	Standardwert = Null Wert: Beliebiger gültiger Name einer Maschine mit Windows 95 oder Windows NT

Tabelle 30. Verschiedene Variablen (Forts.)

Variablenname	**Betriebssystem**	**Werte**
Beschreibung		
Definiert den Namen der fernen Maschine, die die Win32-Registrierdatenbankliste von DB2-Exemplarprofilen und DB2-Exemplaren enthält. Der Wert für DB2REMOTEPREG sollte nur einmal nach der Installation von DB2 definiert und anschließend nicht mehr geändert werden. Verwenden Sie diese Variable mit großer Vorsicht.		
DB2ROUTINE_DEBUG	AIX und Windows NT	Standardwert = OFF Werte: ON, OFF
Gibt an, ob die Fehlerbehebungsfunktion (Debug) für gespeicherte Java-Prozeduren aktiviert wird. Wenn sie nicht gerade Fehler in gespeicherten Java-Prozeduren beheben, sollten Sie den Standardwert OFF verwenden. Die Aktivierung des Fehlerbehebungsmodus hat Auswirkungen auf die Leistung. Weitere Informationen zur Debug-Funktion für gespeicherte Java-Prozeduren finden Sie im Handbuch *Application Development Guide*.		
DB2SORCVBUF	Windows 95 und Windows NT	Standardwert = 32767
Definiert den Wert von TCP/IP-Empfangspuffern auf den Betriebssystemen Windows 95 und Windows NT.		
DB2SORT	Alle, nur Server	Standardwert = Null
Definiert die Speicherposition einer während der Laufzeit durch das Dienstprogramm LOAD zu ladenden Bibliothek. Die Bibliothek enthält den Eingangspunkt für Funktionen, die beim Sortieren von Indexdaten verwendet werden. Verwenden Sie DB2SORT, um Sortierprogrammprodukte anderer Lieferanten mit dem Dienstprogramm LOAD zur Generierung von Tabellenindizes zu nutzen. Der angegebene Pfad muss relativ zum Datenbank-Server definiert werden.		
DB2SYSTEM	Windows NT, Windows 95, OS/2 und UNIX	Standardwert = Null

Tabelle 30. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		
Definiert den Namen, der von Ihren Benutzern und Datenbankadministratoren zur Identifizierung des DB2-Server-Systems verwendet wird. Dieser Name sollte nach Möglichkeit innerhalb Ihres Netzwerks eindeutig sein. Dieser Name wird auf der Systemebene der Objektbaumstruktur in der Steuerzentrale angezeigt, um Administratoren bei der Identifizierung von Server-Systemen zu helfen, die von der Steuerzentrale aus verwaltet werden können. Bei Verwendung der Funktion zum Durchsuchen des Netzwerks der „Client-Konfiguration - Unterstützung" gibt DB2-Discovery diesen Namen zurück und er wird auf der Systemebene der sich ergebenden Baumstruktur angezeigt. Dieser Name unterstützt Benutzer bei der Identifizierung des Systems, das die Datenbank enthält, auf die Sie zugreifen wollen. Bei der Installation wird für DB2SYSTEM wie folgt ein Wert festgelegt: • Unter Windows NT oder Windows 95 definiert das SETUP-Programm für diesen Wert den Computernamen, der für das Windows-System angegeben ist. • Unter OS/2 wird der Benutzer während der Installation aufgefordert, den Namen für das DB2-System (DB2SYSTEM) einzugeben. • Auf UNIX-Systemen wird für diesen Parameter der TCP/IP-Host-Name des UNIX-Systems definiert.		
DB2UPMPR	OS/2	Standardwert =ON Werte: ON oder OFF
Definiert, ob die Anmeldeanzeige der Benutzerprofilverwaltung (UPM) auf dem Bildschirm angezeigt wird, wenn der Benutzer eine falsche Benutzer-ID oder ein falsches Kennwort unter OS/2 eingibt.		
DB2_VENDOR_INI	AIX, HP-UX, Sun Solaris und Windows NT	Standardwert = Null Werte: eine beliebige gültige Angabe für Pfad und Datei
Verweist auf eine Datei, die alle herstellerspezifischen Umgebungseinstellungen enthält. Der Wert wird beim Start des Datenbankmanagers erfasst.		
DB2_XBSA_LIBRARY	AIX, HP-UX, Sun Solaris und Windows NT	Standardwert = Null Werte: eine beliebige gültige Angabe für Pfad und Datei

Tabelle 30. Verschiedene Variablen (Forts.)

Variablenname	Betriebssystem	Werte
Beschreibung		

Verweist auf die werkseitige XBSA-Bibliothek. Unter AIX muss die Einstellung das gemeinsam benutzte Objekt umfassen, sofern es nicht shr.o heißt. Für HP-UX, Sun Solaris und Windows NT wird der Name des gemeinsam benutzten Objekts nicht benötigt. Wenn z. B. das NetWorker Business Suite Module für DB2 von Legato verwendet werden soll, muss die Registrierungsvariable wie folgt eingestellt werden:
`db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"`
Die XBSA-Schnittstelle kann mit dem Befehl BACKUP DATABASE oder RESTORE DATABASE aufgerufen werden. Beispiel:
`db2 backup db sample use XBSA`
`db2 restore db sample use XBSA`		

Anhang B. EXPLAIN-Tabellen und Definitionen

Die EXPLAIN-Tabellen erfassen Zugriffspläne, wenn das Programm EXPLAIN aktiviert ist. Folgende EXPLAIN-Tabellen und Definitionen werden im vorliegenden Abschnitt beschrieben:

- „Die Tabelle EXPLAIN_ARGUMENT" auf Seite 610
- „Die Tabelle EXPLAIN_INSTANCE" auf Seite 614
- „Die Tabelle EXPLAIN_OBJECT" auf Seite 617
- „Die Tabelle EXPLAIN_OPERATOR" auf Seite 620
- „Die Tabelle EXPLAIN_PREDICATE" auf Seite 622
- „Die Tabelle EXPLAIN_STATEMENT" auf Seite 624
- „Die Tabelle EXPLAIN_STREAM" auf Seite 627
- „Die Tabelle ADVISE_INDEX" auf Seite 629
- „Die Tabelle ADVISE_WORKLOAD" auf Seite 633

Die EXPLAIN-Tabellen müssen erstellt werden, bevor EXPLAIN aufgerufen werden kann. Erstellen Sie diese Tabellen mit der Beispieleingabeprozedur des Befehlszeilenprozessors, die sich in der Datei EXPLAIN.DDL im Unterverzeichnis 'misc' des Verzeichnisses 'sqllib' befindet. Stellen Sie die Verbindung zu der Datenbank her, in der die EXPLAIN-Tabellen benötigt werden. Geben Sie dann den Befehl `db2 -tf EXPLAIN.DDL` ein, um die Tabellen zu erstellen. Weitere Informationen finden Sie in „Tabellendefinitionen für EXPLAIN-Tabellen" auf Seite 634.

Das Auffüllen der EXPLAIN-Tabellen mit Werten durch das Programm EXPLAIN aktiviert weder Auslöser noch referenzielle Integritätsbedingungen oder Prüfungen auf Integritätsbedingungen. Wenn zum Beispiel ein Einfügeauslöser in der Tabelle EXPLAIN_INSTANCE definiert wäre und EXPLAIN-Informationen über eine auswählbare Anweisung ausgegeben würden, würde der Auslöser nicht aktiviert.

Weitere Hinweise zur Einrichtung EXPLAIN finden Sie in „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251.

Legende für die EXPLAIN-Tabellen:

Überschrift	**Erläuterung**
Spaltenname	Name der Spalte
Datentyp	Datentyp der Spalte

Nullwert?	Ja: Nullwerte sind zulässig Nein: Nullwerte sind nicht zulässig
Schlüs- sel?	PK: Die Spalte ist Teil eines Primärschlüssels FK: Die Spalte ist Teil eines Fremdschlüssels
Beschreibung	Beschreibung der Spalte

Die Tabelle EXPLAIN_ARGUMENT

Die Tabelle EXPLAIN_ARGUMENT enthält gegebenenfalls vorhandene, eindeutige Kenndaten für jeden einzelnen Operator.

Tabelle 31. Die Tabelle EXPLAIN_ARGUMENT

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen.
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
OPERATOR_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Operator in dieser Abfrage
ARGUMENT_TYPE	CHAR(8)	Nein	Nein	Die Argumentart für diesen Operator
ARGUMENT_VALUE	VARCHAR(1024)	Ja	Nein	Der Argumentwert für diesen Operator. NULL, wenn sich der Wert in LONG_ARGUMENT_VALUE befindet.
LONG_ARGUMENT_VALUE	CLOB(1M)	Ja	Nein	Der Argumentwert für diesen Operator, wenn der Text nicht in ARGUMENT_VALUE passt. NULL, wenn sich der Wert in ARGUMENT_VALUE befindet.

Tabelle 32. Spaltenwerte für ARGUMENT_TYPE und ARGUMENT_VALUE

Wert für ARGUMENT_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	Partielle Bezugswerte für Spaltenberechnung
BITFLTR	TRUE FALSE	Die Hash-Verknüpfung verwendet einen Bitfilter zur Leistungsverbesserung.
CSETEMP	TRUE FALSE	Temporäre Tabelle über Markierung für allgemeinen Unterausdruck (Common Subexpression)
DIRECT	TRUE	Anzeiger für direkten Abruf
DUPLWARN	TRUE FALSE	Markierung für Warnung, dass gleiche Werte auftraten
EARLYOUT	TRUE FALSE	Anzeiger für frühes Verlassen (Early out)
ENVVAR	Jede Zeile dieses Typs enthält die folgenden Angaben: • Name der Umgebungsvariable • Wert der Umgebungsvariable	Sich auf das Optimierungsprogramm auswirkende Umgebungsvariable
FETCHMAX	IGNORE INTEGER	Außerkraftsetzen des Werts für das Argument MAXPAGES im Operator FETCH
GROUPBYC	TRUE FALSE	Gibt an, ob Spalten vorhanden sind, auf die die Klausel Group By angewandt wurde.
GROUPBYN	Ganze Zahl	Anzahl der Vergleichsspalten
GROUPBYR	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte in der Klausel Group By (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte	Anforderung der Klausel Group By
INNERCOL	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte in der Reihenfolge (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte • Wert für Reihenfolge **(A)** Aufsteigend **(D)** Absteigend	Innere Ordnungsspalten
ISCANMAX	IGNORE INTEGER	Außerkraftsetzen des Werts für das Argument MAXPAGES im Operator ISCAN

Tabelle 32. Spaltenwerte für ARGUMENT_TYPE und ARGUMENT_VALUE (Forts.)

Wert für ARGUMENT_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
JN_INPUT	INNER OUTER	Gibt an, ob Operator der Operator ist, der INNER oder OUTER für eine JOIN-Operation bereitstellt.
LISTENER	TRUE FALSE	Anzeiger für empfangsbereite Tabellenwarteschlange (Listener)
MAXPAGES	ALL NONE INTEGER	Maximale Anzahl der erwarteten Seiten für Vorablesezugriff
MAXRIDS	NONE INTEGER	Die maximale Anzahl der Zeilen-IDs muss in jeder Anforderung für Vorablesezugriff über Listen enthalten sein.
NUMROWS	INTEGER	Anzahl der erwarteten, zu sortierenden Zeilen
ONEFETCH	TRUE FALSE	Anzeiger für eine Abrufoperation (FETCH)
OUTERCOL	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte in der Reihenfolge (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte • Wert für Reihenfolge **(A)** Aufsteigend **(D)** Absteigend	Äußere Ordnungsspalten
OUTERJN	LEFT RIGHT	Anzeiger für äußere Verknüpfung
PARTCOLS	Name der Spalte	Partitionierungsspalten für den Operator
PREFETCH	LIST NONE SEQUENTIAL	Art des verfügbaren Vorablesezugriffs
RMTQTEXT	Abfragetext	Ferner Abfragetext
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	Vorgesehene Zeilensperre (Row Lock Intent)
ROWWIDTH	INTEGER	Länge der zu sortierenden Zeile
SCANDIR	FORWARD REVERSE	Suchrichtung

Tabelle 32. Spaltenwerte für ARGUMENT_TYPE und ARGUMENT_VALUE (Forts.)

Wert für ARGUMENT_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
SCANGRAN	INTEGER	Partitionsinterne Parallelität; Granularität der partitionsinternen Parallelsuche, ausgedrückt in SCANUNITs
SCANTYPE	LOCAL PARALLEL	Partitionsinterne Parallelität, Index- oder Tabellensuche
SCANUNIT	ROW PAGE	Partitionsinterne Parallelität, Einheit für Suchgranularität
SERVER	Ferner Server	Ferner Server
SHARED	TRUE	Partitionsinterne Parallelität, Anzeiger für gemeinsam benutztes TEMP
SLOWMAT	TRUE FALSE	Markierung für langsame Materialisierung (Slow Materialization)
SNGLPROD	TRUE FALSE	<-- Änderung aufgrund Antwort auf Append, AM, 21.7.98 --> Sortierung durch partitionsinterne Parallelität oder Tabelle TEMP von einzelnem Agenten erstellt.
SORTKEY	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte im Schlüssel (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte • Wert für Reihenfolge **(A)** Aufsteigend **(D)** Absteigend	Sortierschlüsselspalten
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	Partitionsinterne Parallelität, Sortiertyp
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	Vorgesehene Tabellensperre (Table Lock Intent)
TQDEGREE	INTEGER	Partitionsinterne Parallelität, Anzahl von Subagenten, die auf die Tabellenwarteschlange zugreifen
TQMERGE	TRUE FALSE	Angabe für (sortiertes) Mischen von Tabellenwarteschlangen

Tabelle 32. Spaltenwerte für ARGUMENT_TYPE und ARGUMENT_VALUE (Forts.)

Wert für ARGUMENT_TYPE	Gültige Werte für ARGUMENT_VALUE	Beschreibung
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	Lesemerkmal für Tabellenwarteschlange
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	Sendemerkmal für Tabellenwarteschlange
TQTYPE	LOCAL	Partitionsinterne Parallelität, Tabellenwarteschlange
TRUNCSRT	TRUE	Einschränkendes Sortieren (beschränkt die Anzahl erzeugter Zeilen)
UNIQUE	TRUE FALSE	Anzeiger für Eindeutigkeit
UNIQKEY	Jede Zeile dieses Typs enthält die folgenden Angaben: • Ordinalzahl der Spalte im Schlüssel (gefolgt von einem Doppelpunkt und einem Leerzeichen) • Name der Spalte	Eindeutige Schlüsselspalten
VOLATILE	TRUE	Flüchtige Tabelle

Die Tabelle EXPLAIN_INSTANCE

Die Tabelle EXPLAIN_INSTANCE ist die Hauptsteuertabelle für alle EXPLAIN-Informationen. Jede Datenzeile in den EXPLAIN-Tabellen ist explizit mit einer eindeutigen Zeile in dieser Tabelle verbunden. Die Tabelle EXPLAIN_INSTANCE enthält grundlegende Informationen über die Quelle der SQL-Anweisungen, für die EXPLAIN-Informationen ausgegeben werden, sowie Informationen über die Umgebung, in der die Ausgabe von EXPLAIN-Informationen stattfand.

Die Definition dieser Tabelle finden Sie in „Tabellendefinition EXPLAIN_INSTANCE" auf Seite 635.

Tabelle 33. Tabelle EXPLAIN_INSTANCE

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	PK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	PK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung

Tabelle 33. Tabelle EXPLAIN_INSTANCE (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
SOURCE_NAME	VARCHAR(128)	Nein	PK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	PK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_OPTION	CHAR(1)	Nein	Nein	Gibt an, welche EXPLAIN-Informationen bei dieser Anforderung angefordert wurden. Die folgenden Werte sind möglich: **P** PLANAUSWAHL
SNAPSHOT_TAKEN	CHAR(1)	Nein	Nein	Gibt an, ob eine EXPLAIN-Momentaufnahme für diese Anforderung erstellt wurde. Die folgenden Werte sind möglich: **Y** Es wurde eine oder mehrere EXPLAIN-Momentaufnahmen erstellt und in der Tabelle EXPLAIN_STATEMENT gespeichert. Darüber hinaus wurden auch reguläre EXPLAIN-Informationen erfasst. **N** Es wurde keine EXPLAIN-Momentaufnahme erstellt. Die regulären EXPLAIN-Informationen wurden jedoch erfasst. **O** Es wurde nur eine EXPLAIN-Momentaufnahme erstellt. Die regulären EXPLAIN-Informationen wurden nicht erfasst.
DB2_VERSION	CHAR(7)	Nein	Nein	Release-Nummer des Produkts DB2 Universal Database, das diese EXPLAIN-Anforderung verarbeitete. Das Format ist vv.rr.m, wobei Folgendes gilt: **vv** Versionsnummer **rr** Release-Nummer **m** Nummer des Wartungs-Release
SQL_TYPE	CHAR(1)	Nein	Nein	Gibt an, ob das EXPLAIN-Exemplar für statisches oder dynamisches SQL galt. Die folgenden Werte sind möglich: **S** Statisches SQL **D** Dynamisches SQL
QUERYOPT	INTEGER	Nein	Nein	Gibt die vom SQL-Compiler zum Zeitpunkt des EXPLAIN-Aufrufs verwendete Optimierungsklasse an. Der Wert gibt an, welche Stufe der Abfrageoptimierung durch den SQL-Compiler für die mit EXPLAIN bearbeiteten SQL-Anweisungen ausgeführt wurde.

Tabelle 33. Tabelle EXPLAIN_INSTANCE (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
BLOCK	CHAR(1)	Nein	Nein	Gibt an, welche Art von Cursor-Blockung beim Kompilieren der SQL-Anweisungen ausgeführt wurde. Weitere Informationen finden Sie in der Spalte BLOCK in der Katalogsicht SYSCAT.PACKAGES. Die folgenden Werte sind möglich: **N** Keine Blockung **U** Blockung eindeutiger Cursor **B** Blockung aller Cursor
ISOLATION	CHAR(2)	Nein	Nein	Gibt an, welche Isolationsstufe beim Kompilieren der SQL-Anweisungen verwendet wurde. Weitere Informationen finden Sie in der Spalte ISOLA-TION in der Katalogsicht SYSCAT.PACKAGES. Die folgenden Werte sind möglich: **RR** Wiederholtes Lesen (Repeatable Read) **RS** Lesestabilität (Read Stability) **CS** Cursorstabilität (Cursor Stability) **UR** Nicht festgeschriebener Lesevorgang (Uncommitted Read)
BUFFPAGE	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters BUFFPAGE für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
AVG_APPLS	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters AVG_APPLS zum Zeitpunkt des Aufrufs von EXPLAIN.
SORTHEAP	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters SORTHEAP für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
LOCKLIST	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters LOCKLIST für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
MAXLOCKS	SMALLINT	Nein	Nein	Enthält den Wert des Konfigurationsparameters MAXLOCKS für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
LOCKS_AVAIL	INTEGER	Nein	Nein	Enthält die Anzahl der Sperren, die nach Annahme des Optimierungsprogramms für jeden Benutzer verfügbar sind. (Von den Konfigurationsparametern LOCKLIST und MAXLOCKS abgeleitet.)
CPU_SPEED	DOUBLE	Nein	Nein	Enthält den Wert des Konfigurationsparameters CPUSPEED für den Datenbankmanager zum Zeit-punkt des Aufrufs von EXPLAIN.
REMARKS	VARCHAR(254)	Ja	Nein	Vom Benutzer angegebener Kommentar

Tabelle 33. Tabelle EXPLAIN_INSTANCE (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
DBHEAP	INTEGER	Nein	Nein	Enthält den Wert des Konfigurationsparameters DBHEAP für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
COMM_SPEED	DOUBLE	Nein	Nein	Enthält den Wert des Konfigurationsparameters COMM_BANDWIDTH für die Datenbank zum Zeitpunkt des Aufrufs von EXPLAIN.
PARALLELISM	CHAR(2)	Nein	Nein	Die folgenden Werte sind möglich: • N=Keine Parallelität • P=Partitionsinterne Parallelität • IP=Partitionsübergreifende Parallelität • BP=Partitionsinterne Parallelität und partitionsübergreifende Parallelität
DATAJOINER	CHAR(1)	Nein	Nein	Die folgenden Werte sind möglich: • N=Zugriffsplan von Systemen mit nicht zusammengeschlossenen Datenbanken • Y=Zugriffsplan von Systemen mit zusammengeschlossenen Datenbanken

Die Tabelle EXPLAIN_OBJECT

Die Tabelle EXPLAIN_OBJECT identifiziert die Datenobjekte, die für den Zugriffsplan erforderlich sind, der zur Ausführung der SQL-Anweisung generiert wurde.

Tabelle 34. Tabelle EXPLAIN_OBJECT

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen.
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen

Tabelle 34. Tabelle EXPLAIN_OBJECT (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
OBJECT_SCHEMA	VARCHAR(128)	Nein	Nein	Schema, zu dem dieses Objekt gehört.
OBJECT_NAME	VARCHAR(128)	Nein	Nein	Name des Objekts
OBJECT_TYPE	CHAR(2)	Nein	Nein	Beschreibende Kennung für den Objekttyp.
CREATE_TIME	TIMESTAMP	Ja	Nein	Zeitpunkt der Erstellung des Objekts. Null, wenn es sich um eine Tabellenfunktion handelt.
STATISTICS_TIME	TIMESTAMP	Ja	Nein	Zeitpunkt der letzten Statistikaktualisierung für dieses Objekt. Null, wenn keine Statistik für dieses Objekt vorhanden ist.
COLUMN_COUNT	SMALLINT	Nein	Nein	Anzahl der Spalten in diesem Objekt.
ROW_COUNT	INTEGER	Nein	Nein	Geschätzte Anzahl Zeilen in diesem Objekt.
WIDTH	INTEGER	Nein	Nein	Durchschnittliche Länge des Objekts in Byte. Bei einem Index auf -1 gesetzt.
PAGES	INTEGER	Nein	Nein	Geschätzte Anzahl Seiten, die das Objekt im Pufferpool einnimmt. Für eine Tabellenfunktion auf -1 gesetzt.
DISTINCT, Option	CHAR(1)	Nein	Nein	Gibt an, ob die Zeilen in dem Objekt eindeutig sind (d. h. keine gleichen Werte enthalten) Die folgenden Werte sind möglich: **Y** Ja **N** Nein
TABLESPACE_NAME	VARCHAR(128)	Ja	Nein	Name des Tabellenbereichs, in dem dieses Objekt gespeichert ist. Null, wenn kein Tabellenbereich benutzt wird.
OVERHEAD	DOUBLE	Nein	Nein	Geschätzter Gesamtaufwand in Millisekunden für eine einzelne wahlfreie Ein-/Ausgabeoperation mit dem angegebenen Tabellenbereich. Berücksichtigt die Aufwände für die Steuereinheit, für Plattensuchen und die Latenzzeit. Wert -1, wenn kein Tabellenbereich benutzt wird.
TRANSFER_RATE	DOUBLE	Nein	Nein	Geschätzte Zeit (in Millisekunden) zum Lesen einer Datenseite vom angegebenen Tabellenbereich. Wert -1, wenn kein Tabellenbereich benutzt wird.
PREFETCHSIZE	INTEGER	Nein	Nein	Die Anzahl der Datenseiten, die gelesen werden, wenn der Vorablesezugriff (Prefetch) aktiv ist. Für eine Tabellenfunktion auf -1 gesetzt.
EXTENTSIZE	INTEGER	Nein	Nein	Größe des Speicherbereichs in Datenseiten. Dies ist die Anzahl der 4-KB-Seiten, die in einen Behälter des Tabellenbereichs geschrieben werden, bevor zum nächsten Behälter gewechselt wird. Für eine Tabellenfunktion auf -1 gesetzt.

Tabelle 34. Tabelle EXPLAIN_OBJECT (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
CLUSTER	DOUBLE	Nein	Nein	Grad der Datenclusterbildung bezüglich des Index. Wenn>= 1, ist dies der Wert CLUSTER-RATIO. Wenn>= 0 und < 1, ist dies der Wert CLUSTERFACTOR. Für eine Tabelle, Tabellen-funktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
NLEAF	INTEGER	Nein	Nein	Die Anzahl der Blattseiten (Leaf pages), die von den Werten dieses Indexobjekts eingenommen werden. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
NLEVELS	INTEGER	Nein	Nein	Die Anzahl von Indexstufen in der Baumstruktur dieses Indexobjekts. Für eine Tabelle, Tabellen-funktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
FULLKEYCARD	BIGINT	Nein	Nein	Anzahl der unterschiedlichen vollständigen Schlüsselwerte in diesem Indexobjekt. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
OVERFLOW	INTEGER	Nein	Nein	Gesamtzahl der Überlaufsätze in der Tabelle. Für einen Index, eine Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert -1 gesetzt.
FIRSTKEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen ersten Schlüsselwerte. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.
FIRST2KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen ersten Schlüsselwerte bei Verwendung der ersten {2,3,4} Spalten des Index. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.
FIRST3KEYCARD	BIGINT	Nein	Nein	
FIRST4KEYCARD	BIGINT	Nein	Nein	
SEQUENTIAL_PAGES	INTEGER	Nein	Nein	Anzahl der Blattseiten (Leaf pages), die sich mit wenigen oder keinen größeren Abständen nach Indexschlüsseln sortiert auf der Platte befinden. Für eine Tabelle, Tabellenfunktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.
DENSITY	INTEGER	Nein	Nein	Verhältnis der SEQUENTIAL_PAGES zur Anzahl der Seiten im durch den Index belegten Seiten-bereich; ausgedrückt als Prozentwert (ganze Zahl zwischen 0 und 100). Für eine Tabelle, Tabellen-funktion oder wenn diese Statistik nicht verfügbar ist, auf den Wert –1 gesetzt.

Tabelle 35. Mögliche Werte für OBJECT_TYPE

Wert	Beschreibung
IX	Index
TA	Tabelle
TF	Tabellenfunktion

Die Tabelle EXPLAIN_OPERATOR

Die Tabelle EXPLAIN_OPERATOR enthält alle Operatoren, die der SQL-Compiler zum Ausführen der SQL-Anweisung benötigt.

Tabelle 36. Die Tabelle EXPLAIN_OPERATOR

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen.
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
OPERATOR_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Operator in dieser Abfrage
OPERATOR_TYPE	CHAR(6)	Nein	Nein	Beschreibender Kennsatz für den Operatortyp
TOTAL_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Gesamtaufwand (in Timerons) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich
IO_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer E/A-Aufwand (in Datenseitenein-/ausgaben) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich
CPU_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer CPU-Aufwand (in Instruktionen) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich

Tabelle 36. Die Tabelle EXPLAIN_OPERATOR (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
FIRST_ROW_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Aufwand (in Timerons) für das Abrufen der ersten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich. Dieser Wert enthält den gesamten erforderlichen Anfangsaufwand.
RE_TOTAL_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Aufwand (in Timerons) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich
RE_IO_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer E/A-Aufwand (in Datenseitenein-/ausgaben) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich
RE_CPU_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer CPU-Aufwand (in Timerons) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich
COMM_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Übertragungsaufwand (in TCP/IP-Rahmen) für die Ausführung des ausgewählten Zugriffsplans bis zu diesem Operator einschließlich
FIRST_COMM_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Übertragungsaufwand (in TCP/IP-Rahmen) für das Abrufen der nächsten Zeile für den ausgewählten Zugriffsplan bis zu diesem Operator einschließlich. Dieser Wert enthält den gesamten erforderlichen Anfangsaufwand.
BUFFERS	DOUBLE	Nein	Nein	Geschätzte Puffervoraussetzungen für diesen Operator und seine Eingaben
REMOTE_TOTAL_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Gesamtaufwand (in Timerons) für die Ausführung von Operationen für die ferne(n) Datenbank(en)
REMOTE_COMM_COST	DOUBLE	Nein	Nein	Geschätzter kumulativer Übertragungsaufwand für die Ausführung des ausgewählten fernen Zugriffsplans bis zu diesem Operator einschließlich

Tabelle 37. Werte für OPERATOR_TYPE

Wert	Beschreibung
DELETE	Löschen
FETCH	Abrufen
FILTER	Filterzeilen
GENROW	Zeile generieren
GRPBY	Gruppieren nach

Tabelle 37. Werte für OPERATOR_TYPE (Forts.)

Wert	Beschreibung
HSJOIN	Hash-Verknüpfung
INSERT	Einfügen
IXAND	Dynamische logische AND-Verknüpfung von Indizes über Bitzuordnungen (Index ANDing)
IXSCAN	Indexsuche
MSJOIN	Mischverknüpfung (Merge Join)
NLJOIN	Verknüpfung über Verschachtelungsschleifen (Nested Loop Join)
RETURN	Ergebnis
RIDSCN	Durchsuchen von Satz-IDs (RIDs)
RQUERY	Ferne Abfrage
SORT	Sortierung
TBSCAN	Tabellensuche
TEMP	Erstellen einer temporären Tabelle
TQ	Tabellenwarteschlange
UNION	Gesamtverknüpfung
UNIQUE	Eliminierung doppelter Werte
UPDATE	Aktualisierung

Die Tabelle EXPLAIN_PREDICATE

Die Tabelle EXPLAIN_PREDICATE gibt die Vergleichselemente an, die von einem bestimmten Operator angewandt werden.

Tabelle 38. Die Tabelle EXPLAIN_PREDICATE

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen.

Tabelle 38. Die Tabelle EXPLAIN_PREDICATE (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
OPERATOR_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Operator in dieser Abfrage
PREDICATE_ID	INTEGER	Nein	Nein	Eindeutige ID für dieses Vergleichselement für den angegebenen Operator
HOW_APPLIED	CHAR(5)	Nein	Nein	Verwendungsart des angegebenen Operators für dieses Vergleichselement
WHEN_EVALUATED	CHAR(3)	Nein	Nein	Kennzeichnet, wann die in diesem Vergleichs-element verwendete Unterabfrage ausgewertet wird. Die folgenden Werte sind möglich: **Leer** Dieses Vergleichselement enthält keine Unterabfrage. **EAA** Die in diesem Vergleichselement verwendete Unterabfrage wird bei der Anwendung des Vergleichselement (EAA) ausgewertet. D. h., sie wird bei der Anwendung des Vergleichselements für jede vom angegebenen Operator verarbeiteten Zeile erneut ausgewertet. **EAO** Die in diesem Vergleichselement verwendete Unterabfrage wird beim Öffnen des Vergleichselements (EAO) ausgewertet. D. h., sie wird für den angegebenen Operator nur einmal erneut ausgewertet, und ihre Ergebnisse werden bei der Anwendung des Vergleichselements für jede Zeile erneut verwendet. **MUL** Es gibt mehrere Arten von Unterabfragen in diesem Vergleichselement.
RELOP_TYPE	CHAR(2)	Nein	Nein	Der Vergleichsoperatortyp, der in diesem Vergleichselement verwendet wird
SUBQUERY	CHAR(1)	Nein	Nein	Gibt an, ob ein Datenstrom aus einer Unterabfrage für dieses Vergleichselement erforderlich ist. Es können mehrere Datenströme aus Unterabfragen erforderlich sein. Die folgenden Werte sind möglich: **N** Es ist kein Datenstrom aus einer Unterabfrage erforderlich. **Y** Ein oder mehrere Datenströme sind aus einer Unterabfrage erforderlich.

Tabelle 38. Die Tabelle EXPLAIN_PREDICATE (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
FILTER_FACTOR	DOUBLE	Nein	Nein	Der geschätzte Anteil der Zeilen, die durch dieses Vergleichselement qualifiziert wird
PREDICATE_TEXT	CLOB(1M)	Ja	Nein	Der Text des Vergleichselements wie aus der internen Darstellung der SQL-Anweisung erneut erstellt Null, wenn nicht vorhanden

Tabelle 39. Mögliche Werte für HOW_APPLIED

Wert	Beschreibung
JOIN	Zum Verknüpfen von Tabellen verwendet
RESID	Als Restvergleichselement ausgewertet
SARG	Wird ausgewertet als ein als Suchargument verwendbares Vergleichselement für Index oder Datenseite
START	Als Startbedingung verwendet
STOP	Als Stoppbedingung verwendet

Tabelle 40. Gültige Werte für RELOP_TYPE

Wert	Beschreibung
Leerzeichen	Nicht zutreffend
EQ	Gleich
GE	Größer-gleich
GT	Größer als
IN	In Liste
LE	Kleiner-gleich
LK	Ähnlich
LT	Kleiner als
NE	Ungleich
NL	Gleich null
NN	Ungleich null

Die Tabelle EXPLAIN_STATEMENT

Die Tabelle EXPLAIN_STATEMENT enthält den Text der SQL-Anweisung, wie sie für die verschiedenen Stufen der EXPLAIN-Informationen existiert. Die ursprüngliche SQL-Anweisung, wie sie vom Benutzer eingegeben wurde, wird in dieser Tabelle zusammen mit der (vom Optimierungsprogramm) zum Auswählen eines Zugriffsplans für die Ausführung der SQL-Anweisung verwendeten Version gespeichert. Die letztere Version hat oft sehr wenig Ähnlich-

keit mit dem Original, da sie möglicherweise vom SQL-Compiler umgeschrieben und/oder mit zusätzlichen Vergleichselementen versehen wurde.

Die Definition dieser Tabelle finden Sie in „Tabellendefinition EXPLAIN_STATEMENT" auf Seite 639.

Tabelle 41. Tabelle EXPLAIN_STATEMENT

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	PK, FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	PK, FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	PK, FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	PK, FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	PK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist Gültige Werte: **O** Originaltext (wie vom Benutzer eingegeben) **P** PLANAUSWAHL
STMTNO	INTEGER	Nein	PK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen. Bei dynamischen EXPLAIN-SQL-Anweisungen wird dieser Wert auf 1 gesetzt. Bei statischen SQL-Anweisungen gleicht dieser Wert dem für die Katalogsicht SYSCAT.STATEMENTS verwendeten Wert.
SECTNO	INTEGER	Nein	PK	Abschnittsnummer in dem Paket, die diese SQL-Anweisung enthält. Bei dynamischen SQL-Anweisungen EXPLAIN ist dies die Abschnittsnummer, die zum Speichern des Abschnitts für diese Anweisung zur Laufzeit verwendet wird. Bei statischen SQL-Anweisungen gleicht dieser Wert dem für die Katalogsicht SYSCAT.STATEMENTS verwendeten Wert.
QUERYNO	INTEGER	Nein	Nein	Numerische Kennung für eine mit EXPLAIN bearbeitete SQL-Anweisung. Für dynamische SQL-Anweisungen (außer der SQL-Anweisung EXPLAIN), die mit Hilfe von CLP oder CLI angegeben wurden, ist der Standardwert ein sequenziell inkrementierter Wert. Andernfalls ist der Standardwert für statische SQL-Anweisungen der Wert STMTNO und für dynamische SQL-Anweisungen der Wert 1.

Tabelle 41. Tabelle EXPLAIN_STATEMENT (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
QUERYTAG	CHAR(20)	Nein	Nein	Identifizierendes Kennzeichen für jede mit EXPLAIN bearbeitet SQL-Anweisung. Für dynamische SQL-Anweisungen, die mit CLP angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLP'. Für dynamische SQL-Anweisungen, die mit CLI angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLI'. Andernfalls werden Leerzeichen als Standardwert verwendet.
STATEMENT_TYPE	CHAR(2)	Nein	Nein	Beschreibender Kennsatz für den mit EXPLAIN bearbeiteten Abfragetyp. Die folgenden Werte sind möglich: **S** Auswählen **D** Löschen **DC** Löschen bei aktueller Cursorposition **I** Einfügen **U** Aktualisieren **UC** Aktualisieren bei aktueller Cursorposition
UPDATABLE	CHAR(1)	Nein	Nein	Zeigt an, ob diese Anweisung als aktualisierbar angesehen wird. Dies ist besonders wichtig für Anweisungen SELECT, die als potenziell aktualisierbar bestimmt werden können. Die folgenden Werte sind möglich: **' '** Nicht zutreffend (leer) **N** Nein **Y** Ja
DELETABLE	CHAR(1)	Nein	Nein	Zeigt an, ob diese Anweisung als löschbar angesehen wird. Dies ist besonders wichtig für Anweisungen SELECT, die als potenziell löschbar bestimmt werden können. Die folgenden Werte sind möglich: **' '** Nicht zutreffend (leer) **N** Nein **Y** Ja
TOTAL_COST	DOUBLE	Nein	Nein	Der geschätzte Gesamtaufwand (in Timerons) der Ausführung des ausgewählten Zugriffsplans für diese Anweisung; der Wert wird auf 0 (null) gesetzt, wenn EXPLAIN_LEVEL *O* ist (ursprünglicher Text), da zu diesem Zeitpunkt kein Zugriffsplan ausgewählt ist.

