IBM DB2 Universal Database

SQL Erste Schritte

Version 7

SC12-2882-00

IBM DB2 Universal Database

SQL Erste Schritte

Version 7

SC12-2882-00

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Anhang C. Bemerkungen” auf Seite 129 gelesen werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM DB2 Universal Database, SQL Getting Started, Version 7,
IBM Form SC09-2973-00,

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 NLS Center
Kst. 2877
April 2000

Inhaltsverzeichnis

Willkommen

Dieses Handbuch soll Benutzern eine Einführung in SQL (Structured Query Language) und relationale Datenbanken geben. Es

- behandelt die Basiskonzepte von SQL, die im Produkt DB2 angewendet werden,
- erläutert die Durchführung von Tasks zur Datenbankbearbeitung,
- veranschaulicht Tasks mit Hilfe von einfachen Beispielen.

Als Systemadministrator sollten Sie folgendes durchführen, bevor Sie die Beispiele in diesem Handbuch anwenden:

- Installieren und konfigurieren Sie den Server gemäß der Beschreibungen im Handbuch *Einstieg* für Ihr Betriebssystem. Erstellen Sie mit der Option 'Erste Schritte' die Datenbank SAMPLE. Die Datenbank SAMPLE können Sie auch über eine Eingabeaufforderung erstellen. Ausführliche Informationen finden Sie im Handbuch *SQL Reference*. Anmerkung: Bitte stellen Sie keine eigenen Daten in die DB2-Datenbank SAMPLE.
- Erstellen Sie die Benutzer-ID für den DB2-Administrator anhand der Anweisungen im Handbuch *Einstieg*.

Sollten Sie kein Systemadministrator sein, vergewissern Sie sich, daß Sie über eine gültige Benutzer-ID und die entsprechende Berechtigung sowie die geeigneten Zugriffsrechte für die Datenbank SAMPLE verfügen.

Referenzliteratur zu diesem Handbuch

Die folgenden Veröffentlichungen enthalten weitere nützliche Informationen:

Einstieg	Dieses Handbuch enthält die zur Installation und Verwendung des Datenbankmanagers benötigten Informationen.
SQL Reference	In diesem Handbuch finden Sie Referenzinformationen zu SQL.
Systemverwaltung	Dieses Handbuch enthält Informationen, die Sie zum Entwerfen, Implementieren und Verwalten einer Datenbank benötigen, auf die entweder lokal oder in einer Client/Server-Umgebung zugegriffen werden soll.
Application Development Guide	In diesem Handbuch werden der Anwendungsentwicklungsprozeß sowie die Codierung, Kompilierung und Ausführung von Anwendungsprogrammen erörtert, die eingebettetes SQL für den Datenbankzugriff einsetzen oder unter Verwendung der SQL-Prozedursprache (bzw. anderer unterstützter Programmiersprachen) als gespeicherte DB2-Prozeduren ausgeführt werden.

Hervorhebungskonventionen

Dieses Handbuch verwendet die folgenden Konventionen:

Fettdruck	Hiermit werden in Beispielen die durch das System vordefinierten Befehle und Schlüsselwörter gekennzeichnet.
Kursivdruck	Diese Hervorhebung kennzeichnet • die Einführung eines neuen Begriffs oder • einen Verweis auf eine weitere Informationsquelle.
GROSS-BUCHSTABEN	Diese Hervorhebung kennzeichnet • durch das System vordefinierte Befehle und Schlüsselwörter oder • Beispiele für spezifische Datenwerte oder Spaltennamen.

Kapitel 1. Relationale Datenbanken und SQL

In einer *relationalen Datenbank* werden Daten in *Tabellen* gespeichert. Eine Tabelle ist eine Datensammlung, die in *Zeilen* und *Spalten* gegliedert ist. Abb. 1 auf Seite 4 zeigt eine Beispieldarstellung für eine Tabelle. In der Abbildung wurden die (vertikalen) Spalten und (horizontalen) Zeilen gekennzeichnet. Mit *SQL (Structured Query Language)* können Sie Daten abrufen und aktualisieren, indem Sie Spalten, Tabellen und die unterschiedlichen Beziehungen zwischen ihnen angeben.

SQL ist eine standardisierte Sprache für das Definieren und Bearbeiten von Daten in einer relationalen Datenbank. SQL-Anweisungen werden durch einen *Datenbankmanager* ausgeführt. Ein Datenbankmanager ist ein Computerprogramm, das die Daten verwaltet.

Eine *partitionierte* relationale Datenbank ist eine relationale Datenbank, in der die Daten über mehrere Partitionen hinweg verwaltet werden (diese Partitionen werden auch als *Knoten* bezeichnet). Zur Vereinfachung kann man sich eine Partition auch als einen physischen Computer vorstellen. Im vorliegenden Handbuch werden Datenbanken mit nur einer Partition behandelt.

Mit Hilfe des *interaktiven* SQL können Sie auf die Beispieldatenbank zugreifen und alle in diesem Handbuch beschriebenen Beispiele ausprobieren, indem Sie eine Schnittstelle wie beispielsweise den Befehlszeilenprozessor (CLP) oder die Befehlszentrale verwenden.

Kapitel 2. Organisieren von Daten

Im vorliegenden Kapitel werden wichtige konzeptionelle Beschreibungen von *Tabellen*, *Sichten* und *Schemata* vorgestellt. Es enthält eine allgemeine Übersicht über die Verbindung zwischen den unterschiedlichen Bausteinen einer relationalen Datenbank. Der letzte Abschnitt erörtert kurz einige der wichtigen und häufiger verwendeten Datentypen.

Tabellen

Tabellen sind logische Strukturen, die aus einer definierten Anzahl von *Spalten* und einer variablen Anzahl von *Zeilen* bestehen. Eine Spalte ist eine Gruppe von Werten desselben Datentyps. Eine Zeile ist eine Folge von Werten, die einen einzelnen Datensatz in der Tabelle bilden. Zeilen werden in einer Tabelle nicht zwangsläufig sortiert. Wenn die Ergebnismenge sortiert werden soll, müssen Sie dies in der SQL-Anweisung, die die Daten aus der Tabelle auswählt, explizit angeben. Den Schnittpunkt zwischen einer Spalte und einer Zeile bildet jeweils ein Datenelement, das als *Wert* bezeichnet wird. In Abb. 1 auf Seite 4 ist 'Sanders' ein Beispiel für einen Wert in der Tabelle.

Eine *Basistabelle* enthält Benutzerdaten und wird mit der Anweisung CREATE TABLE erstellt. Eine *Ergebnistabelle* ist eine Gruppe von Zeilen, die der Datenbankmanager aus einer oder mehreren Basistabellen auswählt oder generiert, um eine Abfrage zu beantworten.

Abb. 1 zeigt einen Ausschnitt aus einer Tabelle. Spalten und Zeilen wurden gekennzeichnet.

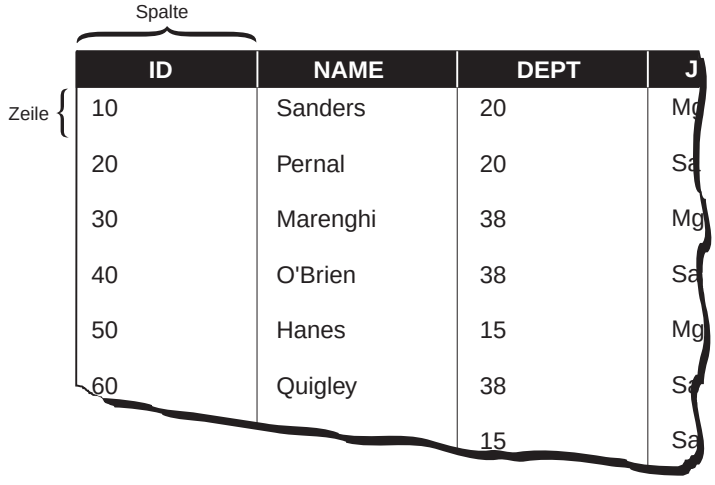


Abbildung 1. Darstellung einer Tabelle

Sichten

Eine *Sicht* bietet eine alternative Betrachtungsweise für die Daten in einer oder mehreren Tabellen. Sie ist ein dynamisches Fenster für Tabellen.

Mit Hilfe von Sichten können mehrere Benutzer unterschiedliche Darstellungen von identischen Daten anzeigen. Beispiel: Unterschiedliche Benutzer können auf eine Tabelle mit Mitarbeiterdaten zugreifen. Eine Führungskraft kann Daten über die Mitarbeiter der eigenen Abteilung, nicht jedoch über Mitarbeiter anderer Abteilungen anzeigen. Ein für die Personalbeschaffung zuständiger Mitarbeiter kann die Einstellungsdaten aller Angestellten, aber nicht deren Gehälter anzeigen, wohingegen ein Finanzsachbearbeiter die Gehälter, aber nicht die Einstellungsdaten anzeigen kann. Jeder dieser Benutzer arbeitet mit einer Sicht, die von der tatsächlichen Tabelle abgeleitet wurde. Jede Sicht wird wie eine Tabelle dargestellt und hat einen eigenen Namen.

Ein Vorteil der Verwendung von Sichten besteht darin, daß mit ihrer Hilfe der Zugriff auf sensible Daten gesteuert werden kann. Unterschiedliche Benutzer können somit auf unterschiedliche Datenspalten oder -zeilen zugreifen.

Schemata

Ein *Schema* ist eine Objektgruppe aus benannten Objekten (z. B. Tabellen und Sichten). Es ermöglicht die logische Klassifikation von Objekten in der Datenbank.

Ein Schema wird implizit erstellt, wenn Sie eine Tabelle, eine Sicht oder ein anderes benanntes Objekt erstellen. Sie können ein Schema natürlich auch mit der Anweisung CREATE SCHEMA explizit erstellen.

Bei der Erstellung eines benannten Objekts können Sie seinem Namen den Namen eines bestimmten Schemas zuordnen (= *qualifizieren*). Die Namen benannter Objekte bestehen aus zwei Teilen. Der erste Namensteil ist der Name des Schemas, dem das Objekt zugeordnet ist. Wenn Sie keinen Schemennamen angeben, wird das Objekt dem Standardschema zugeordnet. (Der Name des Standardschemas ist die *Berechtigungs-ID* des Benutzers, der die Anweisung ausführt.)

Beim interaktiven SQL, also der Methode, die für die Ausführung der Beispiele in diesem Handbuch verwendet wird, ist die Berechtigungs-ID die Benutzer-ID, die in der Anweisung CONNECT angegeben wird. Lautet beispielsweise der Name einer Tabelle STAFF und wird die Benutzer-ID USERXYZ angegeben, ist der qualifizierte Tabellenname USERXYZ.STAFF. Ausführliche Informationen zur Anweisung CONNECT finden Sie unter „Herstellen einer Verbindung zu einer Datenbank” auf Seite 22.

Einige Schemennamen sind reserviert. So befinden sich beispielsweise *integrierte Funktionen* im Schema SYSIBM, während die vorinstallierten *benutzerdefinierten Funktionen* zum Schema SYSFUN gehören. Ausführliche Informationen zur Anweisung CREATE SCHEMA finden Sie im Handbuch *SQL Reference*.

Datentypen

Datentypen definieren die für Konstanten, Spalten, Host-Variablen, Funktionen, Ausdrücke und Sonderregister zulässigen Werte. Dieser Abschnitt beschreibt die Datentypen, auf die in den Beispielen Bezug genommen wird. Eine vollständige Liste und umfassende Beschreibung weiterer Datentypen enthält das Handbuch *SQL Reference*.

Zeichenfolge

Eine *Zeichenfolge* ist eine Folge von Byte. Die Länge der Zeichenfolge entspricht der Anzahl der Byte in der Folge. Ist die Länge gleich Null, wird der Wert als *leere Zeichenfolge* bezeichnet.

Zeichenfolge mit fester Länge

CHAR(x) ist eine Zeichenfolge mit fester Länge. Der Wert für das Längenattribut x muß zwischen 1 und 254 (einschließlich) liegen.

Zeichenfolge mit variabler Länge

Zeichenfolgen mit variabler Länge haben einen der folgenden drei Datentypen: VARCHAR, LONG VARCHAR und CLOB.

Datentypen VARCHAR(x) sind Zeichenfolgen mit variabler Länge. Daher kann eine Zeichenfolge mit der Länge von 9 in VARCHAR(15) eingefügt werden. Sie behält jedoch die Zeichenfolgenlänge 9 bei.

Ausführliche Informationen zum Datentyp CLOB finden Sie unter „Große Objekte (LOB)" auf Seite 80.

Grafikzeichenfolge

Eine *Grafikzeichenfolge* ist eine Folge von Doppelbytezeichendaten.

Grafikzeichenfolge mit fester Länge

GRAPHIC(x) ist eine Zeichenfolge mit fester Länge. Der Wert für das Längenattribut x muß zwischen 1 und 127 (einschließlich) liegen.

Grafikzeichenfolge mit variabler Länge

Grafikzeichenfolgen mit variabler Länge haben einen der folgenden drei Datentypen: VARGRAPHIC, LONG VARGRAPHIC und DBCLOB. Ausführliche Informationen zum Datentyp DBCLOB finden Sie unter „Große Objekte (LOB)" auf Seite 80.

Binärzeichenfolge

Eine *Binärzeichenfolge* ist eine Folge von Byte. Sie enthält nicht konventionelle Daten wie beispielsweise Bilder. BLOB (Binary Large OBject - großes Binärobjekt) ist eine Binärzeichenfolge. Weitere Informationen finden Sie unter „Große Objekte (LOB)” auf Seite 80.

Zahlen

Alle Zahlen haben ein Vorzeichen und eine *Genauigkeit*. Die Genauigkeit gibt die Anzahl der Bit oder Stellen ausschließlich des Vorzeichens an.

SMALLINT
Ein *SMALLINT-Wert (ganze Zahl ohne erweiterte Genauigkeit)* ist eine aus zwei Byte bestehende ganze Zahl mit einer Genauigkeit von 5 Stellen.

INTEGER
Ein *INTEGER-Wert (ganze Zahl mit erweiterter Genauigkeit)* ist eine aus vier Byte bestehende ganze Zahl mit einer Genauigkeit von 10 Stellen.

BIGINT
Ein *BIGINT-Wert (große ganze Zahl)* ist eine aus acht Byte bestehende ganze Zahl mit einer Genauigkeit von 19 Stellen.

REAL Ein *REAL-Wert (Gleitkommazahl mit einfacher Genauigkeit)* ist eine näherungsweise 32-Bit-Entsprechung einer reellen Zahl.

DOUBLE
Ein *DOUBLE-Wert (Gleitkommazahl mit doppelter Genauigkeit)* ist eine näherungsweise 64-Bit-Entsprechung einer reellen Zahl. Der Datentyp DOUBLE wird auch als FLOAT bezeichnet.

DECIMAL(p,s)

Ein *DECIMAL-Wert* ist eine Dezimalzahl. Die Position des Dezimalzeichens wird durch die *Genauigkeit (p)* und die *Anzahl der Kommastellen (s)* der Zahl definiert. Die Genauigkeit gibt die Gesamtzahl der Ziffern an und muß kleiner als 32 sein. Die Anzahl der Kommastellen ist die Anzahl der Ziffern im Bruchteil und immer kleiner-gleich des Genauigkeitswerts. Ein DECIMAL-Wert hat standardmäßig die Genauigkeit 5 und die Anzahl der Kommastellen 0, wenn Genauigkeit und Anzahl der Kommastellen nicht angegeben werden.

Werte für Datum und Uhrzeit

Werte für Datum und Uhrzeit stellen Daten, Uhrzeiten und Zeitmarken dar. (Eine Zeitmarke ist eine 14 Ziffern umfassende Zeichenfolge, die einen gültigen Wert für Datum und Uhrzeit im Format *jjjjxxtthhmmss* angibt). Werte für Datum und Uhrzeit können in bestimmten Arithmetik- und Zeichenfolgeoperationen verwendet werden und sind mit bestimmten Zeichenfolgen kompatibel. Werte für Datum und Uhrzeit sind jedoch weder Zeichenfolgen noch Zahlen.[1]

Datum
Ein *Datum* ist ein dreiteiliger Wert (Jahr, Monat und Tag).

Zeit Eine *Zeit* ist ein dreiteiliger Wert (Stunde, Minute und Sekunde), der eine Uhrzeit in der 24-Stunden-Zeiteinteilung angibt.

Zeitmarke
Eine *Zeitmarke* ist ein siebenteiliger Wert (Jahr, Monat, Tag, Stunde, Minute, Sekunde und Mikrosekunde), der ein Datum und eine Uhrzeit angibt.

Nullwert

Der *Nullwert* ist ein Sonderwert, der sich von allen anderen Werten, die keine Nullwerte sind, unterscheidet. Er gibt an, daß für diese Spalte in der Zeile kein anderer Wert vorhanden ist. Der Nullwert ist bei allen Datentypen möglich.

Die folgende Tabelle hebt die Kenndaten für die Datentypen hervor, die in den Beispielen verwendet werden. Alle numerischen Datentypen sind in einem bestimmten Bereich definiert. Der Bereich der numerischen Datentypen ist in der Tabelle ebenfalls angegeben. Sie können diese Tabelle als Kurzübersicht für den richtigen Einsatz von Datentypen verwenden.

1. Im vorliegenden Handbuch werden Werte für Datum und Uhrzeit gemäß den ISO-Darstellungen angegeben.

Datentyp	Typ	Merkmal	Beispiel oder Bereich
CHAR(15)	Zeichenfolge mit fester Länge	Maximale Länge: 254	'Sonntag '
VARCHAR(15)	Zeichenfolge mit variabler Länge	Maximale Länge: 32672	'Sonntag'
SMALLINT	Zahl	Länge: 2 Byte, Genauigkeit: 5 Ziffern	Bereich: -32768 bis 32767
INTEGER	Zahl	Länge: 4 Byte, Genauigkeit: 10 Ziffern	Bereich: -2147483648 bis 2147483647
BIGINT	Zahl	Länge: 8 Byte, Genauigkeit: 19 Ziffern	Bereich: -9223372036854775808 bis 9223372036854775807
REAL	Zahl	Gleitkommazahl mit einfacher Genauigkeit, näherungsweise 32-Bit-Entsprechung	Bereich: -3.402E+38 bis -1.175E-37 oder 1.175E-37 bis -3.402E+38 oder Null
DOUBLE	Zahl	Gleitkommazahl mit doppelter Genauigkeit, näherungsweise 64-Bit-Entsprechung	Bereich: -1.79769E+308 bis -2.225E-307 oder 2.225E-307 bis 1.79769E+308 oder Null
DECIMAL(5,2)	Zahl	Genauigkeit: 5, Anzahl der Kommastellen: 2	Bereich: -10**31+1 bis 10**31-1
DATE	Wert für Datum und Uhrzeit	Dreiteiliger Wert	1991-10-27
TIME	Wert für Datum und Uhrzeit	Dreiteiliger Wert	13.30.05
TIMESTAMP	Wert für Datum und Uhrzeit	Siebenteiliger Wert	1991-10-27-13.30.05.000000

Weitere Informationen enthält die Tabelle für die Kompatibilität von Datentypen, die Sie im Handbuch *SQL Reference* finden.

Kapitel 3. Erstellen von Tabellen und Sichten

Dieses Kapitel beschreibt, wie Sie Tabellen und Sichten in DB2 Universal Database erstellen und bearbeiten können. Die Beziehung zwischen Tabellen und Sichten wird anhand von Abbildungen und Beispielen erläutert.

Die folgenden Themen werden behandelt:

- Erstellen von Tabellen und Erstellen von Sichten
- Einfügen von Daten
- Ändern von Daten
- Löschen von Daten
- Verwenden von Sichten zur Datenbearbeitung

Erstellen von Tabellen

Mit der Anweisung CREATE TABLE können Sie Ihre eigenen Tabellen erstellen. Hierbei werden die Spaltennamen, die Spaltentypen sowie *Integritätsbedingungen* angegeben. Integritätsbedingungen werden im Abschnitt „Durchsetzen von Geschäftsregeln mit Hilfe von Integritätsbedingungen und Auslösern” auf Seite 61 erörtert.