Tabelle 41. Tabelle EXPLAIN_STATEMENT (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
STATEMENT_TEXT	CLOB(1M)	Nein	Nein	Text oder Teil des Textes der mit EXPLAIN bearbeiteten SQL-Anweisung. Der Text, der für die Planauswahlstufe von EXPLAIN gezeigt wird, wurde aus der internen Darstellung rekonstruiert und ist SQL-ähnlich; das bedeutet, dass nicht garantiert wird, dass die rekonstruierte Anweisung der gültigen SQL-Syntax entspricht.
SNAPSHOT	BLOB(10M)	Ja	Nein	Momentaufnahme der internen Darstellung für diese SQL-Anweisung beim angezeigten Explain_Level. Diese Spalte ist für die Verwendung mit DB2 Visual Explain konzipiert. Die Spalte wird auf null gesetzt, wenn der Wert für EXPLAIN_LEVEL gleich *0* ist (ursprüngliche Anweisung), da zu dem Zeitpunkt, zu dem diese spezifische Version der Anweisung erfasst wird, kein Zugriffsplan ausgewählt ist.
QUERY_DEGREE	INTEGER	Nein	Nein	Gibt den Grad der partitionsinternen Parallelität zum Zeitpunkt des Aufrufs von EXPLAIN an. Für die Originalanweisung enthält diese Angabe den gesteuerten Grad der partitionsinternem Parallelität. Bei PLAN SELECTION enthält sie den Grad der partitionsinternen Parallelität, der für den zu verwendenden Zugriffsplan generiert wurde.

Die Tabelle EXPLAIN_STREAM

Die Tabelle EXPLAIN_STREAM zeigt den Eingabe- und Ausgabedatenstrom zwischen einzelnen Operatoren und Datenobjekten. Die Datenobjekte selbst werden in der Tabelle EXPLAIN_OBJECT beschrieben. Die an einem Datenstrom beteiligten Operatoren befinden sich in der Tabelle EXPLAIN_OPERATOR.

Tabelle 42. Die Tabelle EXPLAIN_STREAM

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	FK	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	FK	Initialisierungszeitpunkt für die EXPLAIN-Anforderung
SOURCE_NAME	VARCHAR(128)	Nein	FK	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.

Tabelle 42. Die Tabelle EXPLAIN_STREAM (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
SOURCE_SCHEMA	VARCHAR(128)	Nein	FK	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	FK	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	FK	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen.
SECTNO	INTEGER	Nein	FK	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
STREAM_ID	INTEGER	Nein	Nein	Eindeutige ID für diesen Datenstrom im angegebenen Operator
SOURCE_TYPE	CHAR(1)	Nein	Nein	Kennzeichnet die Quelle dieses Datenstroms: **O** Operator **D** Datenobjekt
SOURCE_ID	SMALLINT	Nein	Nein	Eindeutige ID für den Operator in dieser Abfrage, die die Quelle dieses Datenstroms ist. Wird auf -1 gesetzt, wenn SOURCE_TYPE 'D' ist.
TARGET_TYPE	CHAR(1)	Nein	Nein	Kennzeichnet das Ziel dieses Datenstroms: **O** Operator **D** Datenobjekt
TARGET_ID	SMALLINT	Nein	Nein	Eindeutige ID für den Operator in dieser Abfrage, die das Ziel dieses Datenstroms ist. Wird auf -1 gesetzt, wenn TARGET_TYPE 'D' ist.
OBJECT_SCHEMA	VARCHAR(128)	Ja	Nein	Schema, zu dem das betreffende Datenobjekt gehört. Wird auf null gesetzt, wenn SOURCE_TYPE und TARGET_TYPE gleich 'O' sind
OBJECT_NAME	VARCHAR(128)	Ja	Nein	Name des Objekts, das das Subjekt des Datenstroms ist. Wird auf null gesetzt, wenn SOURCE_TYPE und TARGET_TYPE gleich 'O' sind
STREAM_COUNT	DOUBLE	Nein	Nein	Geschätzte Kardinalität des Datenstroms
COLUMN_COUNT	SMALLINT	Nein	Nein	Anzahl der Spalten im Datenstrom
PREDICATE_ID	INTEGER	Nein	Nein	Wenn dieser Datenstrom Teil einer Unterabfrage für ein Vergleichselement ist, wird hier die Vergleichselement-ID wiedergegeben; andernfalls wird die Spalte auf -1 gesetzt.

Tabelle 42. Die Tabelle EXPLAIN_STREAM (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
COLUMN_NAMES	CLOB(1M)	Ja	Nein	Diese Spalte enthält die Namen und die Informationen zur Reihenfolge der von diesem Datenstrom betroffenen Spalten. Diese Namen haben folgendes Format: `NAME1(A)+NAME2(D)+NAME3+NAME4` Dabei gibt *(A)* eine aufsteigend sortierte Spalte an, *(D)* gibt eine absteigend sortierte Spalte an, und fehlende Sortierangaben bedeuten, dass die Spalte entweder nicht sortiert oder die Sortierfolge nicht relevant ist.
PMID	SMALLINT	Nein	Nein	ID der Partitionierungszuordnung
SINGLE_NODE	CHAR(5)	Ja	Nein	Gibt an, ob dieser Datenstrom sich auf einer einzigen oder auf mehreren Partitionen befindet: **MULT** Auf mehreren Partitionen **COOR** Auf Koordinatorknoten **HASH** Weiterleitung durch Hash-Verfahren **RID** Weiterleitung durch Zeilen-ID **FUNC** Weiterleitung durch Funktion (PARTITION() oder NODENUMBER()) **CORR** Weiterleitung durch Korrelationswert **Numerisch** Auf vorbestimmten einzigen Knoten gerichtet
PARTITION_COLUMNS	CLOB(64K)	Ja	Nein	Liste der Spalten, über die dieser Datenstrom partitioniert ist

Die Tabelle ADVISE_INDEX

Die Tabelle ADVISE_INDEX zeigt die empfohlenen Indizes.

Tabelle 43. Die Tabelle ADVISE_INDEX

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
EXPLAIN_REQUESTER	VARCHAR(128)	Nein	Nein	Berechtigungs-ID des Initiators dieser EXPLAIN-Anforderung
EXPLAIN_TIME	TIMESTAMP	Nein	Nein	Initialisierungszeitpunkt für die EXPLAIN-Anforderung

Tabelle 43. Die Tabelle ADVISE_INDEX (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
SOURCE_NAME	VARCHAR(128)	Nein	Nein	Name des Pakets, das ausgeführt wurde, als die dynamische Anweisung mit EXPLAIN bearbeitet wurde, oder der Name der Quellendatei, als das statische SQL mit EXPLAIN bearbeitet wurde.
SOURCE_SCHEMA	VARCHAR(128)	Nein	Nein	Schema oder Qualifikationsmerkmal der Quelle der EXPLAIN-Anforderung
EXPLAIN_LEVEL	CHAR(1)	Nein	Nein	Ebene der EXPLAIN-Informationen, für die diese Zeile relevant ist
STMTNO	INTEGER	Nein	Nein	Anweisungsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen.
SECTNO	INTEGER	Nein	Nein	Abschnittsnummer im Paket, auf die sich diese EXPLAIN-Informationen beziehen
QUERYNO	INTEGER	Nein	Nein	Numerische Kennung für eine mit EXPLAIN bearbeitete SQL-Anweisung. Für dynamische SQL-Anweisungen (außer der SQL-Anweisung EXPLAIN), die mit Hilfe von CLP oder CLI angegeben wurden, ist der Standardwert ein sequenziell inkrementierter Wert. Andernfalls ist der Standardwert für statische SQL-Anweisungen der Wert STMTNO und für dynamische SQL-Anweisungen der Wert 1.
QUERYTAG	CHAR(20)	Nein	Nein	Identifizierendes Kennzeichen für jede mit EXPLAIN bearbeitet SQL-Anweisung. Für dynamische SQL-Anweisungen, die mit CLP angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLP'. Für dynamische SQL-Anweisungen, die mit CLI angegeben wurden (außer der EXPLAIN-SQL-Anweisung), ist der Standardwert 'CLI'. Andernfalls werden Leerzeichen als Standardwert verwendet.
NAME	VARCHAR(128)	Nein	Nein	Name des Index.
CREATOR	VARCHAR(128)	Nein	Nein	Qualifikationsmerkmal des Indexnamens.
TBNAME	VARCHAR(128)	Nein	Nein	Name oder Kurzname der Tabelle, für die der Index definiert ist
TBCREATOR	VARCHAR(128)	Nein	Nein	Qualifikationsmerkmal des Tabellennamens
COLNAMES	CLOB(64K)	Nein	Nein	Liste mit Spaltennamen.
UNIQUERULE	CHAR(1)	Nein	Nein	Regel zur Eindeutigkeit: D=Gleiche Werte zulässig P=Primärindex U=Nur eindeutige Einträge zulässig
COLCOUNT	SMALLINT	Nein	Nein	Anzahl der Spalten im Schlüssel plus Anzahl der INCLUDE-Spalten, wenn vorhanden
IID	SMALLINT	Nein	Nein	Interne Index-ID

Tabelle 43. Die Tabelle ADVISE_INDEX (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
NLEAF	INTEGER	Nein	Nein	Anzahl der Blattseiten (Leaf pages); –1, wenn keine statistischen Daten erfasst wurden.
NLEVELS	SMALLINT	Nein	Nein	Anzahl der Indexstufen; –1, wenn keine statistischen Daten erfasst wurden.
FULLKEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen vollständigen Schlüsselwerte; –1, wenn keine statistischen Daten erfasst wurden.
FIRSTKEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen ersten Schlüsselwerte; –1, wenn keine statistischen Daten erfasst wurden.
CLUSTERRATIO	SMALLINT	Nein	Nein	Grad der Datenclusterbildung bezüglich des Index; –1, wenn keine statistischen Daten erfasst wurden oder wenn detaillierte statistische Daten zum Index erfasst wurden (in diesem Fall wird CLUSTERFACTOR verwendet).
CLUSTERFACTOR	DOUBLE	Nein	Nein	Feinere Erfassung des Grades der Clusterbildung oder –1, wenn keine detaillierte Indexstatistik gesammelt wurde oder wenn der Index für einen Kurznamen definiert ist.
USERDEFINED	SMALLINT	Nein	Nein	Vom Benutzer definiert.
SYSTEM_REQUIRED	SMALLINT	Nein	Nein	1, wenn dieser Index für eine Integritätsbedingung über Primärschlüssel oder eindeutige Schlüssel erforderlich ist ODER wenn er der Index für die Objektbezeichnerspalte (OID-Spalte) einer typisierten Tabelle ist. 2, wenn dieser Index für eine Integritätsbedingung über Primärschlüssel oder eindeutige Schlüssel erforderlich ist UND wenn er der Index für die Objektbezeichnerspalte (OID-Spalte) einer typisierten Tabelle ist. Ansonsten 0.
CREATE_TIME	TIMESTAMP	Nein	Nein	Zeitmarke für den Zeitpunkt der Indexerstellung.
STATS_TIME	TIMESTAMP	Ja	Nein	Zeitmarke für den Zeitpunkt der letzten Änderung an den für diesen Index aufgezeichneten Statistikdaten. Null, wenn keine Statistikdaten verfügbar sind.
PAGE_FETCH_PAIRS	VARCHAR(254)	Nein	Nein	Eine Liste von Paaren ganzer Zahlen, die in Zeichenform dargestellt sind. Jedes Paar zeigt die Anzahl der Seiten in einem angenommenen Puffer, und die Anzahl der Abrufoperationen für Seiten (Page fetches), die erforderlich wäre, um die Tabelle mit diesem Index unter Verwendung des angenommenen Puffers zu durchsuchen. (Eine Zeichenfolge der Länge 0, wenn keine Daten verfügbar sind.)
REMARKS	VARCHAR(254)	Ja	Nein	Ein Kommentar des Benutzers oder null.

Tabelle 43. Die Tabelle ADVISE_INDEX (Forts.)

Spaltenname	Datentyp	Null-wert?	Schlüs-sel?	Beschreibung
DEFINER	VARCHAR(128)	Nein	Nein	Der Benutzer, der den Index erstellt hat.
CONVERTED	CHAR(1)	Nein	Nein	Zur zukünftigen Verwendung reserviert.
SEQUENTIAL_PAGES	INTEGER	Nein	Nein	Anzahl der Blattseiten (Leaf pages), die sich mit wenigen oder keinen größeren Abständen nach Indexschlüsseln sortiert auf der Platte befinden. (–1, wenn keine statistischen Daten verfügbar sind.)
DENSITY	INTEGER	Nein	Nein	Verhältnis der SEQUENTIAL_PAGES zur Anzahl der Indexseiten ausgedrückt als Prozentwert (ganze Zahl zwischen 0 und 100, –1, wenn keine statistischen Daten verfügbar sind).
FIRST2KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen Schlüssel in den ersten beiden Spalten des Index (–1, wenn keine statistischen Daten gesammelt werden oder wenn dies nicht zutrifft)
FIRST3KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen Schlüssel in den ersten drei Spalten des Index (–1, wenn keine statistischen Daten gesammelt werden oder wenn dies nicht zutrifft)
FIRST4KEYCARD	BIGINT	Nein	Nein	Anzahl der eindeutigen Schlüssel in den ersten vier Spalten des Index (–1, wenn keine statistischen Daten gesammelt werden oder wenn dies nicht zutrifft)
PCTFREE	SMALLINT	Nein	Nein	Prozentsatz jeder äußeren Seite (Blattseite) des Index, der beim Erstellen des Index für spätere Einfügungen reserviert wird.
UNIQUE_COLCOUNT	SMALLINT	Nein	Nein	Anzahl der für einen eindeutigen Schlüssel erforderlichen Spalten. Immer <=COLCOUNT. < COLCOUNT nur, wenn INCLUDE-Spalten vorhanden sind. –1, wenn der Index keinen eindeutigen Schlüssel hat (gleiche Werte möglich).
MINPCTUSED	SMALLINT	Nein	Nein	Wenn nicht Null, dann ist die Online-Indexreorganisation aktiviert, und der Wert ist die Schwelle des minimal belegten Speicherbereichs vor dem Mischen der Seiten.
REVERSE_SCANS	CHAR(1)	Nein	Nein	Y=Index unterstützt umgekehrtes Durchsuchen N=Index unterstützt nicht umgekehrtes Durchsuchen
USE_INDEX	CHAR(1)	Ja	Nein	Y=Index empfohlen oder ausgewertet N=Index wird nicht empfohlen
CREATION_TEXT	CLOB(1M)	Nein	Nein	Die zum Erstellen des Index verwendete SQL-Anweisung.
PACKED_DESC	BLOB(20M)	Ja	Nein	Interne Beschreibung der Tabelle.

Die Tabelle ADVISE_WORKLOAD

Die Tabelle ADVISE_WORKLOAD zeigt die Anweisung, die die Auslastung bildet. Weitere Einzelheiten zur Auslastung finden Sie im Handbuch *Systemverwaltung: Optimierung*.

Tabelle 44. Die Tabelle ADVISE_WORKLOAD

Spaltenname	Datentyp	Nullwert?	Schlüssel?	Beschreibung
WORKLOAD_NAME	CHAR(128)	Nein	Nein	Name der Objektgruppe von SQL-Anweisungen (Auslastung), zu der diese Anweisung gehört.
STATEMENT_NO	INTEGER	Nein	Nein	Anweisungsnummer in der Auslastung, auf die sich diese EXPLAIN-Informationen beziehen.
STATEMENT_TEXT	CLOB(1M)	Nein	Nein	Inhalt der SQL-Anweisung.
STATEMENT_TAG	VARCHAR(256)	Nein	Nein	Identifizierendes Kennzeichen für jede mit EXPLAIN bearbeitete SQL-Anweisung.
FREQUENCY	INTEGER	Nein	Nein	Gibt an, wie oft diese Anweisung in der Auslastung erscheint.
IMPORTANCE	DOUBLE	Nein	Nein	Wichtigkeit der Anweisung.
COST_BEFORE	DOUBLE	Ja	Nein	Der Aufwand (in Timerons) der Abfrage, wenn die empfohlenen Indizes nicht erstellt werden.
COST_AFTER	DOUBLE	Ja	Nein	Der Aufwand (in Timerons) der Abfrage, wenn die empfohlenen Indizes erstellt werden.

Tabellendefinitionen für EXPLAIN-Tabellen

Die EXPLAIN-Tabellen müssen erstellt werden, bevor EXPLAIN aufgerufen werden kann. Die folgenden Definitionen geben an, wie die erforderlichen EXPLAIN-Tabellen erstellt werden können:

- „Tabellendefinition EXPLAIN_ARGUMENT"
- „Tabellendefinition EXPLAIN_INSTANCE" auf Seite 635
- „Tabellendefinition EXPLAIN_OBJECT" auf Seite 636
- „Tabellendefinition EXPLAIN_OPERATOR" auf Seite 637
- „Tabellendefinition EXPLAIN_PREDICATE" auf Seite 638
- „Tabellendefinition EXPLAIN_STATEMENT" auf Seite 639
- „Tabellendefinition EXPLAIN_STREAM" auf Seite 640
- „Tabellendefinition ADVISE_INDEX" auf Seite 641
- „Tabellendefinition ADVISE_WORKLOAD" auf Seite 643

Erstellen Sie diese Tabellen alternativ mit der Beispieleingabeprozedur des Befehlszeilenprozessors, die sich in der Datei EXPLAIN.DDL im Unterverzeichnis 'misc' des Verzeichnisses 'sqllib' befindet. Stellen Sie die Verbindung zu der Datenbank her, in der die EXPLAIN-Tabellen benötigt werden. Geben Sie dann den Befehl `db2 -tf EXPLAIN.DDL` ein, um die Tabellen zu erstellen.

Tabellendefinition EXPLAIN_ARGUMENT

```
CREATE TABLE EXPLAIN_ARGUMENT ( EXPLAIN_REQUESTER   VARCHAR(128)  NOT NULL,
                                EXPLAIN_TIME        TIMESTAMP     NOT NULL,
                                SOURCE_NAME         VARCHAR(128)  NOT NULL,
                                SOURCE_SCHEMA       VARCHAR(128)  NOT NULL,
                                EXPLAIN_LEVEL       CHAR(1)       NOT NULL,
                                STMTNO              INTEGER       NOT NULL,
                                SECTNO              INTEGER       NOT NULL,
                                OPERATOR_ID         INTEGER       NOT NULL,
                                ARGUMENT_TYPE       CHAR(8)       NOT NULL,
                                ARGUMENT_VALUE      VARCHAR(1024) NOT NULL,
                                LONG_ARGUMENT_VALUE CLOB(1M)      NOT LOGGED,
                                   FOREIGN KEY (EXPLAIN_REQUESTER,
                                                EXPLAIN_TIME,
                                                SOURCE_NAME,
                                                SOURCE_SCHEMA,
                                                EXPLAIN_LEVEL,
                                                STMTNO,
                                                SECTNO)
                                   REFERENCES EXPLAIN_STATEMENT
                                   ON DELETE CASCADE )
```

Tabellendefinition EXPLAIN_INSTANCE

```
CREATE TABLE EXPLAIN_INSTANCE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_OPTION    CHAR(1)      NOT NULL,
                                SNAPSHOT_TAKEN    CHAR(1)      NOT NULL,
                                DB2_VERSION       CHAR(7)      NOT NULL,
                                SQL_TYPE          CHAR(1)      NOT NULL,
                                QUERYOPT          INTEGER      NOT NULL,
                                BLOCK             CHAR(1)      NOT NULL,
                                ISOLATION         CHAR(2)      NOT NULL,
                                BUFFPAGE          INTEGER      NOT NULL,
                                AVG_APPLS         INTEGER      NOT NULL,
                                SORTHEAP          INTEGER      NOT NULL,
                                LOCKLIST          INTEGER      NOT NULL,
                                MAXLOCKS          SMALLINT     NOT NULL,
                                LOCKS_AVAIL       INTEGER      NOT NULL,
                                CPU_SPEED         DOUBLE       NOT NULL,
                                REMARKS           VARCHAR(254),
                                DBHEAP            INTEGER      NOT NULL,
                                COMM_SPEED        DOUBLE       NOT NULL,
                                PARALLELISM       CHAR(2)      NOT NULL,
                                DATAJOINER        CHAR(1)      NOT NULL,
                                     PRIMARY KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                 SOURCE_SCHEMA))
```

Tabellendefinition EXPLAIN_OBJECT

```
CREATE TABLE EXPLAIN_OBJECT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128) NOT NULL,
                              OBJECT_NAME       VARCHAR(128) NOT NULL,
                              OBJECT_TYPE       CHAR(2)      NOT NULL,
                              CREATE_TIME       TIMESTAMP,
                              STATISTICS_TIME   TIMESTAMP,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              ROW_COUNT         INTEGER      NOT NULL,
                              WIDTH             INTEGER      NOT NULL,
                              PAGES             INTEGER      NOT NULL,
                              DISTINCT          CHAR(1)      NOT NULL,
                              TABLESPACE_NAME   VARCHAR(128),
                              OVERHEAD          DOUBLE       NOT NULL,
                              TRANSFER_RATE     DOUBLE       NOT NULL,
                              PREFETCHSIZE      INTEGER      NOT NULL,
                              EXTENTSIZE        INTEGER      NOT NULL,
                              CLUSTER           DOUBLE       NOT NULL,
                              NLEAF             INTEGER      NOT NULL,
                              NLEVELS           INTEGER      NOT NULL,
                              FULLKEYCARD       BIGINT       NOT NULL,
                              OVERFLOW          INTEGER      NOT NULL,
                              FIRSTKEYCARD      BIGINT       NOT NULL,
                              FIRST2KEYCARD     BIGINT       NOT NULL,
                              FIRST3KEYCARD     BIGINT       NOT NULL,
                              FIRST4KEYCARD     BIGINT       NOT NULL,
                              SEQUENTIAL_PAGES  INTEGER      NOT NULL,
                              DENSITY           INTEGER      NOT NULL,
                                      FOREIGN KEY (EXPLAIN_REQUESTER,
                                                   EXPLAIN_TIME,
                                                   SOURCE_NAME,
                                                   SOURCE_SCHEMA,
                                                   EXPLAIN_LEVEL,
                                                   STMTNO,
                                                   SECTNO)
                                   REFERENCES EXPLAIN_STATEMENT
                                   ON DELETE CASCADE )
```

Tabellendefinition EXPLAIN_OPERATOR

```
CREATE TABLE EXPLAIN_OPERATOR ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                STMTNO            INTEGER      NOT NULL,
                                SECTNO            INTEGER      NOT NULL,
                                OPERATOR_ID       INTEGER      NOT NULL,
                                OPERATOR_TYPE     CHAR(6)      NOT NULL,
                                TOTAL_COST        DOUBLE       NOT NULL,
                                IO_COST           DOUBLE       NOT NULL,
                                CPU_COST          DOUBLE       NOT NULL,
                                FIRST_ROW_COST    DOUBLE       NOT NULL,
                                RE_TOTAL_COST     DOUBLE       NOT NULL,
                                RE_IO_COST        DOUBLE       NOT NULL,
                                RE_CPU_COST       DOUBLE       NOT NULL,
                                COMM_COST         DOUBLE       NOT NULL,
                                FIRST_COMM_COST   DOUBLE       NOT NULL,
                                REMOTE_TOTAL_COST DOUBLE       NOT NULL,
                                REMOTE_COMM_COST  DOUBLE       NOT NULL,
                                     FOREIGN KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA,
                                                  EXPLAIN_LEVEL,
                                                  STMTNO,
                                                  SECTNO)
                                     REFERENCES EXPLAIN_STATEMENT
                                     ON DELETE CASCADE )
```

Tabellendefinition EXPLAIN_PREDICATE

```
CREATE TABLE EXPLAIN_PREDICATE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 OPERATOR_ID       INTEGER      NOT NULL,
                                 PREDICATE_ID      INTEGER      NOT NULL,
                                 HOW_APPLIED       CHAR(5)      NOT NULL,
                                 WHEN_EVALUATED    CHAR(3)      NOT NULL,
                                 RELOP_TYPE        CHAR(2)      NOT NULL,
                                 SUBQUERY          CHAR(1)      NOT NULL,
                                 FILTER_FACTOR     DOUBLE       NOT NULL,
                                 PREDICATE_TEXT    CLOB(1M)     NOT LOGGED,
                                      FOREIGN KEY (EXPLAIN_REQUESTER,
                                                   EXPLAIN_TIME,
                                                   SOURCE_NAME,
                                                   SOURCE_SCHEMA,
                                                   EXPLAIN_LEVEL,
                                                   STMTNO,
                                                   SECTNO)
                                      REFERENCES EXPLAIN_STATEMENT
                                      ON DELETE CASCADE )
```

Tabellendefinition EXPLAIN_STATEMENT

```
CREATE TABLE EXPLAIN_STATEMENT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 QUERYNO           INTEGER      NOT NULL,
                                 QUERYTAG          CHAR(20)     NOT NULL,
                                 STATEMENT_TYPE    CHAR(2)      NOT NULL,
                                 UPDATABLE         CHAR(1)      NOT NULL,
                                 DELETABLE         CHAR(1)      NOT NULL
                                 TOTAL_COST        DOUBLE       NOT NULL,
                                 STATEMENT_TEXT    CLOB(1M)     NOT NULL
                                                                NOT LOGGED,
                                 SNAPSHOT          BLOB(10M)    NOT LOGGED,
                                 QUERY_DEGREE      INTEGER      NOT NULL,
                                     PRIMARY KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA,
                                                  EXPLAIN_LEVEL,
                                                  STMTNO,
                                                  SECTNO),
                                     FOREIGN KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA)
                                     REFERENCES EXPLAIN_INSTANCE
                                     ON DELETE CASCADE )
```

Tabellendefinition EXPLAIN_STREAM

```
CREATE TABLE EXPLAIN_STREAM ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              STREAM_ID         INTEGER      NOT NULL,
                              SOURCE_TYPE       CHAR(1)      NOT NULL,
                              SOURCE_ID         SMALLINT     NOT NULL,
                              TARGET_TYPE       CHAR(1)      NOT NULL,
                              TARGET_ID         SMALLINT     NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128),
                              OBJECT_NAME       VARCHAR(128),
                              STREAM_COUNT      DOUBLE       NOT NULL,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              PREDICATE_ID      INTEGER      NOT NULL,
                              COLUMN_NAMES      CLOB(1M)     NOT LOGGED,
                              PMID              SMALLINT     NOT NULL,
                              SINGLE_NODE       CHAR(5),
                              PARTITION_COLUMNS CLOB(64K)    NOT LOGGED,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

Tabellendefinition ADVISE_INDEX

```
CREATE TABLE ADVISE_INDEX (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           EXPLAIN_TIME      TIMESTAMP    NOT NULL
                              WITH DEFAULT CURRENT TIMESTAMP,
                           SOURCE_NAME       VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           SOURCE_SCHEMA     VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           EXPLAIN_LEVEL     CHAR(1)      NOT NULL
                              WITH DEFAULT '',
                           STMTNO            INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           SECTNO            INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           QUERYNO           INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           QUERYTAG          CHAR(20)     NOT NULL
                              WITH DEFAULT '',
                           NAME              VARCHAR(128) NOT NULL,
                           CREATOR           VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           TBNAME            VARCHAR(128) NOT NULL,
                           TBCREATOR         VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           COLNAMES          CLOB(64K)    NOT NULL,
                           UNIQUERULE        CHAR(1)      NOT NULL
                              WITH DEFAULT '',
                           COLCOUNT          SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           IID               SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           NLEAF             INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           NLEVELS           SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           FIRSTKEYCARD      BIGINT       NOT NULL
                              WITH DEFAULT 0,
                           FULLKEYCARD       BIGINT       NOT NULL
                              WITH DEFAULT 0,
                           CLUSTERRATIO      SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           CLUSTERFACTOR     DOUBLE       NOT NULL
                              WITH DEFAULT 0,
                           USERDEFINED       SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           SYSTEM_REQUIRED   SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           CREATE_TIME       TIMESTAMP    NOT NULL
                              WITH DEFAULT CURRENT TIMESTAMP,
                           STATS_TIME        TIMESTAMP
                              WITH DEFAULT CURRENT TIMESTAMP,
                           PAGE_FETCH_PAIRS  VARCHAR(254) NOT NULL
                              WITH DEFAULT '',
                           REMARKS           VARCHAR(254)
```

```
                        WITH DEFAULT '',
DEFINER           VARCHAR(128) NOT NULL
                        WITH DEFAULT '',
CONVERTED          CHAR(1) NOT NULL
                        WITH DEFAULT '',
SEQUENTIAL_PAGES  INTEGER      NOT NULL
                        WITH DEFAULT 0,
DENSITY           INTEGER      NOT NULL
                        WITH DEFAULT 0,
FIRST2KEYCARD     BIGINT       NOT NULL
                        WITH DEFAULT 0,
FIRST3KEYCARD     BIGINT       NOT NULL
                        WITH DEFAULT 0,
FIRST4KEYCARD     BIGINT       NOT NULL
                        WITH DEFAULT 0,
PCTFREE           SMALLINT     NOT NULL
                        WITH DEFAULT -1,
UNIQUE_COLCOUNT   SMALLINT     NOT NULL
                        WITH DEFAULT -1,
MINPCTUSED        SMALLINT     NOT NULL
                        WITH DEFAULT 0,
REVERSE_SCANS     CHAR(1)      NOT NULL
                        WITH DEFAULT 'N',
USE_INDEX         CHAR(1),
CREATION_TEXT     CLOB(1M)     NOT NULL
                        NOT LOGGED WITH DEFAULT '',
PACKED_DESC       BLOB(1M)     NOT LOGGED)
```

Tabellendefinition ADVISE_WORKLOAD