Die folgende Anweisung erstellt eine Tabelle namens PERS. Diese Tabelle ähnelt der Tabelle STAFF, enthält jedoch eine zusätzliche Spalte BIRTH_DATE für das Geburtsdatum.

```
CREATE TABLE PERS
    ( ID              SMALLINT        NOT NULL,
      NAME            VARCHAR(9),
      DEPT            SMALLINT WITH DEFAULT 10,
      JOB             CHAR(5),
      YEARS           SMALLINT,
      SALARY          DECIMAL(7,2),
      COMM            DECIMAL(7,2),
      BIRTH_DATE      DATE)
```

Die mit dieser Anweisung erstellte Tabelle enthält keine Daten. Im nächsten Abschnitt wird beschrieben, wie Daten in eine neue Tabelle eingefügt werden.

Wie aus dem Beispiel ersichtlich wird, geben Sie für jede Spalte sowohl einen Namen als auch einen Datentyp an. Die Datentypen werden im Abschnitt „Datentypen” auf Seite 6 erläutert. Die Angabe NOT NULL ist wahlfrei. Mit ihr kann festgelegt werden, daß Nullwerte in einer Spalte nicht zulässig sind. Auch die Angabe von Standardwerten ist wahlfrei.

In einer Anweisung CREATE TABLE können Sie noch viele weitere Optionen angeben, z. B. *eindeutige Integritätsbedingungen oder referentielle Integritätsbedingungen*. Weitere Informationen zu allen Optionen finden Sie im Abschnitt über die Anweisung CREATE TABLE im Handbuch *SQL Reference*.

Einfügen von Daten

Wenn Sie eine neue Tabelle erstellen, enthält diese keine Daten. Mit der Anweisung INSERT können Sie neue Zeilen in eine Tabelle eingeben. Diese Anweisung kann in zwei allgemeinen Formaten verwendet werden:

- Bei dem einen Format geben Sie Werte für die Spalten einer oder mehrerer Zeilen mit einer Klausel VALUES an. In den drei folgenden Beispielen wird dieses allgemeine Format verwendet, um Daten in Tabellen einzufügen.
- Bei dem anderen Format verwenden Sie anstelle einer Klausel VALUES eine *Gesamtauswahl,* die Spalten aus Zeilen in anderen Tabellen und/oder Sichten angibt.

Eine Gesamtauswahl ist eine in Anweisungen INSERT oder CREATE VIEW verwendete bzw. auf ein Prädikat folgende Anweisung SELECT. Eine in runde Klammern gesetzte Gesamtauswahl wird im allgemeinen als *Unterabfrage* bezeichnet.

Abhängig von den Standardoptionen, die Sie bei der Erstellung der Tabelle ausgewählt haben, stellen Sie für jede Zeile, die Sie einfügen, entweder einen Wert zur Verfügung, oder Sie übernehmen einen Standardwert. Die Standardwerte für die unterschiedlichen Datentypen werden im Handbuch *SQL Reference* behandelt.

Die nächste Anweisung verwendet eine Klausel VALUES, um eine Datenzeile in die Tabelle PERS einzufügen:

```
INSERT INTO PERS
   VALUES (12, 'Harris', 20, 'Sales', 5, 18000, 1000, '1950-1-1')
```

Die folgende Anweisung verwendet die Klausel VALUES, um drei Zeilen in die Tabelle PERS einzufügen, wobei lediglich die IDs, die Namen und die Tätigkeitsbezeichnungen bekannt sind. Ist eine Spalte als NOT NULL definiert und enthält sie keinen Standardwert, müssen Sie einen Wert für diese Spalte angeben.

In einer Anweisung CREATE TABLE kann die Klausel NOT NULL in einer Spaltendefinition durch den Zusatz WITH DEFAULT erweitert werden. Wenn eine Spalte als NOT NULL WITH DEFAULT oder ein Konstantenstandardwert wie beispielsweise WITH DEFAULT 10 definiert ist und Sie die Spalte in der Spaltenliste nicht angeben, wird in der eingefügten Zeile der Standardwert in

diese Spalte eingefügt. Beispiel: In der Anweisung CREATE TABLE wurde lediglich für die Spalte DEPT ein Standardwert angegeben und mit 10 definiert. Daher wird die Abteilungsnummer (Spalte DEPT) auf den Wert 10 gesetzt. Alle anderen Spalten, für die nicht explizit ein Wert angegeben wird, werden auf NULL gesetzt.

```
INSERT INTO PERS (NAME, JOB, ID)
   VALUES ('Swagerman', 'Prgmr', 500),
          ('Limoges', 'Prgmr', 510),
          ('Li', 'Prgmr', 520)
```

Die folgende Anweisung gibt das Ergebnis der Einfügungen zurück:

```
SELECT *
   FROM PERS

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM      BIRTH_DATE
------ --------- ------ ----- ------ --------- --------- ----------
    12 Harris        20 Sales      5  18000.00   1000.00 01/01/1950
   500 Swagerman     10 Prgmr      -         -         - -
   510 Limoges       10 Prgmr      -         -         - -
   520 Li            10 Prgmr      -         -         - -
```

Bitte beachten Sie, daß in diesem Fall nicht für alle Spalten Werte angegeben wurden. Nullwerte werden als Bindestrich (–) angezeigt. Damit dies funktioniert, muß die Liste der Spaltennamen sowohl hinsichtlich der Reihenfolge als auch in Bezug auf die Datentypen mit den Werten übereinstimmen, die in der Klausel VALUES angegeben werden. Wird die Liste der Spaltennamen übergangen (wie im ersten Beispiel), muß die nach VALUES angegebene Liste der Datenwerte dieselbe Reihenfolge wie die Spalten in der Tabelle aufweisen, in die die Werte eingefügt werden. Außerdem muß die Anzahl der Werte mit der Anzahl der Spalten in der Tabelle identisch sein.

Jeder Wert muß mit dem Datentyp der Spalte kompatibel sein, in die er eingefügt wird. Wenn für eine Spalte definitionsgemäß einen Nullwert enthalten kann und für diese Spalte kein Wert angegeben wird, erhält diese Spalte in der eingefügten Zeile den Wert NULL.

Im folgenden Beispiel wird in die Spalten YEARS, COMM und BIRTH_DATE ein Nullwert eingefügt, da für diese Spalten in der Zeile keine Werte angegeben wurden.

```
INSERT INTO PERS (ID, NAME, JOB, DEPT, SALARY)
   VALUES (410, 'Perna', 'Sales', 20, 20000)
```

Das zweite Format der Anweisung INSERT eignet sich besonders gut, um eine Tabelle mit Werten aus Zeilen in einer anderen Tabelle zu füllen. Wie bereits beschrieben, verwenden Sie hierbei anstelle einer Klausel VALUES eine Gesamtauswahl, um die Spalten aus Zeilen in anderen Tabellen und/oder Sichten anzugeben.

Im folgenden Beispiel werden Daten aus der Tabelle STAFF für die Mitarbeiter der Abteilung 38 ausgewählt und in die Tabelle PERS eingefügt:

```
INSERT INTO PERS (ID, NAME, DEPT, JOB, YEARS, SALARY)
   SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
      FROM STAFF
      WHERE DEPT = 38
```

Im Anschluß an diese Einfügung erstellt die folgende Anweisung SELECT ein Ergebnis, das mit der Gesamtauswahl in der Anweisung INSERT identisch ist.

```
SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
   FROM PERS
   WHERE DEPT = 38
```

Das Ergebnis lautet:

```
ID     NAME      DEPT   JOB   YEARS  SALARY
------ --------- ------ ----- ------ ---------
    30 Marenghi      38 Mgr        5  17506.75
    40 O'Brien       38 Sales      6  18006.00
    60 Quigley       38 Sales      -  16808.30
   120 Naughton      38 Clerk      -  12954.75
   180 Abrahams      38 Clerk      3  12009.75
```

Ändern von Daten

Zum Ändern der Daten in einer Tabelle verwenden Sie die Anweisung UPDATE. Mit dieser Anweisung können Sie den Wert einer oder mehrerer Spalten für alle Zeilen ändern, die die Suchbedingung der Klausel WHERE erfüllen.

Das folgende Beispiel aktualisiert die Informationen zum Mitarbeiter mit der ID 410:

```
UPDATE PERS
   SET JOB='Prgmr', SALARY = SALARY + 300
   WHERE ID = 410
```

Die Klausel SET gibt die Spalten an, die aktualisiert werden sollen, und stellt die Werte zur Verfügung.

Die Klausel WHERE ist wahlfrei. Sie gibt die Zeilen an, die aktualisiert werden sollen. Wird die Klausel WHERE übergangen, aktualisiert der Datenbankmanager alle Zeilen in der Tabelle oder Sicht mit den von Ihnen angegeben Werten.

Im vorangegangenen Beispiel wird zunächst die Tabelle (PERS) benannt und anschließend eine Bedingung für die zu aktualisierende Zeile angegeben. Die Informationen für die Personalnummer 410 haben sich geändert. Die Tätigkeitsbezeichnung für den Mitarbeiter wurde in 'Prgmr' geändert, und das Gehalt des Mitarbeiters wurde um $ 300 erhöht.

Wenn Sie Daten in mehr als einer Zeile ändern wollen, können Sie zu diesem Zweck eine Klausel WHERE einfügen, die auf zwei oder mehr Zeilen angewendet wird. Im folgenden Beispiel werden die Gehälter aller Verkaufsmitarbeiter (Tätigkeitsbeschreibung 'Sales') um 15% erhöht:

```
UPDATE PERS
   SET SALARY = SALARY * 1.15
   WHERE JOB = 'Sales'
```

Löschen von Daten

Mit der Anweisung DELETE können Sie Datenzeilen aus einer Tabelle löschen. Basis dieser Löschoperation ist die Suchbedingung, die in der Klausel WHERE angegeben wird. Im folgenden Beispiel wird die Zeile mit der Mitarbeiter-ID 120 gelöscht:

```
DELETE FROM PERS
   WHERE ID = 120
```

Die Klausel WHERE ist wahlfrei. Sie gibt die Zeilen an, die gelöscht werden sollen. Wird die Klausel WHERE übergangen, löscht der Datenbankmanager alle Zeilen in der Tabelle oder Sicht.

Mit der Anweisung DELETE können Sie auch mehrere Zeilen löschen. Im folgenden Beispiel werden alle Zeilen gelöscht, in denen für einen Mitarbeiter die Abteilungsnummer (DEPT) 20 angegeben ist:

```
DELETE FROM PERS
   WHERE DEPT = 20
```

Beim Löschen einer Zeile werden keine spezifischen Spaltenwerte aus einer Zeile entfernt, sondern die gesamte Zeile wird gelöscht.

Wenn Sie die Definition einer Tabelle sowie deren gesamten Inhalt löschen wollen, setzen Sie die Anweisung DROP TABLE ab. Eine entsprechende Beschreibung finden Sie im Handbuch *SQL Reference*.

Erstellen von Sichten

Wie bereits unter „Sichten” auf Seite 4 erläutert, bietet eine Sicht eine alternative Betrachtungsweise für die Daten in einer oder mehreren Tabellen. Durch das Erstellen von Sichten können Sie die Informationen einschränken, die für unterschiedliche Benutzer sichtbar sein sollen. Die folgende Abbildung verdeutlicht die Beziehung zwischen Sichten und Tabellen.

In Abb. 2 schränkt die Sicht View_A den Zugriff ausschließlich auf die Spalten AC1 und AC2 der Tabelle TABLE_A ein.

Die Sicht View_AB läßt den Zugriff auf die Spalte AC3 in der Tabelle TABLE_A und die Spalte BC2 in der Tabelle TABLE_B zu.

Durch das Erstellen der Sicht View_A schränken Sie den Zugriff ein, der für Benutzer auf die Tabelle TABLE_A möglich sein soll. Mit der Sicht VIEW_AB beschränken Sie den Zugriff auf bestimmte Spalten in beiden Tabellen.

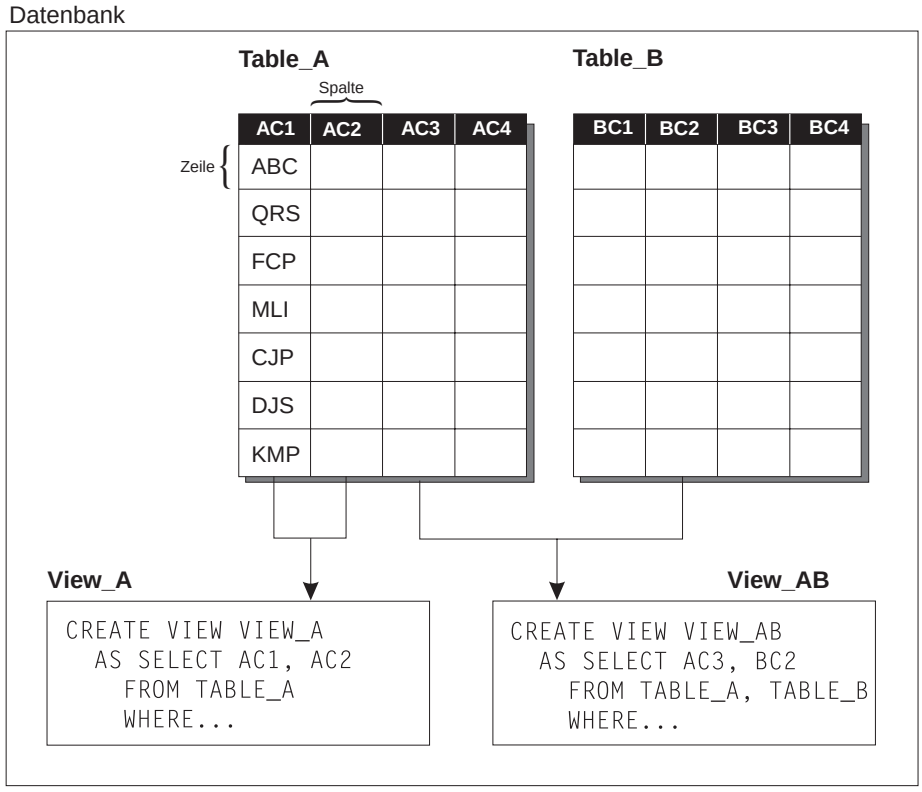


Abbildung 2. Beziehung zwischen Tabellen und Sichten

Die folgende Anweisung erstellt eine Sicht der nicht zur Abteilungsleitung ('Mgr') gehörenden Mitarbeiter der Abteilung 20 in der Tabelle STAFF, wobei die Angaben über Gehalt und Provision aus der Basistabelle nicht angezeigt werden.

```
CREATE VIEW STAFF_ONLY
   AS SELECT ID, NAME, DEPT, JOB, YEARS
         FROM STAFF
         WHERE JOB <> 'Mgr' AND DEPT=20
```

Nachdem die Sicht erstellt wurde, zeigt die folgende Anweisung den Inhalt der Sicht an:

```
SELECT *
   FROM STAFF_ONLY
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
ID     NAME      DEPT   JOB   YEARS
------ --------- ------ ----- ------
    20 Pernal        20 Sales      8
    80 James         20 Clerk      -
   190 Sneider       20 Clerk      8
```

In einem weiteren Beispiel wird anhand der Tabellen STAFF und ORG eine Sicht erstellt, die die Namen aller Abteilungen sowie den Namen des jeweiligen Abteilungsleiters auflistet. Diese Sicht wird durch die folgende Anweisung erstellt:

```
CREATE VIEW DEPARTMENT_MGRS
   AS SELECT NAME, DEPTNAME
         FROM STAFF, ORG
         WHERE MANAGER = ID
```

Mit Hilfe der Klausel WITH CHECK OPTION können Sie beim Erstellen einer Sicht zusätzliche Integritätsbedingungen für Einfügungen und Aktualisierungen einer Tabelle über eine Sicht angeben. Diese Klausel weist den Datenbankmanager an, alle Aktualisierungen oder Einfügungen, die für die Sicht vorgenommen werden, dahingehend auszuwerten, ob sie der Definition der Sicht entsprechen, sowie alle Aktualisierungen und Einfügungen zurückzuweisen, bei denen dies nicht der Fall ist. Wenn Sie diese Klausel übergehen, werden Einfügungen und Aktualisierungen nicht mit der Definition der Sicht abgeglichen. Ausführliche Informationen zur Funktionsweise von WITH CHECK OPTION finden Sie im Abschnitt über die Anweisung CREATE VIEW des Handbuchs *SQL Reference*.

Verwenden von Sichten zur Datenbearbeitung

Wie die Anweisung SELECT werden auch die Anweisungen INSERT, DELETE und UPDATE so auf eine Sicht angewendet, als ob es sich um eine echte Tabelle handelte. Die Anweisungen bearbeiten die Daten in der/den zugrundeliegenden Basistabelle(n). Sobald Sie erneut auf die Sicht zugreifen, wird sie daher anhand der aktuellsten Basistabelle(n) ausgewertet. Wenn Sie die Klausel WITH CHECK OPTION nicht verwenden, werden die unter Verwendung einer Sicht geänderten Daten unter Umständen bei wiederholten Zugriffen auf die Sicht nicht angezeigt, da die Daten möglicherweise nicht mehr mit der ursprünglichen Definition der Sicht übereinstimmen.

Im folgenden Beispiel wird eine Aktualisierung auf die Sicht FIXED_INCOME angewendet:

```
CREATE VIEW FIXED_INCOME (LNAME, DEPART, JOBTITLE, NEWSALARY)
   AS SELECT NAME, DEPT, JOB, SALARY
         FROM PERS
         WHERE JOB <> 'Sales' WITH CHECK OPTION

UPDATE FIXED_INCOME
   SET NEWSALARY = SALARY * 1.10
   WHERE LNAME = 'Li'
```

Die Aktualisierung in der vorherigen Sicht entspricht (mit Ausnahme der Prüfoption) dem Aktualisieren der Basistabelle PERS:

```
UPDATE PERS
   SET SALARY = SALARY * 1.10
   WHERE NAME = 'Li'
     AND JOB <> 'Sales'
```

Bitte beachten Sie, daß aufgrund der Erstellung der Sicht mit WITH CHECK OPTION für die Integritätsbedingung JOB <> 'Sales' in der Anweisung CREATE VIEW FIXED_INCOME die folgende Aktualisierung nicht zulässig ist, sobald der Mitarbeiter 'Limoges' in die Verkaufsabteilung ('Sales') wechselt:

```
UPDATE FIXED_INCOME
   SET JOBTITLE = 'Sales'
   WHERE LNAME = 'Limoges'
```

Durch Ausdrücke wie beispielsweise SALARY + COMM oder SALARY * 1.25 definierte Spalten können nicht aktualisiert werden. Wenn Sie eine Sicht definieren, die eine oder mehrere solcher Spalten enthält, erhält der Eigner das Zugriffsrecht UPDATE für diese Spalten nicht. Anweisungen INSERT sind für Sichten mit solchen Spalten nicht zulässig. Anweisungen DELETE können jedoch ausgeführt werden.

Bei einer angenommenen Tabelle PERS, in der keine Spalten als NOT NULL definiert sind, könnten Sie Zeilen sogar dann über die Sicht FIXED_INCOME in die Tabelle PERS einfügen, wenn die Sicht nicht die Spalten ID, YEARS, COMM oder BIRTHDATE aus der zugrundeliegenden Tabelle PERS enthält. Spalten, die in der Sicht nicht angezeigt werden, würden in diesem Fall auf NULL bzw. den Standardwert gesetzt.

In der Tabelle PERS ist jedoch die Spalte ID als NOT NULL definiert. Wenn Sie versuchen, eine Zeile über die Sicht FIXED_INCOME einzufügen, versucht das System, in alle Spalten der Tabelle PERS, die in der Sicht „nicht sichtbar" sind, Nullwerte einzufügen. Da die Spalte ID in der Sicht nicht enthalten ist und Nullwerte in dieser Spalte nicht zulässig sind, läßt das System das Einfügen über die Sicht nicht zu.

Angaben zu Regeln und Einschränkungen für das Ändern von Sichten können Sie dem Abschnitt über die Anweisung CREATE VIEW des Handbuchs *SQL Reference* entnehmen.

Kapitel 4. Verwenden von SQL-Anweisungen für den Zugriff auf Daten

Dieser Abschnitt beschreibt, wie eine Verbindung zu einer Datenbank hergestellt wird und wie Daten unter Verwendung von SQL-Anweisungen abgerufen werden.