```
CREATE TABLE ADVISE_WORKLOAD (WORKLOAD_NAME  CHAR(128)    NOT NULL
                                WITH DEFAULT 'WK0',
                              STATEMENT_NO   INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              STATEMENT_TEXT CLOB(1M)     NOT NULL NOT LOGGED,
                              STATEMENT_TAG  VARCHAR(256) NOT NULL
                                WITH DEFAULT '',
                              FREQUENCY      INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              IMPORTANCE     DOUBLE       NOT NULL
                                WITH DEFAULT 1,
                              COST_BEFORE    DOUBLE,
                              COST_AFTER     DOUBLE)
```

Anhang C. EXPLAIN-Programme (SQL)

Das Tool **db2expln** beschreibt den Zugriffsplan, der für statische SQL-Anweisungen in den Paketen ausgewählt wurde, die in den Systemkatalogtabellen gespeichert werden. Mit diesem Tool kann eine schnelle Beschreibung des ausgewählten Zugriffsplans für Pakete angefordert werden, für die keine EXPLAIN-Daten beim Binden erfasst wurden.

Das Tool **dynexpln** beschreibt den Zugriffsplan, der für dynamische Anweisungen ausgewählt wurde. Es erstellt ein statisches Paket für die Anweisungen und verwendet dann das Tool **db2expln**, um sie zu beschreiben.

Sie können diese EXPLAIN-Tools verwenden, um sich einen Einblick in den für eine bestimmte SQL-Anweisung ausgewählten Plan zu verschaffen. Sie können aber auch die integrierte EXPLAIN-Einrichtung („Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251) zusammen mit Visual Explain verwenden, wenn Sie eine Beschreibung für den für eine bestimmte SQL-Anweisung ausgewählten Zugriffsplan erhalten möchten. Mit der EXPLAIN-Einrichtung können sowohl dynamische als auch statische SQL-Anweisungen bearbeitet werden. Ein Unterschied zu den EXPLAIN-Tools besteht darin, dass mit Visual Explain die EXPLAIN-Informationen in einem grafischen Format dargestellt werden. Ansonsten ist die Detaillierungsebene beider Methoden gleichwertig.

Damit Sie die Ausgabe von `db2expln` und `dynexpln` optimal verwenden können, müssen Sie mit folgenden Punkten vertraut sein:

- Die verschiedenen unterstützten SQL-Anweisungen und die für diese Anweisungen verwendete Terminologie (z. B. Vergleichselemente in einer Anweisung SELECT).
- Der Zweck eines Pakets (Zugriffsplans). (Siehe „Datenzugriffskonzepte und Optimierung" auf Seite 187.)
- Der Zweck und Inhalt der Systemkatalogtabellen. (Diese Informationen finden Sie im Handbuch *SQL Reference*.)
- Weitere Konzepte, die in „Teil 2. Optimieren der Anwendungsleistung" auf Seite 45 beschrieben werden.

Unter folgenden Themen finden Sie Informationen über `db2expln` und `dynexpln`:

- Ausführen von db2expln und dynexpln
- Syntax und Parameter für db2expln
- Hinweise zur Verwendung von db2expln

- Syntax und Parameter für dynexpln
- Hinweise zur Verwendung von dynexpln
- Beschreibung der Ausgabe von db2expln und dynexpln
- Beispiele für die Ausgabe von db2expln und dynexpln.

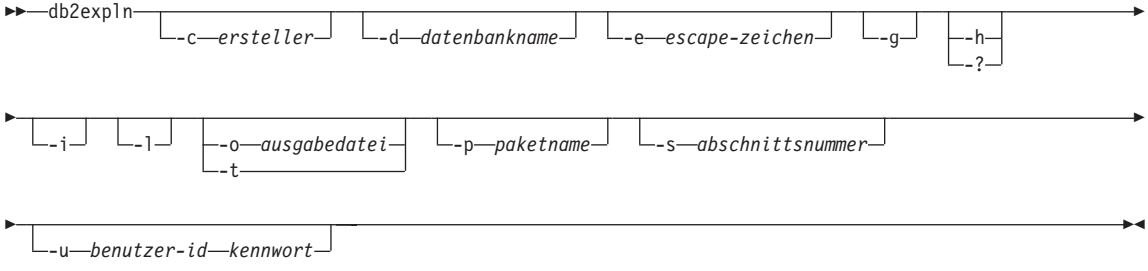
Ausführen von db2expln und dynexpln

Die EXPLAIN-Programme (`db2expln` und `dynexpln`) befinden sich im Unterverzeichnis `misc` des Verzeichnisses `sqllib` für Ihr Exemplar. Wenn sich `db2expln` und `dynexpln` nicht in Ihrem aktuellen Verzeichnis befinden, müssen sie sich in einem Verzeichnis befinden, das in Ihrer Umgebungsvariablen PATH definiert ist.

Das Programm `db2expln` stellt eine Verbindung zu einer Datenbank her und bindet sich mit Hilfe der Datei `db2expln.bnd` an die Datenbank, wenn zum ersten Mal auf die Datenbank zugegriffen wird. Die Datei `db2expln.bnd` befindet sich im Unterverzeichnis `bnd` Ihres Verzeichnisses `sqllib`.

Zum Ausführen von `db2expln` müssen Sie über das Zugriffsrecht SELECT für die Systemkatalogsichten sowie über die Ausführungsberechtigung (EXECUTE) für das `db2expln`-Paket verfügen. Für die Ausführung von `dynexpln` sind folgende Voraussetzungen erforderlich: Sie benötigen die Berechtigung BINDADD für die Datenbank, das Schema, das Sie für die Verbindungsherstellung zur Datenbank verwenden, muss existieren oder Sie müssen über die Berechtigung EXPLICIT_SCHEMA für die Datenbank verfügen und außerdem benötigen Sie alle Zugriffsrechte für die SQL-Anweisungen, die vom EXPLAIN-Programm bearbeitet werden. (Wenn Sie über SYSADM- oder DBADM-Berechtigung verfügen, haben Sie automatisch alle erforderlichen Berechtigungsstufen.)

Syntax und Parameter für db2expln



Dabei gilt folgendes:

-c ersteller
Die Benutzer-ID des Paketerstellers

Wenn Sie diese Option nicht angeben, werden Sie vom Programm zur Eingabe der entsprechenden Informationen aufgefordert.

Sie können zur Angabe des Namens des Erstellers die Platzhalterzeichen Prozentzeichen (%) und Unterstreichungszeichen (_) verwenden, die in einem Vergleichselement LIKE benutzt werden können.

-d datenbankname
Der Name der Datenbank, die die mit EXPLAIN zu bearbeitenden Pakete enthält

Wenn Sie diese Option nicht angeben, werden Sie vom Programm zur Eingabe der entsprechenden Informationen aufgefordert.

-e escape-zeichen
Dient zur Angabe des Zeichens, das als Escape-Zeichen und nicht als Platzhalterzeichen zu interpretieren ist.

Beispiel: Der `db2expln`-Befehl zur Bearbeitung des Pakets TESTID-.CALC% lautet `db2expln -c TESTID -p CALC%`. Mit diesem Befehl würden jedoch auch alle anderen Pläne, die mit `CALC` beginnen, von EXPLAIN bearbeitet. Wenn in diesem Fall nur das Paket TESTID-.CALC% bearbeitet werden soll, muss das Escape-Zeichen verwendet werden. Der Befehl könnte dann wie folgt geändert werden: `db2expln -c TESTID -e | -p CALC|%`. Sie definieren zunächst das Zeichen | als Escape-Zeichen, so dass anschließend |% als % interpretiert wird.

-g Anzeigen der Plandiagramme des Optimierungsprogramms. Jeder Abschnitt wird untersucht, und das ursprüngliche Plandiagramm des Optimierungsprogramms (wie von Visual Explain dargestellt) wird erstellt. Beachten Sie, dass das generierte Diagramm eventuell nicht mit dem ursprünglichen Plan übereinstimmt.

-h oder -?
Abrufen eines Hilfetextes zu den Eingabeparametern. Bei Angabe dieser Option werden alle anderen Optionen außer Kraft gesetzt.

-i Anzeigen der Operator-IDs im mit EXPLAIN bearbeiteten Plan. Die Operator-IDs ermöglichen das Abgleichen der Ausgabe von `db2expln` mit der Ausgabe der EXPLAIN-Einrichtung.

-l Wenn diese Option angegeben wird, kann der Paketname entweder in Kleinbuchstaben oder in gemischter Groß-/Kleinschreibung angegeben werden. Wenn die Option **-l** nicht angegeben wird, wird der Paketname in Großbuchstaben umgesetzt.

-o ausgabedatei
Der Name der Datei, in die `db2expln` die Ergebnisse schreibt

Wenn Sie **-o** ohne einen Dateinamen angeben, werden Sie aufgefordert, einen Dateinamen einzugeben. Der Standarddateiname ist `db2expln.out`.

-t Die Ausgabe wird an das Terminal geleitet.

Wenn **-o** oder **-t** nicht angegeben wird, werden Sie zur Eingabe eines Dateinamens aufgefordert; standardmäßig wird die Ausgabe am Terminal angezeigt.

-p paketname
Der Name des mit EXPLAIN zu bearbeitenden Pakets

Wenn Sie diese Option nicht angeben, werden Sie vom Programm zur Eingabe der entsprechenden Informationen aufgefordert.

Sie können zur Angabe des Paketnamens die Platzhalterzeichen Prozentzeichen (%) und Unterstreichungszeichen (_) verwenden, die in einem Vergleichselement LIKE benutzt werden können.

-s abschnittsnummer
Die Nummer des Abschnitts innerhalb des Pakets, der mit EXPLAIN bearbeitet werden soll. Wenn alle Abschnitte im Paket bearbeitet werden sollen, kann die Nummer 0 angegeben werden. Wenn die Argumente für den Paketersteller (**-c**) oder Paketnamen (**-p**) erkennen lassen, dass mehrere Pakete und daher auch mehrere Abschnitte bearbeitet werden, wird der Wert für den Abschnitt, falls ein solcher angegeben wurde, mit Null (0) überschrieben.

Wenn Sie diese Option nicht angeben, werden Sie vom Programm zur Eingabe der entsprechenden Informationen aufgefordert.

Abschnittsnummern können durch Abfragen des Systemkatalogs SYSCAT.STATEMENTS ermittelt werden (eine Beschreibung der Systemkatalogtabellen finden Sie in *SQL Reference*).

-u benutzer-id kennwort
Mit dieser Option werden eine Benutzer-ID und ein Kennwort für die Verbindung zu einer Datenbank angegeben.

Sowohl die Benutzer-ID als auch das Kennwort müssen gemäß den DB2-Namenskonventionen gültig sein und von der Datenbank erkannt werden.

Einige der in der obigen Tabelle aufgeführten Markierungen können eine besondere Bedeutung für Ihr Betriebssystem haben und infolgedessen in der Befehlszeile für den Befehl db2expln nicht richtig interpretiert werden. Diese Zeichen können jedoch möglicherweise eingegeben werden, wenn ihnen ein Escape-Zeichen vorangestellt wird. Weitere Informationen hierzu finden Sie im Benutzerhandbuch zu Ihrem Betriebssystem.

Hilfenachrichten und Nachrichten zum Anfangsstatus, die von db2expln erstellt werden, werden an die Standardausgabeeinheit geleitet. Alle Dialognachrichten und anderen Statusnachrichten, die durch das EXPLAIN-Programm erzeugt werden, werden an die Standardausgabeeinheit für Fehler geleitet. EXPLAIN-Text wird je nach der gewählten Ausgabeoption an die Standardausgabeeinheit oder in eine Datei geleitet.

Über die Optionen **-p** und **-c** können mehrere Pläne durch einen Programmaufruf bearbeitet werden, indem Zeichenfolgekonstanten für Pakete und Ersteller mit LIKE-Mustern angegeben werden. Das Unterstreichungszeichen (_) kann also als Platzhalterzeichen für ein Zeichen und das Prozentzeichen (%) als Platzhalterzeichen für null oder mehr Zeichen verwendet werden.

Wenn zum Beispiel alle Abschnitte für alle Pakete in der Datenbank SAMPLE bearbeitet und die Ergebnisse in der Datei **my.exp** gespeichert werden sollen, geben Sie Folgendes ein:

```
db2expln -d SAMPLE -p % -c % -s 0 -o my.exp
```

Hinweise zur Verwendung von db2expln

Es folgen einige allgemeine Nachrichten, die db2expln ausgeben kann:

- `No packages found for database <datenbank>, package pattern: <ersteller>.<paket>.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn keine Pakete in der Datenbank gefunden wurden, die dem angegebenen Muster entsprachen.

- `Bind messages can be found in db2expln.msg`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn das Binden von db2expln.bnd nicht erfolgreich war. Weitere Informationen über die aufgetretenen Fehler finden Sie in der Datei db2expln.msg im aktuellen Verzeichnis.

- `Section number overridden to 0 for potential multiple packages.`

  Die Nachricht, dass die Abschnittsnummer mit 0 überschrieben wurde, wird in der Ausgabe angezeigt, wenn von db2expln mehrere Pakete bearbeitet werden können. Diese Aktion wird ausgeführt, wenn eines der Platzhalterzeichen in den Eingabeargumenten für Pakete oder Ersteller verwendet wird.

- `No static sections qualify from package.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn das angegebene Paket nur dynamische SQL-Anweisungen enthält, was bedeutet, dass keine statischen Abschnitte existieren.
- `Database <datenbank>, package <ersteller>.<paket> is not valid. Rebind and then rerun db2expln.`

  Die Nachricht wird in der Ausgabe angezeigt, wenn das angegebene Paket momentan ungültig ist. Führen Sie, wie in der Nachricht angegeben, den Befehl BIND oder REBIND für den Plan neu aus, um erneut ein gültiges Paket in der Datenbank zu erstellen, und wiederholen Sie anschließend `db2expln`.
- `Section not processed: Produced by unsupported release.`

  Diese Nachricht wird ebenfalls in der Ausgabe angezeigt, wenn der momentan bearbeitete Abschnitt durch ein DB2-Release erstellt wurde, das von der vorliegenden Version von `db2expln` nicht unterstützt wird. Verwenden Sie in diesem Fall die Kopie von `db2expln` aus dem DB2-Release, das den Abschnitt erstellt hat.

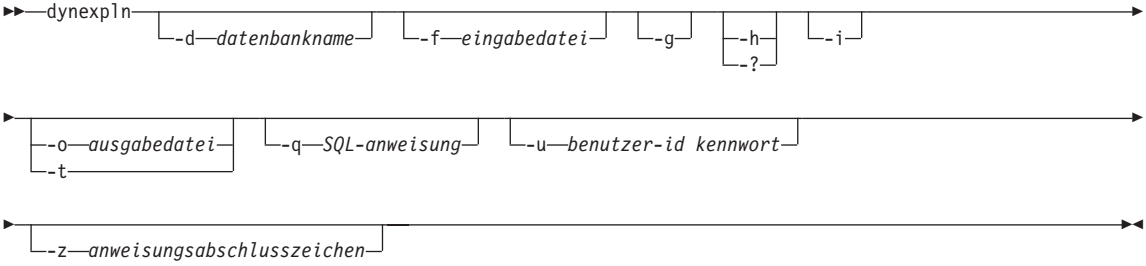
Von der Ausgabe ausgeschlossene SQL-Anweisungen: Die folgenden Anweisungen werden von EXPLAIN nicht bearbeitet:

- BEGIN/END DECLARE SECTION
- BEGIN/END COMPOUND
- INCLUDE
- WHENEVER
- COMMIT und ROLLBACK
- CONNECT
- OPEN Cursor
- FETCH
- CLOSE Cursor
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- DESCRIBE
- Dynamisches DECLARE CURSOR
- SQL-Steueranweisungen

Jede Unteranweisung innerhalb einer Compound-SQL-Anweisung kann über einen eigenen Abschnitt verfügen, der von **db2expln** bearbeitet werden kann.

Syntax und Parameter für dynexpln

Dabei gilt folgendes:



-d datenbankname
Der Name der Datenbank, die die mit EXPLAIN zu bearbeitenden Pakete enthält

Wenn Sie diese Option nicht angeben, werden Sie vom Programm zur Eingabe der entsprechenden Informationen aufgefordert.

-f eingabedatei
Der Name der Datei, die die mit EXPLAIN zu bearbeitenden SQL-Anweisungen enthält.

Sofern Sie die Option (**-z**) für das Abschlusszeichen nicht verwenden, darf jede Zeile der Datei nur eine SQL-Anweisung enthalten. SQL-Kommentare können in die Datei geschrieben werden. Ein SQL-Kommentar beginnt mit `--` und geht bis zum Ende der Zeile.

-g Anzeigen der Plandiagramme des Optimierungsprogramms. Jeder Abschnitt wird untersucht, und das ursprüngliche Plandiagramm des Optimierungsprogramms (wie von Visual Explain dargestellt) wird erstellt. Beachten Sie, dass das generierte Diagramm eventuell nicht mit dem ursprünglichen Plan übereinstimmt.

-h oder -?
Abrufen eines Hilfetextes zu den Eingabeparametern. Bei Angabe dieser Option werden alle anderen Optionen außer Kraft gesetzt.

-i Anzeigen der Operator-IDs im mit EXPLAIN bearbeiteten Plan. Die Operator-IDs ermöglichen das Abgleichen der Ausgabe von `db2expln` mit der Ausgabe der EXPLAIN-Einrichtung.

-o ausgabedatei
Der Name der Datei, in die `db2expln` die Ergebnisse schreibt

-t Die Ausgabe wird an das Terminal geleitet.

Wenn sowohl die Ausgabeoption (**-o**) als auch die Option **-t** angegeben werden, wird die Ausgabe an den Terminal geleitet.

Wenn weder die Option für die Ausgabedatei (**-o**) noch die Option **-t** angegeben werden, werden Sie zur Eingabe eines Dateinamens aufgefordert, wobei die Standardoption die Anzeige der Ausgabe am Terminal ist.

-q SQL-anweisung
Die SQL-Anweisung, die mit EXPLAIN bearbeitet werden soll.

Wenn Sie diese Option bzw. den wahlfreien Parameter (**-f**) für die Eingabedatei nicht angeben, werden Sie vom Programm zur Eingabe der zu bearbeitenden SQL-Anweisung aufgefordert.

Wenn Sie sowohl diese Option als auch den wahlfreien Parameter (**-f**) für die Eingabedatei angeben, beschreibt `dynexpln` zunächst die Anweisungen, die durch die SQL-Anweisungsoption (**-s**) angegeben wurden, und anschließend die Anweisungen in der Eingabedatei (**-f**).

-u benutzer-id kennwort
Mit dieser Option werden eine Benutzer-ID und ein Kennwort für die Verbindung zu einer Datenbank angegeben.

Sowohl die Benutzer-ID als auch das Kennwort müssen gemäß den DB2-Namenskonventionen gültig sein und von der Datenbank erkannt werden.

-z anweisungsabschlusszeichen
Das Zeichen, das dazu verwendet wird, das Ende einer SQL-Anweisung anzuzeigen.

Standardmäßig wird kein Anweisungsabschlusszeichen verwendet. Wird diese Option nicht verwendet, wird jede Zeile der Datei als getrennte SQL-Anweisung interpretiert. Wenn Sie diese Option angegen, verwendet `dynexpln` das angegebene Abschlusszeichen zur Trennung der Anweisungen.

Einige der in der obigen Tabelle aufgeführten Markierungen können eine besondere Bedeutung für Ihr Betriebssystem haben und infolgedessen in der Befehlszeile für den Befehl `dynexpln` nicht richtig interpretiert werden. Diese Zeichen können jedoch möglicherweise eingegeben werden, wenn ihnen ein Escape-Zeichen vorangestellt wird. Weitere Informationen hierzu finden Sie im Benutzerhandbuch zu Ihrem Betriebssystem.

Wenn Sie die Option (**-z**) für das Anweisungsabschlusszeichen verwenden, können Sie mit Hilfe der SQL-Anweisungsoption (**-s**) mehrere Anweisungen eingeben. In diesem Fall müssen Sie die einzelnen Anweisungen mit dem Abschlusszeichen trennen.

Hilfenachrichten und Nachrichten zum Anfangsstatus, die von `dynexpln` erstellt werden, werden an die Standardausgabeeinheit geleitet. Alle Dialog-

nachrichten und anderen Statusnachrichten, die durch das EXPLAIN-Programm erzeugt werden, werden an die Standardausgabeeinheit für Fehler geleitet. EXPLAIN-Text wird je nach der gewählten Ausgabeoption an die Standardausgabeeinheit oder in eine Datei geleitet.

Wenn zum Beispiel die Verbindung zu einer Datenbank SAMPLE hergestellt, alle Anweisungen in der Datei TRYIT mit EXPLAIN bearbeitet und die Ergebnisse in einer Datei my.exp gespeichert werden sollen, kann Folgendes eingegeben werden:

```
dynexpln -d SAMPLE -f TRYIT -o my.exp
```

Hinweise zur Verwendung von dynexpln

Zur Bearbeitung dynamischer Anweisungen erstellt dynexpln eine statische Anwendung für die Anweisungen und ruft anschließend db2expln auf. Zur Erstellung der statischen Anweisungen generiert dynexpln ein triviales C-Programm mit den Anweisungen und ruft den DB2-Precompiler zur Erstellung des Pakets auf. (Das generierte C-Programm ist nicht vollständig und kann nicht kompiliert werden. Es enthält nur so viele Informationen, dass der Precompiler das Paket erstellen kann.)

Es folgen einige allgemeine Nachrichten, die dynexpln ausgeben kann:

- Alle Fehlernachrichten aus db2expln

  Da dynexpln das Programm db2expln aufruft, können die Mehrzahl der Fehlernachrichten von db2expln in der Ausgabe auftreten.

- `Error connecting to the database.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn ein Fehler bei der Herstellung der Verbindung zur Datenbank aufgetreten ist. In einer weiteren CLI-Fehlernachricht wird die Ursache für das Fehlschlagen der Verbindung angezeigt. Beheben Sie die Fehlerursache, und führen Sie dynexpln erneut aus.

- `The file "<dateiname>" must be removed before dynexpln will run.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn die angegebene Datei zur Zeit der Ausführung von dynexpln existiert. Entfernen Sie die Datei, oder ändern Sie den Wert für die Umgebungsvariable DYNEXPLN_PACKAGE, um den Namen der Datei zu ändern, die erstellt wird, und führen Sie dynexpln erneut aus.

- `The package "<ersteller>.<paket>" must be dropped before dynexpln will run.`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn das angegebene Paket zur Zeit der Ausführung von dynexpln existiert. Löschen Sie das Paket, oder ändern Sie den Wert für die Umgebungsvariable DYNEXPLN_PACKAGE, um den Namen des Pakets zu ändern, das erstellt wird, und führen Sie dynexpln erneut aus.

- `Error writing file "<dateiname>".`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn in die Datei nicht geschrieben werden kann. Stellen Sie sicher, dass `dynexpln` in die Dateien des aktuellen Verzeichnisses schreiben kann, und führen Sie `dynexpln` erneut aus.
- `Error reading input file "<dateiname>".`

  Diese Nachricht wird in der Ausgabe angezeigt, wenn die mit der Option **-f** angegebene Datei nicht gelesen werden kann. Stellen Sie sicher, dass die Datei vorhanden ist und dass `dynexpln` sie lesen kann. Führen Sie anschließend `dynexpln` erneut aus.

Umgebungsvariablen: Es gibt zwei verschiedene Umgebungsvariablen, die in Verbindung mit `dynexpln` verwendet werden können:

- **DYNEXPLN_OPTIONS** definiert die Optionen für den SQL-Precompiler, die Sie bei der Erstellung des Pakets für Ihre Anweisungen verwenden. Verwenden Sie dieselbe Syntaxvariable wie bei der Eingabe eines Befehls PREP über CLP.

  Beispiel: `DYNEXPLN_OPTIONS="OPTLEVEL 5 BLOCKING ALL"`
- **DYNEXPLN_PACKAGE** definiert den Namen des Pakets, das in der Datenbank erstellt wird. Die mit EXPLAIN zu bearbeitenden Anweisungen werden in dieses Paket geschrieben. Falls diese Variable nicht definiert ist, erhält das Paket den Standardnamen **DYNEXPLN**. (Es werden nur die ersten acht Zeichen des Namens in dieser Umgebungsvariablen verwendet.)

  Der Name wird auch zur Generierung von Namen für von `dynexpln` benötigte temporäre Dateien verwendet.

Beschreibung der Ausgabe von db2expln und dynexpln

In der Ausgabe werden die EXPLAIN-Informationen für jedes Paket in zwei Komponenten unterteilt:

- Paketinformationen wie das Datum des Bindevorgangs und relevante Bindeoptionen
- Abschnittsinformationen wie die Abschnittsnummer, gefolgt von der mit EXPLAIN bearbeiteten SQL-Anweisung. Unter den Abschnittsinformationen befindet sich die EXPLAIN-Ausgabe des für die angezeigte SQL-Anweisung ausgewählten Zugriffsplans.

Die Schritte eines Zugriffsplans oder Abschnitts werden in der Reihenfolge aufgeführt, in der sie vom Datenbankmanager ausgeführt werden. Jeder Hauptschritt wird als linksbündig ausgerichtete Überschrift angezeigt. Informationen zum Schritt werden darunter eingerückt angezeigt. In der linken Spalte der EXPLAIN-Ausgabe für den Zugriffsplan befinden sich Einrückungsbalken. Diese Balken zeigen auch den ″Bereich″ der Operation an. Operationen niedrigerer Einrückungsstufe innerhalb derselben Operation

(d. h. solche, die weiter rechts angezeigt werden) werden verarbeitet, bevor zur vorherigen Einrückungsstufe zurückgekehrt wird.

Es ist zu beachten, dass der ausgewählte Zugriffsplan auf einer verbesserten Version der ursprünglichen SQL-Anweisung (der in der Ausgabe gezeigten) basiert. Beispielsweise kann die ursprüngliche Anweisung die Aktivierung einer beliebigen Anzahl von Auslösern und Einschränkungen bewirken. Die SQL-Anweisung kann nun von der Komponente zum Umschreiben von Abfragen des SQL-Compilers in ein gleichwertiges, aber effizienteres Format umgesetzt werden. Alle diese Faktoren sind in den Informationen enthalten, die an das Optimierungsprogramm weitergeleitet werden, wenn es den effizientesten Plan zum Ausführen der Anweisung ermittelt. Daher kann sich der in der EXPLAIN-Ausgabe angezeigte Zugriffsplan erheblich von dem unterscheiden, der für die ursprüngliche SQL-Anweisung vermutet werden könnte. Die integrierte EXPLAIN-Einrichtung zeigt die tatsächlich für die Optimierung verwendete SQL-Anweisung in Form einer Anweisung im SQL-Format an, die durch Rückübersetzung der internen Darstellung der Abfrage erstellt wird (eine Beschreibung der integrierten EXPLAIN-Einrichtung finden Sie in „Kapitel 7. Die SQL-EXPLAIN-Einrichtung" auf Seite 251).

Beim Vergleichen der Ausgabe von `db2expln` oder `dynexpln` mit der Ausgabe der EXPLAIN-Einrichtung kann die Option (**-i**) für die Operator-ID sehr nützlich sein. Jedesmal, wenn `db2expln` bzw. `dynexpln` die Verarbeitung eines neuen Operators aus der EXPLAIN-Einrichtung beginnt, wird die Operator-ID links neben dem mit EXPLAIN bearbeiteten Plan ausgegeben. Die Operator-ID kann zum Abgleichen der Schritte in den verschiedenen Darstellungen des Zugriffsplans verwendet werden. Beachten Sie, dass es nicht immer eine Eins-zu-eins-Entsprechung gibt zwischen den Operatoren in der Ausgabe der EXPLAIN-Einrichtung und den Operationen, die in der Ausgabe von `db2expln` und `dynexpln` angezeigt werden.

Unter den folgenden Themen wird der EXPLAIN-Text beschrieben, der von `db2expln` und `dynexpln` generiert werden kann:

- Zugriff auf Tabellen
- Temporäre Tabellen
- Verknüpfungen
- Datenströme
- Einfügen, Aktualisieren und Löschen
- Vorbereitung von Satzkennungen (Satz-IDs)
- Spaltenberechnungen
- Parallelverarbeitung
- Verarbeitung von Anweisungen in zusammengeschlossenen Datenbanken
- Verschiedene Angaben.

Zugriff auf Tabellen

Diese Angabe zeigt den Namen und den Typ der Tabelle an, auf die zugegriffen wird. Es werden zwei Formate verwendet:

1. Drei Typen von regulären Tabellen:
   - Zugriff auf Tabellenname (Access Table Name):

     ```
     Access Table Name = schema.name  ID = ts,n
     ```

     Dabei gilt folgendes:
     - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
     - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.
   - Zugriff auf Hierarchietabellenname (Access Hierarchy Table Name):

     ```
     Access Hierarchy Table Name = schema.name  ID = ts,n
     ```

     Dabei gilt folgendes:
     - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
     - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.
   - Zugriff auf Übersichtstabellenname (Access Summary Table Name):

     ```
     Access Summary Table Name = schema.name  ID = ts,n
     ```

     Dabei gilt folgendes:
     - *schema.name* ist der vollständig qualifizierte Name der Tabelle, auf die zugegriffen wird.
     - *ID* ist die entsprechende TABLESPACEID und TABLEID aus dem Katalog SYSCAT.TABLES für die Tabelle.
2. Zwei Typen von temporären Tabellen:
   - Zugriff auf ID für temporäre Tabellen (Access Temporary Table ID):

     ```
     Access Temp Table  ID = tn
     ```

     Dabei gilt folgendes:
     - *ID* ist die entsprechende Kennung, die von `db2expln` zugeordnet wurde.
   - Zugriff auf ID für deklarierte temporäre Tabellen (Access Declared Global Temporary Table ID):

     ```
     Access Global Temp Table  ID = ts,tn
     ```

Dabei gilt folgendes:

– *ID* ist die entsprechende TABLESPACEID aus dem Katalog SYSCAT-.TABLES für die Tabelle (`ts`) und die entsprechende von `db2expln` zugeordnete Kennung (`tn`)

Nach der Angabe über den Zugriff auf die Tabelle folgen weitere Angaben, die den Zugriff weiter beschreiben. Diese Angaben werden unter der Angabe über den Zugriff auf die Tabelle eingerückt. Folgende Angaben sind möglich:

- Anzahl von Spalten
- Paralleles Suchen
- Suchrichtung
- Zugriffsmethode für Zeilen
- Geplante Sperren
- Vergleichselemente
- Verschiedene Tabellenangaben.

Anzahl von Spalten

Die folgende Angabe zeigt die Anzahl von Spalten an, die aus jeder Zeile der Tabelle verwendet werden:

```
#Columns = n
```

Paralleles Suchen

Die folgende Angabe zeigt an, dass der Datenbankmanager mehrere Subagenten zum parallelen Lesen aus der Tabelle verwendet:

```
Parallel Scan
```

Wenn diese Angabe fehlt, wird die Tabelle nur von einem Agenten (bzw. Subagenten) gelesen.

Suchrichtung

Die folgende Angabe zeigt an, dass der Datenbankmanager Zeilen in umgekehrter Reihenfolge liest:

```
Scan Direction = Reverse
```

Wenn dieser Text nicht angezeigt wird, wird in Vorwärtsrichtung gesucht (dies ist der Standardwert).

Zugriffsmethode für Zeilen

Eine der folgenden Angaben zur Art und Weise des Zugriffs auf die Zeilen in der Tabelle, die den angegebenen Kriterien entsprechen, wird angezeigt:

- Die Angabe `Relation Scan` bedeutet, dass die Tabelle sequenziell durchsucht wird, um die den angegebenen Kriterien entsprechenden Zeilen zu ermitteln.
  – Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf Daten erfolgt:

```
   Relation Scan
|  Prefetch: None
```

– Die folgende Angabe bedeutet, dass das Optimierungsprogramm die Anzahl der Seiten, die vorabgelesen werden, vorbestimmt hat:

```
   Relation Scan
|  Prefetch: n Pages
```

– Die folgende Angabe weist darauf hin, dass die Daten wahrscheinlich vorabgelesen werden:

```
Relation Scan
|  Prefetch: Eligible
```

– Die folgende Angabe bedeutet, dass die Identifizierung der Zeilen, die den angegebenen Kriterien entsprechen, und der Zugriff auf sie über einen Index erfolgen:

```
Index Scan:  Name = schema.name  ID = xx
|  Index Columns:
```

Dabei gilt folgendes:

- *schema.name* ist der vollständig qualifizierte Name des Index, der durchsucht wird.
- *ID* ist die entsprechende Spalte IID in der Katalogsicht SYSCAT.INDEXES.

Diesen Angaben folgen jeweils eine Zeile für jede Spalte im Index. Jede Spalte im Index wird in einer der folgenden Formen aufgeführt:

```
n: column_name (Ascending)
n: column_name (Descending)
n: column_name (Include Column)
```

Die folgenden Angaben präzisieren die Art der Indexsuche:

- Die bereichsbegrenzenden Vergleichselemente für den Index werden folgendermaßen angegeben:

```
#Key Columns = n
   |  Start Key: xxxxx
   |  Stop Key: xxxxx
```

Dabei ist `xxxxx` eine der folgenden Angaben:

- `Start of Index`
- `End of Index`
- `Inclusive Value:` oder `Exclusive Value:`

  Ein inklusiver Schlüsselwert wird in die Indexsuche mit einbezogen. Ein exklusiver Schlüsselwert wird in der Suchoperation nicht berücksichtigt. Der Wert für den Schlüssel wird durch eine der folgenden Zeilen für jeden Teil des Schlüssels angegeben:

```
n: 'zeichenfolge'
n: nnn
n: jjjj-mm-tt
n: ss:mm:ss
n: jjjj-mm-tt ss:mm:ss.uuuuuu
n: NULL
n: ?
```

Wenn eine Literalzeichenfolge angezeigt wird, werden nur die ersten 20 Zeichen angegeben. Ist die Zeichenfolge länger als 20 Zeichen, wird dieses durch **...** am Ende der Zeichenfolge angezeigt. Einige Schlüssel können erst bestimmt werden, wenn der Abschnitt (section) ausgeführt wird. Dies wird durch den Wert **?** angezeigt.

- `Index-Only Access`

  Wenn alle benötigten Spalten aus dem Indexschlüssel abgerufen werden können, wird diese Angabe für reinen Indexzugriff angezeigt, und es wird nicht auf Tabellendaten zugegriffen.

- Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf die Indexseiten erfolgt:

  ```
  Index Prefetch: None
  ```

- Die folgende Angabe bedeutet, dass Indexseiten wahrscheinlich vorabgelesen werden:

  ```
  Index Prefetch: Eligible
  ```

- Die folgende Angabe bedeutet, dass kein Vorablesezugriff auf Datenseiten erfolgt:

  ```
  Data Prefetch: None
  ```

- Die folgende Angabe weist darauf hin, dass Datenseiten wahrscheinlich vorabgelesen werden:

  ```
  Data Prefetch: Eligible
  ```

- Wenn Vergleichselemente existieren, die an den Indexmanager übermittelt werden können, um bei der Qualifizierung von Indexeinträgen zu helfen, zeigt die folgende Angabe die Anzahl der Vergleichselemente:

  ```
  Sargable Index Predicate(s)
  |  #Predicates = n
  ```

– Die Angabe `Fetch Direct` bedeutet, dass auf die Zeilen, die den Kriterien entsprechen, über Satz-IDs (RID) zugegriffen wird, die zuvor im Zugriffsplan vorbereitet wurden.

Geplante Sperren

Für jeden Tabellenzugriff wird die Art der Sperre, die auf Tabellen- und Zeilenebene aktiviert wird, mit der folgenden Angabe angezeigt:

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

Folgende Werte sind für eine Tabellensperre möglich:

- Exclusive
- Intent Exclusive
- Intent None
- Intent Share
- Share
- Share Intent Exclusive
- Super Exclusive
- Update

Folgende Werte sind für eine Zeilensperre möglich:

- Exclusive
- Next Key Exclusive (wird nicht in der Ausgabe von `db2expln` angezeigt)
- None
- Share
- Next Key Share
- Update
- Next Key Weak Exclusive
- Weak Exclusive

Eine Erläuterung dieser Arten von Sperren finden Sie in „Attribute von Sperren" auf Seite 58.

Vergleichselemente

Es gibt zwei Angaben, die Informationen über die in einem Zugriffsplan verwendeten Vergleichselemente enthalten:

1. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die ausgewertet werden, nachdem die Daten zurückgegeben wurden (d. h. Restvergleichselemente):

   ```
   Residual Predicate(s)
   |  #Predicates = n
   ```

2. Die folgende Angabe zeigt die Anzahl von Vergleichselementen, die ausgewertet werden, während auf die Daten zugegriffen wird. Die Anzahl der Vergleichselemente lässt verschobene (Pushdown) Operationen wie Spaltenberechnung oder Sortierung unberücksichtigt.

   ```
   Sargable Predicate(s)
   |  #Predicates = n
   ```

Die Anzahl der in den obigen Angaben gezeigten Vergleichselemente spiegelt möglicherweise die Anzahl der Vergleichselemente der SQL-Anweisung aus folgenden Gründen nicht genau wider:

- Vergleichselemente können mehrmals in derselben Abfrage verwendet werden.
- Vergleichselemente können durch Hinzufügung impliziter Vergleichselemente während der Optimierung der Abfrage umgewandelt und erweitert worden sein.
- Vergleichselemente können während der Optimierung der Abfrage in weniger Vergleichselemente umgewandelt und komprimiert worden sein.

Verschiedene Tabellenangaben

- Die folgende Angabe weist darauf hin, dass nur auf eine Zeile zugegriffen wird:

  ```
  Single Record
  ```

- Die folgende Angabe wird angezeigt, wenn die für diesen Tabellenzugriff verwendete Isolationsstufe nicht der Isolationsstufe des Pakets entspricht:

  ```
  Isolation Level: xxxx
  ```

  Eine andere Isolationsstufe kann aus einer Reihe von Gründen verwendet werden, zum Beispiel:

  – Ein Paket wurde mit wiederholtem Lesen (RR) gebunden und hat Auswirkungen auf referenzielle Integritätsbedingungen. Der Zugriff der übergeordneten Tabelle zum Prüfen referenzieller Integritätsbedingungen wird auf die Isolationsstufe Cursorstabilität (CS) heruntergestuft, um unnötige Sperren für diese Tabelle zu vermeiden.

  – Ein mit der Isolationsstufe für nicht festgeschriebenen Lesevorgang (UR) gebundenes Paket gibt eine Anweisung DELETE oder UPDATE aus. Der Tabellenzugriff für den tatsächlichen Löschvorgang wird auf Cursorstabilität (CS) hochgestuft.

- Die folgende Angabe weist darauf hin, dass einige oder alle Zeilen, die aus der temporären Tabelle gelesen wurden, außerhalb des Pufferpools zwischengespeichert werden, wenn genügend Sortierspeicher verfügbar ist:

  ```
  Keep Rows In Private Memory
  ```

- Wenn für die Tabelle das Attribut für flüchtige Kardinalität definiert wurde, wird dies folgendermaßen angezeigt:

  ```
  Volatile Cardinality
  ```

Temporäre Tabellen

Eine temporäre Tabelle wird von einem Zugriffsplan während dessen Ausführung zum Speichern von Daten in einer temporären Arbeitstabelle oder Übergangstabelle verwendet. Die Tabelle existiert nur, solange der Zugriffsplan ausgeführt wird. Allgemein werden temporäre Tabellen verwendet, wenn Unterabfragen frühzeitig im Zugriffsplan ausgewertet werden müssen oder wenn Zwischenergebnisse nicht in den vorhandenen Speicher passen.

Wenn eine temporäre Tabelle erstellt werden muss, kann eine von zwei möglichen Angaben angezeigt werden. Diese Angaben weisen darauf hin, dass eine temporäre Tabelle erstellt und Zeilen in sie eingefügt werden. Die ID ist eine Kennung, die aus praktischen Gründen von `db2expln` zugeordnet wird, wenn auf die temporäre Tabelle Bezug genommen wird. Dieser ID wird als Präfix der Buchstabe 't' vorangestellt, um anzuzeigen, dass es sich um eine temporäre Tabelle handelt.

- Die folgende Angabe bedeutet, dass eine normale temporäre Tabelle erstellt wird:

  ```
  Insert Into Temp Table  ID = tn
  ```

- Die folgende Angabe bedeutet, dass eine normale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:

  ```
  Insert Into Shared Temp Table  ID = tn
  ```

- Die folgende Angabe bedeutet, dass eine sortierte temporäre Tabelle erstellt wird:

  ```
  Insert Into Sorted Temp Table  ID = tn
  ```

- Die folgende Angabe bedeutet, dass eine sortierte temporäre Tabelle von mehreren Subagenten parallel erstellt wird:

  ```
  Insert Into Sorted Shared Temp Table  ID = tn
  ```

- Die folgende Angabe bedeutet, dass eine deklarierte globale temporäre Tabelle erstellt wird:

  ```
  Insert Into Global Temp Table  ID = ts,tn
  ```

- Die folgende Angabe bedeutet, dass eine deklarierte globale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:

  ```
  Insert Into Shared Global Temp Table  ID = ts,tn
  ```

- Die folgende Angabe bedeutet, dass eine sortierte deklarierte globale temporäre Tabelle erstellt wird:

  ```
  Insert Into Sorted Global Temp Table  ID = ts,tn
  ```

- Die folgende Angabe bedeutet, dass eine sortierte deklarierte globale temporäre Tabelle von mehreren Subagenten parallel erstellt wird:

  ```
  Insert Into Sorted Shared Global Temp Table  ID = ts,tn
  ```

Nach jeder der oben gezeigten Angaben folgt die Angabe:

```
#Columns = n
```

Diese Angabe zeigt die Anzahl der Spalten für jede Zeile, die in die temporäre Tabelle eingefügt wird.

Sortierte temporäre Tabellen

Sortierte temporäre Tabellen treten z. B. als Ergebnis folgender Operationen auf:

- ORDER BY
- DISTINCT
- GROUP BY
- Mischverknüpfung (Merge Join)
- '= ANY' Unterabfrage
- '<> ALL' Unterabfrage
- INTERSECT oder EXCEPT
- UNION (ohne das Schlüsselwort ALL)

Eine Reihe zusätzlicher Angaben kann auf die ursprüngliche Angabe über die Erstellung für eine sortierte temporäre Tabelle folgen:

- Die folgende Angabe zeigt die Anzahl der bei der Sortierung verwendeten Schlüsselspalten:

  ```
  #Sort Key Columns = n
  ```

  Für jede Spalte im Sortierschlüssel wird eine der folgenden Zeilen angezeigt:

  ```
  Key n: column_name (Ascending)
  Key n: column_name (Descending)
  Key n: (Ascending)
  Key n: (Descending)
  ```

- Die folgenden Angaben enthalten Schätzwerte für die Anzahl und die Größe der Zeilen, so dass die optimale Größe für den Sortierspeicher zur Laufzeit zugeordnet werden kann.