In den Beispielen ist zunächst die einzugebende Anweisung angegeben. Darauf folgen (in den meisten Fällen) die Ergebnisse, die angezeigt werden, wenn diese Anweisung für die Beispieldatenbank abgesetzt wird. In den Beispielen sind die Anweisungen zwar in Großbuchstaben dargestellt, bei der Eingabe der Anweisungen muß jedoch die Groß-/Kleinschreibung nicht berücksichtigt werden. Eine Ausnahme bilden Anweisungen, die in einzelne (') oder doppelte ('') Anführungszeichen gesetzt sind.

Die in DB2 Universal Database enthaltene Datenbank SAMPLE besteht aus mehreren Tabellen, die in „Anhang A. Tabellen der Beispieldatenbank" auf Seite 85, aufgelistet sind. Diese Datenbank kann mit dem Installationsassistenten **Erste Schritte** erstellt werden. Sie können die Datenbank SAMPLE auch über die Befehlszeile erstellen. Weitere Informationen finden Sie im Handbuch *SQL Reference*.

Bitte beachten Sie auch die zusätzlichen Beispieldatenbanken in DB2 Universal Database, die die Funktionalität der Data Warehouse-Zentrale und des OLAP Starter Kits veranschaulichen. Die Beispiele im vorliegenden Handbuch verwenden ausschließlich die allgemeine Beispieldatenbank SAMPLE.

Abhängig davon, wie Ihre Datenbank konfiguriert wurde, müssen Sie unter Umständen die verwendeten Tabellennamen qualifizieren, indem Sie ihnen den Schemennamen und einen Punkt voranstellen. Die Beispiele in diesem Handbuch gehen davon aus, daß das Standardschema USERID lautet. Daher könnten Sie auf die Tabelle ORG auch durch die Angabe USERID.ORG verweisen. Bei Ihrem Administrator können Sie erfragen, ob dies erforderlich ist oder nicht.

Die folgenden Themen werden behandelt:

- Herstellen einer Verbindung zu einer Datenbank
- Untersuchen von Fehlern
- Auswählen von Spalten und Auswählen von Zeilen
- Sortieren von Zeilen und Entfernen von doppelten Zeilen
- Reihenfolge der Operationen

- Verwenden von Ausdrücken zum Berechnen von Werten
- Benennen von Ausdrücken
- Auswählen von Daten aus mehreren Tabellen
- Verwenden einer Unterabfrage
- Verwenden von Funktionen
- Gruppierung

Herstellen einer Verbindung zu einer Datenbank

Sie müssen eine Verbindung zu einer Datenbank herstellen herstellen, bevor sie mit Hilfe von SQL-Anweisungen abgefragt oder bearbeitet werden kann. Die Anweisung CONNECT ordnet einer Datenbankverbindung einen Benutzernamen zu.

Um beispielsweise eine Verbindung zur Datenbank SAMPLE herzustellen, geben Sie den folgenden Befehl im DB2-Befehlszeilenprozessor ein:

```
CONNECT TO SAMPLE USER USERID USING PASSWORD
```

(Vergewissern Sie sich, daß die ausgewählten Werte für Benutzer-ID und Kennwort auf dem Server-System gültig sind.)

Im vorliegenden Beispiel wird für USER der Wert USERID und für USING der Wert PASSWORD verwendet.

Die folgende Nachricht teilt Ihnen mit, daß die Verbindung erfolgreich hergestellt wurde:

```
Datenbankverbindungsinformationen

   Datenbankprodukt                = DB2/NT 7.1.0
   SQL-Berechtigungs-ID            = USERID
   Aliasname der lokalen Datenbank = SAMPLE
```

Sobald Sie eine Verbindung hergestellt haben, können Sie mit der Bearbeitung der Datenbank beginnen. Ausführliche Informationen zu Verbindungen finden Sie im Abschnitt über die Anweisung CONNECT des Handbuchs *SQL Reference*.

Untersuchen von Fehlern

Immer dann, wenn Sie bei der Eingabe eines der Beispiele einen Fehler machen oder wenn während der Ausführung einer SQL-Anweisung ein Fehler auftritt, gibt der Datenbankmanager eine Fehlernachricht zurück. Die Fehlernachricht besteht aus einer Nachrichten-ID, einer kurzen Erläuterung und einem SQLSTATE-Wert.

Fehler mit einem SQLSTATE-Wert sind Fehlercodes, die von der gesamten DB2-Produktfamilie verwendet werden. Fehler mit SQLSTATE-Werten entsprechen dem ISO / ANSI-Standard SQL92.

Würde beispielsweise in der Anweisung CONNECT eine falsche Angabe für Benutzer-ID oder Kennwort gemacht, würde der Datenbankmanager die Nachrichten-ID SQL1403N und den SQLSTATE-Wert 08004 zurückgeben. Die Nachricht lautet folgendermaßen:

```
SQL1403N Angegebene Benutzer-ID und/oder Kennwort sind/ist
         falsch.   SQLSTATE=08004
```

Zusätzliche Informationen zur Fehlernachricht können Sie aufrufen, indem Sie ein Fragezeichen (?) und anschließend die Nachrichten-ID oder den SQLSTATE-Wert eingeben:

```
        ? SQL1403N
ODER
        ? SQL1403
ODER
        ? 08004
```

Bitte beachten Sie, daß die zweite letzte Zeile der Beschreibung des Fehlers SQL1403N angibt, daß der SQLCODE-Wert -1403 lautet. Der SQLCODE-Wert ist ein produktspezifischer Fehlercode. Nachrichten-IDs, die mit einem N (Notification - Hinweis) oder einem C (Critical - Kritisch) enden, stehen für Fehler und haben negative SQLCODE-Werte. Nachrichten-IDs, die mit einem W (Warning - Warnung) enden, stehen für eine Warnung und haben positive SQLCODE-Werte.

Auswählen von Spalten

Mit der Anweisung SELECT können Sie spezifische Spalten aus einer Tabelle auswählen. Geben Sie in der Anweisung eine Liste von Spaltennamen an, die durch Kommata voneinander getrennt werden. Diese Liste wird als *SELECT-Liste* bezeichnet.

Die folgende Anweisung wählt die Abteilungsnamen (DEPTNAME) und die Abteilungsnummern (DEPTNUMB) aus der Tabelle ORG der Datenbank SAMPLE aus:

```
SELECT DEPTNAME, DEPTNUMB
   FROM ORG
```

Die vorstehende Anweisung erzeugt das folgende Ergebnis:

```
DEPTNAME       DEPTNUMB
-------------- --------
Head Office          10
New England          15
Mid Atlantic         20
South Atlantic       38
Great Lakes          42
Plains               51
Pacific              66
Mountain             84
```

Durch die Verwendung eines Sterns (*) können Sie alle Spalten aus der Tabelle auswählen. Im nächsten Beispiel werden alle Spalten und Zeilen aus der Tabelle ORG aufgelistet:

```
SELECT *
   FROM ORG
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
DEPTNUMB DEPTNAME       MANAGER DIVISION   LOCATION
-------- -------------- ------- ---------- -------------
      10 Head Office        160 Corporate  New York
      15 New England         50 Eastern    Boston
      20 Mid Atlantic        10 Eastern    Washington
      38 South Atlantic      30 Eastern    Atlanta
      42 Great Lakes        100 Midwest    Chicago
      51 Plains             140 Midwest    Dallas
      66 Pacific            270 Western    San Francisco
      84 Mountain           290 Western    Denver
```

Auswählen von Zeilen

Zur Auswahl von spezifischen Zeilen aus einer Tabelle geben Sie nach der Anweisung SELECT mit einer Klausel WHERE die Bedingung bzw. die Bedingungen an, die eine Zeile erfüllen muß, um ausgewählt zu werden. Ein Kriterium für die Auswahl von Zeilen aus einer Tabelle wird als *Suchbedingung* bezeichnet.

Eine Suchbedingung besteht aus einem oder mehreren *Prädikaten*. Ein Prädikat gibt eine Bedingung an, die für eine Zeile wahr oder falsch (bzw. unbekannt) ist. Zur Angabe von Bedingungen in der Klausel WHERE können Sie die folgenden Prädikate BASIC verwenden:

Prädikat	**Funktion**
x = y	x ist gleich y
x <> y	x ist ungleich y
x < y	x ist kleiner als y
x > y	x ist größer als y
x <= y	x ist kleiner-gleich y
x >= y	x ist größer-gleich y
IS NULL/IS NOT NULL	prüft auf Nullwerte

Beim Erstellen von Suchbedingungen müssen Sie darauf achten, daß Rechenoperationen nur für numerische Datentypen und Vergleiche nur zwischen kompatiblen Datentypen durchgeführt werden. Beispielsweise können Sie Zeichenfolgen nicht mit Zahlen vergleichen.

Wenn Sie Zeilen anhand eines Zeichenwerts auswählen, muß dieser Wert in einfache Anführungszeichen gesetzt werden (z. B. `WHERE JOB = 'Clerk'`), und jeder Zeichenwert muß exakt so eingegeben werden, wie er in der Datenbank vorhanden ist. Besteht der in der Datenbank enthaltene Datenwert aus Kleinbuchstaben, werden keine Zeilen ausgewählt, wenn Sie ihn in Großbuchstaben eingeben. Bei der Auswahl von Zeilen anhand eines numerischen Werts darf dieser Wert nicht in Anführungszeichen gesetzt werden (Beispiel: `WHERE DEPT = 20`).

Im folgenden Beispiel werden nur die Zeilen für die Abteilung 20 aus der Tabelle STAFF ausgewählt:

```
SELECT DEPT, NAME, JOB
   FROM STAFF
   WHERE DEPT = 20
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
 DEPT   NAME      JOB
 ------ --------- -----
     20 Sanders   Mgr
     20 Pernal    Sales
     20 James     Clerk
     20 Sneider   Clerk
```

Im nächsten Beispiel wird der Operator AND eingesetzt, um mehr als eine Bedingung anzugeben. Sie können beliebig viele Bedingungen angeben. Das folgende Beispiel wählt die Sachbearbeiter ('Clerk') der Abteilung 20 aus der Tabelle STAFF aus:

```
SELECT DEPT, NAME, JOB
   FROM STAFF
   WHERE JOB = 'Clerk'
   AND DEPT = 20
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
DEPT   NAME      JOB
------ --------- -----
    20 James     Clerk
    20 Sneider   Clerk
```

Ein Nullwert tritt auf, wenn kein Wert eingegeben wird und die Spalte keinen Standardwert unterstützt. Er kann außerdem auftreten, wenn der Wert ausdrücklich auf Null gesetzt ist. Ein Nullwert kann nur in Spalten auftreten, die für die Unterstützung von Nullwerten definiert sind. Das Definieren und Unterstützen von Nullwerten in Tabellen wird im Abschnitt „Erstellen von Tabellen" auf Seite 11 erörtert.

Mit den Prädikaten IS NULL und IS NOT NULL können Sie überprüfen, ob ein Nullwert vorhanden ist.

Die folgende Anweisung listet die Mitarbeiter auf, deren Provision (Spalte COMM) nicht bekannt ist:

```
SELECT ID, NAME
   FROM STAFF
   WHERE COMM IS NULL
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
    50 Hanes
   100 Plotz
   140 Fraye
   160 Molinare
   210 Lu
   240 Daniels
   260 Jones
   270 Lea
   290 Quill
```

Ein Wert 0 (Null) ist nicht dasselbe wie ein Nullwert. Die folgende Anweisung wählt alle Mitarbeiter aus, deren Provision die Höhe 0 (Null) aufweist:

```
SELECT ID, NAME
   FROM STAFF
   WHERE COMM = 0
```

Da die Beispieltabelle in der Spalte COMM keine Werte 0 enthält, ist die zurückgegebene Ergebnismenge leer.

Im nächsten Beispiel werden alle Zeilen in der Tabelle STAFF ausgewählt, deren Wert in der Spalte YEARS größer als 9 ist:

```
SELECT NAME, SALARY, YEARS
   FROM STAFF
   WHERE YEARS > 9
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME      SALARY    YEARS
--------- --------- ------
Hanes      20659.80     10
Lu         20010.00     10
Jones      21234.00     12
Quill      19818.00     10
Graham     21000.00     13
```

Sortieren von Zeilen

Manchmal ist es günstig, die Informationen in einer spezifischen Reihenfolge zurückgeben zu lassen. Mit der Klausel ORDER BY können Sie die Informationen anhand der Werte in einer oder mehreren Spalten sortieren.

Die folgende Anweisung zeigt die Mitarbeiter der Abteilung 84 sortiert nach der Länge der Betriebszugehörigkeit (Spalte YEARS) an:

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY YEARS
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME      JOB   YEARS
--------- ----- ------
Davis     Sales      5
Gafney    Clerk      5
Edwards   Sales      7
Quill     Mgr       10
```

Die Klausel ORDER BY wird als letzte Klausel der gesamten Anweisung SELECT angegeben. Bei den in dieser Klausel angegebenen Spalten kann es sich um Ausdrücke oder um beliebige Spalten der Tabelle handeln. Die Spaltennamen in der Klausel ORDER BY müssen nicht in der SELECT-Liste angegeben sein.

Die Zeilen können in aufsteigender oder in absteigender Reihenfolge sortiert werden. Hierzu geben Sie in der Klausel ORDER BY explizit ASC (=aufsteigend) oder DESC (=absteigend) an. Enthält die Klausel keine dieser Angaben, werden die Zeilen automatisch in aufsteigender Reihenfolge sortiert. Die folgende Anweisung zeigt die Mitarbeiter der Abteilung 84 nach der Länge der Betriebszugehörigkeit, und zwar in absteigender Reihenfolge sortiert, an:

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY YEARS DESC
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME      JOB   YEARS
--------- ----- ------
Quill     Mgr       10
Edwards   Sales      7
Davis     Sales      5
Gafney    Clerk      5
```

Sie können Zeilen nach Zeichenwerten wie auch nach numerischen Werten sortieren. Die folgende Anweisung zeigt die Mitarbeiter der Abteilung 84 in alphabetischer Reihenfolge der Namen an:

```
SELECT NAME, JOB, YEARS
   FROM STAFF
   WHERE DEPT = 84
   ORDER BY NAME
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME      JOB   YEARS
--------- ----- ------
Davis     Sales      5
Edwards   Sales      7
Gafney    Clerk      5
Quill     Mgr       10
```

Entfernen von doppelten Zeilen

Bei der Verwendung der Anweisung SELECT möchten Sie vielleicht vermeiden, daß Informationen doppelt zurückgegeben werden. Beispiel: Die Tabelle STAFF enthält eine Spalte DEPT, in der mehrere Abteilungsnummern einige aufgelistet sind. Außerdem enthält diese Tabelle eine Spalte JOB, in der einige Tätigkeitsbezeichnungen mehrfach aufgeführt sind.

Mit der Option DISTINCT in der Klausel SELECT können Sie doppelte Zeilen eliminieren. Wenn Sie beispielsweise DISTINCT in die Anweisung einfügen, wird jede Tätigkeitsbezeichnung nur einmal pro Abteilung aufgelistet:

```
SELECT DISTINCT DEPT, JOB
   FROM STAFF
   WHERE DEPT < 30
   ORDER BY DEPT, JOB
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
DEPT   JOB
------ -----
    10 Mgr
    15 Clerk
    15 Mgr
    15 Sales
    20 Clerk
    20 Mgr
    20 Sales
```

Mit DISTINCT wurden alle Zeilen eliminiert, die in der Gruppe der Spalten, welche in der Anweisung SELECT angegeben war, doppelte Daten enthalten.

Reihenfolge der Operationen

Die Reihenfolge der Operationen muß unbedingt berücksichtigt werden. Die Ausgabe einer Klausel stellt die Eingabe für die nächste Klausel dar, was aus der folgenden Liste deutlich wird. Ein Beispiel, in dem die Reihenfolge der Operationen berücksichtigt wurde, finden Sie unter „Benennen von Ausdrücken” auf Seite 31.

Die folgende Operationsfolge entspricht nicht notwendigerweise der Reihenfolge, in der die Operationen im DB2-Code ausgeführt werden. Diese einfache Erläuterung soll lediglich eine intuitivere Vorstellung von Abfragen ermöglichen. Die Operationsfolge lautet:

1. Klausel FROM
2. Klausel WHERE
3. Klausel GROUP BY
4. Klausel HAVING
5. Klausel SELECT
6. Klausel ORDER BY

Verwenden von Ausdrücken zum Berechnen von Werten

Ein *Ausdruck* ist eine Berechnung oder Funktion, die in eine Anweisung eingefügt wird. Die folgende Anweisung berechnet das Gehalt, das sich für jeden Mitarbeiter der Abteilung 38 bei Zuteilung einer Bonuszahlung von $ 500 ergeben würde:

```
SELECT DEPT, NAME, SALARY + 500
   FROM STAFF
   WHERE DEPT = 38
   ORDER BY 3
```

Das Ergebnis lautet:

```
DEPT   NAME      3
------ --------- ----------------
    38 Abrahams          12509.75
    38 Naughton          13454.75
    38 Quigley           17308.30
    38 Marenghi          18006.75
    38 O'Brien           18506.00
```

Auffällig ist, daß der Spaltenname der dritten Spalte eine Nummer ist. Hierbei handelt es sich um eine durch das System generierte Nummer, da der Ausdruck SALARY+500 keinen Spaltennamen angibt. Später wird diese Nummer in der Klausel ORDER BY verwendet, um die dritte Spalte zu bezeichnen. Im Abschnitt „Benennen von Ausdrücken” auf Seite 31 wird erläutert, wie Ausdrücke mit aussagekräftigen Namen versehen werden.

Zur Erstellung von arithmetischen Ausdrücken können Sie die arithmetischen Basisoperatoren für Addition (+), Subtraktion (−), Multiplikation (*) und Division (/) verwenden.

Die Operatoren können auf numerische Werte aus unterschiedlichen Operandenarten angewendet werden, wie beispielsweise
- Spaltennamen (wie in RATE * HOURS)
- Konstantenwerte (wie in RATE * 1.07)
- Skalarfunktionen (wie in LENGTH(NAME) + 1).

Benennen von Ausdrücken

Mit der wahlfreien Klausel AS können Sie einem Ausdruck einen aussagekräftigen Namen zuordnen, der den Rückbezug auf den Ausdruck vereinfacht. Sie können eine Klausel AS verwenden, um für jedes beliebige Element in der SELECT-Liste einen Namen zu vergeben.

Die folgende Anweisung zeigt alle Mitarbeiter an, deren Gehalt (Spalte SALARY) zuzüglich Provision weniger als $ 13.000 beträgt. Der Ausdruck SALARY + COMM wird mit PAY benannt:

```
SELECT NAME, JOB, SALARY + COMM AS PAY
   FROM STAFF
   WHERE (SALARY + COMM) < 13000
   ORDER BY PAY
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME      JOB   PAY
--------- ----- ----------
Yamaguchi Clerk   10581.50
Burke     Clerk   11043.50
Scoutten  Clerk   11592.80
Abrahams  Clerk   12246.25
Kermisch  Clerk   12368.60
Ngan      Clerk   12714.80
```

Durch die Verwendung der Klausel AS können Sie die Klausel ORDER BY auf einen bestimmten Spaltennamen anstelle der durch das System generierten Nummer beziehen. Im dargestellten Beispiel wird in der Klausel WHERE der Vergleich zwischen (SALARY + COMM) und 13000 vorgenommen, anstatt den Namen PAY zu verwenden. Dies ergibt sich aus der Reihenfolge der Operationen. Die Klausel WHERE wird ausgewertet, bevor für (SALARY + COMM) der Name PAY vergeben wird, weil die Klausel SELECT nach der Klausel WHERE ausgeführt wird. Daher kann PAY im Prädikat nicht verwendet werden.

Auswählen von Daten aus mehreren Tabellen

Mit der Anweisung SELECT können Sie Berichte erzeugen, die Informationen aus zwei oder mehr Tabellen enthalten. Dies wird gemeinhin als *Verknüpfung* bezeichnet. Sie können beispielsweise Daten aus den Tabellen STAFF und ORG verknüpfen und so eine neue Tabelle bilden. Zum Verknüpfen zweier Tabellen geben Sie in der Klausel SELECT die anzuzeigenden Spalten, in einer Klausel FROM die Tabellennamen und in der Klausel WHERE die Suchbedingung an. Die Verwendung der Klausel WHERE ist wahlfrei.

Das nächste Beispiel ordnet den Namen der jeweiligen Abteilungsleiter einen Abteilungsnamen zu. Zu diesem Zweck müssen Informationen aus zwei Tabellen ausgewählt werden, da die Mitarbeiterdaten in der Tabelle STAFF und die Abteilungsinformationen in der Tabelle ORG separat gespeichert sind. Die folgende Abfrage wählt die Spalten NAME und DEPTNAME aus den Tabellen STAFF bzw. ORG aus. Die Suchbedingung schränkt die Auswahl auf diejenigen Zeilen ein, in denen die Werte in der Spalte MANAGER mit den Werten in der Spalte ID identisch sind:

```
SELECT DEPTNAME, NAME
   FROM ORG, STAFF
   WHERE MANAGER = ID
```

Abb. 3 veranschaulicht, wie Spalten in zwei unterschiedlichen Tabellen verglichen werden. Die eingerahmten Werte weisen auf eine Übereinstimmung hin, die die Suchbedingung erfüllt.

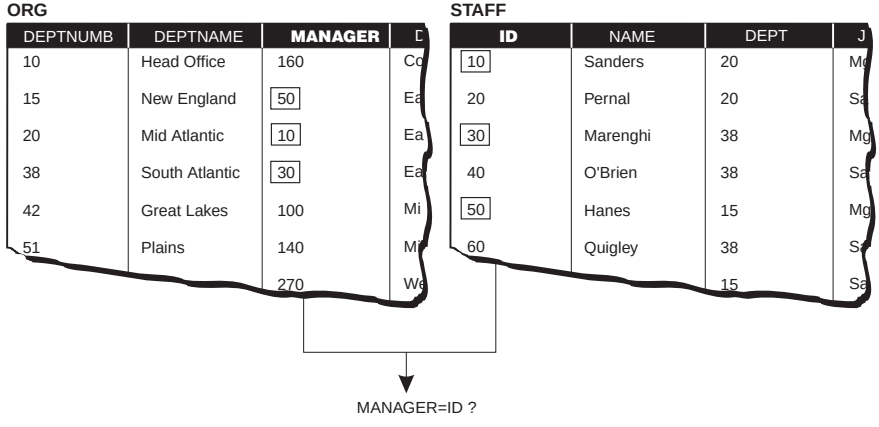


Abbildung 3. Auswählen aus den Tabellen STAFF und ORG

Die Anweisung SELECT erzeugt das folgende Ergebnis:

```
DEPTNAME       NAME
-------------- ---------
Mid Atlantic   Sanders
South Atlantic Marenghi
New England    Hanes
Great Lakes    Plotz
Plains         Fraye
Head Office    Molinare
Pacific        Lea
Mountain       Quill
```

Das Ergebnis listet die Namen der Abteilungsleiter sowie die entsprechende Abteilung auf.

Verwenden einer Unterabfrage

Beim Schreiben einer SQL-Anweisung SELECT können Sie zusätzliche Anweisungen SELECT in die Klausel WHERE stellen. Jede zusätzliche Anweisung SELECT startet eine Unterabfrage. Eine Unterabfrage kann dann wiederum eine weitere, separate Unterabfrage enthalten, deren Ergebnis in die Klausel WHERE der ursprünglichen Unterabfrage eingesetzt wird. Außerdem kann eine Klausel WHERE Unterabfragen in mehreren Suchbedingungen enthalten. Die Unterabfrage kann sich auf andere Tabellen und Spalten als die Hauptabfrage beziehen.

Die folgende Anweisung wählt den Unternehmensbereich (Spalte DIVISION) und den Standort (Spalte LOCATION) aus der Tabelle ORG für den Mitarbeiter aus, dessen ID in der Tabelle STAFF 280 lautet:

```
SELECT DIVISION, LOCATION
   FROM ORG
   WHERE DEPTNUMB = (SELECT DEPT
                        FROM STAFF
                        WHERE ID = 280)
```

Beim Verarbeiten einer Anweisung bestimmt DB2 zunächst das Ergebnis der Unterabfrage. Das Ergebnis der Unterabfrage aus dem Beispiel ist 66, da der Mitarbeiter mit der ID 280 zur Abteilung 66 gehört. Das Endergebnis wird anschließend aus der Zeile der Tabelle ORG entnommen, deren Spalte DEPTNUMB den Wert 66 hat. Das Endergebnis lautet:

```
DIVISION   LOCATION
---------- -------------
Western    San Francisco
```

Wenn Sie eine Unterabfrage verwenden, wertet der Datenbankmanager sie aus und setzt den resultierenden Wert direkt in die Klausel WHERE ein. Unterabfragen werden im Abschnitt „Unterabfragen mit Korrelationsbezug” auf Seite 46 näher erläutert.

Verwenden von Funktionen

Dieser Abschnitt enthält eine kurze Einführung in die Funktionen, die in den Beispielen in diesem Handbuch verwendet werden. Eine *Datenbankfunktion* ist die Beziehung zwischen einer Gruppe von Eingabedatenwerten und einem Ergebniswert.

Funktionen können *integriert* oder *benutzerdefiniert* sein. DB2 Universal Database bietet viele integrierte und vorinstallierte benutzerdefinierte Funktionen.

Die integrierten Funktionen finden Sie im Schema SYSIBM. Die vorinstallierten benutzerdefinierten Funktionen befinden sich im Schema SYSFUN. SYSIBM und SYSFUN sind reservierte Schemata.

Nicht alle Benutzeranforderungen können mit Hilfe der integrierten und vorinstallierten benutzerdefinierten Funktionen erfüllt werden. Daher müssen Anwendungsentwickler unter Umständen eine eigene Funktionsgruppe erstellen, die speziell auf deren Anwendungen abgestimmt ist. Ermöglicht wird dies durch benutzerdefinierte Funktionen, die den Bereich von DB2 Universal Database so erweitern, daß beispielsweise angepaßte Geschäfts- oder wissenschaftliche Funktionen integriert werden. Dieser Aspekt wird unter „Benutzerdefinierte Funktionen” auf Seite 78 näher erläutert.

Spaltenfunktionen

Spaltenfunktionen werden auf eine Gruppe von Werten in einer Spalte angewendet, um einen einzelnen Ergebniswert abzuleiten. Im folgenden werden nur einige Beispiele von Spaltenfunktionen dargestellt. Eine vollständige Liste finden Sie im Handbuch *SQL Reference*.

AVG Gibt die Summe der Werte in einer Gruppe dividiert durch die Anzahl der Werte in dieser Gruppe zurück.

COUNT Gibt die Anzahl der Zeilen oder Werte in einer Gruppe von Zeilen oder Werten zurück.

MAX Gibt den größten Wert aus einer Gruppe von Werten zurück.

MIN Gibt den kleinsten Wert aus einer Gruppe von Werten zurück.

Die folgende Anweisung wählt das höchste Gehalt (Spalte SALARY) aus der Tabelle STAFF aus:

```
SELECT MAX(SALARY)
   FROM STAFF
```

Diese Anweisung gibt den Wert 22959.20 aus der Beispieltabelle STAFF zurück.

Das nächste Beispiel wählt die Namen und Gehälter der Mitarbeiter aus, deren Einkommen über dem Durchschnittseinkommen liegt, die jedoch eine geringere als die durchschnittliche Betriebszugehörigkeitsdauer aufweisen.

```
SELECT NAME, SALARY
   FROM STAFF
   WHERE SALARY > (SELECT AVG(SALARY) FROM STAFF)
   AND YEARS < (SELECT AVG(YEARS) FROM STAFF)
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
 NAME      SALARY
--------- ---------
Marenghi   17506.75
Daniels    19260.25
Gonzales   16858.20
```

Im vorstehenden Beispiel ist die Spaltenfunktion in die Klausel WHERE in Form einer *Unterabfrage* integriert, statt direkt implementiert zu sein (z. B. WHERE SALARY > AVG(SALARY)). In der Klausel WHERE können keine Spaltenfunktionen angegeben werden. Dies ergibt sich aus der Reihenfolge der Operationen. Die Klausel WHERE wird vor der Klausel SELECT ausgewertet. Infolgedessen kann bei Auswertung der Klausel WHERE die Spaltenfunktion nicht auf die Gruppe von Werten zugreifen. Diese Gruppe von Werten wird zu einem späteren Zeitpunkt durch die Klausel SELECT ausgewählt.

Das Element DISTINCT kann als Bestandteil des Arguments einer Spaltenfunktion verwendet werden, um gleiche Werte zu eliminieren, bevor eine Funktion angewendet wird. Durch die Angabe COUNT(DISTINCT WORKDEPT) wird somit die Anzahl der unterschiedlichen Abteilungen berechnet.

Skalarfunktionen

Eine *Skalarfunktion* führt eine Operation für einen Einzelwert aus und gibt einen anderen Einzelwert zurück. Im folgenden werden nur einige Beispiele für die Skalarfunktionen von DB2 Universal Database dargestellt.

ABS Gibt den absoluten Wert einer Zahl zurück.

HEX Gib die hexadezimale Darstellung eines Werts zurück.

LENGTH Gibt die Anzahl der Byte in einem Argument zurück (bei einer Grafikzeichenfolge wird die Anzahl der Doppelbytezeichen zurückgegeben).

YEAR Extrahiert den Abschnitt für das Jahr aus einem Wert für Datum und Uhrzeit.

Eine ausführliche Liste und Beschreibung der Skalarfunktionen finden Sie im Handbuch *SQL Reference*.

Die folgende Anweisung gibt die Abteilungsnamen aus der Tabelle ORG zusammen mit der Länge dieser Namen zurück:

```
SELECT DEPTNAME, LENGTH(DEPTNAME)
   FROM ORG
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
DEPTNAME       2
-------------- -----------
Head Office             11
New England             11
Mid Atlantic            12
South Atlantic          14
Great Lakes             11
Plains                   6
Pacific                  7
Mountain                 8
```

Anmerkung: Da der Angabe LENGTH(DEPTNAME) nicht mit Hilfe der Klausel AS ein aussagekräftiger Name zugeordnet wurde, wird in der zweiten Spalte eine durch das System generierte Nummer angezeigt.

Tabellenfunktionen

Tabellenfunktionen geben Spalten einer Tabelle zurück. Das Ergebnis gleicht der Tabelle, die durch eine einfache Anweisung CREATE TABLE erstellt wird. Eine Tabellenfunktion kann nur in der Klausel FROM einer SQL-Anweisung verwendet werden.

Die einzige in DB2 Universal Database gegenwärtig unterstützte Tabellenfunktion ist die Funktion SQLCACHE_SNAPSHOT.

SQLCACHE_SNAPSHOT
Gibt die Ergebnisse einer Momentaufnahme des DB2-Caches für dynamische SQL-Anweisungen in Form einer Tabelle zurück.

Gruppierung

DB2 Universal Database kann Daten basierend auf bestimmten Spalten einer Tabelle analysieren.

Sie können Zeilen gemäß einer Gruppierungsstruktur organisieren, die in einer Klausel GROUP BY definiert wird. In ihrer einfachsten Form besteht eine *Gruppe* aus einer Reihe von Zeilen mit identischen Werten in den Spalten, die in der Klausel GROUP BY angegeben sind. Die Spaltennamen in der Klausel SELECT müssen entweder Gruppierungsspalten oder Spaltenfunktionen angeben. Spaltenfunktionen geben einen Wert für jede Gruppe zurück, die in der Klausel GROUP BY definiert ist. Jede Gruppe wird in der

Ergebnismenge durch eine Zeile dargestellt. Das folgende Beispiel erzeugt ein Ergebnis, das das höchste Gehalt für jede Abteilungsnummer auflistet:

```
SELECT DEPT, MAX(SALARY) AS MAXIMUM
   FROM STAFF
   GROUP BY DEPT
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
DEPT   MAXIMUM
------ ---------
    10  22959.20
    15  20659.80
    20  18357.50
    38  18006.00
    42  18352.80
    51  21150.00
    66  21000.00
    84  19818.00
```

Bitte beachten Sie, daß MAX(SALARY) für jede Abteilung, eine durch die Klausel GROUP BY definierte Gruppe, und nicht für das gesamte Unternehmen berechnet wird.

Verwenden einer Klausel WHERE mit einer Klausel GROUP BY

Eine Gruppierungsabfrage kann eine Standardklausel WHERE enthalten, die Zeilen, welche das Qualifikationsmerkmal nicht erfüllen, eliminiert, bevor die Gruppen gebildet und die Spaltenfunktionen berechnet werden. Die Klausel WHERE muß *vor* der Klausel GROUP BY angegeben werden. Beispiel:

```
SELECT WORKDEPT, EDLEVEL, MAX(SALARY) AS MAXIMUM
   FROM EMPLOYEE
   WHERE HIREDATE > '1979-01-01'
   GROUP BY WORKDEPT, EDLEVEL
   ORDER BY WORKDEPT, EDLEVEL
```

Das Ergebnis lautet:

```
WORKDEPT EDLEVEL MAXIMUM
-------- ------- -----------
D11           17    18270.00
D21           15    27380.00
D21           16    36170.00
D21           17    28760.00
E11           12    15340.00
E21           14    26150.00
```

Bitte beachten Sie, daß jeder in der Anweisung SELECT angegebene Spaltenname auch in der Klausel GROUP BY enthalten ist. Werden die Spaltennamen nicht in beiden Positionen angegeben, erhalten Sie einen Fehler. Die Klausel GROUP BY gibt eine Zeile für jede eindeutige Kombination aus WORKDEPT und EDLEVEL zurück.

Verwenden der Klausel HAVING nach der Klausel GROUP BY

Sie können eine Qualifizierungsbedingung auf Gruppen anwenden. DB2 gibt dann nur für die Gruppen ein Ergebnis zurück, die die Bedingung erfüllen. Zu diesem Zweck fügen Sie *nach* der Klausel GROUP BY eine Klausel HAVING ein. Eine Klausel HAVING kann eines oder mehrere Prädikate enthalten, die durch die Operatoren AND und OR verbunden sind. Jedes Prädikat vergleicht ein Merkmal der Gruppe (wie z. B. AVG(SALARY)) mit einem der folgenden Elemente:

- Anderes Merkmal in der Gruppe

  Beispiel:

  ```
  HAVING AVG(SALARY) > 2 * MIN(SALARY)
  ```

- Konstante

  Beispiel:

  ```
  HAVING AVG(SALARY) > 20000
  ```

Beispielsweise sucht die folgende Abfrage nach dem höchsten und dem niedrigsten Gehalt in Abteilungen mit mehr als 4 Mitarbeitern:

```
SELECT WORKDEPT, MAX(SALARY) AS MAXIMUM, MIN(SALARY) AS MINIMUM
   FROM EMPLOYEE
   GROUP BY WORKDEPT
   HAVING COUNT(*) > 4
   ORDER BY WORKDEPT
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
WORKDEPT MAXIMUM     MINIMUM
-------- ----------- -----------
D11         32250.00    18270.00
D21         36170.00    17250.00
E11         29750.00    15340.00
```

Es ist möglich (wenngleich unüblich), daß eine Abfrage eine Klausel HAVING, aber keine Klausel GROUP BY enthält. In diesem Fall behandelt DB2 die gesamte Tabelle als eine Gruppe. Da die Tabelle als Gruppe behandelt wird, können Sie höchstens eine Ergebniszeile erhalten. Wenn die in der Klausel HAVING angegebene Bedingung für die Tabelle als ganzes wahr (= zutreffend) ist, wird das ausgewählte Ergebnis (das vollständig aus Spaltenfunktionen bestehen muß) zurückgegeben. Andernfalls werden keine Zeilen zurückgegeben.

Kapitel 5. Ausdrücke und Unterabfragen

DB2 bietet Flexibilität beim Ausdrücken von Abfragen. Das vorliegende Kapitel beschreibt einige der wichtigen Methoden, mit denen komplexe Abfragen ausgedrückt werden können.

Es enthält eine umfassende Beschreibung der folgenden Aspekte:

- Skalare Gesamtauswahl
- Umsetzen von Datentypen
- Ausdrücke CASE
- Tabellenausdrücke
- Korrelationsnamen

Skalare Gesamtauswahl

Eine Gesamtauswahl ist eine Form der Abfrage, die in SQL-Anweisungen verwendet werden kann. Eine skalare Gesamtauswahl ist eine Gesamtauswahl, die eine Zeile mit nur einem Wert zurückgibt. Eine skalare Gesamtauswahl ist dann sinnvoll, wenn Datenwerte aus der Datenbank abgerufen werden sollen, um sie in einem Ausdruck zu verwenden.

- Das folgende Beispiel listet die Namen der Mitarbeiter (Tabelle EMPLOYEE) auf, deren Gehalt das durchschnittliche Gehalt aller Mitarbeiter übersteigt. Die skalare Gesamtauswahl innerhalb der Abfrage ist die in runde Klammern gesetzte Anweisung SELECT.

```
SELECT LASTNAME, FIRSTNME
   FROM EMPLOYEE
   WHERE  SALARY > (SELECT AVG(SALARY)
                    FROM EMPLOYEE)
```

- Das nächste Beispiel sucht nach dem Durchschnittsgehalt der Mitarbeiter in der Tabelle STAFF und nach dem Durchschnittsgehalt der Mitarbeiter in der Tabelle EMPLOYEE.

```
SELECT AVG(SALARY) AS "Average_Employee",
   (SELECT AVG(SALARY) AS "Average_Staff" FROM STAFF)
      FROM EMPLOYEE
```

Umsetzen von Datentypen

Manchmal müssen Werte aus einem Datentyp in einen anderen Datentyp umgesetzt werden, beispielsweise ein numerischer Wert in eine Zeichenfolge. Zur Umsetzung eines Werts in einen anderen Datentyp verwenden Sie die Spezifikation CAST.

Ein anderer Verwendungszweck für eine Spezifikation CAST ist das Abschneiden einer sehr langen Zeichenfolge. In der Tabelle EMP_RESUME hat die Spalte RESUME den Datentyp CLOB(5K). Unter Umständen wollen Sie nur die ersten 370 Zeichen mit den persönlichen Daten des Mitarbeiters anzeigen. Um die ersten 370 Zeichen aus den Lebensläufen im ASCII-Format aus der Tabelle EMP_RESUME anzuzeigen, setzen Sie die folgende Abfrage ab:

```
SELECT EMPNO, CAST(RESUME AS VARCHAR(370))
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
```

Es wird eine Warnung ausgegeben. Sie informiert Sie darüber, daß Werte mit einer Länge von mehr als 370 Zeichen abgeschnitten werden.

Sie können Nullwerte in andere Datentypen umsetzen, die in einer Abfrage besser bearbeitet werden können. Unter „Allgemeine Tabellenausdrücke” auf Seite 44 ist eine Beispiel für die Verwendung der Umsetzung zu diesem Zweck dargestellt.

Ausdrücke CASE

Mit Ausdrücken CASE in SQL-Anweisungen können Sie die Datendarstellung einer Tabelle ganz leicht bearbeiten. Hierdurch verfügen Sie über eine leistungsstarke Funktion für Bedingungsausdrücke, die in ihrem Konzept den Anweisungen CASE in bestimmten Programmiersprachen ähnelt.