  ```
  Sortheap Allocation Parameters:
  |  #Rows     = n
  |  Row Width = n
  ```

- Wenn lediglich die ersten Zeilen des sortierten Ergebnisses benötigt werden, wird Folgendes angezeigt:

  ```
  Sort Limited To Estimated Row Count
  ```

- Für Sortiervorgänge in einer symmetrischen Mehrprozessorumgebung (SMP-Umgebung) wird die Art der durchzuführenden Sortierung durch eine der folgenden Angaben angezeigt:

  ```
  Use Partitioned Sort
  Use Shared Sort
  Use Replicated Sort
  Use Round-Robin Sort
  ```

  Eine Beschreibung der verschiedenen Sortiertechniken finden Sie in „Strategien für paralleles Sortieren" auf Seite 229.

- Die folgenden Angaben zeigen an, ob das Ergebnis der Sortierung im Sortierspeicher verbleibt:

```
Piped
```

und

```
Not Piped
```

Wenn eine über Pipe geleitete Sortierung angegeben wird, behält der Datenbankmanager die sortierte Ausgabe im Speicher, anstatt das sortierte Ergebnis in eine andere temporäre Tabelle zu schreiben. (Eine Beschreibung der über Pipe geleiteten und nicht über Pipe geleiteten Sortiervorgänge finden Sie in „Einfluss des Sortierens auf das Optimierungsprogramm" auf Seite 225.)

- Die folgende Angabe bedeutet, dass doppelte Werte während der Sortierung entfernt werden:

  ```
  Duplicate Elimination
  ```

- Wenn Spaltenberechnungen in der Sortierung durchgeführt werden, wird dies durch eine der folgenden Angaben angezeigt:

  ```
  Partial Aggregation
  Intermediate Aggregation
  Buffered Partial Aggregation
  Buffered Intermediate Aggregation
  ```

Abschließen der temporären Tabelle

Nach einem Tabellenzugriff, der eine verschobene (Pushdown) Operation zum Erstellen einer temporären Tabelle enthält (d. h. eine Operation zum Erstellen einer temporären Tabelle, die im Rahmen eines Tabellenzugriffs auftritt), folgt eine Abschlussangabe (Completion), die das Dateiende verarbeitet, indem sie die temporäre Tabelle darauf vorbereitet, Zeilen für den nachfolgenden Zugriff auf die temporäre Tabelle zur Verfügung zu stellen. Es wird eine der folgenden Zeilen angezeigt:

```
Temp Table Completion  ID = tn
Shared Temp Table Completion  ID = tn
Sorted Temp Table Completion  ID = tn
Sorted Shared Temp Table Completion  ID = tn
```

Tabellenfunktionen

Tabellenfunktionen sind benutzerdefinierte Funktionen (UDFs), die Daten in Form einer Tabelle an die Anweisung zurückgeben. Im Handbuch *SQL Reference* finden Sie weitere Informationen zu Tabellenfunktionen. Tabellenfunktionen werden durch die folgende Angabe angezeigt:

```
Access User Defined Table Function
|   Name = schema.funcname
|   Language = xxxx
|   Fenced   Deterministic   NULL Call   Disallow Parallel
```

Die Sprache (C, OLE oder Java), in der die Tabellenfunktion geschrieben ist, wird zusammen mit den Attributen der Tabellenfunktion angegeben.

Verknüpfungen

Es gibt drei Arten von Verknüpfungen (eine Beschreibung dieser Verknüpfungen finden Sie in „Verknüpfungskonzepte" auf Seite 203):

- Hash-Verknüpfung (Hash Join)
- Mischverknüpfung (Merge Join)
- Verknüpfung über Verschachtelungsschleife (Nested Loop Join)

Wenn bei der Ausführung eines Abschnitts der Zeitpunkt kommt, zu dem eine Verknüpfung ausgeführt wird, wird eine der folgenden Angaben angezeigt:

```
Hash Join
```

oder

```
Merge Join
```

oder

```
Nested Loop Join
```

Es ist möglich, dass eine linke äußere Verknüpfung durchgeführt wird. Eine linke äußere Verknüpfung wird durch eine der folgenden Angaben angezeigt:

```
Left Outer Hash Join
```

oder

```
Left Outer Merge Join
```

oder

```
Left Outer Nested Loop Join
```

Bei Mischverknüpfungen und Verknüpfungen über Verschachtelungsschleife ist die äußere Tabelle der Verknüpfung die Tabelle, auf die in der vorherigen Zugriffsangabe, die in der Ausgabe angezeigt wird, verwiesen wird. Die innere Tabelle bei dieser Verknüpfung ist die Tabelle, auf die in der Zugriffsangabe verwiesen wird, die sich im Bereich der Verknüpfungsangabe befindet. Bei Hash-Verknüpfungen sind die Zugriffsangaben umgekehrt, d. h. die äußere Tabelle ist im Verknüpfungsbereich enthalten, und die innere Tabelle wird vor der Verknüpfung angezeigt.

Bei einer Hash- oder Mischverknüpfung können folgende weitere Angaben auftreten:

- In einigen Fällen muss bei einer Verknüpfung lediglich festgestellt werden, ob eine Zeile der inneren Tabelle mit der aktuellen Zeile in der äußeren Tabelle übereinstimmt. Dies wird durch folgende Angabe angezeigt:

  ```
  Early Out: Single Match Per Outer Row
  ```

- Es ist möglich, nach Abschluss der Verknüpfung Vergleichselemente anzuwenden. Die Anzahl der Vergleichselemente, die angewandt werden, wird folgendermaßen angezeigt:

```
Residual Predicate(s)
   |  #Predicates = n
```

Bei einer Hash-Verknüpfung können folgende weitere Angaben auftreten:

- Die Hash-Tabelle wird aus der inneren Tabelle erstellt. Wenn die Erstellung der Hash-Tabelle in ein Vergleichselement im Zugriff auf die innere Tabelle verschoben wurde, wird darauf durch die folgende Angabe im Zugriff der inneren Tabelle hingewiesen:

  ```
  Process Hash Table For Join
  ```

- Beim Zugriff der äußeren Tabelle kann eine Prüftabelle erstellt werden, um die Leistung der Verknüpfung zu steigern. Auf die Erstellung der Prüftabelle wird durch die folgende Angabe im Zugriff der äußeren Tabelle hingewiesen:

  ```
  Process Probe Table For Hash Join
  ```

- Die geschätzte erforderliche Anzahl Byte zum Erstellen der Hash-Tabelle wird durch folgende Angabe dargestellt:

  ```
  Estimated Build Size: n
  ```

- Die geschätzte erforderliche Anzahl Byte für die Prüftabelle wird durch folgende Angabe dargestellt:

  ```
  Estimated Probe Size: n
  ```

Bei einer Verknüpfung über Verschachtelungsschleife kann die folgende zusätzliche Angabe direkt nach der Verknüpfungsangabe angezeigt werden:

```
Piped Inner
```

Diese Angabe bedeutet, dass die innere Tabelle der Verknüpfung das Ergebnis einer anderen Reihe von Operationen ist. Dies wird auch als eine *zusammengesetzte innere (composite inner)* Tabelle bezeichnet.

Wenn eine Verknüpfung mehr als zwei Tabellen umfasst, müssen die EXPLAIN-Schritte von oben nach unten gelesen werden. Angenommen, die EXPLAIN-Ausgabe hat folgende Struktur:

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

Die Ausführung würde in diesem Fall in folgenden Schritten ablaufen:

1. Abrufen einer den Kriterien entsprechenden Zeile aus `W`
2. Verknüpfen der Zeile aus `W` mit (der nächsten) Zeile aus `X` und Benennung des Ergebnisses als `P1` (für Verknüpfungsteilergebnis Nr. 1)
3. Verknüpfung von `P1` mit der (nächsten) Zeile aus `Y` zum Erstellen von `P2`
4. Verknüpfung von `P2` mit der (nächsten) Zeile aus `Z` zum Erstellen einer vollständigen Ergebniszeile
5. Wenn weitere Zeilen in `Z` sind, weiter mit Schritt 4
6. Wenn weitere Zeilen in `Y` sind, weiter mit Schritt 3
7. Wenn weitere Zeilen in `X` sind, weiter mit Schritt 2
8. Wenn weitere Zeilen in `W` sind, weiter mit Schritt 1

Datenströme

In einem Zugriffsplan ist es oft erforderlich, die Erstellung und den Fluss von Daten von einer Reihe von Operationen zur anderen zu steuern. Das Konzept des Datenstroms erlaubt es, eine Gruppe von Operationen innerhalb eines Zugriffsplans als Einheit zu steuern. Der Beginn eines Datenstroms wird durch folgende Angabe gekennzeichnet:

```
Data Stream n
```

Hierbei ist `n` eine eindeutige Kennung, die zur leichteren Bezugnahme von `db2expln` zugeordnet wird. Das Ende des Datenstroms wird durch folgende Angabe gekennzeichnet:

```
End of Data Stream n
```

Alle Operationen zwischen diesen Angaben werden als Teil desselben Datenstroms angesehen. Ein Datenstrom hat eine Anzahl von Merkmalen, und auf die einleitende Datenstromangabe können eine oder mehrere Angaben folgen, um diese Merkmale zu beschreiben:

- Wenn die Verarbeitung des Datenstroms von einem Wert abhängt, der früher im Zugriffsplan generiert wurde, wird der Datenstrom mit folgender Angabe markiert:

  ```
  Correlated
  ```

- Ähnlich wie bei einer sortierten temporären Tabelle zeigen die folgenden Angaben, ob die Ergebnisse des Datenstroms im Speicher behalten werden:

  ```
  Piped
  ```

  und

  ```
  Not Piped
  ```

  Wie bei temporären Tabellen kann ein über eine Pipe geleiteter Datenstrom auf die Platte geschrieben werden, wenn zur Ausführungszeit zu wenig Speicher vorhanden ist. Der Zugriffsplan sieht beide Möglichkeiten vor.

- Die folgende Angabe bedeutet, dass nur ein einziger Satz aus diesem Datenstrom benötigt wird:

  ```
  Single Record
  ```

Wenn auf einen Datenstrom zugegriffen wird, wird die folgende Angabe in der Ausgabe angezeigt:

```
Access Data Stream n
```

Einfügen, Aktualisieren und Löschen

Der EXPLAIN-Text für diese SQL-Anweisungen ist selbsterklärend. Nachfolgend sind mögliche Texte für diese SQL-Operationen dargestellt:

- `Insert: Table Name = schema.name ID = ts,n`
- `Update: Table Name = schema.name ID = ts,n`
- `Delete: Table Name = schema.name ID = ts,n`
- `Insert: Hierarchy Table Name = schema.name ID = ts,n`
- `Update: Hierarchy Table Name = schema.name ID = ts,n`
- `Delete: Hierarchy Table Name = schema.name ID = ts,n`
- `Insert: Summary Table Name = schema.name ID = ts,n`
- `Update: Summary Table Name = schema.name ID = ts,n`
- `Delete: Summary Table Name = schema.name ID = ts,n`
- `Insert: Global Temporary Table ID = ts, tn`
- `Update: Global Temporary Table ID = ts, tn`
- `Delete: Global Temporary Table ID = ts, tn`

Vorbereitung von Satzkennungen (Satz-IDs)

Für einige Zugriffspläne ist es effizienter, wenn die den Kriterien entsprechenden Satz-IDs (RIDs) sortiert und mehrfach auftretende Werte entfernt werden (bei OR-Verknüpfung von Indizes) bzw. wenn eine Technik verwendet wird, mit der RIDs, die in allen betroffenen Indizes auftreten (bei AND-Verknüpfung von Indizes), identifiziert werden, bevor der tatsächliche Zugriff auf die Tabelle erfolgt. Es gibt drei Hauptgründe für die Vorbereitung von Satz-IDs, auf die in den EXPLAIN-Angaben hingewiesen wird:

- Die folgende Angabe bedeutet, dass die OR-Verknüpfung „Index ORing" zum Vorbereiten der Liste der den Kriterien entsprechenden Satz-IDs (RIDs) verwendet wird:

  ```
  Index ORing RID Preparation
  ```

  Index ORing (OR-Verknüpfung von Indizes) bezeichnet ein Verfahren, bei dem mehrere Indexzugriffe vorgenommen und die Ergebnisse kombiniert werden, so dass die eindeutigen Satz-IDs, die mindestens in einem der Indizes auftreten, auf die zugegriffen wird, zusammengefasst werden. Das Optimierungsprogramm erwägt die Verwendung dieses Verfahrens, wenn Vergleichselemente durch Schlüsselwörter OR verknüpft werden oder ein Vergleichselement IN existiert. Die Zugriffe können auf den gleichen Index oder auf verschiedene Indizes erfolgen.

- Ein weiterer Grund für die Vorbereitung von Satz-IDs ist die Vorbereitung der Eingabedaten, die während des Vorablesezugriffs über Listen verwendet werden sollen, wie dies durch folgende Angabe angezeigt wird:

  ```
  List Prefetch RID Preparation
  ```

- *Index ANDing* (AND-Verknüpfung von Indizes) bezeichnet ein Verfahren, beim dem mehrere Indexzugriffe vorgenommen und die Ergebnisse kombiniert werden, so dass die Satz-IDs, die jeweils in allen Indizes auftreten, auf die zugegriffen wird, zusammengefasst werden. Die folgende Angabe zeigt an, dass die Verarbeitung einer AND-Verknüpfung von Indizes gestartet wird:

  ```
  Index ANDing
  ```

  Wenn das Optimierungsprogramm die Größe der Ergebnismenge abgeschätzt hat, wird der Schätzwert mit der folgenden Angabe angezeigt:

  ```
  Optimizer Estimate of Set Size: n
  ```

  Die Filteroperationen bei AND-Verknüpfungen von Indizes verarbeiten die Satz-IDs und verwenden Bit-Filtermethoden, um die Satz-IDs zu bestimmen, die in allen Indizes vorkommen, auf die zugegriffen wird. Die folgenden Angaben weisen darauf hin, dass Satz-IDs zur AND-Verknüpfung von Indizes verarbeitet werden:

  ```
  Index ANDing Bitmap Build
  Index ANDing Bitmap Probe
  Index ANDing Bitmap Build and Probe
  ```

  Wenn das Optimierungsprogramm die Größe der Ergebnismenge für eine Bit-Abgleichung (Bitmap) abgeschätzt hat, wird der Schätzwert mit der folgenden Angabe angezeigt:

  ```
  Optimizer Estimate of Set Size: n
  ```

Bei jeder Art von Satz-ID-Vorbereitung wird die Möglichkeit, dass ein Vorablesezugriff über Listen erfolgen kann, mit folgender Angabe angezeigt:

```
Prefetch: Enabled
```

Spaltenberechnungen

Spaltenberechnungen (Aggregation) werden für die Zeilen vorgenommen, die den etwaigen, in den Vergleichselementen der SQL-Anweisungen festgelegten Kriterien entsprechen. Wenn eine Art von Spaltenfunktion auszuführen ist, wird eine der folgenden Angaben angezeigt:

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
```

```
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

Die Angabe *Predicate aggregation* besagt, dass die Spaltenberechnungsoperation zur Verarbeitung als Vergleichselement, wenn auf die Daten wirklich zugegriffen wird, verschoben wurde.

Unter jeder der oben aufgeführten Angaben über Spaltenberechnungen befindet sich eine Angabe, die die Art der durchgeführten Spaltenfunktion anzeigt:

- `Group By`
- `Column Function(s)`
- `Single Record`

Die spezifische Spaltenfunktion kann von der ursprünglichen SQL-Anweisung abgeleitet werden. Ein einzelner Satz (Single Record) wird aus einem Index abgerufen, um eine Operation MIN oder MAX auszuführen.

Wenn eine Spaltenberechnung über Vergleichselemente verwendet wird, folgt auf die Angabe über den Tabellenzugriff, in der die Spaltenberechnung auftrat, eine Angabe über den Abschluss der Spaltenberechnung, die alle erforderlichen Verarbeitungsschritte bei Gruppenende bzw. bei Dateiende vornimmt. Dazu wird eine der folgenden Zeilen in der Ausgabe angezeigt:

```
Aggregation Completion
Partial Aggregation Completion
Intermediate Aggregation Completion
Final Aggregation Completion
```

Parallelverarbeitung

Für die parallele Ausführung einer SQL-Anweisung (entweder mit partitionsinterner oder partitionsübergreifender Parallelität) sind einige besondere Operationen erforderlich. Die Operationen für Parallelpläne werden im Folgenden beschrieben.

- Bei der Ausführung eines partitionsinternen Parallelplans werden Abschnitte des Plans gleichzeitig mit Hilfe verschiedener Subagenten ausgeführt. Die Erstellung der Subagenten wird durch folgende Angabe angezeigt:

  ```
  Process Using n Subagents
  ```

- Bei der Ausführung eines partitionsübergreifenden parallelen Zugriffsplans wird der Abschnitt (Section) in mehrere Teilbereiche (Subsections) geteilt. Jeder Teilbereich wird zur Ausführung an einen oder mehrere Knoten gesendet. Ein wichtiger Teilbereich ist der *Koordinatorteilbereich*. Der Koordinatorteilbereich ist der erste Teilbereich in jedem Plan. Er erhält

zunächst die Steuerung und ist dann für die Verteilung der anderen Teilbereiche und die Rückgabe von Ergebnissen an die aufrufende Anwendung zuständig.

Die Verteilung von Teilbereichen wird durch folgende Angabe angezeigt:

```
Distribute Subsection #n
```

Die Knoten, die einen Teilbereich empfangen, können durch eine von acht Methoden bestimmt werden:

- Die folgende Angabe bedeutet, dass der Teilbereich abhängig vom Wert der Spalten zu einem Knoten innerhalb der Knotengruppe gesendet wird.

  ```
  Directed by Hash
  |  #Columns = n
  |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an einen vorbestimmten Knoten gesendet wird. (Dies tritt häufig auf, wenn die Anweisung mit der Funktion NODENUMBER() arbeitet.)

  ```
  Directed by Node Number
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an den Knoten gesendet wird, der einer vorbestimmten Partitionsnummer in der angegebenen Knotengruppe entspricht. (Dies tritt häufig auf, wenn die Anweisung mit der Funktion PARTITION() arbeitet.)

  ```
  Directed by Partition Number
  |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an den Knoten gesendet wird, der die aktuelle Zeile für den Cursor der Anwendung verfügbar gemacht hat.

  ```
  Directed by Position
  ```

- Die folgende Angabe bedeutet, dass nur ein einziger Knoten, der bei der Kompilierung der Anweisung festgelegt wird, den Teilbereich empfängt.

  ```
  Directed to Single Node
  |  Node Number = n
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich auf dem Koordinatorknoten ausgeführt wird.

  ```
  Directed to Coordinator Node
  ```

- Die folgende Angabe bedeutet, dass der Teilbereich an alle aufgeführten Knoten gesendet wird.

  ```
  Broadcast to Node List
  |  Nodes = n1, n2, n3, ...
  ```

- Die folgende Angabe bedeutet, dass nur ein einziger Knoten, der bei der Ausführung der Anweisung festgelegt wird, den Teilbereich empfängt.

  ```
  Directed to Any Node
  ```

- Tabellenwarteschlangen werden zum Versetzen von Daten zwischen Teilbereichen in einer Umgebung mit partitionierten Datenbanken oder zwischen Subagenten in einer symmetrischen Mehrprozessorumgebung (SMP) verwendet. Tabellenwarteschlangen werden folgendermaßen beschrieben:
  - Die folgenden Angaben zeigen an, dass Daten in eine Tabellenwarteschlange eingefügt werden:

    ```
    Insert Into Synchronous Table Queue  ID = qn
    Insert Into Asynchronous Table Queue  ID = qn
    Insert Into Synchronous Local Table Queue  ID = qn
    Insert Into Asynchronous Local Table Queue  ID = qn
    ```

  - Bei Tabellenwarteschlangen einer Datenbankpartition wird das Ziel für Zeilen, die in die Tabellenwarteschlange eingefügt werden, durch eine der folgenden Angaben angezeigt:

    ```
    Broadcast to Coordinator Node
    ```

    Alle Zeilen werden an den Koordinatorknoten gesendet.

    ```
    Broadcast to All Nodes of Subsection n
    ```

    Alle Zeilen werden an jede Datenbankpartition gesendet, auf der der angegebene Teilbereich ausgeführt wird.

    ```
    Hash to Specific Node
    ```

    Jede Zeile wird abhängig von den Werten in der Zeile an eine Datenbankpartition gesendet.

    ```
    Send to Specific Node
    ```

    Jede Zeile wird an eine Datenbankpartition gesendet, die während der Ausführung der Anweisung bestimmt wird.

    ```
    Send to Random Node
    ```

    Jede Zeile wird an eine zufällig bestimmte Datenbankpartition gesendet.
  - In einigen Fällen ist es möglich, dass die Tabellenwarteschlange der Datenbankpartition einige Zeilen wegen Kapazitätsüberschreitung in eine temporäre Tabelle versetzen muss. Diese Möglichkeit wird durch folgende Angabe angezeigt:

    ```
    Rows Can Overflow to Temporary Table
    ```

  - Auf einen Tabellenzugriff, der eine verschobene Operation (Pushdown) zum Einfügen von Zeilen in eine Tabellenwarteschlange enthält, folgt eine Abschlussangabe (Completion), die die Zeilen handhabt, die nicht unmittelbar gesendet werden konnten. Dazu wird eine der folgenden Zeilen in der Ausgabe angezeigt:

    ```
    Insert Into Synchronous Table Queue Completion  ID = qn
    Insert Into Asynchronous Table Queue Completion  ID = qn
    Insert Into Synchronous Local Table Queue Completion  ID = qn
    Insert Into Asynchronous Local Table Queue Completion  ID = qn
    ```

– Die folgenden Angaben zeigen an, dass Daten aus einer Tabellenwarteschlange abgerufen werden:

```
Access Table Queue  ID = qn
Access Local Table Queue  ID = qn
```

Diesen Nachrichten folgt stets eine Angabe der Anzahl der Zeilen, die abgerufen werden.

```
#Columns = n
```

– Wenn die Tabellenwarteschlange die Zeilen auf der Empfängerseite sortiert, wird der Angabe über den Tabellenwarteschlangenzugriff außerdem eine der folgenden Nachrichten beigefügt:

```
Output Sorted
Output Sorted and Unique
```

Diesen Nachrichten folgt eine Angabe der Anzahl der Schlüsselspalten, die für die Sortierung verwendet wurden.

```
#Key Columns = n
```

Für jede Spalte im Sortierschlüssel wird eine der folgenden Angaben angezeigt:

```
Key n: (Ascending)
Key n: (Descending)
```

– Wenn Vergleichselemente auf der Empfängerseite der Tabellenwarteschlange auf die Zeilen angewendet werden, wird folgende Nachricht in der Ausgabe angezeigt:

```
Residual Predicate(s)
|  #Predicates = n
```

- Einige Teilbereiche in Umgebungen mit partitionierten Datenbanken springen explizit zurück an den Start des Teilbereichs. Dies wird durch folgende Angabe angezeigt:

```
Jump Back to Start of Subsection
```

Verarbeitung von Anweisungen in zusammengeschlossenen Datenbanken

Für die Ausführung einer SQL-Anweisung in einer zusammengeschlossenen Datenbank ist die Fähigkeit erforderlich, Teile der betreffenden Anweisung in Bezug auf andere Datenquellen auszuführen.

Im folgenden wird angegeben, dass auf eine Datenquelle zugegriffen wird:

```
Distributed Subquery #n
|  #Columns = n
```

Es ist möglich, dass Vergleichselemente auf die Daten angewandt werden, die von der verteilten Unterabfrage (Distributed Subquery) zurückgegeben wurden.

Die Anzahl der Vergleichselemente (Predicates), die angewandt werden, wird folgendermaßen angezeigt:

```
Residual Predicate(s)
   |  #Predicates = n
```

Die Einzelheiten für die verteilten Unterabfragen werden getrennt bereitgestellt. Die Optionen für verteilte Unterabfragen werden nachfolgend beschrieben:

- Die Datenquelle für die Unterabfrage wird durch eine der folgenden Angaben gezeigt:

  ```
  Server: server_name (type, version)
  Server: server_name (type)
  Server: server_name
  ```

- Die SQL-Anweisung für die Unterabfrage (Subquery SQL Statement) wird folgendermaßen angezeigt:

  ```
  Subquery SQL Statement:
  statement
  ```

- Die Kurznamen (Nicknames), auf die in der Unterabfrage verwiesen wird, werden folgendermaßen aufgelistet:

  ```
  Nickname Referenced:
  Schema.nickname Base = baseschema.basetable
  ```

- Wenn die Werte vom Server mit zusammengeschlossenen Datenbanken an die Datenquelle übergeben werden, bevor die Unterabfrage ausgeführt wird, wird die Zahl der Werte folgendermaßen gezeigt:

  ```
  #Input Columns: n
  ```

- Wenn die Werte von der Datenquelle an den Server mit zusammengeschlossenen Datenbanken übergeben werden, nachdem die Unterabfrage ausgeführt wird, wird die Zahl der Werte folgendermaßen gezeigt:

  ```
  #Output Columns: n
  ```

Verschiedene Angaben

- Abschnitte für Anweisungen der Datendefinitionssprache (DDL - Data Definition Language) werden in der Ausgabe folgendermaßen gekennzeichnet:

  ```
  DDL Statement
  ```

  Für DDL-Anweisungen wird keine weitere EXPLAIN-Ausgabe bereitgestellt.

- Abschnitte für SET-Anweisungen für die aktualisierbaren Sonderregister wie **CURRENT EXPLAIN SNAPSHOT** werden in der Ausgabe folgendermaßen gekennzeichnet:

  ```
  SET Statement
  ```

  Für SET-Anweisungen wird keine weitere EXPLAIN-Ausgabe bereitgestellt.

- Wenn die SQL-Anweisung die Klausel DISTINCT enthält, kann der folgende Text in der Ausgabe angezeigt werden:

  ```
  Distinct Filter  #Columns = n
  ```

  Dabei ist `n` die Anzahl von Spalten, die beim Abrufen eindeutiger Zeilen verwendet wird. Zum Abrufen eindeutiger Zeilenwerte müssen die Zeilen geordnet sein, so dass gleiche Werte übersprungen werden können. Diese Angabe wird nicht angezeigt, wenn der Datenbankmanager gleiche Werte nicht explizit entfernen muss wie in folgenden Fällen:
  - Es existiert ein eindeutiger Index, und alle Spalten im Indexschlüssel sind Teil der Operation für die Klausel DISTINCT.
  - Die mehrfach auftretenden Werte können beim Sortieren entfernt werden.
- Die folgende Angabe wird angezeigt, wenn die nächste Operation von einer bestimmten Satzkennung (RID) abhängig ist:

  ```
  Positioned Operation
  ```

  Diese Angabe wird für jede SQL-Anweisung angezeigt, die die Syntax WHERE CURRENT OF verwendet.
- Die folgende Angabe wird angezeigt, wenn Vergleichselemente vorhanden sind, die auf das Ergebnis angewendet werden müssen, aber die nicht als Teil einer anderen Operation angewendet werden konnten (Restvergleichselemente):

  ```
  Residual Predicate Application
  |  #Predicates = n
  ```
- Die folgende Angabe wird angezeigt, wenn ein Operator UNION in der SQL-Anweisung enthalten ist:

  ```
  UNION
  ```
- Die folgende Angabe wird angezeigt, wenn sich im Zugriffsplan eine Operation befindet, deren einziger Zweck das Erstellen von Zeilenwerten zur Verwendung durch nachfolgende Operationen ist:

  ```
  Table Constructor
  |  n-Row(s)
  ```

  Table Constructors können verwendet werden, um Werte in einer Menge in eine Reihe von Zeilen umzuwandeln, die anschließend an nachfolgende Operationen übergeben werden. Wenn die nächste Zeile von einem Table Constructor angefordert wird, wird die folgende Angabe angezeigt:

  ```
  Access Table Constructor
  ```

- Die folgende Angabe wird angezeigt, wenn eine Operation existiert, die nur unter bestimmten Bedingungen verarbeitet wird:

```
Conditional Evaluation
|  Condition #n:
|  |  #Predicates = n
|  Action #n:
```

  Durch bedingte Auswertung werden Aktivitäten wie die SQL-Anweisung CASE oder interne Mechanismen wie referenzielle Integritätsbedingungen oder Auslöser implementiert. Wenn keine Aktion angezeigt wird, werden nur Datenbearbeitungsoperationen verarbeitet, wenn die Bedingung wahr ist.
- Eine der folgenden Angaben wird angezeigt, wenn eine Unterabfrage nach ALL, ANY oder EXISTS im Zugriffsplan verarbeitet wird:
  - `ANY/ALL Subquery`
  - `EXISTS Subquery`
  - `EXISTS SINGLE Subquery`
- Vor bestimmten UPDATE- und DELETE-Operationen muss die Position einer speziellen Zeile in der Tabelle bestimmt werden. Dies wird durch folgende Angabe gekennzeichnet:

```
Establish Row Position
```

- Die folgende Angabe wird angezeigt, wenn Zeilen an die Anwendung zurückgegeben werden:

```
   Return Data to Application
|  #Columns = n
```

  Wenn die Operation in einen Tabellenzugriff verschoben (Pushdown) wurde, wird eine Abschlussphase (Completion) erforderlich. Diese Phase wird wie folgt angezeigt:

```
Return Data Completion
```

Beispiele für die Ausgabe von db2expln und dynexpln

Zum besseren Verständnis finden Sie nachfolgend fünf Beispiele, die den Aufbau und das Format der Ausgabe von `db2expln` und `dynexpln` verdeutlichen. Diese Beispiele wurden für die Beispieldatenbank SAMPLE ausgeführt, die mit DB2 ausgeliefert wird. Jedes Beispiel wird kurz erläutert. Die signifikanten Unterschiede von einem Beispiel zum nächsten wurden **fett** hervorgehoben.

Beispiel 1: Plan ohne Parallelität

In diesem Beispiel wird einfach eine Liste aller Namen von Mitarbeitern (employee), ihrer Aufgaben (job), ihrer Abteilungen (department) und Standorte (location) sowie der Namen der Projekte, an denen sie arbeiten, abgerufen. Das wesentliche Merkmal dieses Zugriffsplans besteht darin, dass über Operationen zur Mischverknüpfung die relevanten Daten aus allen angegebenen Tabellen verknüpft werden. Da keine Indizes verfügbar sind, führt der Zugriffsplan eine Tabellensuche in allen Tabellen aus, und jede Tabelle muss sortiert werden, bevor sie mit einer anderen verknüpft werden kann.

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:47:58

Bind Timestamp = 2000-01-03-15.47.58.607455

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno


Estimated Cost        = 126
Estimated Cardinality = 153

Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|  #Columns = 3
|  Relation Scan
```

```
|  |  Prefetch: Eligible
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Insert Into Sorted Temp Table  ID = t1
|  |  #Columns = 3
|  |  #Sort Key Columns = 1
|  |  |  Key 1: DEPTNO (Ascending)
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 40
|  |  |  Row Width = 48
|  |  Piped
Sorted Temp Table Completion  ID = t1
Access Temp Table  ID = t1
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
Merge Join
|  Access Table Name = DOOLE.PROJECT  ID = 2,7
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t2
|  |  |  #Columns = 2
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 38
|  |  |  |  Row Width = 28
|  |  |  Piped
|  Sorted Temp Table Completion  ID = t2
|  Access Temp Table  ID = t2
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
Merge Join
|  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
|  |  #Columns = 3
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t3
|  |  |  #Columns = 3
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: WORKDEPT (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 63
|  |  |  |  Row Width = 32
|  |  |  Piped
|  Sorted Temp Table Completion  ID = t3
```

```
    |  Access Temp Table  ID = t3
|   |  #Columns = 3
|   |  Relation Scan
|   |  |  Prefetch: Eligible
    Return Data to Application
|   #Columns = 5

End of section


Optimizer Plan:

              RETURN
              (   1)
                 |
              MSJOIN
              (   2)
              /     \
         MSJOIN      TBSCAN
         (   3)      (  12)
        /     \        |
    TBSCAN    TBSCAN   SORT
    (   4)    (   8)   (  13)
      |         |        |
    SORT      SORT     TBSCAN
    (   5)    (   9)   (  14)
      |         |        |
    TBSCAN    TBSCAN  Table:
    (   6)    (  10)   DOOLE
      |         |     EMPLOYEE
  Table:      Table:
  DOOLE       DOOLE
  DEPARTMENT  PROJECT
```

Im ersten Teil des Plans wird auf die Tabellen DEPARTMENT und PROJECT zugegriffen, die über eine Mischverknüpfung verknüpft werden. Das Ergebnis dieser Verknüpfung wird mit der Tabelle EMPLOYEE verknüpft. Die Ergebniszeilen werden an die Anwendung zurückgegeben.

Beispiel 2: Zugriffsplan für Datenbank mit einer einzelnen Partition und partitionsinterner Parallelität

In diesem Beispiel wird dieselbe SQL-Anweisung wie in „Beispiel 1: Plan ohne Parallelität" auf Seite 677, gezeigt, jedoch wurde diese Abfrage für eine 4-Wege-SMP-Maschine kompiliert.

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:48:51

Bind Timestamp = 2000-01-03-15.48.51.402403
Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5
Partition Parallel       = No
Intra-Partition Parallel = Yes (Bind Degree = 4)
Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 142
Estimated Cardinality = 153

Process Using 4 Subagents
|  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|  |  #Columns = 3
|  |  Parallel Scan
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Shared Temp Table  ID = t1
|  |  |  #Columns = 3
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  |  Use Round-Robin Sort
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  |  #Rows     = 40
|  |  |  |  |  Row Width = 48
|  |  |  Piped
```

```
Sorted Shared Temp Table Completion  ID = t1
  Access Temp Table  ID = t1
  #Columns = 3
  Relation Scan
  | Prefetch: Eligible
  Merge Join
  Access Table Name = DOOLE.PROJECT  ID = 2,7
  | #Columns = 2
  | Parallel Scan
  | | Relation Scan
  | | | Prefetch: Eligible
  | Lock Intents
  | | | Table: Intent Share
  | | | Row  : Next Key Share
  | | Insert Into Sorted Shared Temp Table  ID = t2
  | | #Columns = 2
  | | | #Sort Key Columns = 1
  | | | | Key 1: DEPTNO (Ascending)
  | | Use Replicated Sort
  | | | Sortheap Allocation Parameters:
  | | | #Rows     = 38
  | | | Row Width = 28
  | | | Piped
  | Sorted Shared Temp Table Completion  ID = t2
  | Access Temp Table  ID = t2
  | #Columns = 2
  | | Relation Scan
  | | | Prefetch: Eligible
Insert Into Sorted Shared Temp Table  ID = t3
  #Columns = 5
  #Sort Key Columns = 1
  | Key 1: (Ascending)
  Use Partitioned Sort
  Sortheap Allocation Parameters:
  | #Rows     = 61
  | Row Width = 72
  Piped
  Access Temp Table  ID = t3
  #Columns = 5
  Relation Scan
  | Prefetch: Eligible
  Merge Join
  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
  | #Columns = 3
  | Parallel Scan
  | | Relation Scan
  | | | Prefetch: Eligible
  | Lock Intents
  | | | Table: Intent Share
  | | | Row  : Next Key Share
  | | Insert Into Sorted Shared Temp Table  ID = t4
  | | | #Columns = 3
  | | | #Sort Key Columns = 1
  | | | Key 1: WORKDEPT (Ascending)
  | | | Use Partitioned Sort
```

```
   |  |  |  |  Sortheap Allocation Parameters:
|  |  |  |  |  #Rows     = 63
|  |  |  |  |  Row Width = 32
   |  |  |  |  Piped
|  |  Sorted Shared Temp Table Completion  ID = t4
   |  |  Access Temp Table  ID = t4
|  |  |  #Columns = 3
   |  |  |  Relation Scan
   |  |  |  |  Prefetch: Eligible
|  Insert Into Asynchronous Local Table Queue  ID = q1
Access Local Table Queue  ID = q1  #Columns = 5
   Return Data to Application
|  #Columns = 5

End of section


Optimizer Plan:

             RETURN
             (   1)
               |
              LTQ
             (   2)
               |
             MSJOIN
              (   3)
             /      \
       TBSCAN         TBSCAN
        (   4)        (  15)
          |             |
         SORT           SORT
        (   5)        (  16)
          |             |
        MSJOIN         TBSCAN
        (   6)        (  17)
       /      \         |
   TBSCAN    TBSCAN  Table:
   (   7)    (  11)  DOOLE
     |         |     EMPLOYEE
    SORT           SORT
   (   8)    (  12)
     |         |
    TBSCAN         TBSCAN
   (   9)    (  13)
     |         |
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

Dieser Plan stimmt mit dem Plan des ersten Beispiels weitgehend überein. Der Unterschied besteht in der Erstellung von vier Subagenten beim ersten Starten des Plans und der Tabellenwarteschlange am Ende des Plans, um die Ergebnisse der Arbeit aller Subagenten aufzunehmen, bevor sie an die Anwendung zurückgegeben werden.

Außerdem ist die Beobachtung interessant, dass vor der Verknüpfung mit EMPLOYEE eine weitere Sortierung erforderlich ist. Dies ist nötig, weil die Subagenten, die die Mischverknüpfung zwischen DEPARTMENT und PROJECT verarbeiten, verknüpfte Zeilen in falscher Reihenfolge hervorbringen können.

Beispiel 3: Zugriffsplan für eine Datenbank mit mehreren Partitionen und partitionsübergreifender Parallelität

Dieses Beispiel zeigt wiederum dieselbe SQL-Anweisung wie „Beispiel 1: Plan ohne Parallelität" auf Seite 677, aber hier wurde die Abfrage auf einer partitionierten Datenbank kompiliert, die aus drei Datenbankpartitionen besteht.

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:21:29

Bind Timestamp = 2000-01-03-15.21.29.990983

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = Yes
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno


Estimated Cost        = 118
Estimated Cardinality = 263

Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
```

```
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #3
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #1
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Access Table Queue  ID = q1  #Columns = 5
   Return Data to Application
   |  #Columns = 5

Subsection #1:
   Access Table Queue  ID = q2  #Columns = 3
   |  Output Sorted
   |  |  #Key Columns = 1
   |  |  |  Key 1: (Ascending)
   Merge Join
   |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   |  |  Lock Intents
   |  |  |  Table: Intent Share
   |  |  |  Row  : Next Key Share
   |  |  Insert Into Sorted Temp Table  ID = t1
   |  |  |  #Columns = 3
   |  |  |  #Sort Key Columns = 1
   |  |  |  |  Key 1: DEPTNO (Ascending)
   |  |  |  Sortheap Allocation Parameters:
   |  |  |  |  #Rows     = 40
   |  |  |  |  Row Width = 48
   |  |  |  Piped
   |  Sorted Temp Table Completion  ID = t1
   |  Access Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   Merge Join
   |  Access Table Queue  ID = q3  #Columns = 2
   |  |  Output Sorted
   |  |  |  #Key Columns = 1
   |  |  |  |  Key 1: (Ascending)
   Insert Into Asynchronous Table Queue  ID = q1
   |  Broadcast to Coordinator Node
   |  Rows Can Overflow to Temporary Table

Subsection #2:
   Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
   |  #Columns = 3
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Lock Intents
   |  |  Table: Intent Share
   |  |  Row  : Next Key Share
   |  Insert Into Sorted Temp Table  ID = t2
```

```
    |  |  #Columns = 3
    |  |  #Sort Key Columns = 1
    |  |  |  Key 1: WORKDEPT (Ascending)
    |  |  Sortheap Allocation Parameters:
    |  |  |  #Rows     = 27
    |  |  |  Row Width = 32
    |  |  Piped
   Sorted Temp Table Completion  ID = t2
   Access Temp Table  ID = t2
   |  #Columns = 3
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Insert Into Asynchronous Table Queue  ID = q2
   |  |  Hash to Specific Node
   |  |  Rows Can Overflow to Temporary Tables
   Insert Into Asynchronous Table Queue Completion  ID = q2

Subsection #3:
   Access Table Name = DOOLE.PROJECT  ID = 2,7
   |  #Columns = 2
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Lock Intents
   |  |  Table: Intent Share
   |  |  Row  : Next Key Share
   |  Insert Into Sorted Temp Table  ID = t3
   |  |  #Columns = 2
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: DEPTNO (Ascending)
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 38
   |  |  |  Row Width = 28
   |  |  Piped
   Sorted Temp Table Completion  ID = t3
   Access Temp Table  ID = t3
   |  #Columns = 2
   |  Relation Scan
   |  |  Prefetch: Eligible
   |  Insert Into Asynchronous Table Queue  ID = q3
   |  |  Hash to Specific Node
   |  |  Rows Can Overflow to Temporary Tables
   Insert Into Asynchronous Table Queue Completion  ID = q3
End of section

Optimizer Plan:
              RETURN
              (   1)
                 |
               BTQ
               (   2)
                 |
              MSJOIN
               (   3)
              /      \
       MSJOIN          MDTQ
```

```
      (   4)          ( 14)
     /     \            |
   MDTQ     TBSCAN    TBSCAN
  (   5)    (  10)    (  15)
    |         |         |
  TBSCAN     SORT      SORT
  (   6)    (  11)    (  16)
    |         |         |
   SORT     TBSCAN    TBSCAN
  (   7)    (  12)    (  17)
    |         |         |
  TBSCAN   Table:     Table:
  (   8)   DOOLE      DOOLE
    |      DEPARTMENT PROJECT
 Table:
 DOOLE
 EMPLOYEE
```

Dieser Plan enthält dieselben Bestandteile wie der Plan im ersten Beispiel, aber der Bereich wurde in vier Teilbereiche (Subsection) unterteilt. Die Teilbereiche haben folgende Aufgaben:

- **Coordinator Subsection (Koordinatorteilbereich)**. Dieser Teilbereich koordiniert die anderen Teilbereiche. In diesem Plan sorgt er dafür, dass die anderen Teilbereiche verteilt werden, und er verwendet anschließend eine Tabellenwarteschlange, um die Ergebnisse zu sammeln, die an die Anwendung zurückgegeben werden sollen.
- **Subsection #1**. Dieser Teilbereich durchsucht die Tabellenwarteschlange q2 und verwendet eine Mischverknüpfung, um sie mit der Tabelle DEPARTMENT zu verknüpfen. Eine zweite Mischverknüpfung fügt anschließend die Daten aus Tabellenwarteschlange q3 ein. Die verknüpften Zeilen werden dann über Tabellenwarteschlange q1 an den Koordinatorteilbereich gesendet.
- **Subsection #2**. Dieser Teilbereich durchsucht die Tabelle EMPLOYEE, sortiert sie und verteilt sie per Hash-Verfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.
- **Subsection #3**. Dieser Teilbereich durchsucht die Tabelle PROJECT, sortiert sie und verteilt sie per Hash-Verfahren mit den Ergebnissen an einen bestimmten Knoten. Diese Ergebnisse werden von Subsection #1 gelesen.

Beispiel 4: Zugriffsplan für die Datenbank mit mehreren Partitionen sowie partitionsinterner und partitionsübergreifender Parallelität

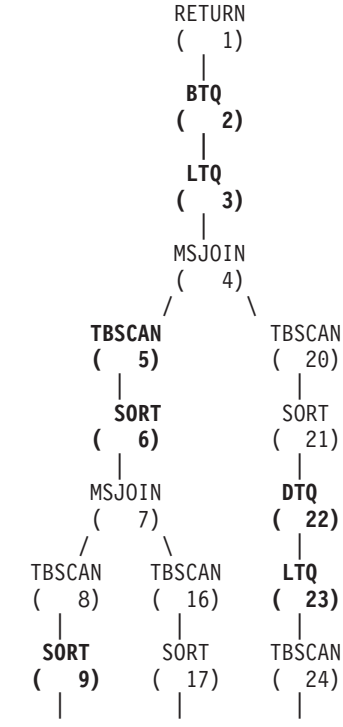
In diesem Beispiel wird dieselbe SQL-Anweisung wie in „Beispiel 1: Plan ohne Parallelität" auf Seite 677 gezeigt, aber die Abfrage wurde auf einer partitionierten Datenbank mit drei Datenbankpartitionen kompiliert, die jeweils auf einer 4-Wege-SMP-Maschine sind.

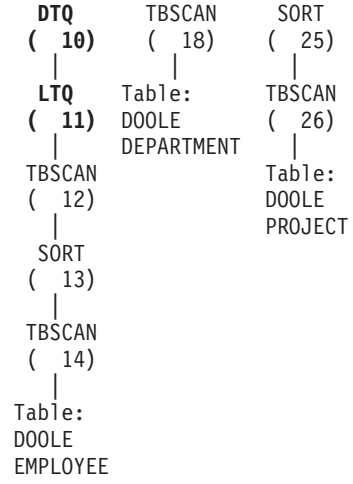
```
****************** PACKAGE ************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:22:14

Bind Timestamp = 2000-01-03-15.22.14.659970

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = Yes
Intra-Partition Parallel = Yes (Bind Degree = 4)

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 140
Estimated Cardinality = 263

Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #3
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #1
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Access Table Queue  ID = q1  #Columns = 5
   Return Data to Application
   |  #Columns = 5
```

```
Subsection #1:
   Process Using 4 Subagents
   |  Access Table Queue  ID = q3  #Columns = 3
   |  Insert Into Sorted Shared Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: (Ascending)
   |  |  Use Partitioned Sort
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 27
   |  |  |  Row Width = 32
   |  |  Piped
   |  Access Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   |  Merge Join
   |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
   |  |  |  #Columns = 3
   |  |  |  Parallel Scan
   |  |  |  Relation Scan
   |  |  |  |  Prefetch: Eligible
   |  |  |  Lock Intents
   |  |  |  |  Table: Intent Share
   |  |  |  |  Row  : Next Key Share
   |  |  |  Insert Into Sorted Shared Temp Table  ID = t2
   |  |  |  |  #Columns = 3
   |  |  |  |  #Sort Key Columns = 1
   |  |  |  |  |  Key 1: DEPTNO (Ascending)
   |  |  |  |  Use Partitioned Sort
   |  |  |  |  Sortheap Allocation Parameters:
   |  |  |  |  |  #Rows     = 40
   |  |  |  |  |  Row Width = 48
   |  |  |  |  Piped
   |  |  Sorted Shared Temp Table Completion  ID = t2
   |  |  Access Temp Table  ID = t2
   |  |  |  #Columns = 3
   |  |  |  Relation Scan
   |  |  |  |  Prefetch: Eligible
   |  Insert Into Sorted Shared Temp Table  ID = t3
   |  |  #Columns = 6
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: (Ascending)
   |  |  Use Partitioned Sort
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 44
   |  |  |  Row Width = 76
   |  |  Piped
   |  Access Temp Table  ID = t3
   |  |  #Columns = 6
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   |  Merge Join
   |  |  Access Table Queue  ID = q5  #Columns = 2
   |  |  Insert Into Sorted Shared Temp Table  ID = t4
```

```
    |  |  |  #Columns = 2
    |  |  |  #Sort Key Columns = 1
    |  |  |  |  Key 1: (Ascending)
    |  |  |  Use Partitioned Sort
    |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 38
    |  |  |  |  Row Width = 28
    |  |  |  Piped
    |  |  Access Temp Table  ID = t4
    |  |  |  #Columns = 2
    |  |  |  Relation Scan
    |  |  |  |  Prefetch: Eligible
    |  Insert Into Asynchronous Local Table Queue  ID = q2
    Access Local Table Queue  ID = q2  #Columns = 5
    Insert Into Asynchronous Table Queue  ID = q1
    |  Broadcast to Coordinator Node
    |  Rows Can Overflow to Temporary Table

Subsection #2:
    Process Using 4 Subagents
    |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
    |  |  #Columns = 3
    |  |  Parallel Scan
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    |  |  Lock Intents
    |  |  |  Table: Intent Share
    |  |  |  Row  : Next Key Share
    |  |  Insert Into Sorted Shared Temp Table  ID = t5
    |  |  |  #Columns = 3
    |  |  |  #Sort Key Columns = 1
    |  |  |  |  Key 1: WORKDEPT (Ascending)
    |  |  |  Use Round-Robin Sort
    |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 27
    |  |  |  |  Row Width = 32
    |  |  |  Piped
    |  Sorted Shared Temp Table Completion  ID = t5
    |  Access Temp Table  ID = t5
    |  |  #Columns = 3
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    |  Insert Into Asynchronous Local Table Queue  ID = q4
    Access Local Table Queue  ID = q4  #Columns = 3
    Insert Into Asynchronous Table Queue  ID = q3
    |  Hash to Specific Node
    |  Rows Can Overflow to Temporary Tables

Subsection #3:
    Process Using 4 Subagents
    |  Access Table Name = DOOLE.PROJECT  ID = 2,7
    |  |  #Columns = 2
    |  |  Parallel Scan
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
```

```
    |  |  Lock Intents
    |  |  |  Table: Intent Share
    |  |  |  Row  : Next Key Share
    |  |  Insert Into Sorted Shared Temp Table  ID = t6
    |  |  |  #Columns = 2
    |  |  |  #Sort Key Columns = 1
    |  |  |  |  Key 1: DEPTNO (Ascending)
    |  |  |  Use Round-Robin Sort
    |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 38
    |  |  |  |  Row Width = 28
    |  |  |  Piped
    |  Sorted Shared Temp Table Completion  ID = t6
    |  Access Temp Table  ID = t6
    |  |  #Columns = 2
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    |  Insert Into Asynchronous Local Table Queue  ID = q6
  Access Local Table Queue  ID = q6  #Columns = 2
  Insert Into Asynchronous Table Queue  ID = q5
    |  Hash to Specific Node
    |  Rows Can Overflow to Temporary Tables

End of section


Optimizer Plan:
```





Dieser Plan ist ähnlich wie der in „Beispiel 3: Zugriffsplan für eine Datenbank mit mehreren Partitionen und partitionsübergreifender Parallelität" auf Seite 683, nur dass mehrere Subagenten jeden Teilbereich ausführen. Außerdem sammelt am Ende eines jeden Teilbereichs eine lokale Tabelle die Ergebnisse von allen Subagenten, bevor die entsprechenden Zeilen in die zweite Tabellenwarteschlange eingefügt werden, die per Hash-Verfahren an einen bestimmten Knoten gesendet wird.

Beispiel 5: Zugriffsplan für eine zusammengeschlossene Datenbank

In diesem Beispiel wird dieselbe SQL-Anweisung wie in „Beispiel 1: Plan ohne Parallelität" auf Seite 677 gezeigt. Die Abfrage wurde jedoch auf einer zusammengeschlossenen Datenbank kompiliert, bei der sich die Tabellen DEPARTMENT und PROJECT auf einer Datenquelle befinden und die Tabelle EMPLOYEE auf dem Server mit zusammengeschlossenen Datenbanken ist.

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 16:29:01

Bind Timestamp = 2000-01-03-16.29.01.479230

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Estimated Cost        = 1954
Estimated Cardinality = 100800

Distribute Subquery #2
   |  #Columns = 3
   Insert Into Sorted Shared Temp Table  ID = t1
   |  #Columns = 3
   |  #Sort Key Columns = 1
   |  |  Key 1: Remote Query #2, Output Column 1 (Ascending)
   |  Sortheap Allocation Parameters:
   |  |  #Rows     = 1000
   |  |  Row Width = 56
   |  Piped
Access Temp Table  ID = t1
   |  #Columns = 3
   |  Relation Scan
   |  |  Prefetch: Eligible
   Merge Join
   |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,5
```

```
    |  #Columns = 3
    |  Relation Scan
    |  |  Prefetch: Eligible
    |  |  Lock Intents
    |  |  |  Table: Intent Share
    |  |  |  Row  : Next Key Share
|   |  Insert Into Sorted Temp Table  ID = t2
    |  |  |  #Columns = 3
    |  |  |  #Sort Key Columns = 1
    |  |  |  |  Key 1: WORKDEPT (Ascending)
    |  |  |  Sortheap Allocation Parameters:
|   |  |  |  #Rows     = 63
    |  |  |  |  Row Width = 32
    |  |  |  Piped
|   Sorted Temp Table Completion  ID = t2
|   Access Temp Table  ID = t2
    |  |  #Columns = 3
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    Merge Join
    |  Distribute Subquery #1
    |  |  #Columns = 2
    |  Insert Into Sorted Temp Table  ID = t3
    |  |  #Columns = 2
    |  |  |  Key 1: Remote Query #1, Output Column 1 (Ascending)
    |  |  Sortheap Allocation Parameters:
    |  |  |  #Rows     = 1000
    |  |  |  Row Width = 36
    |  |  Piped
    |  Access Temp Table  ID = t3
    |  |  #Columns = 2
    |  |  Relation Scan
    |  |  |  Prefetch: Eligible
    Return Data to Application
    |  #Columns = 5

    Distributed Subquery #1:
    Server: REMOTE_SAMPLE  (DB2/CS 7.1)
    Subquery SQL Statement:

         SELECT A0."DEPTNO", A0."PROJNAME"
         FROM "DOOLE"."PROJECT" A0

    Nicknames Referenced:
       REMOTE.PROJECT  ID = 7  Base = DOOLE.PROJECT
    #Output Columns = 2

    Distributed Subquery #2:
    Server: REMOTE_SAMPLE  (DB2/CS 7.1)
    Subquery SQL Statement:

         SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
         FROM "DOOLE"."DEPARTMENT" A0

    Nicknames Referenced:
```

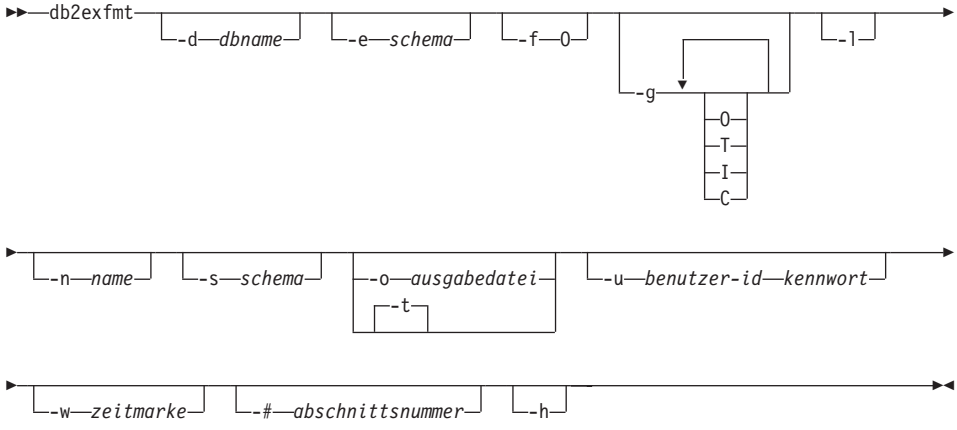
```
     REMOTE.DEPARTMENT  ID = 4  Base = DOOLE.DEPARTMENT
   #Output Columns = 3

End of section


Optimizer Plan:

             RETURN
             (   1)
                |
             MSJOIN
             (   2)
             /       \
       MSJOIN        TBSCAN
       (   3)          (  13)
      /     \            |
   TBSCAN    TBSCAN     SORT
  (   4)     (   9)     (  14)
    |          |         |
   SORT        SORT     DSBQRY
  (   5)     (  10)     (  15)
    |          |         |
  DSBQRY      TBSCAN   Nickname:
  (   6)     (  11)    REMOTE
    |          |       PROJECT
 Nickname:   Table:
 REMOTE      DOOLE
 DEPARTMENT  EMPLOYEE
```

Dieser Plan enthält dieselben Bestandteile wie der Plan im ersten Beispiel. Ausgenommen hiervon sind die Daten für zwei der Tabellen, die von Datenquellen kommen. Auf diese zwei Tabellen wird über verteilte Unterabfragen zugegriffen, die in diesem Fall einfach alle Zeilen der betreffenden Tabellen auswählen. Wenn die Daten zum Server mit zusammengeschlossenen Datenbanken zurückgegeben werden, werden sie mit den Daten aus der lokalen Tabelle verknüpft.

Anhang D. db2exfmt - EXPLAIN-Tool für Tabellenformat

Mit dem Tool `db2exfmt` können Sie den Inhalt der EXPLAIN-Tabellen formatieren. Dieses Tool befindet sich im Unterverzeichnis `misc` des Verzeichnisses `sqllib` für Ihr Exemplar.

Sie müssen über Lesezugriff auf die zu formatierenden EXPLAIN-Tabellen verfügen, um dieses Tool verwenden zu können.



-d dbname
Name der Datenbank mit den Paketen

-e schema
EXPLAIN-Tabellenschema

-f Formatierungsmarkierungen. In diesem Release wird nur der Wert `O` (Operatorzusammenfassung) unterstützt.

-g Diagrammzugriffsplan. Wenn nur `-g` angegeben wird, wird ein Diagramm gefolgt von den Formatierungsangaben für alle Tabellen generiert. Ansonsten kann eine beliebige Kombination der folgenden gültigen Werte angegeben werden:

- **O** Nur ein Diagramm wird generiert. Der Tabelleninhalt wird nicht formatiert.
- **T** In das Diagramm wird der Gesamtaufwand unter jedem Operator aufgenommen.
- **I** In das Diagramm wird der Ein-/Ausgabeaufwand unter jedem Operator aufgenommen.
- **C** In das Diagramm wird die erwartete Ausgabekardinalität (Anzahl Tupel) für jeden Operator aufgenommen.

-l Die Groß-/Kleinschreibung wird während der Verarbeitung der Paketnamen beachtet.

-n name
Name für die Quelle der EXPLAIN-Anforderung (SOURCE_NAME)

-s schema
Schema oder Qualifikationsmerkmal für die Quelle der EXPLAIN-Anforderung (SOURCE_SCHEMA)

-o ausgabedatei
Name der Ausgabedatei

-t Leiten der Ausgabe an die Workstation

-u benutzer-ID kennwort
Mit dieser Option werden eine Benutzer-ID und ein Kennwort für die Verbindung zu einer Datenbank angegeben.

Sowohl die Benutzer-ID als auch das Kennwort müssen gemäß den Namenskonventionen gültig sein und von der Datenbank erkannt werden.

-w zeitmarke
EXPLAIN-Zeitmarke. Geben Sie `-1` an, um die letzte EXPLAIN-Anforderung abzurufen.

-# abschnittsnummer
Abschnittsnummer in der Quelle. Geben Sie null an, um alle Abschnitte anzufordern.

-h Anzeigen des Hilfetexts. Wenn diese Option angegeben wird, werden alle anderen Optionen ignoriert, und lediglich der Hilfetext wird angezeigt.

Sie werden aufgefordert, die nicht angegebenen Parameterwerte anzugeben, oder Ihnen wird mitgeteilt, dass nicht alle angegeben wurden, ausgenommen im Fall der Optionen `-h` und `-l`.

Wenn kein EXPLAIN-Tabellenschema angegeben wird, wird der Wert der Umgebungsvariable **USER** als Standardwert verwendet. Wenn diese Variable nicht gefunden wird, wird der Benutzer aufgefordert, ein EXPLAIN-Tabellenschema anzugeben.

Der Quellenname, das Quellenschema und die EXPLAIN-Zeitmarke können in Form eines LIKE-Vergleichselements angegeben werden, in dem das Prozentzeichen (%) und das Unterstreichungszeichen (_) als Platzhalterzeichen zur Auswahl mehrerer Quellen bei einem Aufruf verwendet werden können. Für die letzte mit EXPLAIN bearbeitete Anweisung kann die EXPLAIN-Zeit als `-1` angegeben werden.

Wenn `-o` ohne Dateiname und `-t` nicht angegeben wird, wird der Benutzer aufgefordert, einen Dateinamen anzugeben (der Standardname ist `db2exfmt.out`). Wenn weder `-o` noch `-t` angegeben wird, wird der Benutzer aufgefordert, einen Dateinamen anzugeben (die Standardoption ist Terminalausgabe). Wenn sowohl `-o` als auch `-t` angegeben werden, wird die Ausgabe an das Terminal geleitet.

Anhang E. Verwenden der DB2-Bibliothek

Die Bibliothek für DB2 Universal Database besteht aus Online-Hilfe, Handbüchern (PDF und HTML) und Beispielprogrammen in HTML-Format. Im Folgenden wird beschrieben, welche Informationen bereitgestellt werden und wie Sie darauf zugreifen können.

Über **Information - Unterstützung** können Sie online auf die Produktinformationen zugreifen. Weitere Informationen finden Sie in „Zugreifen auf Informationen mit ″Information - Unterstützung″“ auf Seite 718. Sie können sich im Web Informationen zu Tasks und zur Fehlerbehebung sowie DB2-Bücher, Beispielprogramme und DB2-Informationen anzeigen lassen.

PDF-Dateien und gedruckte Bücher für DB2

Informationen zu DB2

In der folgenden Tabelle sind die DB2-Handbücher in vier Kategorien unterteilt:

DB2-Benutzerhandbücher und -Referenzinformationen
Diese Bücher enthalten die allgemeinen DB2-Informationen für alle Plattformen.

DB2-Installations- und -Konfigurationsinformationen
Diese Bücher gelten für DB2 auf einer bestimmten Plattform. So steht beispielsweise jeweils ein separates Handbuch *Einstieg* (Quick Beginnings) für DB2 auf OS/2-, Windows- und UNIX-Plattformen zur Verfügung.

Plattformübergreifende Beispielprogramme in HTML
Bei diesen Beispielen handelt es sich um die HTML-Versionen der mit Application Development Client installierten Beispielprogramme. Sie dienen zur Information und können die Programme selbst nicht ersetzen.

Release-Informationen
Diese Dateien enthalten die neuesten Informationen, die in die DB2-Handbücher nicht mehr aufgenommen werden konnten.

Die Installationshandbücher, Release-Informationen und Lernprogramme können im HTML-Format direkt von der Produkt-CD-ROM angezeigt werden. Die meisten Handbücher stehen auf der Produkt-CD-ROM im HTML-Format zur Verfügung und können angezeigt werden. Auf der CD-ROM mit DB2-Veröffentlichungen stehen die Handbücher im PDF-Format zur Verfügung und

können mit Adobe Acrobat angezeigt und gedruckt werden. Darüber hinaus können Sie gedruckte Veröffentlichungen bei IBM bestellen. Siehe hierzu „Bestellen der gedruckten Handbücher" auf Seite 713. Die folgende Tabelle enthält eine Liste der Bücher, die bestellt werden können.

Auf OS/2- und Windows-Plattformen können Sie die HTML-Dateien im Verzeichnis `sqllib\doc\html` installieren. Die DB2-Informationen werden in verschiedene Sprachen übersetzt, jedoch nicht alle Informationen in alle Sprachen. Sind bestimmte Informationen in einer Sprache nicht verfügbar, wird stattdessen die englische Version dieser Informationen zur Verfügung gestellt.

Auf UNIX-Plattformen können Sie die HTML-Dateien in mehreren Sprachen installieren, und zwar in den Unterverzeichnissen `doc/%L/html`, wobei *%L* für den Code der jeweiligen Landessprache steht. Weitere Informationen finden Sie im entsprechenden Handbuch *Einstieg*.

Es gibt verschiedene Möglichkeiten, auf DB2-Bücher und -Informationen zuzugreifen:

- „Anzeigen von Online-Informationen" auf Seite 717
- „Suchen nach Online-Informationen" auf Seite 722
- „Bestellen der gedruckten Handbücher" auf Seite 713
- „Drucken der PDF-Handbücher" auf Seite 712

Tabelle 45. Informationen zu DB2

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
DB2-Benutzerhandbücher und -Referenzinformationen			
Systemverwaltung	*Systemverwaltung: Konzept*. Dieses Handbuch enthält eine Übersicht über Datenbankkonzepte, Informationen zu Aspekten des Datenbankentwurfs (wie z. B. zum logischen und physischen Datenbankentwurf) sowie eine Erläuterung zur hohen Verfügbarkeit. *Systemverwaltung: Implementierung*. Dieses Handbuch enthält Informationen zu Implementierungsaspekten, wie beispielsweise zur Implementierung des Datenbankentwurfs, zum Zugriff auf Datenbanken sowie zu Prüfungs-, Sicherungs- und Wiederherstellungsverfahren. *Systemverwaltung: Optimierung*. Dieses Handbuch enthält Informationen zur Datenbankumgebung sowie zur Auswertung und Optimierung der Anwendungsleistung. Sie können die drei Bände des Handbuchs *Systemverwaltung* in englischer Sprache in den USA und Kanada über die Formnummer SBOF-8934 bestellen.	SC12-2879 db2d1g70 SC12-2878 db2d2g70 SC12-2877 db2d3g70	db2d0
Administrative API Reference	Dieses Handbuch enthält eine Beschreibung zu den DB2-Anwendungsprogrammierschnittstellen (APIs) und -Datenstrukturen, die Sie zum Verwalten Ihrer Datenbank verwenden können. Darüber hinaus wird in diesem Handbuch erläutert, wie Sie APIs von Ihren Anwendungen aus aufrufen können.	SC09-2947 db2b0e70	db2b0
Application Building Guide	Dieses Handbuch umfasst Informationen zur Umgebungskonfiguration sowie Anweisungsschritte zum Kompilieren, Verbinden und Ausführen von DB2-Anwendungen auf Windows-, OS/2- und UNIX-Plattformen.	SC09-2948 db2axe70	db2ax

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
APPC, CPI-C, and SNA Sense Codes	Dieses Handbuch enthält Basisinformationen zu APPC-, CPI-DFV- und SNA-Prüfcodes, die bei der Arbeit mit DB2 Universal Database-Produkten ausgegeben werden können. Nur im HTML-Format verfügbar.	Keine Formnummer db2ape70	db2ap
Application Development Guide	Dieses Handbuch enthält eine Erläuterung zur Entwicklung von Anwendungen, die mit Hilfe von eingebettetem SQL bzw. JAVA (JDBC und SQLJ) auf DB2-Datenbanken zugreifen. Unter anderem wird das Schreiben von gespeicherten Prozeduren, das Schreiben von benutzerdefinierten Funktionen, das Erstellen von benutzerdefinierten Typen, das Verwenden von Auslösern und das Entwickeln von Anwendungen in partitionierten Umgebungen oder mit Systemen zusammengeschlossener Datenbanken beschrieben.	SC09-2949 db2a0e70	db2a0
CLI Guide and Reference	Dieses Handbuch erklärt die Entwicklung von Anwendungen, die für den Zugriff auf DB2-Datenbanken DB2 Call Level Interface verwenden, eine aufrufbare SQL-Schnittstelle, die mit der Microsoft-ODBC-Spezifikation kompatibel ist.	SC09-2950 db2l0e70	db2l0
Command Reference	Dieses Handbuch enthält eine Erläuterung zur Verwendung des Befehlszeilenprozessors und eine Beschreibung der DB2-Befehle für die Datenbankverwaltung.	SC09-2951 db2n0e70	db2n0

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Konnektivität Ergänzung	Dieses Handbuch enthält Konfigurations- und Referenzinformationen zur Verwendung von DB2 für AS/400, DB2 für OS/390, DB2 für MVS oder DB2 für VM als DRDA-Anwendungs-Requester mit DB2 Universal Database-Servern. Darüber hinaus enthält dieses Handbuch Informationen zur Verwendung von DRDA-Anwendungs-Servern mit DB2 Connect-Anwendungs-Requestern. Dieses Buch ist lediglich im HTML- und PDF-Format verfügbar.	Keine Formnummer db2h1g70	db2h1
Versetzen von Daten Dienstprogramme und Referenz	Dieses Handbuch enthält eine Erläuterung zur Verwendung der DB2-Dienstprogramme, wie beispielsweise IMPORT, EXPORT, LOAD, AUTOLOADER und DPROP, die das Verschieben von Daten vereinfachen.	SC12-2881 db2dmg70	db2dm
Data Warehouse-Zentrale Verwaltung	Dieses Handbuch enthält Informationen zur Erstellung und Verwaltung eines Data Warehouse mit Hilfe der Data Warehouse-Zentrale.	SC12-2885 db2ddg70	db2dd
Data Warehouse Center Application Integration Guide	Dieses Handbuch enthält Informationen, die Programmierer bei der Integration von Anwendungen in die Data Warehouse-Zentrale sowie in den Information Catalog Manager unterstützen.	SC26-9994 db2ade70	db2ad
DB2 Connect Benutzerhandbuch	Dieses Handbuch enthält eine Beschreibung der Konzepte der DB2 Connect-Produkte, allgemeine Informationen zur Verwendung sowie Informationen zur Programmierung dieser Produkte.	SC12-2880 db2c0g70	db2c0
DB2 Query Patroller Administration Guide	Dieses Handbuch enthält eine Übersicht über den Betrieb des DB2 Query Patroller-Systems, spezifische Informationen zum Systembetrieb und zur Verwaltung sowie Task-Informationen zu den GUI-Verwaltungsdienstprogrammen.	SC09-2958 db2dwe70	db2dw
DB2 Query Patroller User's Guide	In diesem Handbuch wird die Verwendung der Tools und Funktionen von DB2 Query Patroller beschrieben.	SC09-2960 db2wwe70	db2ww

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Glossar	Dieses Handbuch enthält Definitionen zu den in DB2 und den zugehörigen Komponenten verwendeten Begriffen. Es ist im Handbuch *SQL Reference* enthalten und steht außerdem separat im HTML-Format zur Verfügung.	Keine Formnummer db2t0g70	db2t0
DB2 UDB Image, Audio und Video Extender Verwaltung und Programmierung	Dieses Handbuch enthält Basisinformationen zu DB2 Extender, Informationen zur Verwaltung und Konfiguration von IAV Extender sowie Informationen zur Programmierung mit Hilfe von IAV Extender. Es enthält Referenzinformationen, Diagnoseinformationen (mit Nachrichten) und Beispiele.	SC12-2892 dmbu7g70	dmbu7
Information Catalog Manager Systemverwaltung	Dieses Handbuch enthält eine Anleitung zur Verwaltung von Informationskatalogen.	SC12-2886 db2dig70	db2di
Information Catalog Manager Programming Guide and Reference	Dieses Handbuch enthält Definitionen für die Architekturschnittstellen für Information Catalog Manager.	SC26-9997 db2bie70	db2bi
Information Catalog Manager Benutzerhandbuch	Dieses Handbuch enthält Informationen zur Verwendung der Information Catalog Manager-Benutzerschnittstelle.	SC12-2887 db2aig70	db2ai
Installation und Konfiguration Ergänzung	Dieses Handbuch enthält Anweisungen zur Planung, Installation und Konfiguration von plattformspezifischen DB2-Clients. Darüber hinaus enthält es Informationen zu Bindevorgängen, zum Einrichten der Client/Server-Kommunikation, zu DB2-GUI-Tools, zu DRDR-AS, zur verteilten Installation, zur Konfiguration von verteilten Anforderungen sowie zum Zugriff auf heterogene Datenquellen.	GC12-2864 db2iyg70	db2iy