- Um die Abteilungsnummern in der Spalte DEPTNAME der Tabelle ORG in aussagekräftige Begriffe zu ändern, geben Sie die folgende Abfrage ein:

```
SELECT DEPTNAME,
   CASE DEPTNUMB
     WHEN 10 THEN 'Marketing'
     WHEN 15 THEN 'Research'
     WHEN 20 THEN 'Development'
     WHEN 38 THEN 'Accounting'
     ELSE 'Sales'
   END  AS FUNCTION
   FROM ORG
```

Das Ergebnis lautet:

```
DEPTNAME       FUNCTION
-------------- -----------
Head Office    Marketing
New England    Research
Mid Atlantic   Development
South Atlantic Accounting
Great Lakes    Sales
Plains         Sales
Pacific        Sales
Mountain       Sales
```

- Mit Ausdrücken CASE können Sie einen Schutz vor Ausnahmebedingungen wie beispielsweise die Division durch 0 einrichten. Im folgenden Beispiel verhindert die Bedingung der Anweisung einen Fehler, indem bei Mitarbeitern ohne Bonus- oder Provisionszahlung die Division verhindert wird:

```
SELECT LASTNAME, WORKDEPT FROM EMPLOYEE
   WHERE(CASE
            WHEN BONUS+COMM=0 THEN NULL
            ELSE SALARY/(BONUS+COMM)
         END ) > 10
```

- Mit einem Ausdruck CASE können Sie ein Verhältnis der Summe einer Untergruppe von Werten aus einer Spalte zur Summe aller Werte aus dieser Spalte herstellen. Dieses Verhältnis kann in einer einzigen Anweisung enthalten sein, die einen Ausdruck CASE verwendet. Dieser Ausdruck benötigt lediglich einen einzigen Arbeitsgang in den Daten. Ohne einen Ausdruck CASE sind mindestens zwei Arbeitsgänge erforderlich, um dieselbe Berechnung auszuführen.

  Das folgende Beispiel berechnet das Verhältnis der Summe der Gehälter in der Abteilung 20 zur Gesamtsumme aller Gehälter und verwendet zu diesem Zweck einen Ausdruck CASE:

```
SELECT CAST(CAST (SUM(CASE
                        WHEN DEPT = 20 THEN SALARY
                        ELSE 0
                      END) AS DECIMAL(7,2))/
                   SUM(SALARY) AS DECIMAL (3,2))
   FROM STAFF
```

  Das Ergebnis lautet 0.11. Bitte beachten Sie, daß die Funktionen CAST sicherstellen, daß die Genauigkeit des Ergebnisses erhalten bleibt.

- Mit einem Ausdruck CASE können Sie eine einfache Funktion auswerten, anstatt die Funktion selbst aufzurufen, was einen zusätzlichen Systemaufwand erfordern würde. Beispiel:

```
CASE
   WHEN X<0 THEN -1
   WHEN X=0 THEN 0
   WHEN X>0 THEN 1
END
```

  Dieser Ausdruck führt zu demselben Ergebnis wie die benutzerdefinierte Funktion SIGN im Schema SYSFUN.

Tabellenausdrücke

Wenn Sie die Definition einer Sicht lediglich für eine einzige Abfrage benötigen, können Sie einen *Tabellenausdruck* verwenden.

Tabellenausdrücke sind temporäre Elemente und nur für die Dauer der SQL-Anweisung gültig. Anders als Sichten können sie nicht gemeinsam benutzt werden. Sie bieten jedoch im Vergleich zu Sichten eine größere Flexibilität.

Dieser Abschnitt beschreibt, wie allgemeine Tabellenausdrücke und verschachtelte Tabellenausdrücke in Abfragen eingesetzt werden.

Verschachtelte Tabellenausdrücke

Ein verschachtelter Tabellenausdruck ist eine temporäre Sicht, wobei die Definition in der Klausel FROM der Hauptabfrage *verschachtelt* (= direkt definiert) ist.

Die folgende Abfrage verwendet einen verschachtelten Tabellenausdruck, um den durchschnittlichen Gesamtverdienst, die Bildungsstufe und das Einstellungsjahr für Mitarbeiter mit einer höheren Bildungsstufe als 16 zu ermitteln:

```
SELECT EDLEVEL, HIREYEAR, DECIMAL(AVG(TOTAL_PAY),7,2)
   FROM (SELECT EDLEVEL, YEAR(HIREDATE) AS HIREYEAR,
                SALARY+BONUS+COMM AS TOTAL_PAY
           FROM EMPLOYEE
           WHERE EDLEVEL > 16)  AS PAY_LEVEL
   GROUP BY EDLEVEL, HIREYEAR
   ORDER BY EDLEVEL, HIREYEAR
```

Das Ergebnis lautet folgendermaßen:

```
EDLEVEL HIREYEAR    3
------- ----------- ---------
     17        1967  28850.00
     17        1973  23547.00
     17        1977  24430.00
     17        1979  25896.50
     18        1965  57970.00
     18        1968  32827.00
     18        1973  45350.00
     18        1976  31294.00
     19        1958  51120.00
     20        1975  42110.00
```

In dieser Abfrage wird zunächst das Einstellungsjahr mit einem verschachtelten Tabellenausdruck aus der Spalte HIREDATE extrahiert, damit es anschließend in der Klausel GROUP BY verwendet werden kann. In diesem Fall ist es unter Umständen sinnvoller, keine Sicht zu erstellen, wenn ähnliche Abfragen mit unterschiedlichen Werten für EDLEVEL ausgeführt werden sollen.

Das vorstehende Beispiel verwendet die integrierte Skalarfunktion DECIMAL. DECIMAL gibt die Dezimaldarstellung einer Zahl oder einer Zeichenfolge zurück. Ausführliche Informationen zu Funktionen finden Sie im Handbuch *SQL Reference*.

Allgemeine Tabellenausdrücke

Ein *allgemeiner Tabellenausdruck* ist ein Tabellenausdruck, der zur Verwendung in einer komplexen Abfrage erstellt wird. Er wird am Beginn der Abfrage in einer Klausel WITH definiert und benannt. Bei wiederholtem Bezug auf einen allgemeinen Tabellenausdruck wird dieselbe Ergebnismenge verwendet. Im Vergleich hierzu würde bei der Verwendung von verschachtelten Tabellenausdrücken oder Sichten die Ergebnismenge jedesmal erneut und möglicherweise mit unterschiedlichen Ergebnissen generiert.

Das folgende Beispiel listet alle Mitarbeiter im Unternehmen auf, die eine höhere Bildungsstufe als 16 aufweisen und deren Verdienst unter dem Durchschnittsverdienst der Mitarbeiter liegt, die in demselben Jahr eingestellt wurden und dieselbe Bildungsstufe haben. Die einzelnen Bestandteile der Abfrage werden im Anschluß an die Abfrage ausführlicher beschrieben.

```
1
WITH
      PAYLEVEL AS
         (SELECT EMPNO, EDLEVEL, YEAR(HIREDATE) AS HIREYEAR,
                 SALARY+BONUS+COMM AS TOTAL_PAY
             FROM EMPLOYEE
             WHERE EDLEVEL > 16),

2
      PAYBYED (EDUC_LEVEL, YEAR_OF_HIRE, AVG_TOTAL_PAY) AS
         (SELECT EDLEVEL, HIREYEAR, AVG(TOTAL_PAY)
             FROM PAYLEVEL
             GROUP BY EDLEVEL, HIREYEAR)

3
SELECT EMPNO, EDLEVEL, YEAR_OF_HIRE, TOTAL_PAY, DECIMAL(AVG_TOTAL_PAY,7,2)
   FROM PAYLEVEL, PAYBYED
   WHERE EDLEVEL = EDUC_LEVEL
     AND HIREYEAR = YEAR_OF_HIRE
     AND TOTAL_PAY < AVG_TOTAL_PAY
```

1 Dies ist ein allgemeiner Tabellenausdruck mit dem Namen PAYLEVEL. Die entsprechende Ergebnistabelle enthält eine Personalnummer, das Einstellungsjahr, den Gesamtverdienst des Mitarbeiters und seine bzw. ihre Bildungsstufe. In die Tabelle werden Zeilen nur für Mitarbeiter mit einer höheren Bildungsstufe als 16 aufgenommen.

2 Dies ist ein allgemeiner Tabellenausdruck namens PAYBYED (PAY BY EDucation = Gehalt nach Bildung). Er verwendet die Tabelle PAYLEVEL, die mit dem vorangegangenen allgemeinen Tabellenausdruck erstellt wurde. Er bestimmt die Bildungsstufe, das Einstellungsjahr und den Durchschnittsverdienst der Mitarbeiter innerhalb jeder Bildungsstufe, die in demselben Jahr eingestellt wurden. Die durch

diese Tabelle zurückgegebenen Spalten wurden anders benannt (z. B. mit EDUC_LEVEL) als die Spaltennamen, die in der SELECT-Liste verwendet wurden. Die erzeugte Ergebnismenge namens PAYBYED ist mit dem Ergebnis identisch, das im Beispiel des Abschnitts „Verschachtelte Tabellenausdrücke" dargestellt ist.

3 Abschließend folgt die eigentliche Abfrage, die das gewünschte Ergebnis erzeugt. Die beiden Tabellen (PAYLEVEL, PAYBYED) werden verknüpft, um die Personen zu ermitteln, deren Gesamtverdienst geringer als der durchschnittliche Verdienst der Mitarbeiter ist, die in demselben Jahr eingestellt wurden. Bitte beachten Sie, daß die Tabelle PAYBYED auf der Tabelle PAYLEVEL basiert. Daher wird auf die Tabelle PAYLEVEL in der gesamten Anweisung tatsächlich zweimal zugegriffen. Bei beiden Zugriffen wird dieselbe Gruppe von Zeilen zur Auswertung der Abfrage verwendet.

Das Endergebnis lautet folgendermaßen:

```
EMPNO  EDLEVEL YEAR_OF_HIRE TOTAL_PAY     5
------ ------- ------------ ------------- ---------
000210      17         1979      20132.00  25896.50
```

Korrelationsnamen

Ein *Korrelationsname* ist eine Kennung, mit der die einzelnen Verwendungsarten eines Objekts unterschieden werden. Ein Korrelationsname kann in der Klausel FROM einer Abfrage und in der ersten Klausel einer Anweisung UPDATE oder DELETE definiert werden. Ihm kann eine Tabelle, eine Sicht oder ein verschachtelter Tabellenausdruck zugeordnet sein, jedoch nur in dem definierten Kontext.

Beispiel: Die Klausel FROM STAFF S, ORG O legt S und O als Korrelationsnamen für STAFF bzw. ORG fest.

```
SELECT NAME, DEPTNAME
   FROM STAFF S, ORG O
   WHERE O.MANAGER = S.ID
```

Nachdem Sie einen Korrelationsnamen definiert haben, können Sie zur Qualifizierung des Objekts *ausschließlich* den Korrelationsnamen verwenden. Im vorgenannten Beispiel würde beispielsweise die Angabe ORG.MANAGER= STAFF.ID zum Fehlschlagen der Anweisung führen.

Ein Korrelationsname kann auch als Kurzname für den Verweis auf ein Datenbankobjekt verwendet werden. (Der Buchstabe S kann schneller eingegeben werden als der Name STAFF.)

Durch den Einsatz von Korrelationsnamen können Sie **Duplikate** eines Objekts erstellen. Dies ist nützlich, wenn Sie Einträge einer Tabelle mit der Tabelle selbst vergleichen müssen. Im folgenden Beispiel wird die Tabelle EMPLOYEE mit einem weiteren Exemplar dieser Tabelle verglichen, um die Abteilungsleiter für alle Mitarbeiter zu ermitteln. Es werden die Namen aller Mitarbeiter mit einer anderen Tätigkeitsbezeichnung als DESIGNER sowie der entsprechende Abteilungsleiter und die Abteilungsnummer angezeigt.

```
SELECT E2.FIRSTNME, E2.LASTNAME, E2.JOB, E1.FIRSTNME AS MGR_FIRSTNAME,
                    E1.LASTNAME AS MGR_LASTNAME, E1.WORKDEPT
  FROM EMPLOYEE E1, EMPLOYEE E2
  WHERE E1.WORKDEPT = E2.WORKDEPT
    AND E1.JOB = 'MANAGER'
    AND E2.JOB <> 'MANAGER'
    AND E2.JOB <> 'DESIGNER'
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
FIRSTNME     LASTNAME   JOB      MGR_FIRSTNAME MGR_LASTNAME    WORKDEPT
------------ ---------- -------- ------------- --------------- --------
DOLORES      QUINTANA   ANALYST  SALLY         KWAN            C01
HEATHER      NICHOLLS   ANALYST  SALLY         KWAN            C01
JAMES        JEFFERSON  CLERK    EVA           PULASKI         D21
SALVATORE    MARINO     CLERK    EVA           PULASKI         D21
DANIEL       SMITH      CLERK    EVA           PULASKI         D21
SYBIL        JOHNSON    CLERK    EVA           PULASKI         D21
MARIA        PEREZ      CLERK    EVA           PULASKI         D21
ETHEL        SCHNEIDER  OPERATOR EILEEN        HENDERSON       E11
JOHN         PARKER     OPERATOR EILEEN        HENDERSON       E11
PHILIP       SMITH      OPERATOR EILEEN        HENDERSON       E11
MAUDE        SETRIGHT   OPERATOR EILEEN        HENDERSON       E11
RAMLAL       MEHTA      FIELDREP THEODORE      SPENSER         E21
WING         LEE        FIELDREP THEODORE      SPENSER         E21
JASON        GOUNOT     FIELDREP THEODORE      SPENSER         E21
```

Unterabfragen mit Korrelationsbezug

Eine Unterabfrage, die sich auf alle zuvor angegebenen Tabellen beziehen kann, wird als *Unterabfrage mit Korrelationsbezug* bezeichnet. Man spricht in diesem Zusammenhang auch davon, daß die Unterabfrage einen *Korrelationsbezug* auf eine Tabelle in der Hauptabfrage hat.

Im folgenden Beispiel wird eine Unterabfrage ohne Korrelationsbezug verwendet, um die Namen und Personalnummern der Mitarbeiter in der Abteilung 'A00' aufzulisten, deren Gehalt über dem Durchschnittsgehalt in dieser Abteilung liegt:

```
SELECT EMPNO, LASTNAME
   FROM EMPLOYEE
   WHERE WORKDEPT = 'A00'
     AND SALARY > (SELECT AVG(SALARY)
                   FROM EMPLOYEE
                   WHERE WORKDEPT = 'A00')
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
EMPNO  LASTNAME
------ ---------------
000010 HAAS
000110 LUCCHESSI
```

Wenn für jede Abteilung das Durchschnittsgehalt ermittelt werden soll, muß die Unterabfrage für jede Abteilung einmal separat ausgewertet werden. Dies können Sie durch die Korrelationsfähigkeit von SQL erreichen. Sie ermöglicht das Schreiben einer Unterabfrage, die wiederholt ausgeführt wird, nämlich einmal für jede Zeile in der Tabelle, die in der übergeordneten Abfrage angegeben ist.

Im folgenden Beispiel wird eine Unterabfrage mit Korrelationsbezug verwendet, um alle Mitarbeiter aufzulisten, deren Gehalt über dem Durchschnittsgehalt ihrer Abteilung liegt:

```
SELECT E1.EMPNO, E1.LASTNAME, E1.WORKDEPT
   FROM EMPLOYEE E1
   WHERE SALARY > (SELECT AVG(SALARY)
                   FROM EMPLOYEE E2
                   WHERE E2.WORKDEPT = E1.WORKDEPT)
   ORDER BY E1.WORKDEPT
```

In dieser Abfrage wird die Unterabfrage einmal pro Abteilung ausgewertet. Das Ergebnis lautet:

```
EMPNO  LASTNAME        WORKDEPT
------ --------------- --------
000010 HAAS            A00
000110 LUCCHESSI       A00
000030 KWAN            C01
000060 STERN           D11
000150 ADAMSON         D11
000170 YOSHIMURA       D11
000200 BROWN           D11
000220 LUTZ            D11
000070 PULASKI         D21
000240 MARINO          D21
000270 PEREZ           D21
000090 HENDERSON       E11
```

```
000280 SCHNEIDER       E11
000100 SPENSER         E21
000330 LEE             E21
000340 GOUNOT          E21
```

Beim Schreiben einer Abfrage, die eine Unterabfrage mit Korrelationsbezug enthält, verwenden Sie dasselbe Basisformat wie bei einer herkömmlichen übergeordneten Abfrage mit Unterabfrage. Hierbei geben Sie jedoch in der Klausel FROM der übergeordneten Abfrage direkt nach dem Tabellennamen einen Korrelationsnamen an. Die Unterabfrage kann dann Spaltenbezüge enthalten, die durch den Korrelationsnamen qualifiziert werden. Wenn beispielsweise E1 ein Korrelationsname ist, steht E1.WORKDEPT für den Wert WORKDEPT der aktuellen Zeile der Tabelle in der übergeordneten Abfrage. Die Unterabfrage wird konzeptionsgemäß für jede Zeile der Tabelle in der übergeordneten Abfrage erneut ausgewertet.

Durch die Verwendung einer Unterabfrage mit Korrelationsbezug können Sie das System für sich arbeiten lassen und den Umfang des Codes reduzieren, den Sie in ihrer Anwendung schreiben müssen.

Korrelationsbezüge ohne Qualifikationsmerkmal sind in DB2 zulässig. Die Tabelle EMPLOYEE enthält beispielsweise eine Spalte namens LASTNAME, die Tabelle SALES jedoch enthält eine Spalte namens SALES_PERSON und keine Spalte mit dem Namen LASTNAME.

```
SELECT LASTNAME, FIRSTNME, COMM
   FROM EMPLOYEE
   WHERE 3 > (SELECT AVG(SALES)
                FROM SALES
                WHERE LASTNAME = SALES_PERSON)
```

In diesem Beispiel überprüft das System die am weitesten untergeordnete Klausel FROM im Hinblick auf eine Spalte LASTNAME. Da keine gefunden wird, überprüft es anschließend die in der Hierarchie nächsthöhere Klausel FROM (die in diesem Fall die übergeordnete Klausel FROM ist). Auch wenn Korrelationsbezüge mit Qualifikationsmerkmal nicht immer erforderlich sind, empfiehlt sich doch ihre Verwendung, weil auf diese Weise die Lesbarkeit der Abfrage verbessert und sichergestellt wird, daß das beabsichtigte Ergebnis erzielt wird.

Implementieren einer Unterabfrage mit Korrelationsbezug

In welchen Situationen sollte eine Unterabfrage mit Korrelationsbezug verwendet werden? Manchmal ist die Verwendung einer Spaltenfunktion ein Anhaltspunkt.

Angenommen, Sie wollen die Mitarbeiter auflisten, deren Bildungsstufe über dem entsprechenden Durchschnittswert ihrer Abteilung liegt.

Zunächst müssen Sie die Elemente der SELECT-Liste bestimmen. Die Aufgabenstellung lautet „Mitarbeiter auflisten". Dies impliziert, daß die Spalte LASTNAME aus der Tabelle EMPLOYEE ausreichen sollte, um die Mitarbeiter eindeutig zu anzugeben. Die Aufgabenstellung beinhaltet außerdem die Bildungsstufe (EDLEVEL) und die Abteilungen der Mitarbeiter (WORKDEPT) als Bedingungen. Die Aufgabenstellung erfordert zwar nicht explizit das Anzeigen von Spalten, aber durch deren Aufnahme in die SELECT-Liste wird die Lösung anschaulicher. Nun kann ein Teil der Abfrage konstruiert werden:

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE
```

Anschließend wird eine Suchbedingung (Klausel WHERE) benötigt. Die Aufgabenstellung lautet „...deren Bildungsstufe über dem entsprechenden Durchschnittswert ihrer Abteilung liegt". Dies bedeutet, daß für jeden Mitarbeiter in der Tabelle die durchschnittliche Bildungsstufe in seiner Abteilung berechnet werden muß. Diese Anweisung entspricht der Beschreibung einer Unterabfrage mit Korrelationsbezug. Für jede Zeile wird ein bestimmtes unbekanntes Merkmal (die durchschnittliche Bildungsstufe in der Abteilung des aktuellen Mitarbeiters) berechnet. Für die Tabelle EMPLOYEE wird ein Korrelationsname benötigt:

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE E1
```

Die erforderliche Unterabfrage ist einfach. Sie berechnet die durchschnittliche Bildungsstufe für jede Abteilung. Die vollständige SQL-Anweisung lautet folgendermaßen:

```
SELECT LASTNAME, WORKDEPT, EDLEVEL
   FROM EMPLOYEE E1
   WHERE EDLEVEL > (SELECT AVG(EDLEVEL)
                    FROM EMPLOYEE  E2
                    WHERE E2.WORKDEPT = E1.WORKDEPT)
```

Das Ergebnis lautet:

```
LASTNAME        WORKDEPT EDLEVEL
--------------- -------- -------
HAAS            A00           18
KWAN            C01           20
PULASKI         D21           16
HENDERSON       E11           16
LUCCHESSI       A00           19
PIANKA          D11           17
SCOUTTEN        D11           17
JONES           D11           17
LUTZ            D11           18
MARINO          D21           17
JOHNSON         D21           16
SCHNEIDER       E11           17
MEHTA           E21           16
GOUNOT          E21           16
```

Angenommen, Sie wollen anstelle der Abteilungsnummer eines Mitarbeiters den Namen der Abteilung auflisten. Die benötigten Informationen (DEPTNAME) befinden sich in einer separaten Tabelle (DEPARTMENT). Die übergeordnete Abfrage, die eine Korrelationsvariable definiert, kann auch eine Verknüpfungsabfrage sein (ausführliche Informationen finden Sie unter „Auswählen von Daten aus mehreren Tabellen” auf Seite 32).