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Fehlernachrichten	Dieses Handbuch enthält eine Liste der Nachrichten und Codes, die von DB2, vom Information Catalog Manager und von der Data Warehouse-Zentrale ausgegeben werden, sowie eine Beschreibung der jeweils erforderlichen Benutzeraktionen. Sie können beide Bände des Handbuchs *Fehlernachrichten* in englischer Sprache in den USA und Kanada unter der Formnummer SBOF-8932 bestellen.	Band 1 GC12-2875 db2m1g70 Band 2 GC12-2888 db2m2g70	db2m0
OLAP Integration Server Administration Guide	Dieses Handbuch enthält eine Erläuterung zur Verwendung der Komponente Administration Manager von OLAP Integration Server.	SC27-0782 db2dpe70	n/v
OLAP Integration Server Metaoutline User's Guide	Dieses Handbuch enthält eine Erläuterung zum Erstellen und Ausfüllen von OLAP-Metastrukturen mit Hilfe der OLAP Metaoutline-Standardschnittstelle (nicht mit Hilfe des OLAP Metaoutline Assistant).	SC27-0784 db2upe70	n/v
OLAP Integration Server Model User's Guide	Diese Handbuch enthält eine Erläuterung zum Erstellen von OLAP-Modellen mit Hilfe der OLAP Model-Standardschnittstelle (nicht mit Hilfe des OLAP Model Assistant).	SC27-0783 db2lpe70	n/v
OLAP Konfiguration und Benutzerhandbuch	Dieses Handbuch enthält Informationen zur Konfiguration und Einrichtung von OLAP Starter Kit.	SC12-2889 db2ipg70	db2ip
OLAP Tabellenkalkulations-Add-In Benutzerhandbuch für Excel	Dieses Handbuch enthält eine Beschreibung zur Verwendung des Tabellenkalkulationsprogramms Excel zum Analysieren von OLAP-Daten.	SC12-2890 db2epg70	db2ep
OLAP Tabellenkalkulations-Add-In Benutzerhandbuch für Lotus 1-2-3	Dieses Handbuch enthält eine Beschreibung zur Verwendung des Tabellenkalkulationsprogramms Lotus 1-2-3 zum Analysieren von OLAP-Daten.	SC12-2891 db2tpg70	db2tp
Replikation Benutzer- und Referenzhandbuch	Dieses Handbuch enthält Informationen zur Planung, Konfiguration, Verwaltung und Verwendung der mit DB2 gelieferten Replikations-Tools.	SC12-2884 db2e0g70	db2e0

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Spatial Extender Benutzer- und Referenzhandbuch	Diese Handbuch enthält Informationen zur Installation, Konfiguration, Verwaltung, Programmierung und Fehlerbehebung für den Spatial Extender. Darüber hinaus enthält es zentrale Beschreibungen räumlicher Datenkonzepte sowie spezifische Referenzinformationen (Nachrichten und SQL) für den Spatial Extender.	SC12-2894 db2sbg70	db2sb
SQL Erste Schritte	Dieses Handbuch enthält eine Einführung in die SQL-Konzepte sowie Beispiele für eine Reihe von Konstrukten und Tasks.	SC12-2882 db2y0g70	db2y0
SQL Reference, Band 1 und Band 2	Dieses Handbuch beschreibt die Syntax, die Semantik und die Regeln von SQL. Darüber hinaus enthält das Handbuch Informationen zu Inkompatibilitäten zwischen Release-Ständen, Produkteinschränkungen und Katalogsichten. Sie können beide Bände des Handbuchs *SQL Reference* in englischer Sprache in den USA und Kanada unter der Formnummer SBOF-8933 bestellen.	Band 1 SC09-2974 db2s1e70 Band 2 SC09-2975 db2s2e70	db2s0
System Monitor Guide and Reference	Dieses Handbuch enthält eine Beschreibung zum Sammeln unterschiedlicher Informationen zu Datenbanken und dem Datenbankmanager. In diesem Buch wird erläutert, wie Sie mit Hilfe dieser Informationen einen Einblick in Datenbankaktivitäten erhalten, die Leistung verbessern und Fehlerursachen feststellen können.	SC09-2956 db2f0e70	db2f0
Text Extender Verwaltung und Programmierung	Dieses Handbuch enthält Basisinformationen zu DB2 Extender, Informationen zur Verwaltung und Konfiguration von Text Extender sowie zur Programmierung mit Hilfe von Text Extender. Es bietet Referenzinformationen, Diagnoseinformationen (mit Nachrichten) und Beispiele.	SC12-2893 desu9g70	desu9

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
Troubleshooting Guide	Dieses Handbuch hilft Ihnen bei der Bestimmung von Fehlerquellen, bei der Fehlerbehebung sowie bei der Verwendung von Diagnose-Tools, wenn Sie den DB2-Kundendienst in Anspruch nehmen.	GC09-2850 db2p0e70	db2p0
Neue Funktionen	Dieses Handbuch enthält eine Beschreibung der neuen Einrichtungen, Funktionen und Erweiterungen in DB2 Universal Database Version 7.	SC12-2883 db2q0g70	db2q0
DB2-Installations- und -Konfigurationsinformationen			
DB2 Connect Enterprise Edition für OS/2 und Windows Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation und Konfiguration für DB2 Connect Enterprise Edition unter OS/2 und 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2863 db2c6g70	db2c6
DB2 Connect Enterprise Edition für UNIX Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation, Konfiguration und Ausführung von Tasks für DB2 Connect Enterprise Edition auf UNIX-Plattformen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2862 db2cyg70	db2cy
DB2 Connect Personal Edition Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation, Konfiguration und Ausführung von Tasks für DB2 Connect Personal Edition unter OS/2 und 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für alle unterstützten Clients.	GC12-2869 db2c1g70	db2c1
DB2 Connect Personal Edition für Linux Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Connect Personal Edition für alle unterstützten Linux-Varianten.	GC12-2865 db2c4g70	db2c4

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
DB2 Data Links Manager Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Konfiguration und Ausführung von Tasks für DB2 Data Links Manager unter AIX und 32-Bit-Windows-Betriebssystemen.	GC12-2868 db2z6g70	db2z6
DB2 Enterprise - Extended Edition für UNIX Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation und Konfiguration für DB2 Enterprise - Extended Edition auf UNIX-Plattformen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2867 db2v3g70	db2v3
DB2 Enterprise - Extended Edition für Windows Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation und Konfiguration für DB2 Enterprise - Extended Edition unter 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2866 db2v6g70	db2v6
DB2 für OS/2 Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation und Konfiguration von DB2 Universal Database für das Betriebssystem OS/2. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2870 db2i2g70	db2i2
DB2 für UNIX Einstieg	Dieses Handbuch enthält Informationen zur Planung, Migration, Installation und Konfiguration von DB2 Universal Database auf UNIX-Plattformen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2872 db2ixg70	db2ix

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
DB2 für Windows Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Universal Database unter 32-Bit-Windows-Betriebssystemen. Darüber hinaus enthält dieses Handbuch Installations- und Konfigurationsinformationen für eine Reihe von unterstützten Clients.	GC12-2873 db2i6g70	db2i6
DB2 Personal Edition Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Universal Database Personal Edition unter OS/2 und 32-Bit-Windows-Betriebssystemen.	GC12-2871 db2i1g70	db2i1
DB2 Personal Edition für Linux Einstieg	Dieses Handbuch enthält Informationen zur Planung, Installation, Migration und Konfiguration für DB2 Universal Database Personal Edition für alle unterstützten Linux-Varianten.	GC12-2874 db2i4g70	db2i4
DB2 Query Patroller Installation Guide	Dieses Handbuch enthält Installationsinformationen zu DB2 Query Patroller.	GC09-2959 db2iwe70	db2iw
DB2 Warehouse Manager Installation	Dieses Handbuch enthält Installationsinformationen für Warehouse-Agenten, Warehouse-Umsetzungsprogramme und den Information Catalog Manager.	GC12-2876 db2ide70	db2id
Plattformübergreifende Beispielprogramme in HTML			
Beispielprogramme in HTML	Dieses Handbuch enthält die Beispielprogramme für die Programmiersprachen auf allen von DB2 unterstützten Plattformen im HTML-Format. Die Beispielprogramme werden lediglich zu Informationszwecken zur Verfügung gestellt. Nicht alle Beispiele sind für alle Programmiersprachen verfügbar. Die HTML-Beispiele stehen nur dann zur Verfügung, wenn der DB2 Application Development Client installiert ist. Weitere Informationen zu den Programmen finden Sie im Handbuch *Application Building Guide*.	Keine Formnummer	db2hs

Tabelle 45. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
	Release-Informationen		
DB2 Connect Release-Informationen	Dieses Dokument enthält die neuesten Informationen, die in die DB2 Connect-Handbücher nicht mehr aufgenommen werden konnten.	Siehe Anmerkung 2.	db2cr
DB2 Installationsinformationen	Dieses Dokument enthält die neuesten Informationen zur Installation, die in die DB2-Handbücher nicht mehr aufgenommen werden konnten.	Nur auf der Produkt-CD-ROM verfügbar.	
DB2-Release-Informationen	Dieses Dokument enthält die neuesten Informationen zu allen DB2-Produkten und -Funktionen, die in die DB2-Handbücher nicht mehr aufgenommen werden konnten.	Siehe Anmerkung 2.	db2ir

Anmerkungen:

1. Das Zeichen an der sechsten Stelle des Dateinamens gibt die Landessprache eines Buchs an. So kennzeichnet der Dateiname `db2d0e70` die englische Version des Handbuchs *Systemverwaltung*, der Dateiname `db2d0f70` kennzeichnet die französische Version des Buchs. Folgende Buchstaben werden an der sechsten Stelle des Dateinamens verwendet, um die Landessprache für ein Handbuch anzugeben:

Sprache	Kennung
Brasilianisches Portugiesisch	b
Bulgarisch	u
Dänisch	d
Deutsch	g
Englisch	e
Finnisch	y
Französisch	f
Griechisch	a
Italienisch	i
Japanisch	j
Koreanisch	k
Niederländisch	q
Norwegisch	n
Polnisch	p
Portugiesisch	v
Russisch	r
Schwedisch	s
Slowenisch	l
Spanisch	z
Trad. Chinesisch	t
Tschechisch	x
Türkisch	m
Ungarisch	h
Vereinf. Chinesisch	c

2. Kurzfristig verfügbare Informationen, die in die DB2-Handbücher nicht mehr aufgenommen werden können, sind in den Release-Informationen enthalten, die im HTML-Format und als ASCII-Datei verfügbar sind. Die HTML-Version steht über 'Information - Unterstützung' und auf den Produkt-CD-ROMs zur Verfügung. Gehen Sie wie folgt vor, um die ASCII-Dateien anzuzeigen:
   - Rufen Sie auf UNIX-Plattformen die Datei `Release.Notes` auf. Diese Datei befindet sich im Verzeichnis `DB2DIR/Readme/%L`. Dabei ist `%L` die länderspezifische Angabe und `DB2DIR` eine der folgenden Angaben:
     - `/usr/lpp/db2_07_01` (unter AIX)
     - `/opt/IBMdb2/V7.1` (unter HP-UX, PTX, Solaris und Silicon Graphics IRIX)
     - `/usr/IBMdb2/V7.1` (unter Linux)
   - Rufen Sie auf anderen Plattformen die Datei `RELEASE.TXT` auf. Diese Datei befindet sich in dem Verzeichnis, in dem das Produkt installiert ist. Auf OS/2-Plattformen können Sie auch den Ordner **IBM DB2** und anschließend das Symbol **Release-Informationen** doppelt anklicken.

Drucken der PDF-Handbücher

Wenn Sie eine gedruckte Version der Handbücher bevorzugen, können Sie die PDF-Dateien auf der CD-ROM mit DB2-Veröffentlichungen ausdrucken. Mit Adobe Acrobat Reader können Sie entweder das gesamte Handbuch oder bestimmte Teile des Handbuchs ausdrucken. Die Namen der einzelnen Handbücher in der Bibliothek finden Sie in Tabelle 45 auf Seite 701.

Die neueste Version von Adobe Acrobat Reader finden Sie auf der Adobe-Web-Site unter http://www.adobe.com.

Die PDF-Dateien befinden sich auf der CD-ROM mit DB2-Veröffentlichungen und haben die Dateierweiterung PDF. Führen Sie folgende Schritte aus, um auf die PDF-Dateien zuzugreifen:

1. Legen Sie die CD-ROM mit DB2-Veröffentlichungen in das CD-ROM-Laufwerk ein. Auf UNIX-Plattformen: Hängen Sie die CD-ROM mit den DB2-Veröffentlichungen an. Das Handbuch *Einstieg* enthält Anweisungen zu den Mount-Prozeduren.
2. Starten Sie Acrobat Reader.
3. Öffnen Sie die gewünschte PDF-Datei von einer der folgenden Positionen aus:
   - Auf OS/2- und Windows-Plattformen:

     Verzeichnis *x*`:\doc\`*sprache* . Dabei gibt *x* das CD-ROM-Laufwerk an, *sprache* den zweistelligen Landescode für die verwendete Sprache (z. B. EN für Englisch).

- Auf UNIX-Plattformen:

  Verzeichnis */cdrom*/doc/%*L* auf der CD-ROM. Dabei gibt */cdrom* den Mount-Punkt der CD-ROM an und %*L* den Namen der gewünschten länderspezifischen Angaben.

Sie können die PDF-Dateien auch von der CD-ROM in ein lokales Laufwerk oder ein Netzlaufwerk kopieren und sie von dort aus lesen.

Bestellen der gedruckten Handbücher

Sie können die gedruckten DB2-Handbücher einzeln bestellen. In den USA und Kanada ist es außerdem möglich, mehrere Bücher als Paket unter einer SBOF-Nummer zu bestellen. Setzen Sie sich mit Ihrem IBM Vertragshändler oder Vertriebsbeauftragten in Verbindung, oder bestellen Sie die Handbücher telefonisch bei IBM Direkt unter der Nummer 0180/55 090. Darüber hinaus können Sie die Handbücher über die Web-Seite mit Veröffentlichungen unter http://www.elink.ibmlink.ibm.com/pbl/pbl bestellen.

Es sind zwei Gruppen von Handbüchern verfügbar. Die Gruppe mit der Formnummer SBOF-8935 umfasst Referenzinformationen und Informationen zur Verwendung für DB2 Warehouse Manager. Die Gruppe mit der Formnummer SBOF-8931 umfasst Referenzinformationen und Informationen zur Verwendung für alle anderen DB2 Universal Database-Produkte und -Funktionen. Der Inhalt der SBOF-Gruppen ist in der folgenden Tabelle aufgeführt.

Tabelle 46. Bestellen der gedruckten Handbücher

SBOF-Nummer	In dieser Gruppe enthaltene Handbücher	
SBOF-8931	• Administration Guide: Planning • Administration Guide: Implementation • Administration Guide: Performance • Administrative API Reference • Application Building Guide • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • Data Warehouse Center Administration Guide • Data Warehouse Center Application Integration Guide • DB2 Connect User's Guide • Installation and Configuration Supplement • Image, Audio, and Video Extenders Administration and Programming • Message Reference, Volumes 1 and 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volumes 1 and 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • What's New
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2-Online-Dokumentation

Zugreifen auf die Online-Hilfefunktion

Die Online-Hilfefunktion ist für alle DB2-Komponenten verfügbar. In der folgenden Tabelle werden die verschiedenen Hilfearten beschrieben.

Hilfearten	Inhalt	Zugriff
Hilfe für Befehl	Erklärt die Syntax von Befehlen im Befehlszeilenprozessor.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein: `? befehl` Dabei stellt *befehl* ein Schlüsselwort bzw. den vollständigen Befehl dar. So kann beispielsweise durch die Eingabe von `? catalog` Hilfe für alle CATALOG-Befehle angezeigt werden, während mit `? catalog database` lediglich Hilfe für den Befehl CATALOG DATABASE angezeigt wird.
Hilfe für **Client-Konfiguration - Unterstützung** *Hilfe für die Befehlszentrale* *Hilfe für die Steuerzentrale* *Hilfe für die Data Warehouse-Zentrale* *Hilfe für* **Event Analyzer** *Hilfe für Information Catalog Manager* *Hilfe für die Satellitenverwaltungszentrale* *Hilfe für die Prozedurenzentrale*	Erläutert die Tasks, die Sie in einem Fenster oder Notizbuch ausführen können. Die Hilfe umfasst Übersichtsinformationen und unbedingt erforderliche Informationen sowie eine Beschreibung zur Verwendung der Steuerelemente im Fenster oder Notizbuch.	Klicken Sie in einem Fenster oder in einem Notizbuch den Druckknopf **Hilfe** an oder drücken Sie die Taste **F1**.

Hilfearten	Inhalt	Zugriff

Nachrichtenhilfe	Beschreibt die Ursache von Nachrichten sowie die auszuführenden Benutzeraktionen.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein:
		`? XXXnnnnn`
		Dabei ist *XXXnnnnn* eine gültige Nachrichtenkennung.
		Bei Eingabe von `? SQL30081` wird z. B. die Hilfe zur Nachricht SQL30081 angezeigt.
		Wenn Sie die Nachrichtenhilfe seitenweise anzeigen möchten, geben Sie den folgenden Befehl ein:
		`? XXXnnnnn | more`
		Geben Sie folgenden Befehl ein, um die Nachrichtenhilfe in einer Datei zu speichern:
		`? XXXnnnnn > datei.erw`
		Dabei ist *datei.erw* die Datei, in der Sie die Nachrichtenhilfe speichern möchten.

Hilfe für SQL	Erklärt die Syntax von SQL-Anweisungen.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein: `help anweisung` Dabei gibt *anweisung* eine SQL-Anweisung an. So kann beispielsweise durch die Eingabe von `help SELECT` die Hilfe zur Anweisung SELECT angezeigt werden. **Anmerkung:** Die Hilfe für SQL ist auf UNIX-Plattformen nicht verfügbar.
SQLSTATE-Hilfe	Erklärt SQLSTATE-Werte und SQL-Klassencodes.	Geben Sie im interaktiven Modus des Befehlszeilenprozessors Folgendes ein: `? sqlstate` oder `? klassencode` Dabei ist *sqlstate* ein gültiger, fünfstelliger SQL-Status, und *klassencode* stellt die ersten zwei Ziffern des SQL-Statuswerts dar. So kann beispielsweise durch die Eingabe von `? 08003` Hilfe für den SQL-Statuswert 08003 angezeigt werden, während mit `? 08` Hilfe für den Klassencode 08 angezeigt wird.

Anzeigen von Online-Informationen

Die zum Lieferumfang dieses Produkts gehörenden Handbücher werden als Softcopy im HTML-Format (HTML - Hypertext Markup Language) bereitgestellt. In einer Softcopy können Sie die Informationen auf einfache Art suchen und anzeigen und über Hypertextverbindungen auf zugehörige Informationen zugreifen. Außerdem wird die gemeinsame Nutzung der Bibliothek in Ihrem gesamten Unternehmen erleichtert.

Sie können die Online-Bücher und Beispielprogramme mit jedem Browser anzeigen, der den Spezifikationen von HTML Version 3.2 entspricht.

Führen Sie die nachfolgend beschriebenen Schritte aus, um Online-Bücher oder Beispielprogramme anzuzeigen:

- Wenn Sie DB2-Verwaltungs-Tools ausführen, verwenden Sie **Information - Unterstützung**.
- Klicken Sie in einem Browser **Datei—>Seite öffnen** an. Die geöffnete Seite enthält eine Übersicht über die DB2-Informationen und Verbindungen (Links) zu diesen Informationen:
  - Öffnen Sie auf UNIX-Plattformen die folgende Seite:

    ```
    INSTHOME/sqllib/doc/%L/html/index.htm
    ```

    Dabei ist *%L* die länderspezifische Angabe.
  - Öffnen Sie auf anderen Plattformen die folgende Seite:

    ```
    sqllib\doc\html\index.htm
    ```

    Der Pfad befindet sich auf dem Laufwerk, auf dem DB2 installiert ist.

    Wenn Sie **Information - Unterstützung** nicht installiert haben, können Sie die Seite öffnen, indem Sie das Symbol **DB2-Informationen** doppelt anklicken. Je nach verwendetem Betriebssystem befindet sich das Symbol im Hauptproduktordner bzw. unter Windows im Menü **Start**.

Installieren des Netscape-Browsers

Wenn Sie nicht bereits einen Web-Browser installiert haben, können Sie Netscape von der im Lieferumfang des Produkts enthaltenen Netscape-CD-ROM aus installieren. Führen Sie folgende Schritte aus, um ausführliche Informationen zur Installation zu erhalten:

1. Legen Sie die Netscape-CD-ROM ein.
2. Nur auf UNIX-Plattformen: Hängen Sie die CD-ROM an. Das Handbuch *Einstieg* enthält Anweisungen zu den Mount-Prozeduren.
3. Installationsanweisungen finden Sie in der Datei `CDNAV`*nn*`.txt`. Dabei ist *nn* die zweistellige Landeskennung. Die Datei befindet sich im Stammverzeichnis der CD-ROM.

Zugreifen auf Informationen mit "Information - Unterstützung"

Information - Unterstützung ermöglicht Ihnen den schnellen Zugriff auf DB2-Produktinformationen. **Information - Unterstützung** ist auf allen Plattformen mit DB2-Verwaltungs-Tools verfügbar.

Sie können 'Information - Unterstützung' öffnen, indem Sie das entsprechende Symbol doppelt anklicken. Abhängig vom verwendeten System befindet sich das Symbol im Hauptproduktordner im Ordner 'Information' bzw. unter Windows im Menü **Start**.

Sie können auf 'Information - Unterstützung' auch zugreifen, indem Sie die Funktionsleiste und das Menü **Hilfe** auf der DB2-Windows-Plattform verwenden.

Unter 'Information - Unterstützung' finden Sie sechs verschiedene Arten von Informationen. Klicken Sie die entsprechende Indexzunge an, um die für diese Informationsart verfügbaren Themen aufzurufen.

Funktionen Die Hauptfunktionen, die Sie mit DB2 ausführen können.

Referenz DB2-Referenzinformationen, wie beispielsweise Schlüsselwörter, Befehle und APIs.

Handbücher DB2-Handbücher.

Fehlerbehebung
Kategorien von Fehlernachrichten sowie die entsprechenden Benutzeraktionen.

Beispielprogramme
Beispielprogramme, die in DB2 Application Development Client enthalten sind. Wenn Sie DB2 Application Development Client nicht installiert haben, wird diese Indexzunge nicht angezeigt.

Web DB2-Informationen im World Wide Web. Sie müssen über Ihr System eine Verbindung zum Web herstellen können, um auf diese Informationen zugreifen zu können.

Wenn Sie einen Eintrag aus einer der Listen auswählen, startet **Information - Unterstützung** eine Funktion zum Anzeigen der Informationen. Bei der Anzeigefunktion kann es sich abhängig von der ausgewählten Informationsart um die Hilfeanzeige des Systems, einen Editor oder einen Web-Browser handeln.

In 'Information - Unterstützung' steht eine Suchfunktion zur Verfügung, mit der Sie nach einem bestimmten Thema suchen können, ohne in den Listen blättern zu müssen.

Rufen Sie über die Hypertextverbindung in 'Information - Unterstützung' das Suchformular **In DB2-Online-Informationen suchen** auf.

Der HTML-Such-Server wird normalerweise automatisch gestartet. Wenn eine Suche in HTML-Informationen fehlschlägt, müssen Sie möglicherweise mit einer der nachfolgend aufgeführten Methoden den Such-Server starten:

Unter Windows
Klicken Sie **Start** an und wählen Sie **Programme —> IBM DB2 —> Informationen —> HTML-Such-Server starten** aus.

Unter OS/2
Klicken Sie den Ordner **DB2 für OS/2** und anschließend das Symbol für **HTML-Such-Server starten** doppelt an.

Falls andere Probleme bei der Suche in HTML-Informationen auftreten, finden Sie möglicherweise entsprechende Hinweise in den Release-Informationen.

Anmerkung: Die Suchfunktion steht in Linux-, PTX- und Silicon Graphics IRIX-Umgebungen nicht zur Verfügung.

Verwenden der DB2-Assistenten

Assistenten unterstützen Sie bei der Ausführung bestimmter Verwaltungsaufgaben, indem sie Sie Schritt für Schritt durch jede Aufgabe führen. Assistenten stehen über die Steuerzentrale und 'Client-Konfiguration - Unterstützung' zur Verfügung. In der folgenden Tabelle sind die einzelnen Assistenten und deren Verwendungszweck aufgeführt.

Anmerkung: In Umgebungen mit partitionierten Datenbanken sind die Assistenten **Datenbank erstellen**, **Index erstellen**, **Aktualisierung auf mehreren Systemen konfigurieren** und **Leistungskonfiguration** verfügbar.

Assistent	Verwendung	Zugriff
Datenbank hinzufügen	Katalogisieren einer Datenbank auf einer Client-Workstation.	Klicken Sie in **Client-Konfiguration - Unterstützung** die Option **Hinzufügen** an.
Datenbank sichern	Festlegen, Erstellen und Terminieren eines Sicherungsplans.	Klicken Sie in der Steuerzentrale die zu sichernde Datenbank mit der rechten Maustaste an und wählen Sie **Sichern —> Datenbank mit Assistent** aus.
Aktualisierung auf mehreren Systemen konfigurieren	Konfigurieren einer Aktualisierung auf mehreren Systemen, einer verteilten Transaktion oder einer zweiphasigen Festschreibung.	Klicken Sie in der Steuerzentrale den Ordner **Datenbanken** mit der rechten Maustaste an und wählen Sie **Aktualisierung auf mehreren Systemen** aus.

Assistent	Verwendung	Zugriff
Datenbank erstellen	Erstellen einer Datenbank und Ausführen einiger grundlegender Konfigurationsfunktionen.	Klicken Sie in der Steuerzentrale den Ordner **Datenbanken** mit der rechten Maustaste an und wählen Sie **Erstellen —> Datenbank mit Assistent** aus.
Tabelle erstellen	Auswählen eines Basisdatentyps und Erstellen eines Primärschlüssels für die Tabelle.	Klicken Sie in der Steuerzentrale das Symbol **Tabellen** mit der rechten Maustaste an und wählen Sie **Erstellen —> Tabelle mit Assistent** aus.
Tabellenbereich erstellen	Erstellen eines neuen Tabellenbereichs.	Klicken Sie in der Steuerzentrale das Symbol **Tabellenbereiche** mit der rechten Maustaste an und wählen Sie **Erstellen —> Tabellenbereich mit Assistent** aus.
Index erstellen	Hinweise zum Erstellen und Löschen von Indizes für Ihre Abfragen.	Klicken Sie in der Steuerzentrale das Symbol **Index** mit der rechten Maustaste an und wählen Sie **Erstellen —> Index mit Assistent** aus.
Leistungskonfiguration	Optimieren der Leistung einer Datenbank durch Aktualisieren der Konfigurationsparameter, so dass sie den Anforderungen Ihres Unternehmens entsprechen.	Klicken Sie in der Steuerzentrale die Datenbank, die optimiert werden soll, mit der rechten Maustaste an und wählen Sie **Leistung mit Assistent konfigurieren** aus. Klicken Sie in einer Umgebung mit partitionierten Datenbanken in der Sicht für Datenbankpartitionen die erste Datenbankpartition, die optimiert werden soll, mit der rechten Maustaste an und wählen Sie **Leistung mit Assistent konfigurieren** aus.
Datenbank wiederherstellen	Wiederherstellen einer Datenbank nach einem Fehler. Dieser Assistent hilft Ihnen, zu entscheiden, welche Sicherungskopie Sie verwenden und welche Protokolle Sie erneut abarbeiten.	Klicken Sie in der Steuerzentrale die Datenbank, die wiederhergestellt werden soll, mit der rechten Maustaste an und wählen Sie **Wiederherstellen —> Datenbank mit Assistent** aus.

Einrichten eines Dokument-Servers

Die DB2-Informationen werden standardmäßig auf Ihrem lokalen System installiert. Das bedeutet, dass alle Benutzer, die Zugriff auf DB2-Informationen benötigen, dieselben Dateien installieren müssen. Führen Sie folgende Schritte aus, um die DB2-Informationen an einer einzigen Position zu speichern:

1. Kopieren Sie alle Dateien und Unterverzeichnisse aus dem Verzeichnis `\sqllib\doc\html` Ihres lokalen Systems auf einen Web-Server. Jedem Handbuch ist ein Unterverzeichnis zugeordnet, das alle erforderlichen HTML- und GIF-Dateien enthält, aus denen das Handbuch besteht. Stellen Sie sicher, dass die Verzeichnisstruktur erhalten bleibt.
2. Konfigurieren Sie den Web-Server so, dass er die Dateien an der neuen Speicherposition sucht. Informationen hierzu finden Sie im Anhang zu NetQuestion im Handbuch *Installation und Konfiguration Ergänzung*.
3. Wenn Sie die Java-Version von **Information - Unterstützung** verwenden, können Sie eine Basis-URL-Adresse für alle HTML-Dateien angeben. Sie sollten die URL-Adresse für das Bücherverzeichnis verwenden.
4. Wenn Sie die Buchdateien anzeigen können, ist es möglich, bei häufig aufgerufenen Themen Lesezeichen zu setzen. Es empfiehlt sich, folgende Seiten mit einem Lesezeichen zu versehen:
   - Bücherverzeichnis
   - Inhaltsverzeichnis häufig verwendeter Handbücher
   - Themen, auf die häufig verwiesen wird, wie beispielsweise zum Ändern von Tabellen
   - Suchformular

Informationen dazu, wie Sie die DB2 Universal Database-Online-Dokumentationsdateien auf einer zentralen Maschine zur Verfügung stellen können, finden Sie im Anhang zu NetQuestion im Handbuch *Installation und Konfiguration Ergänzung*.

Suchen nach Online-Informationen

Verwenden Sie eine der folgenden Methoden, um nach Informationen in den HTML-Dateien zu suchen:

- Klicken Sie im obersten Rahmen auf **Suchen**. Verwenden Sie das Suchformular, um nach einem bestimmten Thema zu suchen. Diese Funktion steht in Linux-, PIX- oder Silicon Graphics IRIX-Umgebungen nicht zur Verfügung.
- Klicken Sie im obersten Rahmen auf **Index**. Mit Hilfe des Indexes können Sie nach einem bestimmten Thema im Buch suchen.
- Rufen Sie das Inhaltsverzeichnis oder den Index der Hilfe oder des HTML-Buchs auf und verwenden Sie die Suchfunktion des Web-Browsers, um nach einem bestimmten Thema im Buch zu suchen.
- Mit Hilfe der Lesezeichenfunktion des Web-Browsers können Sie schnell zu einem bestimmten Thema zurückkehren.
- Mit Hilfe der Suchfunktion von **Information - Unterstützung** können Sie bestimmte Themen suchen. Weitere Informationen finden Sie in „Zugreifen auf Informationen mit ″Information - Unterstützung″“ auf Seite 718.

Anhang F. Bemerkungen

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. An Stelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Dienstleistungen in Verbindung mit Fremdprodukten und Fremddienstleistungen liegt beim Kunden, soweit nicht ausdrücklich solche Verbindungen erwähnt sind.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanfragen sind schriftlich an IBM Europe, Director of Licensing, 92066 Paris La Defense Cedex, France, zu richten. Anfragen an obige Adresse müssen auf englisch formuliert werden.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen bekanntgegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Garantie, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten der IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden, Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHT-LIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Beispielprogramme geschrieben werden. Die in diesem Handbuch aufgeführten Beispiele sollen lediglich der Veranschaulichung und zu keinem anderen Zweck dienen. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name Ihrer Firma) (Jahr). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. _Jahr/Jahre angeben_. Alle Rechte vorbehalten.

Marken

Folgende Namen sind in gewissen Ländern Marken der International Business Machines Corporation.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
IBM System AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extender
DB2 OLAP Server
DB2 Universal Database
Distributed Relational
Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RS/6000
IBM System /370
SP
SQL/DS
SQL/400
System/370
IBM System /390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

Folgende Namen sind in gewissen Ländern Marken oder eingetragene Marken anderer Unternehmen:

Microsoft, Windows und Windows NT sind Marken oder eingetragene Marken von Microsoft Corporation.

Java und alle auf Java basierenden Marken und Logos sowie Solaris sind in gewissen Ländern Marken von Sun Microsystems, Inc.

Tivoli und NetView sind in gewissen Ländern Marken von Tivoli Systems Inc.

UNIX ist eine eingetragene Marke und wird ausschließlich von der X/Open Company Limited lizenziert.

Andere Namen von Unternehmen, Produkten oder Dienstleistungen können Marken anderer Unternehmen sein.

Index

A

Abfragen
  optimieren 92
Ablaufintervall für Data Link-Zugriffs-Token, Konfigurationsparameter 508
Abrufen von Blöcken 90
ACTIVATE DATABASE 319
Address Windowing Extensions (AWE) 293
ADVISE_INDEX, Tabelle
  detaillierte Beschreibung 629
  Erstellung 641
ADVISE_WORKLOAD, Tabelle
  detaillierte Beschreibung 633
ADVISE_WORKLOAD, Tabellendefinition
  Erstellung 643
Advisor
  Index 274
Agent im Bereitschaftsmodus 321
agent_stack_sz, Konfigurationsparameter 432
  Auswirkungen auf den Speicher 289
Agenten
  Agent im Bereitschaftsmodus 320
  Anfangswert für die Anzahl Agenten im Pool (num_initagents), Datenbankmanager-Parameter 473
  Governor ändert Priorität von 336
  Inaktiver Agent 321
  Koordinationsagent 321
  max_coordagents, Datenbankmanager-Parameter 470
  Maximale Anzahl koordinierender Agenten 470
  Poolgröße, Steuern der 472
  Subagent 321
  Verbindungseinträge, Anzahl 538
  Zwischenspeicher für Anwendungssteuerung, maximaler 420
Agentenpool 321
Agentenprozess
  Agentenpriorität (agentpri), Parameter 466
  Maximale Anzahl gleichzeitig aktiver Agenten (maxcagents), Parameter 469
  Maximale Anzahl von Agenten (maxagents), Parameter 468
  Zwischenspeicher für Anwendungen (applheapsz), Parameter 426
  Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz), Parameter 436
agentpri, Konfigurationsparameter 466
Aktualisierende Wiederherstellung 31
Aktualisierung auf mehreren Systemen 47
  Konfigurationsparameter 499
Aktualisierung auf mehreren Systemen konfigurieren, Assistent 719
Aktualisierungscursor
  nicht festgeschriebener Lesevorgang (UR) 52
Als Suchargument verwendbare Vergleichselemente
  Übersicht 201
ALTER TABLESPACE
  Beispiel 111
Anfangswert für die Anzahl Agenten im Pool (num_initagents), Datenbankmanager-Parameter 473
Anforderungsblöcke für Kommunikation zwischen FCM-Dämon und Agent, Anzahl 539
Antwortzeit für Knotenverbindung (conn_elapse) Datenbankmanager Konfigurationsparameter 535
Anweisungsebene, Isolation 55
Anwendungsentwurf
  Aktivieren von Sperren 57
  Außerkraftsetzen von Sperren 73
  gegenseitige Sperren vermeiden 66
Anwendungsentwurf *(Forts.)*
  Sperren, Faktoren 69
  Sperreneskalation 64
  Sperrenkompatibilität, sicherstellen 61
  Sperrenumwandlung 63
  Überlegungen zu Sperren 75
Anwendungsprogramm 47
  Governor zwingt 336
  Maximale Anzahl koordinierender Agenten am Knoten 470
  Zwischenspeicher für Steuerung, Einstellung 420
Anzahl Data Link-Kopien, Konfigurationsparameter 509
Anzahl der Datenbanksicherungen, Konfigurationsparameter 495
Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb), Datenbankmanager-Parameter 539
Anzahl der FCM-Nachrichtenanker (fcm_num_anchors), Datenbankmanager-Parameter 536
Anzeigen
  Online-Informationen 717
app_ctl_heap_sz, Datenbank-Parameter 420
app_ctl_heap_sz, Datenbankkonfigurationsparameter
  Auswirkungen auf den Speicher 289
applheapsz, Konfigurationsparameter 426
  Auswirkungen auf den Speicher 289
Architektur
  Speicher 16
  Übersicht 11
aslheapsz, Konfigurationsparameter 436
  Auswirkungen auf den Speicher 290
Assistent
  Datenbank wiederherstellen 720
Assistenten
  Assistenten 719
  Datenbank erstellen 719
  Datenbank hinzufügen 719, 720
  Datenbank sichern 719

Assistenten *(Forts.)*
Index 720
Konfigurieren von Aktualisierungen auf mehreren Systemen 719
Leistungskonfiguration 720
Tabelle erstellen 720
Tabellenbereich erstellen 720
Tasks ausführen 719
audit_buf_sz, Konfigurationsparameter 446
Aufbewahrungszeitraum für Wiederherstellungsprotokoll (rec_his_retentn), Konfigurationsparameter 496
Auslöser
EXPLAIN-Tabellen 609
Äußere Tabelle, Verknüpfungsmethode 220
authentication, Konfigurationsparameter 561
Automatische Übersichtstabellen 231
autorestart, Datenbankkonfigurationsparameter 492
avg_appls, Konfigurationsparameter 463
Auswirkung auf die Abfrageoptimierung 105

B

backbufsz, Konfigurationsparameter 413
BACKUP DATABASE, Dienstprogramm
Standardgröße für Sicherungspuffer (backbufsz), Parameter 413
backup_pending, Konfigurationsparameter 511
Befehl db2gov 334
Befehle
ACTIVATE DATABASE 319
db2evmon 330
DEACTIVATE DATABASE 319
REORGCHK 313
Behälter 18
Hinweise für parallele E/A 308
Bei Indexsuchen als Suchargument verwendbare Vergleichselemente (Index SARGable)
Übersicht 201
Beispielprogramme
HTML 709
plattformübergreifend 709
Benutzerdefinierte Funktionen (UDFs)
Aktualisieren von Statistiken 162
Berechtigung
für Dienstprogramm REORG 315
Konfigurationsparameter 557
Bereichsbegrenzende Vergleichselemente
Übersicht 201
Binden
Ändern von Konfigurationsparametern 397
Isolationsstufe 54
Standardwert für die Option DEGREE 100
buffpage, Konfigurationsparameter 404
Auswirkung auf die Abfrageoptimierung 104
Auswirkungen auf den Speicher 288
Verwalten mehrerer Pufferpools 298

C

catalog_noauth, Konfigurationsparameter 563
catalogcache_sz, Konfigurationsparameter 409
chngpgs_thresh, Konfigurationsparameter 453
Verwalten des Pufferpools 295
Client-Unterstützung
APPC-Transaktionsprogrammname (tpname), Parameter 523
E/A-Blockgröße für Clients (rqrioblk), Parameter 440
TCP/IP-Servicename (svcename), Parameter 522
Clusterungsindizes 26
codepage, Konfigurationsparameter 507
Codepages
Richtlinien zur Auswahl 96
codeset, Konfigurationsparameter 506
collate_info, Konfigurationsparameter 507
collating_sequence, Server-Option 122
comm_bandwidth, Konfigurationsparameter 549
comm_rate, Server-Option 123
Compiler
Phasen bei zusammengeschlossenen Datenbanken 174
Pushdown-Analyse 174
Übersicht 172
Übersicht über Generierung von fernem SQL 175
Umschreiben einer Abfrage 176
Compound-Anweisungen
dynamische 96
Compound-SQL-Anweisung
Informationen zur Leistung 95
Übersicht 95
conn_elapse, Konfigurationsparameter 535
copyprotect, Konfigurationsparameter 508
country, Konfigurationsparameter 506
cpu_ratio, Server-Option 123
cpuspeed, Konfigurationsparameter 549
Auswirkung auf die Abfrageoptimierung 106
CREATE INDEX
ALLOW REVERSE SCANS 188
CREATE TABLESPACE 20
CURRENT DEGREE, Sonderregister 100
Cursor
Aktualisierung, nicht festgeschriebener Lesevorgang 52
Lesezugriff, nicht festgeschriebener Lesevorgang (UR) 52
Schließen mit Klausel WITH RELEASE 75
Cursorstabilität (CS)
Übersicht 51

D

DARI 98
Data Links-Token-Algorithmus, Konfigurationsparameter 510
Data Links-Token in Großbuchstaben, Konfigurationsparameter 510
Data Links-Zeit nach DROP, Konfigurationsparameter 509
Database Application Remote Interface (DARI) 98
Anfangsanzahl der abgeschirmten DARI-Prozesse im Pool (num_initdaris), Parameter 477
DARI-Prozess beibehalten (keepdari), Parameter 474

Database Application Remote Interface (DARI) *(Forts.)*
Initialisieren von DARI-Prozessen mit JVM (initdari_jvm), Parameter 477
Maximale Anzahl von DARI-Prozessen (maxdari), Parameter 475
database_consistent, Konfigurationsparameter 511
database_level, Konfigurationsparameter 505
datalinks, Konfigurationsparameter 510
Daten
Caching nach Starten der Datenbank 100
Verbindungseinträge zur Weitergabe durch Agenten, Anzahl 538
Verwaltung 23
Datenbank 47
Agenten 321
aktivieren 319
Automatischer Neustart aktiviert (autorestart), Parameter 492
Benutzerausgang für Protokollierung aktivieren (userexit), Parameter 491
Codepage für die Datenbank (codepage), Parameter 507
Codierter Zeichensatz für die Datenbank (codeset), Parameter 506
Daten-Caching nach Starten der Datenbank 100
Datenbank ist konsistent (database_consistent), Parameter 511
Datenbankgebiet (territory), Parameter 506
inaktivieren 319
Informationen zur Sortierfolge (collate_info), Parameter 507
Konfigurationsparameter 396
Konfigurationsparameter, Überblick 398
Landescode der Datenbank (country), Parameter 506
Maximale Anzahl gleichzeitig aktiver Datenbanken (numdb), Parameter 550
Maximale Anzahl offener Datenbankdateien pro Anwendung (maxfilop), Parameter 464
Datenbank *(Forts.)*
Parameterdatei SQLDBCON 396
Release-Stand der Datenbankkonfiguration (release), Parameter 505
Sicherung anstehend (backup_pending), Parameter 511
Speicher für eine Anwendung 285
Standardanzahl von SMS-Behältern (numsegs), Parameter 459
Startaufwand 319
Status des Benutzerausgangs für Protokollierung (user_exit_status), Parameter 512
Datenbank erstellen, Assistent 719
Datenbank hinzufügen, Assistent 719, 720
Datenbank sichern, Assistent 719
Datenbanken
aktivieren 100
Zwischenspeichern von Daten 100
Datenbankkonfiguration
app_ctl_heap_sz, Parameter 420
Datenbankmanager 47
Governor, Auswirkung auf Leistung 348
Knotenart des Systems (nodetype), Parameter 554
Konfigurationsparameter 389
Konfigurationsparameter, Überblick 391
Parameterdatei db2systm 389
Standarddatenbankpfad (dftdbpathk), Parameter 563
Verwendung von Speicher 284
Zeitlimit für DB2START 541
Zeitlimit für DB2STOP 541
Datenbankmanagerkonfiguration
conn_elapse, Parameter 535
fcm_num_anchors, Parameter 536
fcm_num_buffers, Parameter 536
fcm_num_connect, Parameter 538
fcm_num_rqb, Parameter 539
java_heap_sz, Parameter 447
max_connretries, Parameter 540
max_coordagents, Parameter 470
max_time_diff, Parameter 540
num_initagents, Parameter 473
num_poolagents, Parameter 472
Datenbankmanagerkonfiguration *(Forts.)*
start_stop_time, Parameter 541
Datenbankmonitor
verwenden 328
Datenbankpartitionen
hinzufügen, aktives System 352
hinzufügen, System gestoppt 354
hinzufügen, System ohne Datenbanken 352
Hinzufügen zu einem System 351
löschen mit DB2STOP CMD/API 357
Server mit DB2STOP CMD/API löschen 357
Überlegungen zum Löschen eines Servers 358
Datenbanksystemmonitor
fcm_num_rqb, Datenbankmanager-Parameter, Feinabstimmung 540
Konfigurationsparameter 547
Datenbankverwaltung, Konfigurationsparameter 504
Datenbankzugriff
Auswirkung der Optimierungsklasse 76
Übersicht 187, 188
Datenintegrität
Gemeinsamer Zugriff 47
Schützen mit Sperren 57
Datenquellen
CPU-Geschwindigkeit und Leistung 244
E/A-Geschwindigkeit und Leistung 244
Datenseite 23
DB2_ANTIJOIN 587
DB2_AVOID_PREFETCH 590
DB2_AWE 591
DB2-Bibliothek
Assistenten 719
Dokument-Server einrichten 721
Drucken von PDF-Handbüchern 712
gedruckte Handbücher bestellen 713
Handbücher 699
Information - Unterstützung 718
neueste Informationen 712
Online-Hilfefunktion 715
Online-Informationen anzeigen 717

DB2-Bibliothek *(Forts.)*
Online-Informationen suchen 722
Sprachenkennung für Bücher 711
Struktur 699
DB2_BINSORT 591
DB2_BLOCK_ON_LOG_DISK_FULL 572
DB2 Connect
Verringerung der Verbindungszeit 323
DB2_CORRELATED_PREDICATES 587
DB2_DARI_LOOKUP_ALL 598
DB2 Data Links Manager 508
DB2_DISABLE_FLUSH_LOG 573
DB2_DJ_COMM 601
DB2_ENABLE_BUFPD 592
DB2_ENABLE_LDAP 601
DB2_EXTENDED_OPTIMIZATION 592
DB2_FALLBACK 602
DB2_FORCE_FCM_BP 586
DB2_FORCE_NLS_CACHE 579
DB2_FORCE_TRUNCATION 602
DB2_GRP_LOOKUP 602
DB2_HASH_JOIN 587
DB2_INDEX_2BYTEVARLEN 602
DB2_LIC_STAT_SIZE 574
DB2_LIKE_VARCHAR 588
DB2_MMAP_READ 593
DB2_MMAP_WRITE 594
DB2_NEW_CORR_SQ_FF 589
DB2_NEWLOGPATH2 604
DB2_NO_PKG_LOCK 594
DB2_NUM_FAILOVER_NODES 586
DB2_OVERRIDE_BPF 596
db2_override_bpf, Registrierungsvariable 299
DB2_PARALLEL_IO 577
DB2_PINNED_BP 596
DB2_PRED_FACTORIZE 590
DB2_RR_TO_RS 597
DB2_SELECTIVITY 589
DB2_SORT_AFTER_TQ 597
DB2_STPROC_LOOKUP_FIRST 598
DB2_STRIPED_CONTAINERS 578
DB2_UPDATE_PART_KEY 587
DB2_VENDOR_INI 606
DB2_VI_DEVICE 583
DB2_VI_ENABLE 583
DB2_VI_VIPL 583
DB2_XBSA_LIBRARY 607
DB2ACCOUNT 571
DB2ADMINSERVER 600
DB2ATLD_PORTS 585
DB2ATLD_PWFILE 585
db2batch, Vergleichstest-Tool 376
DB2BIDI 572
DB2BPVARS 592
DB2BQTIME 585
DB2BQTRY 585
DB2CHECKCLIENTINTERVAL 578
DB2CHGPWD_EEE 586
DB2CHKPTR 592
DB2CLIENTADPT 584
DB2CLIENTCOMM 584
DB2CLIINIPATH 600
DB2CODEPAGE 572
DB2COMM 579
DB2CONNECT_IN_APP_PROCESS 576
DB2COUNTRY 572
DB2DBDFT 572
DB2DBMSADDR 573
DB2DEFPREP 601
DB2DIRPATHNAME 584
DB2DISCOVERYTIME 573
DB2DMNBCKCTLR 601
DB2DOMAINLIST 576
db2empfa 309
DB2ENVLIST 576
db2exfmt, Tool 267, 695
db2expln 645
db2govlg, Befehl 348
DB2INCLUDE 574
DB2INSTANCE 577
DB2INSTDEF 574
DB2INSTOWNER 574
DB2INSTPROF 577
DB2IQTIME 585
DB2LDAP_BASEDN 603
DB2LDAP_CLIENT_PROVIDER 603
DB2LDAP_SEARCH_SCOPE 604
DB2LDAPCACHE 603
DB2LDAPHOST 603
DB2LIBPATH 577
DB2LOADREC 604
DB2LOCK_TO_RB 604
db2look, Tool
Übersicht 164
DB2MAXFSCRSEARCH 26, 593
DB2MEMDISCLAIM 593
DB2MEMMAXFREE 593
DB2NBADAPTERS 579
DB2NBBRECVNCBS 580
DB2NBCHECKUPTIME 579
DB2NBDISCOVERRCVBUFS 574
DB2NBINTRLISTENS 580
DB2NBRECVBUFFSIZE 580
DB2NBRESOURCES 580
DB2NBSENDNCBS 580
DB2NBSESSIONS 581
DB2NBXTRANCBS 581
DB2NETREQ 581
DB2NODE 577
Exportiert beim Hinzufügen eines Servers 353
db2nodes.cfg, Datei
Entfernen von Datenbankpartitionen bei Umverteilung von Daten 365
Hinzufügen von Datenbankpartitionen bei Umverteilung von Daten 365
db2nodes.cfg, durch den Datenbankmanager aktualisieren 355
db2nodes.cfg, manuell aktualisieren 357
DB2NOEXITLIST 604
DB2NTMEMSIZE 594
DB2NTNOCACHE 592, 595
DB2NTPRICLASS 595
DB2NTWORKSET 595
DB2OPTIONS 574
DB2PATH 578
DB2PORTRANGE 587
DB2PRIORITIES 596
DB2REMOTEPREG 605
DB2RETRY 581
DB2RETRYTIME 582
DB2ROUTE 584
DB2ROUTINE_DEBUG 605
DB2RQTIME 585
DB2SERVICETPINSTANCE 582
DB2SLOGON 574
DB2SORCVBUF 605
DB2SORT 605
DB2SOSNDBUF 582
db2start/db2stop-Zeitlimit (start_stop_time), Datenbankmanager-Parameter 541
DB2SYSPLEX_SERVER 582
DB2SYSTEM 606
DB2TCPCONNMGRS 583
DB2TIMEOUT 575
DB2TRACEFLUSH 575
DB2TRACENAME 575
DB2TRACEON 575
DB2TRCSYSERR 575
DB2UPMPR 606
DB2YIELD 576
dbexpln, Programm
Daten des Compilers 175
dbheap, Konfigurationsparameter 408
Auswirkungen auf den Speicher 288

dbname, Server-Option 123
DEACTIVATE DATABASE 319
DECLARE CURSOR WITH HOLD, Anweisung 89
DEGREE, Bindeoption 100
Deklarierte temporäre Tabellen
  gemeinsamer Zugriff 57
  Sperren 73
Dekorrelierung einer Abfrage 181
Detektor für gegenseitiges Sperren 15
Dezimalrechnung
  3 Kommastellen bei Dezimaldivision (min_dec_div_3), Parameter 438
  Geänderte Seiten protokollieren (trackmod), Parameter 497
dft_account_str, Konfigurationsparameter 555
dft_client_adpt, Konfigurationsparameter 531
dft_client_comm, Konfigurationsparameter 530
dft_degree, Konfigurationsparameter 100, 104, 516
dft_extent_sz, Konfigurationsparameter 459
dft_loadrec_ses, Konfigurationsparameter 494
dft_mon_bufpool, Konfigurationsparameter 548
dft_mon_lock, Konfigurationsparameter 548
dft_mon_sort, Konfigurationsparameter 548
dft_mon_stmt, Konfigurationsparameter 548
dft_mon_table, Konfigurationsparameter 548
dft_mon_uow, Konfigurationsparameter 547
dft_monswitches, Konfigurationsparameter 547
dft_prefetch_sz, Konfigurationsparameter 458
dft_queryopt, Konfigurationsparameter 105, 517
dft_refresh_age, Konfigurationsparameter 518
dft_sqlmathwarn, Konfigurationsparameter 515
dftdbpath, Konfigurationsparameter 563
diaglevel, Konfigurationsparameter 544
diagpath, Konfigurationsparameter 545
Dienstprogramme
  Reorganisation, Prüfung 313
  zur Reorganisation 313
dir_cache, Konfigurationsparameter 444
dir_obj_name, Konfigurationsparameter 528
dir_path_name, Konfigurationsparameter 527
dir_type, Konfigurationsparameter 526
discover, Konfigurationsparameter 532
discover_comm, Konfigurationsparameter 533
discover_db, Konfigurationsparameter 531
discover_inst, Konfigurationsparameter 534
Discover-Server-Exemplar, Konfigurationsparameter 534
Discovery-Kommunikationsprotokoll, Konfigurationsparameter 533
Discovery-Modus, Konfigurationsparameter 532
Distributed Computing Environment (DCE)
  Konfigurationsparameter 526
dl_expint, Konfigurationsparameter 508
dl_num_copies, Konfigurationsparameter 509
dl_time_drop, Konfigurationsparameter 509
dl_token, Konfigurationsparameter 510
dl_upper, Konfigurationsparameter 510
dlchktime, Konfigurationsparameter 448
DLFM_BACKUP_DIR_NAME 598
DLFM_BACKUP_LOCAL_MP 599
DLFM_BACKUP_TARGET 599
DLFM_BACKUP_TARGET_LIBRARY 599
DLFM_ENABLE_STPROC 599
DLFM_FS_ENVIRONMENT 599
DLFM_GC_MODE 599
DLFM_INSTALL_PATH 600
DLFM_LOG_LEVEL 600
DLFM_PORT 600
DLFM_TSM_MGMTCLASS 600
DMS (Database Managed Space, von der Datenbank verwalteter Speicher) 19
DMS-Tabellenbereich
  Informationen zur Leistung 318
  Zwischenspeichern 318
Dokument-Server einrichten 721
dos_rqrioblk, Konfigurationsparameter 441
drda_heap_sz, Konfigurationsparameter 429
  Auswirkungen auf den Speicher 289
Drucken von PDF-Handbüchern 712
Durchführen von Vergleichstests
  db2batch, Tool 376
  Testmethoden 372
  Testprozess 382
  Übersicht 371
  Vorbereitung 373
dyn_query_mgmt, Konfigurationsparameter 504
Dynamische Compound-Anweisungen 96
Dynamische SQL-Abfragen
  Auswerten der Optimierungsklasse 84
  Einstellen der Optimierungsklasse 81
  EXPLAIN-Einrichtung 267, 269
  Verteilungsstatistik 143
dynexpln 645

E

EDU (Engine Dispatchable Unit, zuteilbare Einheit der Steuerkomponente) 13, 40
Ein-/Ausgabe
  Aktivieren paralleler E/A 306
  Konfigurationsparameter 453
  Vorablesezugriff, parallel 304
EMP (Extent Map Pages, Speicherbereichszuordnungsseiten) 21
Entfernen eines Knotens aus dem System
599 bei Umverteilung der Knotengruppe 365
Ereignismomentaufnahmen 329
Erste aktive Protokolldatei (loghead), Parameter 485
Erstellen einer Spiegelkopie für Speicherseiten (Shadow Paging) 32
Erweiterter Speicher 43, 330

Erweiterter Speicher-Cache 330
Erweiterter UNIX-Code (EUC)
Unterstützung von Codepages 98
estore_seg_sz 44
estore_seg_sz, Konfigurationsparameter 460
Auswirkungen auf den Speicher 289
Exclusive, Modus
Verwendung 74
Exemplare
Unterstützung für Parallelität 100
Zeitdifferenz zwischen Knoten, maximale 540
EXPLAIN 268
FOR SNAPSHOT, Klausel 269
Visual 252, 273
WITH SNAPSHOT, Klausel 269
EXPLAIN_ARGUMENT, Tabelle
detaillierte Beschreibung 610
Erstellung 634
EXPLAIN-Einrichtung
Abrufen von Daten 267
Analyse 256
Auswählen eines Tools 252
Daten des Compilers 174
Datenorganisation 260
dbexpln, Programm 175
Entscheidungsfindung 271
Erfassen von Informationen 254, 267
EXPLAIN-Exemplar 260
grafische Darstellung 257
Grundkonzepte 257
Informationen der Momentaufnahmen 264
Informationen über Anweisungen 262
Informationen über Exemplare 261
Objekte 258
Operatoren 259
Schlüsselwörter 263
Tabelleninformationen 265
Übersicht 251
verwenden 255
EXPLAIN-Exemplare 260
EXPLAIN_INSTANCE, Tabelle
detaillierte Beschreibung 614
Erstellung 635
EXPLAIN-Momentaufnahmen 269
EXPLAIN_OBJECT, Tabelle
detaillierte Beschreibung 617
EXPLAIN_OBJECT, Tabelle *(Forts.)*
Erstellung 636
EXPLAIN_OPERATOR, Tabelle
detaillierte Beschreibung 620
Erstellung 637
EXPLAIN_PREDICATE, Tabelle
detaillierte Beschreibung 622
Erstellung 638
EXPLAIN-Programm 645
ausführen 646
Befehlsoptionen 646, 651
Beispiele für die Ausgabe von db2expln und dynexpln 677
Beschreibung der Ausgabe 654
Datenströme 667
Insert-, Update- und Delete-Operationen 668
Parallelverarbeitung 670
Satzkennungen (RID), Vorbereitung 668
Spaltenberechnung 669
Syntax 646, 651
Tabellenzugriff 656
temporäre Tabellen 661
Verarbeitung von Anweisungen in zusammengeschlossenen Datenbanken 673
Verknüpfungen 665
verschiedene Angaben 674
EXPLAIN_STATEMENT, Tabelle
detaillierte Beschreibung 624
Erstellung 639
EXPLAIN_STREAM, Tabelle
detaillierte Beschreibung 627
Erstellung 640
EXPLAIN-Tabellen
Zugriff auf 252
EXPLAIN-Tool für Tabellenformat 695
EXTENTSIZE, Parameter
auswählen 302

F

Fast Communication Manager (FCM)
optimieren 291
Fast Communications Manager (FCM)
Anzahl der FCM-Anforderungsblöcke (fcm_num_rqb), Parameter 539
Anzahl der FCM-Nachrichtenanker (fcm_num_anchors) Datenbankmanager-Parameter 536
fcm_num_buffers, Datenbankmanager-Parameter 536
Fast Communications Manager (FCM) *(Forts.)*
FCM-Verbindungseinträge (fcm_num_connect), Parameter 538
Nachrichtenanker, Anzahl angeben 536
Nachrichtenpuffer, Anzahl angeben 536
FCM (Fast Communications Manager) 40
fcm_num_anchors, Konfigurationsparameter 536
fcm_num_buffers, Konfigurationsparameter 536
fcm_num_connect, Konfigurationsparameter 538
fcm_num_rqb, Datenbankmanager-Konfigurationsparameter 539
FCM-Puffer (fcm_num_buffers), Datenbankmanager-Konfigurationsparameter 536
FCM-Verbindungseinträge (fcm_num_connect), Datenbankmanager-Parameter 538
federated, Konfigurationsparameter 556
Fehlerbehandlung
Konfigurationsparameter 544
Fehlernachrichten
beim Hinzufügen von Knoten zu partitionierten Datenbanken 359
Fehlersuche
Protokolldatei für Datenumverteilung 369
Ferne Datenbankdienste
Knotenname (nname), Parameter 521
Fernes SQL, Generierung 244
Festlegen der äußeren und der inneren Tabelle
Mischverknüpfungen 210
Übersicht 208
Verknüpfung über Verschachtelungsschleife 209
Festschreiben
Anzahl der Gruppenfestschreibungen (mincommit) 486
FETCH FIRST, Klausel 89
fileserver, Konfigurationsparameter 524
fold_id, Server-Option 124
fold_pw, Server-Option 124
FOR FETCH ONLY, Klausel 86, 93

FOR READ ONLY, Klausel 86, 93
FOR UPDATE, Klausel 86, 93
Free Space Control Record (FSCR, Steuersatz für freien Speicherbereich) 24
Frühere Blattzeiger 190
FSCR 24

G

Geclusterte Indizes
  Clusterverhältnis, Statistik 197
Gegenseitige Sperren 15
  erkennen 66
  Konfigurationsparameter 448
  Prüfen auf 448
  Übersicht 66
Gemeinsamer Zugriff
  deklarierte temporäre Tabellen 57
  Steuern mit Sperren 57
  Übersicht 47
Gemeinsamer Zugriff und Granularität
  Auswirkungen von Sperren 60
Gespeicherte Prozeduren
  Aufrufen ferner Prozeduren 98
  Auswirkungen auf die Leistung 98
  Konfigurationsparameter 474
Globale Optimierung
  analysieren 247
  EXPLAIN-Informationen zum Aufwand 247
  Kurznamenmerkmale 246
  Servermerkmale 244
Governor
  Beispielkonfigurationsdatei 344
  Dämon 336
  Datenbankmanager, Leistung 348
  db2gov 334
  db2govlg 348
  erhält Statistik 336
  Fehlerbehandlung 336
  Konfigurationsdatei 337
  Protokolldatei 346
  Protokolldatei abfragen 348
  Regeln 337
  starten 334
  stoppen 334
  Verwendungszweck 333
Größe des Agentenpools (num_poolagents), Datenbankmanager-Parameter 472
Große Objekte (LOBs)
  DMS-Speicher 319

H

Handbücher 699, 713
Hash-Verknüpfung
  Übersicht 207
Häufigkeitswertstatistik
  Aktualisierungsregeln 159
  Gleichheitsvergleichselement 147
  Übersicht 141
  zu sammelnde Anzahl 145
Hauptspeicher
  Einstellen von Parameterwerten 290
  erweitern 330
  Hinweise für den Systemadministrator (SYSADM) 283
  Konfigurationsparameter 285
  Verwendung durch den Datenbankmanager 284
  vollständig zugeordnet 290
  zur Verarbeitung einer Datenbank 285
Hinzufügen eines Knotens zum System
  bei Umverteilung der Knotengruppe 365
  Einschränkungen bei Datenbankoperationen 349
HTML
  Beispielprogramme 709

I

ID
  Satz (RID) 24
IN (Intent None), Modus 58
Inaktiver DRDA-Agent 323
INCLUDE, Klausel 28
Index
  Zeitpunkt für Indexneuerstellung (indexrec), Parameter 493
Index Advisor 113, 274
Index erstellen, Assistent 720
Indexclusterbildung
  Clusterfaktor, Statistik 132
  Clusterverhältnis, Statistik 132
indexrec, Konfigurationsparameter 493
Indexschlüssel
  größere 113
indexsort, Konfigurationsparameter 457
Indexsuchen 24
Indexsuchen *(Forts.)*
  frühere Blattzeiger 190
  geclusterter Index 197
  Sortieren von Daten 194
  Suchvorgänge 189
  Übersicht 188
  Vergleichselemente 191
  Vergleichselementterminologie 200
  verwenden 190
  WHERE-Klausel 191
  zur Begrenzung von Bereichen 190
Indexzeiger 28
Indizes
  Advisor 113
  Clusterbildung 26, 118
  Definition der AND-Verknüpfung von Indizes (Index ANDing) 196
  Definition der OR-Verknüpfung von Indizes 196
  erstellen 117
  Festlegen der äußeren und der inneren Tabelle 209
  größere Schlüssel 113
  Kurznamen, Informationen zur Leistung 246
  mehrere 196
  Nachteile 112
  reiner Indexzugriff 194, 659
  reorganisieren 116, 313, 316
  Richtlinien 114
  Sperrmodus 71
  Strukturen 189
  Suchfunktion, Auswirkungen auf Sperren 69
  Suchoperationen 189
  Tabellen mit und ohne Index 111
  verwalten 28
  Verwaltung 111
  Verwaltung durch den Administrator 117
  Vorablesezugriff (Prefetch) 301
Information - Unterstützung 718
initdari_jvm, Konfigurationsparameter 477
Innere Tabelle, Verknüpfungsmethode 222, 223
Innere und äußere Tabelle, Verknüpfungsmethode 221
Installation
  Netscape-Browser 717

Integritätsbedingungen
EXPLAIN-Tabellen 609
intra_parallel, Konfigurationsparameter 100, 543
io_ratio, Server-Option 125
ipx_socket, Konfigurationsparameter 525
IS (Intent Share), Modus 58
Isolationsstufen 29
angeben 54
Anweisungsebene 55
auswählen 53
Beschreibung 48
Cursorstabilität (CS) 51
Lesestabilität (RS) 50
nicht festgeschriebener Lesevorgang (UR) 52
Wiederholtes Lesen (RR) 49
IX (Intent Exclusive), Modus 59

J

java_heap_sz, Datenbankmanager-Konfigurationsparameter 447
jdk11_path, Datenbankmanager-Konfigurationsparameter 556

K

Kapazitätsverwaltung, Konfigurationsparameter 404
Kartesische Produkte 211
Sternschemas 211
Kataloge
reorganisieren 313
Katalogknoten 47
Verbindung zur Datenumverteilung 366
Katalogsichten
aktualisierbar 155
COLDIST (Spaltenverteilung) 138
COLUMNS (Spalten) 136
Funktionen 162
INDEXES (Indizes) 137
SYSSTAT.COLDIST 138
SYSSTAT.COLUMNS 136
SYSSTAT.FUNCTIONS (UDFs) 162
SYSSTAT.INDEXES 137
SYSSTAT.TABLES 136
TABLES (Tabellen) 136
keepdari 37
keepdari, Konfigurationsparameter 474
Knoten 47
Knoten *(Forts.)*
andere Operationen während der Umverteilung 370
Antwortzeit für Knotenverbindung 535
Bestimmen, wo die Ausführung von RUNSTATS erfolgt 132
Datenumverteilung, Verfahren 366
Koordinierende Agenten, maximale Anzahl 470
Maximale Anzahl Wiederholungen für Knotenverbindungen 540
maximale Zeitdifferenz zwischen 540
Nachrichtenpuffer, Anzahl angeben 536
Umverteilen von Daten auf Datenbankpartitionen 363
Knotengruppen
andere Operationen während der Umverteilung 370
Neuverteilen von Daten 363
Knotenkonfigurationsdateien
Datenbankmanager, Aktualisierung 355
Kollokation (Zusammenfassen von Tabellen)
Datenumverteilung, Beibehaltung 363
replizierte Übersichtstabellen 213
Kommunikation
FCM-Dämon an Agent, Anforderungsblöcke 539
Knoten, Antwortzeit für Knotenverbindung 535
Knoten, Nachrichtenpuffer 536
Wiederholungen für Knotenverbindungen, Anzahl 540
Konfiguration 388
Ändern der Datenbankmanagerparameter 389
Ändern der Datenbankparameter 396
Ändern der Größe 349
Datenbankmanagerparameter 389
Datenbankparameter 396
Hinzufügen von Servern, aktives System 352
Hinzufügen von Servern, System gestoppt 354
Konfiguration *(Forts.)*
Informationen zu Parametern 403
Optimieren von Parametern 388
Parameter 387
Parameterüberblick, Datenbank 398
Parameterüberblick, Datenbankmanager 391
Konfigurationsdateien
Governor 337
Governor, Beispiel 344
Konfigurationsparameter
agent_stack_sz 289
Agentenkommunikationsspeicher 436
Anwendungen und Agenten 461
Anwendungskommunikationsspeicher 436
applheapsz 289
aslheapsz 290
Auswirkung auf das Optimierungsprogramm 103
buffpage 288, 298
chngpgs_thresh 295
Compilereinstellungen 513
Database Application Remote Interface (DARI) 474
Datenbankattribute 505
Datenbankstatus 511
Datenbanksystemmonitor 547
Datenbankverwaltung 504
DB2 Data Links Manager 508
DB2 Discovery 531
dbheap 288
dft_degree 100
Diagnoseinformationen 544
drda_heap_sz 289
Ein-/Ausgabe und Speicher 453
estore_seg_sz 44, 289
eutil_heap_sz 288
Exemplarspeicher des Datenbankmanagers 443
Exemplarsystemverwaltung 557
Exemplarverwaltung 544
Gemeinsam benutzter Speicher der Datenbank 404
Gemeinsamer Anwendungsspeicher 420
gespeicherte Prozedur 474
intra_parallel 100
Kapazitätsverwaltung 404
keepdari 37

Konfigurationsparameter *(Forts.)*
Konfiguration der Übertragungsprotokolle 521
locklist 288
max_querydegree 100
maxagents 42
maxagents, Konfigurationsparameter 321
maxappls 41
multipage_alloc 309
num_estore_segs 44, 289
num_iocleaners 295
num_poolagents, Konfigurationsparameter 321
numdb 41
Parallelbetrieb 535
partitionierte Datenbank 535
pckcachesz 289
Privater Agentenspeicher 422
Protokollaktivität 486
Protokolldateien 478
protokollieren 478
Query Enabler 504
query_heap_sz 289
rqrioblk 290
sheapthres 311
softmax 296
sortheap 289, 311
Sperren 448
stat_heap_sz 289
stmtheap 289
Systemverwaltung 548
Tivoli Storage Manager 492
Übertragung 521
udf_mem_sz 289
verteilte Arbeitseinheit 499
verteilte Services 526
Wiederherstellung 478, 492
Konzentrator 322
Koordinationsagent 13
Maximale Anzahl am Knoten 470
Koordinatordatenbankpartition
Überlegungen zum Löschen 358
Korrelierte Unterabfragen 181
Kurznamen
Erstellen von Indizes 246
globale Optimierung, relevante Merkmale 246
Pushdown-Analyse 235, 239
Sammeln von Statistikdaten 130
Sichtstatistikdaten 130
Tipps zur Leistung von Abfragen 93

L

Langfelder
DMS-Speicher 319
Leistung
Datenbank-Caching 100
Datenquellen, aktualisieren 243
Datenverteilung, Bestimmen mit Hilfe von SQL 364
db2batch, Vergleichstest-Tool 376
Elemente 3
fernes SQL, Generierung 244
Generierung von fernem SQL für Datenquellen 243
globale Optimierung 244
Governor, Auswirkung auf Datenbankmanager 348
Grenzen der Optimierung 7
Katalogstatistiken 247
Konfigurationsparameter 388
Kurznamen, Überlegungen zu Indizes 246
Neuverteilen von Daten 363
num_ioservers, Konfigurationsparameter 306
Optimierung mit Hilfe von EXPLAIN 271
Optimierungsklasse anpassen 76
Plattenspeicher 5
Pushdown-Analyse (System zusammengeschlossener Datenbanken) 234
Richtlinien 4
RUNSTATS, Dienstprogramm 134
schnelle Bestimmung 8
Server-Merkmale 244
Sperren, Auswirkungen 60
Statistiken 129
Systeme zusammengeschlossener Datenbanken 234
Tabellenkollokation, Datenumverteilung 363
Überlegungen zu Anwendungen 47
Überlegungen zum Betrieb 283
Überlegungen zur Programmierung 47
Überlegungen zur Umgebung 103
Umschreiben der Abfrage durch Compiler 176
Verfahren 6
Verwenden der EXPLAIN-Einrichtung 255
Leistung *(Forts.)*
von der Datenbank verwalteter Speicher (DMS) 318
Zeilenblockung, Richtlinien 91
Zusammenfassung des Handbuchs 8
Leistungskonfiguration, Assistent 720
Lesesperren
Anweisung CLOSE CURSOR 75
Lesestabilität (RS)
Übersicht 50
LOCK TABLE, Anweisung
Minimieren von Sperreneskalationen 65
Verwendung, um Sperrmodi außer Kraft zu setzen 73
locklist, Konfigurationsparameter 415
Auswirkung auf die Abfrageoptimierung 105
Auswirkungen auf den Speicher 288
LOCKSIZE, Klausel 29
locktimeout, Konfigurationsparameter 451
log_retain_status, Konfigurationsparameter 512
logbufsz, Konfigurationsparameter 410
logfilsiz, Konfigurationsparameter 478
loghead, Konfigurationsparameter 485
Logische Knoten
mehrere 40
Logische Partitionen
mehrere 40
logpath, Konfigurationsparameter 485
logprimary, Konfigurationsparameter 480
logretain, Konfigurationsparameter 490
logsecond, Konfigurationsparameter 482