Wenn Sie in einer übergeordneten Abfrage Verknüpfungen verwenden, müssen Sie die zu verknüpfenden Tabellen in der Klausel FROM auflisten und den Korrelationsnamen neben den entsprechenden Tabellennamen stellen.

Um die Abfrage so zu modifizieren, daß anstelle der Abteilungsnummer der Name der Abteilung aufgelistet wird, ersetzen Sie in der SELECT-Liste die Angabe WORKDEPT durch DEPTNAME. Die Klausel FROM muß jetzt außerdem die Tabelle DEPARTMENT enthalten, und die Klausel WHERE muß die geeignete Verknüpfungsbedingung ausdrücken.

Die modifizierte Abfrage lautet wie folgt:

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
   FROM EMPLOYEE E1, DEPARTMENT
   WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
   AND EDLEVEL > (SELECT AVG(EDLEVEL)
                  FROM EMPLOYEE E2
                  WHERE E2.WORKDEPT = E1.WORKDEPT)
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
LASTNAME        DEPTNAME                       EDLEVEL
--------------- ------------------------------ -------
HAAS            SPIFFY COMPUTER SERVICE DIV.        18
LUCCHESSI       SPIFFY COMPUTER SERVICE DIV.        19
KWAN            INFORMATION CENTER                  20
PIANKA          MANUFACTURING SYSTEMS               17
SCOUTTEN        MANUFACTURING SYSTEMS               17
JONES           MANUFACTURING SYSTEMS               17
LUTZ            MANUFACTURING SYSTEMS               18
PULASKI         ADMINISTRATION SYSTEMS              16
MARINO          ADMINISTRATION SYSTEMS              17
JOHNSON         ADMINISTRATION SYSTEMS              16
HENDERSON       OPERATIONS                          16
SCHNEIDER       OPERATIONS                          17
MEHTA           SOFTWARE SUPPORT                    16
GOUNOT          SOFTWARE SUPPORT                    16
```

Aus den zuvor dargestellten Beispielen geht hervor, daß der in einer Unterabfrage verwendete Korrelationsname in der Klausel FROM einer Abfrage definiert werden muß, die die Unterabfrage mit Korrelationsbezug enthält. Dieser Einschluß kann jedoch unterschiedliche Verschachtelungsebenen mit sich bringen.

Angenommen, einige Abteilungen umfassen nur wenige Mitarbeiter und weisen daher eine unter Umständen irreführende durchschnittliche Bildungsstufe auf. In einem solchen Fall könnten Sie beschließen, nur Abteilungen mit mindestens fünf Mitarbeitern zu berücksichtigen, damit ein aussagekräftiger Wert für den Mitarbeitervergleich hinsichtlich der durchschnittlichen Bildungsstufe zur Verfügung steht. Daher müssen nun die Mitarbeiter mit einer höheren als der durchschnittlichen Bildungsstufe in ihrer Abteilung aufgelistet werden, wobei nur Abteilungen mit mindestens fünf Mitarbeitern zu berücksichtigen sind.

Die Aufgabenstellung impliziert eine weitere Unterabfrage, da für jeden Mitarbeiter in der übergeordneten Abfrage die Gesamtzahl der Mitarbeiter in seiner Abteilung ermittelt werden muß:

```
SELECT COUNT(*)
   FROM EMPLOYEE E3
   WHERE E3.WORKDEPT = E1.WORKDEPT
```

Nur dann, wenn die Anzahl größer-gleich 5 ist, soll ein Durchschnittswert berechnet werden:

```
SELECT AVG(EDLEVEL)
   FROM EMPLOYEE E2
   WHERE E2.WORKDEPT = E1.WORKDEPT
   AND 5 <= (SELECT COUNT(*)
                FROM EMPLOYEE  E3
                WHERE E3.WORKDEPT = E1.WORKDEPT)
```

Abschließend werden nur die Mitarbeiter in das Ergebnis aufgenommen, deren Bildungsstufe über dem Durchschnitt ihrer Abteilung liegt:

```
SELECT LASTNAME, DEPTNAME, EDLEVEL
   FROM EMPLOYEE E1, DEPARTMENT
   WHERE E1.WORKDEPT = DEPARTMENT.DEPTNO
   AND EDLEVEL >
   (SELECT AVG(EDLEVEL)
       FROM EMPLOYEE E2
       WHERE E2.WORKDEPT = E1.WORKDEPT
       AND 5 <=
       (SELECT COUNT(*)
           FROM EMPLOYEE E3
           WHERE E3.WORKDEPT = E1.WORKDEPT))
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
LASTNAME        DEPTNAME                    EDLEVEL
--------------- --------------------------- -------
PIANKA          MANUFACTURING SYSTEMS            17
SCOUTTEN        MANUFACTURING SYSTEMS            17
JONES           MANUFACTURING SYSTEMS            17
LUTZ            MANUFACTURING SYSTEMS            18
PULASKI         ADMINISTRATION SYSTEMS           16
MARINO          ADMINISTRATION SYSTEMS           17
JOHNSON         ADMINISTRATION SYSTEMS           16
HENDERSON       OPERATIONS                       16
SCHNEIDER       OPERATIONS                       17
```

Kapitel 6. Verwenden von Operatoren und Prädikaten in Abfragen

In DB2 Universal Database können Sie Abfragen mit unterschiedlichen *Gruppenoperatoren* kombinieren und mit Hilfe von *Prädikaten QUANTIFIED* komplexe bedingte Anweisungen konstruieren.

Dieses Kapitel erläutert, wie Sie

- unterschiedliche Tabellen durch die Gruppenoperatoren UNION, EXCEPT und INTERSECT kombinieren,
- komplexe Bedingungen für Abfragen mit Prädikaten QUANTIFIED konstruieren. Die Prädikate BASIC werden unter „Auswählen von Zeilen” auf Seite 25 kurz erläutert.

Kombinieren von Abfragen durch Gruppenoperatoren

Die Gruppenoperatoren UNION, EXCEPT und INTERSECT ermöglichen die Kombination von zwei oder mehr übergeordneten Abfragen in einer einzigen Abfrage. Jede durch diese Gruppenoperatoren verbundene Abfrage wird ausgeführt, und die einzelnen Ergebnisse werden kombiniert. Jeder Operator erzeugt ein anderes Ergebnis.

Operator UNION

Der Operator UNION leitet eine Ergebnistabelle ab, indem zwei andere Ergebnistabellen (z. B. TABLE1 und TABLE2) kombiniert und alle in den Tabellen doppelt vorhandenen Zeilen eliminiert werden. Wird UNION zusammen mit ALL (d. h. UNION ALL) verwendet, werden doppelt vorhandene Zeilen nicht eliminiert. In beiden Fälle stammt jede Zeile der abgeleiteten Tabelle entweder aus der Tabelle TABLE1 oder aus der Tabelle TABLE2.

Im folgenden Beispiel für den Operator UNION gibt die Abfrage die Namen aller Personen zurück, deren Gehalt größer als $ 21.000 ist oder die eine Führungsposition innehaben und vor weniger als 8 Jahren eingestellt wurden:

```
1
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
UNION

2
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
ORDER BY ID
```

Die Ergebnisse der einzelnen Abfragen lauten:

1

```
ID     NAME
------ ---------
   140 Fraye
   160 Molinare
   260 Jones
```

2

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
   100 Plotz
   140 Fraye
   160 Molinare
   240 Daniels
```

Der Datenbankmanager kombiniert die Ergebnisse beider Abfragen, eliminiert gleiche Werte und gibt das Endergebnis aufsteigend sortiert zurück.

```
ID     NAME
------ ---------
    10 Sanders
    30 Marenghi
   100 Plotz
   140 Fraye
   160 Molinare
   240 Daniels
   260 Jones
```

Wenn Sie die Klausel ORDER BY in einer Abfrage mit einem Gruppenoperator verwenden, muß die Klausel nach der letzten Abfrage eingegeben werden. Das System wendet die Sortierung auf die kombinierte Antwortgruppe an.

Wenn sich die Spaltennamen in den beiden Tabellen unterscheiden, enthält die kombinierte Ergebnistabelle keine Namen für die entsprechenden Spalten. Die Spalten werden statt dessen in der Reihenfolge ihres Auftretens numeriert. Wenn Sie daher die Ergebnistabelle sortieren wollen, müssen Sie in der Klausel ORDER BY die Spaltennummer angeben.

Operator EXCEPT

Der Operator EXCEPT leitet eine Ergebnistabelle ab, indem alle Zeilen aufgenommen werden, die in der Tabelle TABLE 1, aber nicht in der Tabelle TABLE2 enthalten sind, und alle doppelt vorhandenen Zeilen eliminiert werden. Wird EXCEPT zusammen mit ALL verwendet (EXCEPT ALL), werden die doppelt vorhandenen Zeilen nicht eliminiert.

Im folgenden Beispiel für den Operator EXCEPT gibt die Abfrage die Namen aller Personen zurück, die mehr als $ 21.000 verdienen, aber keine Führungsposition innehaben und vor 8 oder mehr Jahren eingestellt wurden.

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
EXCEPT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

Die Ergebnisse der einzelnen Abfragen sind im Abschnitt über den Operator UNION aufgeführt. Die vorstehende Anweisung erzeugt das folgende Ergebnis:

```
ID     NAME
------ ---------
   260 Jones
```

Operator INTERSECT

Der Operator INTERSECT leitet eine Ergebnistabelle ab, indem nur Zeilen aufgenommen werden, die sowohl in der Tabelle TABLE1 als auch in der Tabelle TABLE2 enthalten sind, und alle doppelt vorhandenen Zeilen eliminiert werden. Wird INTERSECT zusammen mit ALL verwendet (INTERSECT ALL), werden die doppelt vorhandenen Zeilen nicht eliminiert.

Im folgenden Beispiel für den Operator INTERSECT gibt die Abfrage den Namen und die ID aller Mitarbeiter zurück, deren Gehalt mehr als $ 21.000 beträgt, die eine Führungsposition innehaben und die vor weniger als 8 Jahren eingestellt wurden.

```
SELECT ID, NAME FROM STAFF WHERE SALARY > 21000
INTERSECT
SELECT ID, NAME FROM STAFF WHERE JOB='Mgr' AND YEARS < 8
```

Die Ergebnisse der einzelnen Abfragen sind im Abschnitt über den Operator UNION aufgeführt.

Werden die beiden Abfragen durch den Operator INTERSECT kombiniert, lautet die Ausgabe:

```
ID     NAME
------ ---------
   140 Fraye
   160 Molinare
```

Bei der Verwendung der Operatoren UNION, EXCEPT und INTERSECT sollte folgendes beachtet werden:

- Alle entsprechenden Elemente für die Operatoren in der SELECT-Liste der Abfragen müssen kompatibel sein. Weitere Informationen können Sie der Tabelle zur Kompatibilität von Datentypen im Handbuch *SQL Reference* entnehmen.
- Eine Klausel ORDER BY muß, wenn sie verwendet wird, nach der letzten Abfrage mit einem Gruppenoperator stehen. Der Spaltenname kann nur dann in der Klausel ORDER BY verwendet werden, wenn der Spaltenname für die sich entsprechenden Elemente in der SELECT-Liste der Abfragen für alle Operatoren identisch ist.
- Operationen zwischen Spalten mit demselben Datentyp und derselben Länge erzeugen eine Spalte dieses Typs und dieser Länge. Informationen zu den Ergebnissen der Gruppenoperatoren UNION, EXCEPT und INTERSECT finden Sie im Abschnitt mit den Regeln für Ergebnisdatentypen im Handbuch *SQL Reference*.

Prädikate

Mit Hilfe von Prädikaten können Sie Bedingungen konstruieren, damit nur die Zeilen verarbeitet werden, die diese Bedingungen erfüllen. Die Prädikate BASIC werden unter „Auswählen von Zeilen" auf Seite 25 erläutert. Der vorliegende Abschnitt behandelt die Prädikate IN, BETWEEN, LIKE, EXISTS und Prädikate QUANTIFIED.

Verwenden des Prädikats IN

Mit dem Prädikat IN können Sie einen Wert mit mehreren anderen Werten vergleichen. Beispiel:

```
SELECT NAME
   FROM STAFF
   WHERE DEPT IN (20, 15)
```

Dieses Beispiel entspricht der folgenden Anweisung:

```
SELECT NAME
   FROM STAFF
   WHERE DEPT = 20  OR DEPT = 15
```

Die Operatoren IN und NOT IN können Sie verwenden, wenn eine Unterabfrage eine Gruppe von Werten zurückgibt. Beispielsweise listet die folgende Abfrage die Familiennamen der Mitarbeiter auf, die für die Projekte MA2100 und OP2012 verantwortlich sind:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE EMPNO IN
      (SELECT RESPEMP
          FROM PROJECT
          WHERE PROJNO = 'MA2100'
          OR PROJNO = 'OP2012')
```

Die Unterabfrage wird einmal ausgewertet, und die resultierende Liste wird direkt in die übergeordnete Abfrage eingesetzt. Wenn die oben dargestellte Unterabfrage beispielsweise die Personalnummern 10 und 330 auswählt, wird die übergeordnete Abfrage so ausgewertet, als ob sie die folgende Klausel WHERE enthielte:

```
WHERE EMPNO IN (10, 330)
```

Die durch die Unterabfrage zurückgegebene Liste von Werten kann keinen, einen oder mehrere Werte enthalten.

Verwenden des Prädikats BETWEEN

Das Prädikat BETWEEN vergleicht einen einzelnen Wert mit einem (im Prädikat BETWEEN benannten) inklusiven Wertebereich.

Das folgende Beispiel sucht nach den Namen der Mitarbeiter, die zwischen $ 10.000 und $ 20.000 verdienen:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY BETWEEN 10000 AND 20000
```

Dies entspricht der folgenden Anweisung:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY >= 10000 AND SALARY <= 20000
```

Das nächste Beispiel sucht nach den Namen der Mitarbeiter, die weniger als $ 10.000 oder mehr als $ 20.000 verdienen:

```
SELECT LASTNAME
   FROM EMPLOYEE
   WHERE SALARY NOT BETWEEN 10000 AND 20000
```

Verwenden des Prädikats LIKE

Mit dem Prädikat LIKE können Sie nach Zeichenfolgen suchen, die bestimmte Muster aufweisen. Das Muster wird durch Prozentzeichen und Unterstreichungszeichen angegeben.

- Das Unterstreichungszeichen (_) steht für ein Einzelzeichen.
- Das Prozentzeichen (%) steht für eine Zeichenfolge aus keinem oder mehreren Zeichen.
- Jedes andere Zeichen steht für sich selbst.

Im folgenden Beispiel werden die Namen der Mitarbeiter mit einer Länge von sieben Zeichen und dem Anfangsbuchstaben S ausgewählt:

```
SELECT NAME
   FROM STAFF
   WHERE NAME LIKE 'S _ _ _ _ _ _'
```

Das nächste Beispiel wählt die Namen der Mitarbeiter aus, die nicht mit dem Buchstaben S beginnen:

```
SELECT NAME
   FROM STAFF
   WHERE NAME  NOT LIKE 'S%'
```

Verwenden des Prädikats EXISTS

Mit einer Unterabfrage können Sie überprüfen, ob eine Zeile *vorhanden* ist, die eine bestimmte Bedingung erfüllt. In diesem Fall wird die Unterabfrage durch das Prädikat EXISTS oder NOT EXISTS mit der übergeordneten Abfrage verbunden. Wenn Sie eine Unterabfrage durch ein Prädikat EXISTS mit einer übergeordneten Abfrage verbinden, gibt die Unterabfrage keinen Wert zurück. Statt dessen ist das Prädikat EXISTS dann wahr (= zutreffend), wenn die Antwortgruppe der Unterabfrage eine oder mehrere Zeilen enthält, und falsch (= nicht zutreffend), wenn die Antwortgruppe keine Zeilen enthält.

Das Prädikat EXISTS wird häufig bei Unterabfragen mit Korrelationsbezug verwendet. Das folgende Beispiel listet die Abteilungen auf, für die in der Tabelle PROJECT gegenwärtig keine Einträge vorhanden sind:

```
SELECT DEPTNO, DEPTNAME
   FROM DEPARTMENT X
   WHERE NOT EXISTS
            (SELECT *
                FROM PROJECT
                WHERE DEPTNO = X.DEPTNO)
   ORDER BY DEPTNO
```

Sie können die Prädikate EXISTS und NOT EXISTS mit anderen Prädikaten verbinden, indem Sie die Operatoren AND und OR in der Klausel WHERE der übergeordneten Abfrage verwenden.

Prädikate QUANTIFIED

Ein Prädikate QUANTIFIED vergleicht einen Wert mit einer Gruppe von Werten. Wenn eine Gesamtauswahl mehr als einen Wert zurückgibt, müssen Sie die Vergleichsoperatoren in Ihrem Prädikat durch Anfügen des Suffixes ALL, ANY oder SOME modifizieren. Diese Suffixe legen fest, wie die zurückgegebene Gruppe von Werten im übergeordneten Prädikat behandelt werden soll. Der Vergleichsoperator > wird als Beispiel verwendet (die folgenden Anmerkungen gelten gleichfalls für die anderen Operatoren):

Ausdruck > ALL (Gesamtauswahl)
Das Prädikat ist wahr, wenn der Ausdruck größer als alle Einzelwerte ist, die durch die Gesamtauswahl zurückgegeben werden. Gibt die Gesamtauswahl keine Werte zurück, ist das Prädikat wahr. Das Ergebnis ist falsch, wenn die angegebene Beziehung für mindestens einen Wert falsch ist. Bitte beachten Sie, daß das Prädikat QUANTIFIED <>ALL äquivalent zum Prädikat NOT IN ist.

Das folgende Beispiel verwendet eine Unterabfrage und einen Vergleich > ALL, um den Namen und den Beruf aller Mitarbeiter zu ermitteln, die mehr als alle Führungskräfte verdienen:

```
SELECT LASTNAME, JOB
   FROM EMPLOYEE
   WHERE SALARY > ALL
   (SELECT SALARY
       FROM EMPLOYEE
       WHERE JOB='MANAGER')
```

Ausdruck > ANY (Gesamtauswahl)
Das Prädikat ist wahr, wenn der Ausdruck größer als mindestens einer der Werte ist, die durch die Gesamtauswahl zurückgegeben werden. Gibt die Gesamtauswahl keine Werte zurück, ist das Prädikat falsch. Bitte beachten Sie, daß der Vergleichsoperator =ANY äquivalent zum Prädikat IN ist.

Ausdruck > SOME (Gesamtauswahl)
SOME ist gleichbedeutend mit ANY.

Weitere Informationen zu Prädikaten und Operatoren finden Sie im Handbuch *SQL Reference*.

Kapitel 7. SQL für Fortgeschrittene

Dieses Kapitel behandelt einige Funktionen von DB2 Universal Database, mit denen Sie Abfragen effizienter konzipieren und dabei an Ihre Bedürfnisse anpassen können. In den Abschnitten im vorliegenden Kapitel wird davon ausgegangen, daß Sie die vorangegangenen Informationen grundlegend verstanden haben.

Die folgenden Themen werden behandelt:

- Durchsetzen von Geschäftsregeln mit Hilfe von Integritätsbedingungen und Auslösern
- Verknüpfungen
- Abfragen mit ROLLUP und CUBE und Rekursive Abfragen
- OLAP-Funktionen

Durchsetzen von Geschäftsregeln mit Hilfe von Integritätsbedingungen und Auslösern

In der Geschäftswelt gibt es bestimmte Regeln, deren ständige Beachtung sichergestellt sein muß. Ein mit einem Projekt befaßter Mitarbeiter muß beispielsweise auf der Gehaltsliste stehen. Denkbar sind auch bestimmte Ereignisse, die automatisch eintreten sollen. Dies könnte beispielsweise bei einem Verkaufsmitarbeiter der Fall sein, dessen Provision erhöht werden soll, wenn er einen Verkauf abschließt.

DB2 Universal Database enthält zu diesem Zweck eine Reihe nützlicher Methoden:

- *Eindeutige Integritätsbedingungen* verhindern gleiche Werte in einer oder mehreren Spalten einer Tabelle.
- *Referentielle Integritätsbedingungen* stellen sicher, daß die Daten in den angegebenen Tabellen konsistent bleiben.
- *Prüfungen auf Integritätsbedingung in Tabellen* sind Regeln, die Einschränkungen hinsichtlich der in einer Spalte zulässigen Werte festlegen. Einfügungen und Aktualisierungen schlagen fehl, wenn ein Wert, der einer Spalte zugeordnet wird, die Prüfung(en) auf Integritätsbedingung für diese Spalte nicht besteht.
- *Auslöser* definieren eine Gruppe von Aktionen, die durch eine Lösch-, Einfüge- oder Aktualisierungsoperation für eine bestimmte Tabelle ausgeführt (= ausgelöst) werden. Auslöser können zum Schreiben in andere Tabellen, zum Modifizieren von Eingabewerten und zum Absetzen von Alerts eingesetzt werden.

Der erste Abschnitt bietet Ihnen eine konzeptionelle Übersicht über Schlüssel. Anschließende werden die referentielle Integrität, Integritätsbedingungen und Auslöser durch Beispiele und Abbildungen erläutert.

Schlüssel

Ein *Schlüssel* ist eine Spaltengruppe, mit der Sie eine oder mehrere bestimmte Zeilen angeben oder auf diese zugreifen können.

Ein Schlüssel, der aus mehreren Spalten besteht, wird als *zusammengesetzter Schlüssel* bezeichnet. In einer Tabelle mit einem zusammengesetzten Schlüssel muß die Reihenfolge der Spalten im zusammengesetzten Schlüssel nicht zwangsläufig der Reihenfolge dieser Spalten in der Tabelle entsprechen.

Eindeutige Schlüssel

Ein *eindeutiger Schlüssel* ist als eine Spalte (oder Spaltengruppe) definiert, in der es keine identischen Werte gibt. Die Spalten eines eindeutigen Schlüssels können keine Nullwerte enthalten. Die Integritätsbedingung wird durch den Datenbankmanager während der Ausführung von Anweisungen INSERT und UPDATE durchgesetzt. Eine Tabelle kann mehrere eindeutige Schlüssel enthalten. Eindeutige Schlüssel sind wahlfrei und können in einer Anweisung CREATE TABLE oder ALTER TABLE definiert werden.

Primärschlüssel

Ein *Primärschlüssel* ist ein eindeutiger Schlüssel, der Bestandteil der Tabellendefinition ist. Eine Tabelle kann nur einen Primärschlüssel enthalten. Die Spalten eines Primärschlüssels können keine Nullwerte enthalten. Primärschlüssel sind wahlfrei und können in einer Anweisung CREATE TABLE oder ALTER TABLE definiert werden.

Fremdschlüssel

Ein *Fremdschlüssel* wird in der Definition einer referentiellen Integritätsbedingung angegeben. Eine Tabelle kann keinen oder auch mehrere Fremdschlüssel enthalten. Der Wert des zusammengesetzten Fremdschlüssels ist Null, wenn eine der Wertkomponenten Null ist. Fremdschlüssel sind wahlfrei und können in einer Anweisung CREATE TABLE oder ALTER TABLE definiert werden.

Eindeutige Integritätsbedingungen

Eine eindeutige Integritätsbedingung stellt sicher, daß die Werte eines Schlüssels innerhalb einer Tabelle eindeutig sind. Eindeutige Integritätsbedingungen sind wahlfrei. Sie können in einer Anweisungen CREATE TABLE oder ALTER TABLE durch Angabe der Klausel PRIMARY KEY oder der Klausel UNIQUE definiert werden. Beispielsweise können Sie eine eindeutige Integritätsbedingung für die Spalte mit der Personalnummer in einer Tabelle definieren, um sicherzustellen, daß jeder Mitarbeiter eine eindeutige Personalnummer hat.

Referentielle Integritätsbedingungen

Durch die Definition von eindeutigen Integritätsbedingungen und Fremdschlüsseln können Sie Beziehungen zwischen Tabellen definieren und bestimmte Geschäftsregeln konsequent durchsetzen. Die Kombination aus Integritätsbedingungen über eindeutige Schlüssel und Fremdschlüssel wird gemeinhin als referentielle Integritätsbedingung bezeichnet. Eine eindeutige Integritätsbedingung, auf die in einem Fremdschlüssel verwiesen wird, wird als *übergeordneter Schlüssel* bezeichnet. Ein Fremdschlüssel verweist auf einen spezifischen übergeordneten Schlüssel bzw. ist mit ihm verbunden. Beispiel: Eine Regel besagt, daß jeder Mitarbeiter (Tabelle EMPLOYEE) einer existierenden Abteilung (Tabelle DEPARTMENT) zugeordnet sein muß. Daher wird die Abteilungsnummer in der Tabelle EMPLOYEE als Fremdschlüssel und in der Tabelle DEPARTMENT als Primärschlüssel definiert. Die folgende Abbildung veranschaulicht das Konzept der referentielle Integritätsbedingung.

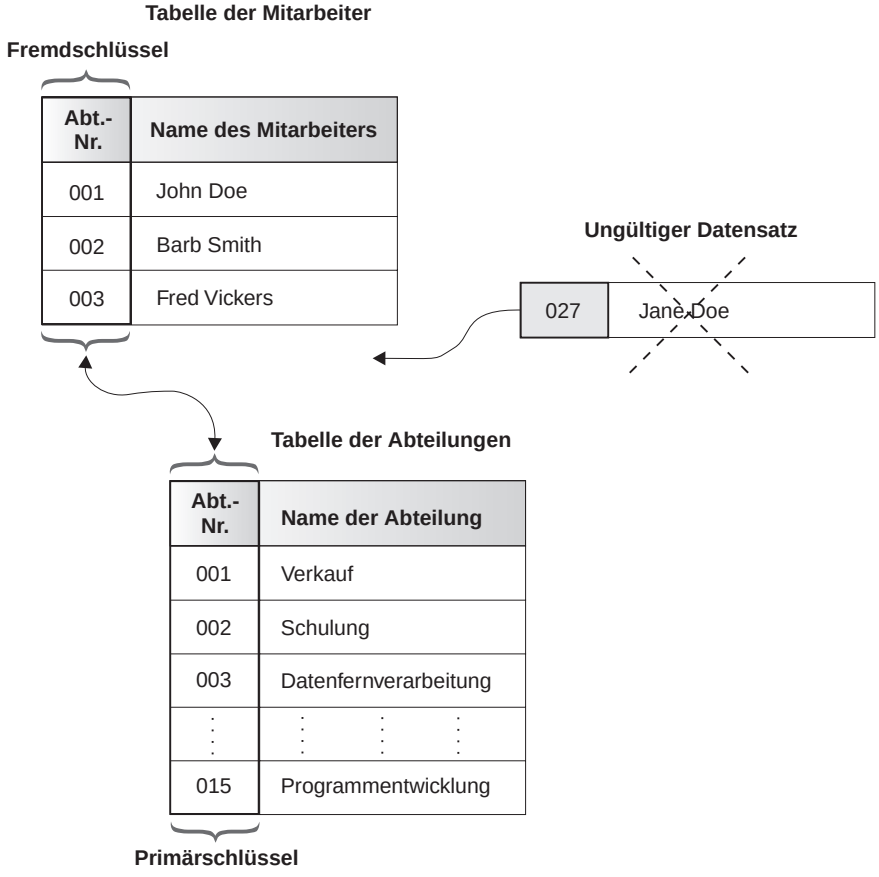


Abbildung 4. Integritätsbedingungen über Primär- und Fremdschlüssel definieren Beziehungen und schützen Daten

Prüfungen auf Integritätsbedingung in Tabellen

Eine *Prüfung auf Integritätsbedingung in Tabellen* gibt Bedingungen an, die für jede Zeile einer Tabelle ausgewertet werden. Sie können Prüfungen auf Integritätsbedingung für einzelne Spalten angeben. Zum Hinzufügen von Prüfungen auf Integritätsbedingung verwenden Sie eine Anweisung CREATE oder ALTER TABLE.

Die folgende Anweisung erstellt eine Tabelle mit den folgenden Integritätsbedingungen:

- Die Abteilungsnummern müssen im Bereich von 10 bis 100 liegen.
- Die Tätigkeit eines Mitarbeiters kann eine der folgenden sein: 'Sales' (Verkauf), 'Mgr' (Leitung) oder 'Clerk' (Sachbearbeiter).
- Jeder Mitarbeiter, der vor 1986 eingestellt wurde, muß mehr als $ 40.500 verdienen.

```
CREATE TABLE EMP
       (ID           SMALLINT NOT NULL,
        NAME         VARCHAR(9),
        DEPT         SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
        JOB          CHAR(5)   CHECK (JOB IN ('Sales', 'Mgr', 'Clerk')),
        HIREDATE     DATE,
        SALARY       DECIMAL(7,2),
        COMM         DECIMAL(7,2),
        PRIMARY KEY (ID),
        CONSTRAINT YEARSAL CHECK
                           (YEAR(HIREDATE) >= 1986 OR SALARY > 40500) )
```

Ein Integritätsbedingung wird nur dann verletzt, wenn die Bedingung als falsch ausgewertet wird. Wenn beispielsweise in einer eingefügten Zeile die Spalte DEPT einen Nullwert enthält, wird die Einfügeoperation ohne Fehler fortgesetzt, obwohl die Werte für DEPT, wie in der Integritätsbedingung definiert, zwischen 10 und 100 liegen sollen.

Die folgende Anweisung fügt eine Integritätsbedingung namens COMP zur Tabelle EMPLOYEE hinzu. Diese Integritätsbedingung gibt an, daß der Gesamtverdienst eines Mitarbeiters mehr als $ 15.000 betragen muß:

```
ALTER TABLE EMP
   ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
```

Die in der Tabelle vorhandenen Zeilen werden dahingehend geprüft, daß die neue Integritätsbedingung nicht verletzt wird. Diese Prüfung können Sie verzögern, indem Sie die Anweisung SET CONSTRAINTS folgendermaßen einsetzen:

```
SET CONSTRAINTS FOR EMP OFF
ALTER TABLE EMP ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
SET CONSTRAINTS FOR EMP IMMEDIATE CHECKED
```

Zunächst wird mit der Anweisung SET CONSTRAINTS die Prüfung auf Integritätsbedingung für die Tabelle verzögert. Anschließend können eine oder mehrere Integritätsbedingungen zur Tabelle hinzugefügt werden, ohne daß eine Prüfung erfolgt. Anschließend wird die Anweisung SET CONSTRAINTS erneut abgesetzt, um die Prüfung der Integritätsbedingungen wieder zu aktivieren und alle verzögerten Prüfungen der Integritätsbedingungen auszuführen.

Auslöser

Ein *Auslöser* definiert eine Gruppe von Aktionen. Auslöser werden durch Operationen aktiviert, die die Daten in einer angegebenen Basistabelle modifizieren.

Einige Einsatzmöglichkeiten für Auslöser:

- Ausführen der Gültigkeitsprüfung für Eingabedaten
- Automatisches Generieren eines Werts für eine neu eingefügte Zeile
- Lesen aus anderen Tabellen zu Querverweiszwecken
- Schreiben in andere Tabellen zu Prüfprotokollzwecken
- Unterstützen von Alerts durch E-Mail-Nachrichten

Die Verwendung von Auslösern ermöglicht eine schnellere Anwendungsentwicklung, die globale Durchsetzung von Geschäftsregeln und eine einfachere Verwaltung von Anwendungen und Daten.

DB2 Universal Database unterstützt verschiedene Arten von Auslösern. Auslöser können so definiert werden, daß sie entweder vor oder nach einer Operation DELETE, INSERT oder UPDATE aktiviert werden. Jeder Auslöser enthält eine Gruppe von SQL-Anweisungen, die als *ausgelöste Aktion* bezeichnet wird und eine wahlfreie Suchbedingung enthalten kann.

Nachauslöser können genauer definiert werden, um die ausgelöste Aktion entweder für jede Zeile oder einmal für die gesamte Anweisung auszuführen. *Vorauslöser* führen die ausgelöste Aktion immer für jede Zeile aus.

Durch die Verwendung eines Auslösers vor einer Anweisung INSERT, UPDATE oder DELETE können bestimmte Bedingungen überprüft werden, bevor eine auslösende Operation ausgeführt wird, oder die Eingabewerte geändert werden, bevor sie in der Tabelle gespeichert werden.

Mit einem Nachauslöser können Werte wie erforderlich weitergegeben oder andere Tasks ausgeführt werden, beispielsweise das Senden einer Nachricht, die als Teil der Auslöseroperation erforderlich ist.

Das folgende Beispiel veranschaulicht eine mögliche Verwendung von Vor- und Nachauslösern. Es wird von einer Anwendung ausgegangen, die Änderungen an Aktienkursen aufzeichnet und protokolliert. Die Datenbank enthält zwei Tabellen namens CURRENTQUOTE (für den aktuellen Kurs) und QUOTEHISTORY (für die Kursentwicklung), die wie folgt definiert sind:

```
CREATE TABLE CURRENTQUOTE
(SYMBOL VARCHAR(10),
 QUOTE DECIMAL(5,2),
 STATUS VARCHAR(9))

CREATE TABLE  QUOTEHISTORY
(SYMBOL VARCHAR(10),
 QUOTE DECIMAL(5,2),
 TIMESTAMP TIMESTAMP)
```

Wird die Spalte QUOTE der Tabelle CURRENTQUOTE mit einer Anweisung wie beispielsweise der folgenden aktiviert:

```
UPDATE CURRENTQUOTE
   SET QUOTE = 68.5
   WHERE SYMBOL = 'IBM'
```

sollte die Spalte STATUS der Tabelle CURRENTQUOTE aktualisiert werden und wiedergeben, ob die Aktie

- im Wert steigt ('Rising'),
- im laufenden Jahr einen neuen Höchstwert erreicht hat ('High'),
- im Wert fällt ('Dropping'),
- im laufenden Jahr einen neuen Tiefstwert erreicht hat ('Low'),
- im Wert stabil ist ('Steady').

Dies wird mit dem folgenden Vorauslöser erreicht:

1

```
CREATE TRIGGER STOCK_STATUS
   NO CASCADE BEFORE UPDATE OF QUOTE ON CURRENTQUOTE
   REFERENCING NEW AS NEWQUOTE OLD AS OLDQUOTE
   FOR EACH ROW MODE DB2SQL
```

2

```
 SET NEWQUOTE.STATUS =
```

3
```
    CASE
```
4
```
      WHEN NEWQUOTE.QUOTE >=
                (SELECT MAX(QUOTE)
                    FROM QUOTEHISTORY
                    WHERE SYMBOL = NEWQUOTE.SYMBOL
                      AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) )
      THEN 'High'
```
5
```
    WHEN NEWQUOTE.QUOTE <=
                (SELECT MIN(QUOTE)
                    FROM QUOTEHISTORY
                    WHERE SYMBOL = NEWQUOTE.SYMBOL
                    AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) )
    THEN 'Low'
```
6
```
    WHEN NEWQUOTE.QUOTE > OLDQUOTE.QUOTE
       THEN 'Rising'
    WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE
       THEN 'Dropping'
    WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE
       THEN 'Steady'
END
```

1 Dieser Codeblock definiert STOCK_STATUS als einen Auslöser, der vor der Aktualisierung der Spalte QUOTE der Tabelle CURRENT-QUOTE aktiviert werden soll. Die zweite Zeile gibt an, daß die ausgelöste Aktion angewendet werden soll, bevor eventuelle Änderungen, die durch die tatsächliche Aktualisierung der Tabelle CURRENT-QUOTE verursacht werden, auf die Datenbank angewendet werden. Die Klausel NO CASCADE besagt, daß die ausgelöste Aktion keine weiteren Auslöser aktiviert. Die dritte Zeile gibt die Namen an, die als Qualifikationsmerkmale des Spaltennamens für die neuen Werte (NEWQUOTE) und die alten Werte (OLDQUOTE) verwendet werden sollen. Spaltennamen, denen dieser Korrelationsnamen (NEWQUOTE und OLDQUOTE) als Qualifikationsmerkmal zugeordnet wird, werden als *Übergangsvariablen* bezeichnet. Die vierte Zeile gibt an, daß die ausgelöste Aktion für jede Zeile ausgeführt werden soll.

2 Hier beginnt die erste und einzige SQL-Anweisung in der ausgelösten Aktion dieses Auslösers. Die Anweisung SET für die Übergangsvariable wird in einem Auslöser verwendet, um einer Spalte in der Zeile der Tabelle, die durch die Anweisung, die den Auslöser aktiviert hat, aktualisiert wird, einen Wert zuzuordnen. Diese Anweisung ordnet der Spalte STATUS der Tabelle CURRENTQUOTE einen Wert zu.

3 Der Ausdruck, der auf der rechten Seite der Zuordnung verwendet wird, ist ein Ausdruck CASE. Der Ausdruck CASE erstreckt sich bis zum Schlüsselwort END.

4 Der erste Teil des Ausdrucks CASE überprüft, ob der neue Wert (NEWQUOTE.QUOTE) den Maximalwert für das Börsenkürzel im laufenden Kalenderjahr überschreitet. Die Unterabfrage verwendet die Tabelle QUOTEHISTORY, die durch den folgenden Nachauslöser aktualisiert wird.

5 Der zweite Teil des Ausdrucks CASE überprüft, ob der neue Wert (NEWQUOTE.QUOTE) den Mindestwert für das Börsenkürzel im laufenden Kalenderjahr unterschreitet. Die Unterabfrage verwendet die Tabelle QUOTEHISTORY, die durch den folgenden Nachauslöser aktualisiert wird.

6 Die letzten drei Teile des Ausdrucks CASE vergleichen den neuen Wert (NEWQUOTE.QUOTE) mit dem Wert, der sich in der Tabelle (OLDQUOTE.QUOTE) befunden hat, und stellen fest, ob er größer, kleiner oder identisch ist. Hier endet die Anweisung SET für die Übergangsvariable.

Zusätzlich zur Aktualisierung des Eintrags in der Tabelle CURRENTQUOTE muß ein Protokolleintrag erstellt werden, indem der neue Wert zusammen mit einer Zeitmarke in die Tabelle QUOTEHISTORY kopiert wird. Dies wird mit dem folgenden Nachauslöser erreicht:

1

```
CREATE TRIGGER RECORD_HISTORY
AFTER UPDATE OF QUOTE ON CURRENTQUOTE
REFERENCING NEW AS NEWQUOTE
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
```

2