M

max_connretries, Datenbankmanager-Konfigurationsparameter 540
max_coordagents, Datenbankmanager-Konfigurationsparameter 470
max_logicagents, Konfigurationsparameter 471

max_querydegree, Konfigurationsparameter 100, 541
max_time_diff, Datenbankmanager-Konfigurationsparameter 540
maxagents 42
maxagents, Konfigurationsparameter 321, 468
  Auswirkung auf den Hauptspeicher 285
maxappls 41
maxappls, Konfigurationsparameter 461
  Auswirkung auf den Hauptspeicher 285
maxcagents, Konfigurationsparameter 469
maxdari, Konfigurationsparameter 475
maxfilop, Konfigurationsparameter 464
Maximale Anzahl koordinierender Agenten (max_coordagents), Datenbankmanager-Parameter 470
Maximale Anzahl Wiederholungen für Knotenverbindungen (max_connretries) 540
Maximale Zeitdifferenz zwischen Knoten (max_time_diff) Datenbankmanager, Parameter 540
Maximaler Grad der Parallelität bei Abfragen, Konfigurationsparameter 106, 541
Maximaler Zwischenspeicher für Anwendungssteuerung (app_ctl_heap_sz), Datenbank-Parameter 420
Maximaler Zwischenspeicher für Java-Interpreter (java_heap_sz), Datenbankmanager-Parameter 447
maxlocks, Konfigurationsparameter 450
  Auswirkung auf die Abfrageoptimierung 105
maxtotfilop, Konfigurationsparameter 465
min_dec_div_3, Konfigurationsparameter 438
min_priv_mem, Konfigurationsparameter 433
mincommit, Konfigurationsparameter 486
MINPCTUSED, Klausel 28
MINPCTUSED, Option 116, 316
Mischverknüpfungen
  Festlegen der äußeren und der inneren Tabelle 210
  Übersicht 206
Momentaufnahmen
  Überwachung zu einem bestimmten Zeitpunkt 328
mon_heap_sz, Konfigurationsparameter 443
Monitorschalter
  aktualisieren 328
multipage_alloc, Konfigurationsparameter 513
  Auswirkung auf den Hauptspeicher 309

N

Nachrichtenanker 536
Netscape-Browser
  installieren 717
Neueste Informationen 712
Neuverteilen von Daten
  andere Operationen während der Umverteilung 370
  Datenbankpartitionen entfernen 365
  Datenbankpartitionen hinzufügen 365
  Datenbankpartitionierung, Prozessübersicht 366
  Datenverteilung, Bestimmen mit Hilfe von SQL 364
  Fehlerbehebung 369
  Operation erfolgreich 367
  Operation fehlgeschlagen 367
  Partitionierungszuordnung, als Ziel angeben 365
  Protokolldatei 369
  replizierte Übersichtstabellen, Einschränkung 364
  Tabelle, Prozessübersicht 367
  Tabellenkollokation 363
  Verbindung zur Katalogdatenbankpartition 366
  Verteilung angeben 364
  Verteilungsdatei 364
  Verwendungszweck 363
newlogpath, Konfigurationsparameter 483
Nicht festgeschriebener Lesevorgang (UR)
  Übersicht 52
nname, Konfigurationsparameter 521
node, Server-Option 125
nodetype, Konfigurationsparameter 554
notifylevel, Konfigurationsparameter 546
NS (Next Key Share), Modus 58
NT_SCATTER_DMSDEVICE 592
NT_SCATTER_DMSFILE 592
NT_SCATTER_SMS 592
num_db_backups, Konfigurationsparameter 495
num_estore_segs 44
num_estore_segs, Konfigurationsparameter 460
  Auswirkungen auf den Speicher 289
num_freqvalues, Konfigurationsparameter 518
num_initagents, Datenbankmanager-Konfigurationsparameter 473
num_initdaris, Konfigurationsparameter 477
num_iocleaners, Konfigurationsparameter 454
  Verwalten des Pufferpools 295
num_ioservers, Konfigurationsparameter 456
  Auswirkungen auf den Vorablesezugriff auf Daten 306
num_poolagents, Datenbankmanager-Konfigurationsparameter 472
num_poolagents, Konfigurationsparameter 321
  Einfluss auf parallele Systeme 326
num_quantiles, Konfigurationsparameter 519
numdb 41
numdb, Konfigurationsparameter 550
  Auswirkung auf den Hauptspeicher 285
numeric_string, Spaltenoption 240
numsegs, Konfigurationsparameter 459
Nur-Lese-Cursor
  nicht festgeschriebener Lesevorgang (UR) 52
NW (Next Key Weak Exclusive), Modus 59
NX (Next Key Exclusive), Modus 59

O

objectname, Konfigurationsparameter 525
Online-Hilfefunktion 715
Online-Indexreorganisation 316
Online-Informationen
  anzeigen 717
  suchen 722
Optimierung, globale 244, 246, 247
Optimierungsklasse
  Ebenen 77
  einstellen 81
  Richtlinien 82
Optimierungsprogramm
  Anpassen des Optimierungsgrads 76
  Auswählen der optimalen Verknüpfung 210
  Auswirkungen der Statistik 129
  Datenbankzugriff 187, 188
  Erstellen von Zugriffsplänen 174
  Sortieren 225
  Verteilungsstatistiken, Auswirkungen 146
  Verwendung von replizierte Übersichtstabellen 213
OPTIMIZE FOR, Klausel 87, 93

P

Pakete
  Isolationsstufen 48
parallel
  Konfigurationsparameter 535
Parallelität
  partitionsinterne 306
partitionierte Datenbank
  Konfigurationsparameter 535
Partitionierte Datenbank
  Datenumverteilung, Fehlerbehebung 369
  Datenumverteilung auf Datenbankpartitionen 366
  Datenumverteilung in Tabellen 367
  Datenverteilung 364
  Dekorrelierung einer Abfrage 181
  Partitionierungszuordnung, als Ziel bei der Datenumverteilung angeben 365
Partitionierte Datenbanken
  Fehler beim Hinzufügen von Knoten 359
Partitionierungszuordnung
  Neuverteilen von Daten 364
  Zielzuordnung, Angeben für die Datenumverteilung 365
Partitionsinterne Parallelität 306
Partitionsinterne Parallelität aktivieren, Konfigurationsparameter 543
password, Server-Option 125
pckcachesz, Konfigurationsparameter 418
  Auswirkungen auf den Speicher 289
PCTFREE, Klausel 27
PDF 712
Performance Monitor
  verwenden 328
plan_hints, Server-Option 126
Planhinweise, Beispiel 245
Pool für abgehende Verbindungen 323
Poolgröße für Agenten, Steuern der 472
priv_mem_thresh, Konfigurationsparameter 434
Protokolldateien
  Einträge für Datenumverteilung 369
  Governor-Protokolldatei 346
Protokolle
  aktualisieren 32
  Anzahl primärer Protokolldateien (logprimary), Parameter 480
  Anzahl sekundärer Protokolldateien (logsecond), Parameter 482
  Datenbankprotokollpfad ändern (newlogpath), Parameter 483
  erste aktive Protokolldatei (loghead), Parameter 485
  Konfigurationsparameter für die Protokollaktivität 486
  Konfigurationsparameter für Protokolldateien 478
  Pfad zu Protokolldateien (logpath), Parameter 485
  Protokolldateigröße (logfilsiz), Parameter 478
  Protokollpuffergröße (logbufsz), Parameter 410
  Protokollspeicherung für Wiederherstellung (logretain), Parameter 490
  Status der Protokollspeicherung für Wiederherstellung (log_retain_status), Parameter 512
Protokolle *(Forts.)*
  Vor dem bedingten Prüfpunkt zu schreibende Protokollsätze (softmax), Parameter 487
Protokollierung 15
  Protokollspeicherung 31
  Umlaufprotokollierung 31
Protokollpuffer 15, 31
Prozesse 34
  aktualisieren 32
  DB2 320
Prozessmodell 34
Prozessoren, einer Maschine hinzufügen 350
Pufferpool
  Binden von Datenbankanwendungen 407
  Größe definieren mit dem Konfigurationsparameter buffpage 404
  Informationen zum Speicher 407
  Informationen zur Leistung 407
Pufferpools 13
  Auswählen der Anzahl 300
  AWE 293
  Festlegen der äußeren und der inneren Tabelle 209
  mehrere 298
  Speicherbedarf 299
  Übersicht 292
  verwalten 296
  von der Datenbank verwalteter Speicher (DMS) 318
Pufferseiten
  Zuordnung mehrerer Pufferseiten 309
pushdown, Server-Option 126
Pushdown-Analyse
  Abfragemerkmale 241
  analysieren 241
  Kurznamenmerkmale 239
  Operatoren für EXPLAIN-Programm 241
  Server-Merkmale 236
  Übersicht 234

Q

Quantilwertstatistik
  Aktualisierungsregeln 159
  Bereichsstatistik 148
  sammeln 145
query_heap_sz, Konfigurationsparameter 428
  Auswirkungen auf den Speicher 289

R

rec_his_retentn, Konfigurationsparameter 496
Registrierungsvariablen 571
  DB2_ANTIJOIN 587
  DB2_AVOID_PREFETCH 590
  DB2_AWE 591
  DB2_BINSORT 591
  DB2_BLOCK_ON_LOG_DISK_FULL 572
  DB2_CORRELATED_PREDICATES 587
  DB2_DARI_LOOKUP_ALL 598
  DB2_DISABLE_FLUSH_LOG 573
  DB2_DJ_COMM 601
  DB2_ENABLE_BUFPD 592
  DB2_ENABLE_LDAP 601
  DB2_EXTENDED_OPTIMIZATION 592
  DB2_FALLBACK 602
  DB2_FORCE_FCM_BP 586
  DB2_FORCE_NLS_CACHE 579
  DB2_FORCE_TRUNCATION 602
  DB2_GRP_LOOKUP 602
  DB2_HASH_JOIN 587
  DB2_INDEX_2BYTEVARLEN 602
  DB2_LIC_STAT_SIZE 574
  DB2_LIKE_VARCHAR 588
  DB2_MMAP_READ 593
  DB2_MMAP_WRITE 594
  DB2_NEW_CORR_SQ_FF 589
  DB2_NEWLOGPATH2 604
  DB2_NO_PKG_LOCK 594
  DB2_NUM_FAILOVER_NODES 586
  DB2_OVERRIDE_BPF 596
  DB2_PARALLEL_IO 577
  DB2_PINNED_BP 596
  DB2_PRED_FACTORIZE 590
  DB2_RR_TO_RS 597
  DB2_SELECTIVITY 589
  DB2_SORT_AFTER_TQ 597
  DB2_STPROC_LOOKUP_FIRST 598
  DB2_STRIPED_CONTAINERS 578
  DB2_UPDATE_PART_KEY 587
  DB2_VENDOR_INI 606
  DB2_VI_DEVICE 583
  DB2_VI_ENABLE 583
  DB2_VI_VIPL 583
  DB2_XBSA_LIBRARY 607
  DB2ACCOUNT 571
  DB2ADMINSERVER 600
  DB2ATLD_PORTS 585
  DB2ATLD_PWFILE 585
  DB2BIDI 572
  DB2BPVARS 592
  DB2BQTIME 585
  DB2BQTRY 585
Registrierungsvariablen *(Forts.)*
  DB2CHECKCLIENTINTERVAL 578
  DB2CHGPWD_EEE 586
  DB2CHKPTR 592
  DB2CLIENTADPT 584
  DB2CLIENTCOMM 584
  DB2CLIINIPATH 600
  DB2CODEPAGE 572
  DB2COMM 579
  DB2CONNECT_IN_APP_PROCESS 576
  DB2COUNTRY 572
  DB2DBDFT 572
  DB2DBMSADDR 573
  DB2DEFPREP 601
  DB2DIRPATHNAME 584
  DB2DISCOVERYTIME 573
  DB2DMNBCKCTLR 601
  DB2DOMAINLIST 576
  DB2ENVLIST 576
  DB2INCLUDE 574
  DB2INSTANCE 577
  DB2INSTDEF 574
  DB2INSTOWNER 574
  DB2INSTPROF 577
  DB2IQTIME 585
  DB2LDAP_BASEDN 603
  DB2LDAP_CLIENT_PROVIDER 603
  DB2LDAP_SEARCH_SCOPE 604
  DB2LDAPCACHE 603
  DB2LDAPHOST 603
  DB2LIBPATH 577
  DB2LOADREC 604
  DB2LOCK_TO_RB 604
  DB2MAXFSCRSEARCH 593
  DB2MEMDISCLAIM 593
  DB2MEMMAXFREE 593
  DB2NBADAPTERS 579
  DB2NBBRECVNCBS 580
  DB2NBCHECKUPTIME 579
  DB2NBDISCOVERRCVBUFS 574
  DB2NBINTRLISTENS 580
  DB2NBRECVBUFFSIZE 580
  DB2NBRESOURCES 580
  DB2NBSENDNCBS 580
  DB2NBSESSIONS 581
  DB2NBXTRANCBS 581
  DB2NETREQ 581
  DB2NODE 577
  DB2NOEXITLIST 604
  DB2NTMEMSIZE 594
  DB2NTNOCACHE 595
  DB2NTPRICLASS 595
  DB2NTWORKSET 595
  DB2OPTIONS 574
  DB2PATH 578
Registrierungsvariablen *(Forts.)*
  DB2PORTRANCE 587
  DB2PRIORITIES 596
  DB2REMOTEPREG 605
  DB2RETRY 581
  DB2RETRYTIME 582
  DB2ROUTE 584
  DB2ROUTINE_DEBUG 605
  DB2RQTIME 585
  DB2SERVICETPINSTANCE 582
  DB2SLOGON 574
  DB2SORCVBUF 605
  DB2SORT 605
  DB2SOSNDBUF 582
  DB2SYSPLEX_SERVER 582
  DB2SYSTEM 606
  DB2TCPCONNMGRS 583
  DB2TIMEOUT 575
  DB2TRACEFLUSH 575
  DB2TRACENAME 575
  DB2TRACEON 575
  DB2TRCSYSERR 575
  DB2UPMPR 606
  DB2YIELD 576
  DLFM_BACKUP_DIR_NAME 598
  DLFM_BACKUP_LOCAL_MP 599
  DLFM_BACKUP_TARGET 599
  DLFM_BACKUP_TARGET_LIBRARY 599
  DLFM_ENABLE_STPROC 599
  DLFM_FS_ENVIRONMENT 599
  DLFM_GC_MODE 599
  DLFM_INSTALL_PATH 600
  DLFM_LOG_LEVEL 600
  DLFM_PORT 600
  DLFM_TSM_MGMTCLASS 600
release, Konfigurationsparameter 505
Release-Informationen 712
REORG, Dienstprogramm
  erforderliche Berechtigung und Zugriffsrechte 315
  Übersicht 313
Reorganisieren des Index
  online 316
REORGCHK 313
replizierte Übersichtstabellen
  umverteilte Knotengruppe, Einschränkung 364
restbufsz, Konfigurationsparameter 414

RESTORE DATABASE, Dienstprogramm
Standardgröße für Wiederherstellungspuffer (restbufsz), Parameter 414
restore_pending, Konfigurationsparameter 513
Restvergleichselemente
Übersicht 202
resync_interval, Konfigurationsparameter 500
REXX
Isolationsstufe angeben 54
RID (Record Identifier, Satz-ID) 24
ROLLFORWARD DATABASE, Dienstprogramm
Aktualisierende Wiederherstellung anstehend (rollfwd_pending), Parameter 512
rollfwd_pending, Konfigurationsparameter 512
route_obj_name, Konfigurationsparameter 529
rqrioblk, Konfigurationsparameter 440
Auswirkungen auf den Speicher 290
Rundgesendete äußere Tabelle, Verknüpfungsmethode 219
Rundgesendete innere Tabelle 222
RUNSTATS, Befehl/API
Knoten der Ausführung 132
RUNSTATS, Dienstprogramm
in einer partitionierten Datenbank 131
verwenden 131
WITH DISTRIBUTION, Klausel 140
zur Reorganisation 132

S

S (Share), Modus 58
Satzkennungen (RID) 668
Seiten
Daten 23
Seitenlöschfunktionen 14
Konfigurationsparameter 295
SELECT, Anweisung
Eliminieren von Klauseln DISTINCT 180
Richtlinien 92
Umschreiben der Abfrage durch Compiler 176
zwei oder mehr Tabellen 94
SELECT-Anweisungen verwenden 92
seqdetect, Konfigurationsparameter 457
Sequenzerkennung 303
Sequenzerkennung 283
Übersicht 303
Server
Optionen 244
Pushdown-Möglichkeiten 236
Server-Optionen
collating_sequence 122
comm_rate 123
connectstring 123
cpu_ratio 123
dbname 123
fold_id 124
fold_pw 124
io_ratio 125
node 125
password 125
plan_hints 126
pushdown 126
varchar_no_trailing_blanks 126
SET CURRENT EXPLAIN MODE 268
SET CURRENT EXPLAIN SNAPSHOT, Anweisung 270
SET CURRENT QUERY OPTIMIZATION, Anweisung 81
Share, Modus
Verwendung 74
sheapthres, Konfigurationsparameter 423
Sichten
Mischen durch das Optimierungsprogramm 177
Verschieben von Vergleichselementen durch das Optimierungsprogramm 181
SIX (Share with Intent Exclusive), Modus 59
Skalieren der Konfiguration 349
SMP (Space Map Pages, Speicherzuordnungsseiten) 20
SMS (System Managed Space, vom System verwalteter Speicher) 19
SMS-Tabellenbereich
Zwischenspeichern 318
softmax, Konfigurationsparameter 487
Verwalten des Pufferpools 296
Sonderregister
CURRENT DEGREE 100
sortheap, Konfigurationsparameter 422
Auswirkung auf die Abfrageoptimierung 105
Auswirkungen auf den Speicher 289
Sortieren
Konfigurationsparameter 309
Leistungsprobleme 310
mit Überlauf 310
nicht über Pipe geleitet 310
ohne Überlauf 310
Optimieren der Leistung 312
Parameter mit Auswirkung 310
Phasen 309
Schwellenwert für Sortierspeicher (sheapthres), Parameter 423
über Pipe geleitet 310
über Pipe geleitete und nicht über Pipe geleitete Sortierungen 225
Zwischenspeicher für Sortierlisten (sortheap), Parameter 422
Sortierfolgen
Systeme zusammengeschlossener Datenbanken 237
Sortierungen nach Überschreiten des Schwellenwerts vermeiden 311
Spaltenoptionen
numeric_string 240
varchar_no_trailing_blanks 241
Speicher
Agentenkommunikationsspeicher 436
Anwendungskommunikationsspeicher 436
Auswirkungen von Sperren 60
Datenbankmanagerexemplar 443
gemeinsamer Anwendungsspeicher 420
gemeinsamer Datenbankspeicher 404
Größe des Paket-Cache (pckcachesz), Parameter 418
privater Agentenspeicher 422
Schwellenwert für Sortierspeicher (sheapthres), Parameter 423
SQL-Anweisungszwischenspeicher (stmtheap), Parameter 425
Zwischenspeicher für Anwendungen (applheapsz), Parameter 426

Speicher *(Forts.)*
Zwischenspeicher für Anwendungsunterstützungsebene (aslheapsz), Parameter 436
Zwischenspeicher für Datenbank (dbheap), Parameter 408
Zwischenspeicher für Sortierlisten (sortheap), Parameter 422
Speicherbedarf
Zwischenspeicher für Anwendungssteuerung 420
Speicherbereich 19
Speicherbereichsverwaltung 25
Speichermodell 40
Sperren
aktivieren 57
Arten 58
Attribute 58
Attribute, Arten der Verarbeitung 69
Dauerattribut 58
Definition 29
deklarierte temporäre Tabellen 73
Erstellen mit Isolationsstufe Cursorstabilität (CS) 51
Erstellen mit Isolationsstufe Wiederholtes Lesen (RR) 49
Eskalation 30, 64
Eskalation und mögliche Maßnahmen 64
Exclusive (X), Modus 58
Faktoren mit Auswirkung auf 69
gegenseitige Sperre, Verwenden der Klausel FOR UPDATE OF 68
Gegenseitige Sperren 15
Intent Exclusive (IX), Modus 58
Intent None (IN), Modus 58
Intent Share (IS), Modus 58
Intervall für Prüfung gegenseitiger Sperren (dlchktime), Parameter 448
Kompatibilität, sicherstellen 61
Konfigurationsparameter 448
Lesestabilität (RS) 50
locktimeout, Konfigurationsparameter 66
Maximale Anzahl von Sperren pro Anwendung (maxlocks), Parameter 450
Maximaler Speicher für Sperrenliste (locklist), Parameter 415
Sperren *(Forts.)*
Modi für Indexsuche 71
Modi für Tabellensuche 70
Modus Exclusive, Verwendung 74
Modus Share, Verwendung 74
Modusattribut 58
Objektattribut 58
Share (S), Modus 58
Share with Intent Exclusive (SIX), Modus 58
Superxclusive (Z), Modus 58
Übersicht 57
Umwandlung 63
Update (U), Modus 58
Verbessern des gemeinsamen Zugriffs 64
Vermeiden globaler gegenseitiger Sperren 66
Verringern von Wartezeiten 66
spm_log_file_sz, Konfigurationsparameter 502
spm_log_path, Konfigurationsparameter 501
spm_max_resync, Konfigurationsparameter 503
spm_name, Konfigurationsparameter 502
Sprachenkennung
Handbücher 711
SQL-Advise-Einrichtung 274
SQL-Anweisungen
Durchführen von Vergleichstests 374
gültig während der Datenumverteilung 370
Optimieren von Abfragen 92
Richtlinien für SELECT-Anweisung 92
SELECT, Anweisung 92
SQL-Anweisungszwischenspeicher (stmtheap), Parameter 425
SQL-Funktionen
NODENUMBER, Datenverteilung bestimmen 364
PARTITION, Datenverteilung bestimmen 364
ss_logon, Konfigurationsparameter 565
Standardtabellenbereiche 18
START
Zeitlimit für Befehl, Einstellen 541
start_stop_time, Datenbankmanager-Konfigurationsparameter 541
stat_heap_sz, Konfigurationsparameter 427
Auswirkungen auf den Speicher 289
Statisches SQL
Auswerten der Optimierungsklasse 85
Einstellen der Optimierungsklasse 81
EXPLAIN-Einrichtung 267, 269
Verteilungsstatistik 143
Statistiken
aktualisieren 154
Aktualisierungsregeln 156, 157, 158, 159
benutzerdefinierte Funktionen (UDFs) 162
Daten modellieren 164
Dienstprogramm RUNSTATS in einer partitionierten Datenbank 131
Häufigkeit der Werte 140
Indexclusterbildung 198
Kopieren aus im Einsatz befindlichen Datenbanken 164
Quantile 140
RUNSTATS, Dienstprogramm 131
Sammeln für Kurznamen 130
Übersicht 129
Verteilung 140
Verteilung, Ermittlungsmethode 141
Zeitpunkt der Erfassung 134
Sternschemas 211
Steuerung des gemeinsamen Zugriffs
Maximale Anzahl aktiver Anwendungen (maxappls), Parameter 461
Maximale Anzahl gleichzeitig aktiver Datenbanken (numdb), Parameter 550
Steuerzentrale
Event Analyzer 328
Performance Monitor 328
Snapshot Monitor 328
stmtheap, Konfigurationsparameter 425
Auswirkung auf die Abfrageoptimierung 106
Auswirkungen auf den Speicher 289

STOP
Zeitlimit für Befehl, Einstellen 541
Suchen
Online-Informationen 719, 722
svcename, Konfigurationsparameter 522
sysadm_group, Konfigurationsparameter 557
sysctrl_group, Konfigurationsparameter 559
sysmaint_group, Konfigurationsparameter 560
Systeme zusammengeschlossener Datenbanken
Sortierfolgen 237
Systemkatalog
RUNSTATS, Dienstprogramm 136
Statistiken 129
Systemverwaltung
Hinweise zum Speicher 283
Konfigurationsparameter 548

T

Tabelle erstellen, Assistent 720
Tabellen
Bestimmen, wo die Ausführung von RUNSTATS erfolgt 132
Datenumverteilung, Verfahren 367
reorganisieren 313
REORGCHK, Befehl 313
Sperrarten 58
sperren 73
Sperrenkompatibilität, sicherstellen 61
Sperrmodus 70
Suchen, Auswirkungen auf Sperren 69
Umverteilung, Fehlerbehebung 369
verknüpfen 204
zwei oder mehr, Auswahlanweisung (SELECT) 94
Tabellenbereich erstellen, Assistent 720
Tabellenbereiche
Auswirkung auf die Abfrageoptimierung 107
Indizes 117
OVERHEAD, Einstellung 108
Sperrarten 58
Standardeinstellung 18
Tabellenbereiche *(Forts.)*
TRANSFERRATE, Einstellung 108
Vergleiche 22
Tabellensuchen 187
Definition 187
Verwendungskriterien 200
Tabellenwarteschlange 224
territory, Konfigurationsparameter 506
Threads 34
DB2 320
Tivoli Storage Manager (TSM)
Konfigurationsparameter 492
tm_database, Konfigurationsparameter 499
Tokens
Steuern der Anzahl 326
tp_mon_name, Konfigurationsparameter 552
tpname, Konfigurationsparameter 523
trackmod, Konfigurationsparameter 497
trust_allclnts, Konfigurationsparameter 565
trust_clntauth, Konfigurationsparameter 566
tsm_mgmtclass, Konfigurationsparameter 497
tsm_nodename, Konfigurationsparameter 498
tsm_owner, Konfigurationsparameter 498
tsm_password, Konfigurationsparameter 497

U

U (Update), Modus 59
Über Pipe geleitete und nicht über Pipe geleitete Sortierungen
Übersicht 225
Überlaufsätze 27
Übersichtstabellen
Beispiel 231
Übertragene äußere Tabelle, Verknüpfung 220
Übertragene innere Tabelle, Verknüpfung 223
Übertragene innere und äußere Tabellen, Verknüpfung 221
Übertragungsbandbreite, Konfigurationsparameter 106
Überwachung
Vorgehensweise 328
Überwachung zu einem bestimmten Zeitpunkt 328
udf_mem_sz, Konfigurationsparameter 430
Auswirkungen auf den Speicher 289
Umgebungsvariablen 571
DB2NODE, beim Hinzufügen eines Servers exportiert 353
Umgekehrte Suchoperationen 188, 192
Umschreiben einer Abfrage
Übersicht 176
Umsetzung
von Sperren, Regeln 63
Unterabfragen
korreliert 181
Unterelementstatistiken 167
Unterstützung der Aktivierung von Data Links, Konfigurationsparameter 510
Unterstützung für Systeme mit zusammengeschlossenen Datenbanken, Konfigurationsparameter 556
Unterstützung von Codepages
Zeichenumsetzung 96
user_exit_status, Konfigurationsparameter 512
userexit, Konfigurationsparameter 491
util_heap_sz, Konfigurationsparameter 412
Auswirkungen auf den Speicher 288

V

varchar_no_trailing_blanks, Server-Option 126
varchar_no_trailing_blanks, Spaltenoption 241
Verarbeitungsagenten
Agent im Bereitschaftsmodus 320
Inaktiver Agent 321
Koordinationsagent 321
Subagent 321
Verbindungen
Antwortzeit 535
Anzahl Wiederholungen 540
Verbindungseintrag 538
Verbundene Anwendungen 322
Vergleichselement
Hinzufügen durch das Optimierungsprogramm 183

Vergleichselement *(Forts.)*
Übersetzen durch das Optimierungsprogramm 182
Vergleichselemente
als Suchargument verwendbare Vergleichselemente 201
anwenden 181
bei Indexsuchen als Suchargument verwendbare Vergleichselemente 201
bereichsbegrenzende 201
Definition 191
einschließende Ungleichheit 193
Restvergleichselemente 202
strenge Ungleichheit 193
Terminologie 200
Übersicht 200
Verteilungsstatistik 147
verwenden 202
Vergleichstestprogramm
Beispielbericht 384
erstellen 375
SQL-Anweisungen 374
Zusammenfassung 385
Verknüpfung über Verschachtelungsschleife
Festlegen der äußeren und der inneren Tabelle 209
Übersicht 204
Verknüpfungen
Aufzählungsalgorithmus 211
Definition 203
Eliminierung von Redundanzen 178
Festlegen der äußeren und der inneren Tabelle 208
gemeinsam benutzte Spaltenberechnung 179
Hash-Verknüpfung 207
kartesische Produkte 211
Mischverknüpfung 206
Optimierungsprogramm, Suchstrategien 210
Tabellen 204
Übersicht 204
Umsetzen von Unterabfragen durch das Optimierungsprogramm 178
Verknüpfung über Verschachtelungsschleife 204
zusammengesetzte Tabellen 213
Verknüpfungsstrategien
äußere Tabelle rundsenden 219
in partitionierten Datenbanken 217
Verknüpfungsstrategien *(Forts.)*
rundgesendete innere Tabelle 222
übertragene äußere Tabelle 220
übertragene innere Tabelle 223
übertragene innere und äußere Tabelle 221
zusammengefasste Tabellen 218
Verringerung der Verbindungszeit 323
Verzeichnis, in dem Java Development Kit 1.1 installiert ist (jkd11_path), Datenbankmanager-Parameter 556
Verzeichnisstruktur 17
Visual Explain 252, 273
Vorablesen von Daten 283
Datenseite 301
E/A-Server 304
Indexseite 301
Optimieren mit Hilfe des Datenbanksystemmonitors 302
partitionsinterne Parallelität 304
PREFETCHSIZE, Klausel 302
Pufferpool 301
Sequenzerkennung 303
sequenziell 301
Vorablesezugriff über Listen 303
Vorablesezugriff (Prefetch)
Clusterung von Seitenleseoperationen 199
Vorablesezugriff auf Indexseiten 301
Vorablesezugriffe 14
Vorausschreibende Protokollierung (Write Ahead Logging, WAL) 31
Vorkompilieren
Isolationsstufe 54

W

W (Weak Exclusive), Modus 59
Weltzeit 541
Wiederherstellen, Assistent 720
Wiederherstellung
Konfigurationsparameter 492
Wiederholtes Lesen (RR)
Übersicht 49

X

X (Exclusive), Modus 59

Z

Z (Superxclusive), Modus 60
Zeichenumsetzung
Informationen zur Leistung 96
Zeilen
Blockung 90
Lesestabilität (RS) 50
schneller Abruf 85
Sperrarten 58
sperren 49, 50, 51
Sperrenkompatibilität, sicherstellen 61
Zeilenblockung
Abrufen von Blöcken 90
Arten 91
Übersicht 90
Zeitdifferenz zwischen Knoten, maximale 540
Zeitlimit, Starten und Stoppen des Datenbankmanagers 541
Zugriffspfad
Auswahl 85
Sperrattribute 69
Zugriffsplan
Aufwandsschätzung 262
db2expln 253
grafische Darstellung 257
Objekte 258
Operatoren 259
Verwenden der EXPLAIN-Einrichtung 255
Visual Explain 273
Zugriffspläne
vom Compiler erstellt 174
Zugriffsrechte
für Dienstprogramm REORG 315
Zugriffssteuerung
Gemeinsamer Zugriff 47
Sperren 57
Zuordnungsseiten
Speicher 20
Speicherbereich 21
Zusammengefasste Verknüpfungen 218
Zusammengeschlossene Datenbanken
Compilerphasen 234
fernes SQL, Generierung 244
Pushdown-Analyse 234
Zusammengesetzte Tabellen
zusammengesetzte äußere Tabelle 213
zusammengesetzte innere Tabelle 213

Zwischenspeicher für Anwendungs-
steuerung
Maximaler Zwischenspeicher für
Anwendungssteuerung
(app_ctl_heap_sz), Datenbank-
Parameter 420

Kontaktaufnahme mit IBM

Bei technischen Problemen lesen Sie bitte die entsprechenden Korrekturmaßnahmen im Handbuch *Troubleshooting Guide* und führen Sie diese aus, bevor Sie sich mit der IBM Kundenunterstützung in Verbindung setzen. Mit Hilfe dieses Handbuchs können Sie Informationen sammeln, die die DB2-Kundenunterstützung zur Fehlerbehebung verwenden kann.

Wenn Sie weitere Informationen benötigen oder eines der DB2 Universal Database-Produkte bestellen möchten, setzen Sie sich mit einem IBM Ansprechpartner in einer lokalen Geschäftsstelle oder einem IBM Software-Vertriebspartner in Verbindung.

Telefonische Unterstützung erhalten Sie unter der folgenden Nummer:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.

Produktinformationen

Telefonische Unterstützung erhalten Sie unter den folgenden Nummern:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0180/55 090 können Sie Handbücher telefonisch bestellen.

http://www.ibm.com/software/data/
Auf den DB2-World Wide Web-Seiten erhalten Sie aktuelle DB2-Informationen wie Neuigkeiten, Produktbeschreibungen, Schulungspläne und vieles mehr.

http://www.ibm.com/software/data/db2/library/
Mit **DB2 Product and Service Technical Library** können Sie auf häufig gestellte Fragen, Berichtigungen, Handbücher und aktuelle technische DB2-Informationen zugreifen.

Anmerkung: Diese Informationen stehen möglicherweise nur auf Englisch zur Verfügung.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
Auf der Web-Site für die Bestellung internationaler Veröffentlichungen (International Publications) finden Sie Informationen zum Bestellverfahren.

http://www.ibm.com/education/certify/
Das 'Professional Certification Program' auf der IBM Web-Site stellt Zertifizierungstestinformationen für eine Reihe von IBM Produkten, u. a. auch DB2, zur Verfügung.

ftp.software.ibm.com
Melden Sie sich als *anonymous* an. Im Verzeichnis `/ps/products/db2` finden Sie Demo-Versionen, Berichtigungen, Informationen und Tools zu DB2 und vielen zugehörigen Produkten.

comp.databases.ibm-db2, bit.listserv.db2-l
Über diese Internet-Newsgroups können DB2-Benutzer Ihre Erfahrungen mit den DB2-Produkten austauschen.

Für Compuserve: GO IBMDB2
Geben Sie diesen Befehl ein, um auf IBM DB2 Family Forums zuzugreifen. Alle DB2-Produkte werden über diese Foren unterstützt.

In Anhang A des Handbuchs *IBM Software Support Handbook* finden Sie Informationen dazu, wie Sie sich mit IBM in Verbindung setzen können. Rufen Sie die folgende Web-Seite auf, um auf dieses Dokument zuzugreifen: http://www.ibm.com/support/. Wählen Sie anschließend die Verbindung zum IBM Software Support Handbook am unteren Rand der Seite aus.

Anmerkung: In einigen Ländern sollten sich die IBM Vertragshändler an die innerhalb ihrer Händlerstruktur vorgesehene Unterstützung wenden, nicht an die IBM Unterstützungsfunktion.

SC12-2877-01