```
INSERT INTO QUOTEHISTORY
   VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
END
```

1 Dieser Codeblock definiert einen Auslöser namens RECORD_HISTORY als Auslöser, der nach der Aktualisierung der Spalte QUOTE der Tabelle CURRENTQUOTE aktiviert werden sollte. Die dritte Zeile gibt den Namen an, der als Qualifikationsmerkmal des Spaltennamens für den neuen Wert (NEWQUOTE) verwendet werden soll. Die vierte Zeile gibt an, daß die ausgelöste Aktion für jede Zeile ausgeführt werden soll.

2 Die ausgelöste Aktion dieses Auslösers enthält eine einzige SQL-Anweisung, die eine Zeile in die Tabelle QUOTEHISTORY einfügt. Zu diesem Zweck werden die Daten aus der Zeile, die aktualisiert wurde (NEWQUOTE.SYMBOL und NEWQUOTE.QUOTE), sowie die aktuelle Zeitmarke verwendet.

CURRENT TIMESTAMP ist ein Sonderregister, das die Zeitmarke enthält. Eine Liste und Erläuterung finden Sie unter „Sonderregister” auf Seite 81.

Verknüpfungen

Der Prozeß, bei dem Daten aus zwei oder mehr Tabellen kombiniert werden, wird als Verknüpfung von Tabellen bezeichnet. Der Datenbankmanager erstellt alle Kombinationen von Zeilen aus den angegebenen Tabellen. Bei jeder Kombination prüft er die *Verknüpfungsbedingung*. Eine Verknüpfungsbedingung ist eine Suchbedingung, die gewissen Einschränkungen unterliegt. Eine Liste dieser Einschränkungen enthält das Handbuch *SQL Reference*.

Bitte beachten Sie, daß die von der Verknüpfungsbedingung betroffenen Spalten nicht identisch sein müssen. Es ist jedoch erforderlich, daß die Zeilen miteinander kompatibel sind. Die Verknüpfungsbedingung wird auf dieselbe Weise wie jede andere Suchbedingung ausgewertet, und es gelten dieselben Regeln für Vergleiche.

Wenn Sie keine Verknüpfungsbedingung angeben, werden alle Kombinationen der Zeilen aus den Tabellen, die in der Klausel FROM aufgelistet sind, zurückgegeben, selbst dann, wenn die Zeilen überhaupt keinen Bezug zueinander aufweisen. Das Ergebnis wird als das *übergreifende Produkt* der beiden Tabellen bezeichnet.

Die Beispiele in diesem Abschnitt basieren auf den beiden folgenden Tabellen. Es handelt sich hierbei um vereinfachte Versionen der Tabellen aus der Beispieldatenbank, die in dieser Form jedoch nicht in der Beispieldatenbank enthalten sind. Mit ihrer Hilfe sollen interessante Aspekte von Verknüpfungen im allgemeinen vorgestellt werden. Die Tabelle SAMP_STAFF listet Namen und Tätigkeitsbeschreibungen der Mitarbeiter auf, die keine freien Mitarbeiter sind. Die Tabelle SAMP_PROJECT enthält die Namen der (festangestellten und freien) Mitarbeiter und die Projekte, die von ihnen bearbeitet werden.

Die Tabellen lauten:

NAME	PROJ
Haas	AD3100
Thompson	PL2100
Walker	MA2112
Lutz	MA2111

Abbildung 5. Tabelle SAMP_PROJECT

NAME	JOB
Haas	PRES
Thompson	MANAGER
Lucchessi	SALESREP
Nicholls	ANALYST

Abbildung 6. Tabelle SAMP_STAFF

Das folgende Beispiel erzeugt das übergreifende Produkt der beiden Tabellen. Es wird keine Verknüpfungsbedingung angegeben. Daher sind alle Kombinationen der Zeilen vorhanden:

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT, SAMP_STAFF
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME       PROJ   NAME       JOB
---------- ------ ---------- --------
Haas       AD3100 Haas       PRES
Thompson   PL2100 Haas       PRES
Walker     MA2112 Haas       PRES
Lutz       MA2111 Haas       PRES
Haas       AD3100 Thompson   MANAGER
Thompson   PL2100 Thompson   MANAGER
Walker     MA2112 Thompson   MANAGER
Lutz       MA2111 Thompson   MANAGER
Haas       AD3100 Lucchessi  SALESREP
Thompson   PL2100 Lucchessi  SALESREP
Walker     MA2112 Lucchessi  SALESREP
Lutz       MA2111 Lucchessi  SALESREP
Haas       AD3100 Nicholls   ANALYST
Thompson   PL2100 Nicholls   ANALYST
Walker     MA2112 Nicholls   ANALYST
Lutz       MA2111 Nicholls   ANALYST
```

Die beiden Hauptarten von Verknüpfungen sind *innere Verknüpfungen* und *erweiterte Verknüpfungen*. Bislang haben alle vorgestellten Beispiele die innere Verknüpfung verwendet. Innere Verknüpfungen behalten nur die Zeilen aus dem übergreifenden Produkt bei, die die Verknüpfungsbedingung erfüllen. Wenn eine Zeile in der einen, nicht jedoch in der anderen Tabelle vorhanden ist, werden die Daten nicht in die Ergebnistabelle übernommen.

Das folgende Beispiel erzeugt die innere Verknüpfung der beiden Tabellen. Die innere Verknüpfung listet die Namen der festangestellten Mitarbeiter auf, die einem Projekt zugeordnet sind:

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT, SAMP_STAFF
  WHERE SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Alternativ können Sie die innere Verknüpfung auch wie folgt angeben:

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT INNER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Das Ergebnis lautet:

```
NAME       PROJ   NAME       JOB
---------- ------ ---------- --------
Haas       AD3100 Haas       PRES
Thompson   PL2100 Thompson   MANAGER
```

Bitte beachten Sie, daß das Ergebnis der inneren Verknüpfung aus Zeilen besteht, deren Werte für die Spalte NAME in der rechten und der linken Tabelle übereinstimmen. 'Haas' und 'Thompson' sind in der Tabelle SAMP-_STAFF mit allen festangestellten Mitarbeitern sowie in der Tabelle SAMP_ PROJECT enthalten, die festangestellte und freie Mitarbeiter auflistet, die einem Projekt zugeordnet sind.

Erweiterte Verknüpfungen kombinieren die innere Verknüpfung mit den Zeilen aus der rechten und/oder linken Tabelle, die in der inneren Verknüpfung fehlen. Wenn Sie eine erweiterte Verknüpfung für zwei Tabellen ausführen, nehmen Sie eine willkürliche Zuordnung der einen Tabelle als linke Tabelle und der anderen Tabelle als rechte Tabelle vor. Es gibt drei Arten von erweiterten Verknüpfungen:

1. Eine ***linke erweiterte Verknüpfung*** enthält die innere Verknüpfung sowie die Zeilen aus der linken Tabelle, die nicht in der inneren Verknüpfung enthalten sind.
2. Eine ***rechte erweiterte Verknüpfung*** enthält die innere Verknüpfung sowie die Zeilen aus der rechten Tabelle, die nicht in der inneren Verknüpfung enthalten sind.
3. Eine ***volle erweiterte Verknüpfung*** enthält die innere Verknüpfung sowie die Zeilen sowohl aus der linken als auch aus der rechten Tabelle, die nicht in der inneren Verknüpfung enthalten sind.

Mit der Anweisung SELECT geben Sie die anzuzeigenden Spalten an. In der Klausel FROM geben Sie den Namen der ersten Tabelle gefolgt von den Schlüsselwörtern LEFT OUTER JOIN, RIGHT OUTER JOIN oder FULL OUTER JOIN an. Anschließend müssen Sie die zweite Tabelle gefolgt von dem Schlüsselwort ON angeben. Nach dem Schlüsselwort ON geben Sie mit der Verknüpfungsbedingung eine Beziehung zwischen den zu verknüpfenden Tabellen an.

Im folgenden Beispiel ist die Tabelle SAMP_STAFF als rechte Tabelle und die Tabelle SAMP_PROJECT als linke Tabelle festgelegt. Durch die Verwendung des Schlüsselworts LEFT OUTER JOIN (= linke erweiterte Verknüpfung) werden Name und Projektnummer aller festangestellten und freien Mitarbeiter (aufgeführt in der Tabelle SAMP_PROJECT) und deren Tätigkeitsbezeichnung aufgelistet, sofern es sich um festangestellte Mitarbeiter (aufgeführt in der Tabelle SAMP_STAFF) handelt:

```
SELECT SAMP_PROJECT.NAME, SAMP_PROJECT.PROJ,
       SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT LEFT OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME       PROJ                NAME       JOB
---------- ------------------- ---------- --------------------
Haas       AD3100              Haas       PRES
Lutz       MA2111              -          -
Thompson   PL2100              Thompson   MANAGER
Walker     MA2112              -          -
```

Das Ergebnis der inneren Verknüpfung sind Zeilen, die in allen Spalten Werte enthalten. Die folgenden Zeilen erfüllen die Verknüpfungsbedingung: 'Haas' und 'Thompson' sind sowohl in der Tabelle SAMP_PROJECT (linke Tabelle) als auch in der Tabelle SAMP_STAFF (rechte Tabelle) enthalten. Bei Zeilen, die die Verknüpfungsbedingung nicht erfüllten, wird in Spalten aus der rechten Tabelle ein Nullwert angezeigt: 'Lutz' und 'Walker' sind freie Mitarbeiter, die in der Tabelle SAMP_PROJECT und nicht in der Tabelle SAMP_STAFF enthalten sind. Bitte beachten Sie, daß die Ergebnismenge alle Zeilen aus der linken Tabelle enthält.

Im nächsten Beispiel wird SAMP_STAFF als rechte Tabelle und SAMP_PROJECT als linke Tabelle festgelegt. Durch die Verwendung des Schlüsselworts RIGHT OUTER JOIN (= rechte erweiterte Verknüpfung) werden Name und Tätigkeitsbezeichnung aller festangestellten Mitarbeiter (aufgeführt in der Tabelle SAMP_STAFF) und deren Projektnummer aufgelistet, sofern sie einem Projekt zugeordnet sind (aufgeführt in der Tabelle SAMP_PROJECT):

```
SELECT SAMP_PROJECT.NAME,
       SAMP_PROJECT.PROJ, SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT RIGHT OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Das Ergebnis lautet:

```
NAME       PROJ                 NAME       JOB
---------- -------------------- ---------- --------------------
Haas       AD3100               Haas       PRES
-          -                    Lucchessi  SALESREP
-          -                    Nicholls   ANALYST
Thompson   PL2100               Thompson   MANAGER
```

Wie bei der linken erweiterten Verknüpfung enthält das Ergebnis Zeilen, die in allen Spalten Werte aufweisen. Die folgenden Zeilen erfüllen die Verknüpfungsbedingung: 'Haas' und 'Thompson' sind sowohl in der Tabelle SAMP_PROJECT (linke Tabelle) als auch in der Tabelle SAMP_STAFF (rechte Tabelle) enthalten. Bei Zeilen, die die Verknüpfungsbedingung nicht erfüllten, wird in Spalten aus der rechten Tabelle ein Nullwert angezeigt: 'Lucchessi' und 'Nicholls' sind festangestellte Mitarbeiter, die keinem Projekt zugeordnet sind. Sie werden zwar in der Tabelle SAMP_STAFF, aber nicht in der Tabelle SAMP_PROJECT aufgeführt. Bitte beachten Sie, daß die Ergebnismenge alle Zeilen aus der rechten Tabelle enthält.

Im nächsten Beispiel wird das Schlüsselwort FULL OUTER JOIN (= volle erweiterte Verknüpfung) mit den Tabellen SAMP_PROJECT und SAMP-_STAFF verwendet. Es listet die Namen aller festangestellten Mitarbeiter, einschließlich der Mitarbeiter, die keinem Projekt zugeordnet sind, sowie die Namen der freien Mitarbeiter auf:

```
SELECT SAMP_PROJECT.NAME, SAMP_PROJECT.PROJ,
       SAMP_STAFF.NAME, SAMP_STAFF.JOB
  FROM SAMP_PROJECT FULL OUTER JOIN SAMP_STAFF
    ON SAMP_STAFF.NAME = SAMP_PROJECT.NAME
```

Das Ergebnis lautet:

```
NAME       PROJ                 NAME       JOB
---------- -------------------- ---------- --------------------
Haas       AD3100               Haas       PRES
-          -                    Lucchessi  SALESREP
-          -                    Nicholls   ANALYST
Thompson   PL2100               Thompson   MANAGER
Lutz       MA2111               -          -
Walker     MA2112               -          -
```

Dieses Ergebnis enthält die linke erweiterte Verknüpfung, die rechte erweiterte Verknüpfung und die innere Verknüpfung. Alle festangestellten und freien Mitarbeiter werden aufgelistet. Wie bei der linken erweiterten Verknüpfung und der rechten erweiterten Verknüpfung wird für Werte, die die Verknüpfungsbedingung nicht erfüllten, in der entsprechenden Spalte ein Nullwert angezeigt. Die Ergebnismenge enthält alle Zeilen aus den Tabellen SAMP_STAFF und SAMP_PROJECT.

Komplexe Abfragen

DB2 Universal Database ermöglicht durch Abfragen mit ROLLUP und CUBE das Gruppieren, Konsolidieren und Anzeigen mehrerer Spalten in einer einzigen Ergebnismenge. Diese neue und leistungsstarke Funktion erweitert und vereinfacht die Datenanalyse auf SQL-Basis.

Es gibt verschiedene Methoden für das Extrahieren von nützlichen Informationen aus der Datenbank. Durch die Implementierung von rekursiven Abfragen können Sie Ergebnistabellen aus vorhandenen Dateien erzeugen.

Abfragen mit ROLLUP und CUBE

Die Operationen ROLLUP und CUBE werden in der Klausel GROUP BY einer Abfrage angegeben.

Die Gruppierung durch ROLLUP erzeugt eine Ergebnismenge, die die regulär gruppierten Zeilen und *Zwischensummenzeilen* enthält. Die Gruppierung durch CUBE erzeugt eine Ergebnismenge, die die Zeilen aus ROLLUP und aus der Kreuztabelle enthält.

Mit Hilfe von ROLLUP könnten Sie daher beispielsweise die Verkäufe pro Mitarbeiter und Monat zusammen mit den monatlichen Gesamtverkäufen und einer Gesamtsumme ermitteln. Bei der Verwendung von CUBE würden zusätzliche Zeilen für die Gesamtverkäufe pro Mitarbeiter hinzugefügt.

Weitere Informationen finden Sie im Handbuch *SQL Reference*.

Rekursive Abfragen

Eine *rekursive Abfrage* ist eine Abfrage, die Ergebnisdaten wiederholt verwendet, um weitere Ergebnisse zu ermitteln. Diesen Vorgang kann man sich auch als das Durcharbeiten einer Baumstruktur oder eines Diagramms vorstellen.

Beispiele für die Anwendung in der Praxis sind Reservierungssysteme, Netzplanung und Zeitplanung.

Eine rekursive Abfrage wird in Form eines allgemeinen Tabellenausdrucks geschrieben, der einen Verweis auf seinen eigenen Namen enthält.

Beispiele für rekursive Abfragen finden Sie im Handbuch *SQL Reference*.

OLAP-Funktionen

OLAP-Funktionen (OnLine Analytical Processing - Online-Analyseverarbeitung) führen eine Spaltenfunktionsoperation für ein *Datenfenster* aus. Dieses Fenster kann eine Zeilenpartitionierung, eine Reihenfolge von Zeilen innerhalb von Partitionen oder eine *Spaltenberechnungsgruppe* angeben. Anhand der Spaltenberechnungsgruppe kann der Benutzer angeben, welche Zeilen in Bezug auf die aktuelle Zeile in die Berechnung einfließen. Die Verwendung eines solchen Fensters ermöglicht Operationen wie kumulierte Summen und gleitende Durchschnitte.

OLAP-Funktionen ermöglichen es dem Benutzer nicht nur, ein Fenster für vorhandene Spaltenfunktionen (wie z. B. SUM und AVG) anzugeben. Mit ihrer Hilfe können außerdem Rangordnungsoperationen (RANK und DENSE_RANK) ausgeführt und Zeilen numeriert (ROW_NUMBER) werden, sofern eine spezifische Partitionierung und Reihenfolge der Zeilen vorliegt.

Die folgende Beispielabfrage ermittelt für die Abteilungen 15 und 38, basierend auf dem Gehalt, die Position eines Mitarbeiters in der Rangordnung der Abteilung und zeigt die kumulierte Summe der Gehälter in der Abteilung an:

```
SELECT NAME, DEPT,
       RANK () OVER (PARTITION BY DEPT ORDER BY SALARY DESC) AS RANK,
       SUM (SALARY) OVER (PARTITION BY DEPT
                          ORDER BY SALARY DESC
                          ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
           AS CUMULATIVE_SUM
  FROM STAFF
  WHERE DEPT IN (15,38)
  ORDER BY DEPT, RANK
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
NAME       DEPT      RANK       CUMULATIVE_SUM
---------- --------- ---------- --------------------
Hanes      15                 1             20659.80
Rothman    15                 2             37162.63
Ngan       15                 3             49670.83
Kermisch   15                 4             61929.33
O'Brien    38                 1             18006.00
Marenghi   38                 2             35512.75
Quigley    38                 3             52321.05
Naughton   38                 4             65275.80
Abrahams   38                 5             77285.55
```

Kapitel 8. Anpassen und Erweitern der Datenbearbeitung

Dieses Kapitel enthält eine kurze Einführung in die *objektorientierten Erweiterungen* in DB2 Universal Database. Die Verwendung objektorientierter Erweiterungen bietet viele Vorteile. *Benutzerdefinierte Typen (UDT)* vergrößern die Gruppe der für Ihre Anwendungen verfügbaren Datentypen. Mit *benutzerdefinierten Funktionen (UDF)* können Sie anwendungsspezifische Funktionen erstellen. UDFs fungieren als *Methoden* für UDTs, indem sie ein konsistentes Verhalten und eine konsistente Kapselung der Typen ermöglichen.

Anschließend werden *Sonderregister* und *Systemkataloge* erläutert. Sonderregister sind Speicherbereiche, die durch den Datenbankmanager definiert werden. Sie werden zum Speichern von Informationen verwendet, auf die SQL-Anweisungen verweisen können. Sonderregister werden zum Zeitpunkt der Verbindung aufgebaut und sind für die Verarbeitung dieser Anwendung spezifisch. Die Systemkataloge enthalten Informationen zur logischen und physischen Struktur von Datenbankobjekten.

Die folgenden Themen werden behandelt:

- Benutzerdefinierte Typen
- Benutzerdefinierte Funktionen
- Große Objekte (LOB)
- Sonderregister
- Einführung in Katalogsichten

Eine ausführliche Erörterung der oben genannten Themen ist im Rahmen dieses Handbuchs leider nicht möglich. Sie finden sie jedoch in den Handbüchern *SQL Reference* und *Systemverwaltung*.

Benutzerdefinierte Typen

Ein *einzigartiger Datentyp* ist ein benutzerdefinierter Datentyp, der seine interne Darstellung mit einem vorhandenen Datentyp (seinem „Quellentyp") gemeinsam benutzt, jedoch für die meisten Operationen als separater und inkompatibler Datentyp gilt. Beispielsweise könnten Sie einen Datentyp für das Alter, einen Datentyp für das Gewicht und einen Datentyp für die Größe definieren, die alle eine relativ unterschiedliche Semantik aufweisen, aber den integrierten Datentyp INTEGER zur internen Darstellung verwenden.

Das folgende Beispiel veranschaulicht die Erstellung eines einzigartigen Datentyps namens PAY:

```
CREATE DISTINCT TYPE PAY AS DECIMAL(9,2) WITH COMPARISONS
```

Obwohl PAY dieselbe Darstellung wie der integrierte Datentyp DECIMAL(9,2) besitzt, wird er als separater Datentyp betrachtet, der weder mit DECIMAL(9,2) noch mit einem anderen Datentyp vergleichbar ist. Er ist nur mit demselben einzigartigen Datentyp vergleichbar. Auch Operatoren und Funktionen, die für den Datentyp DECIMAL verwendet werden können, können auf diesen Datentyp nicht angewendet werden. Ein Wert mit dem Datentyp PAY kann beispielsweise nicht mit einem Wert des Typs INTEGER multipliziert werden. Daher müssen Sie Funktionen schreiben, die nur auf den Datentyp PAY angewendet werden.

Die Verwendung von einzigartigen Datentypen reduziert das Auftreten versehentlicher Fehler. Wurde beispielsweise die Spalte SALARY der Tabelle EMPLOYEE als Datentyp PAY definiert, ist eine Addition zur Spalte COMM selbst dann nicht möglich, wenn beide denselben Quellentyp haben.

Einzigartige Datentypen unterstützen die Datenumsetzung. Ein Quellentyp kann in einen einzigartigen Datentyp und ein einzigartiger Datentyp kann in einen Quellentyp umgesetzt werden. Wurde beispielsweise die Spalte SALARY der Tabelle EMPLOYEE als Datentyp PAY definiert, würde das folgende Beispiel beim Vergleichsoperator nicht fehlschlagen.

```
SELECT * FROM EMPLOYEE
   WHERE DECIMAL(SALARY) = 41250
```

DECIMAL(SALARY) gibt einen Dezimaldatentyp zurück. Umgekehrt kann ein numerischer Datentyp in einen Datentyp PAY umgesetzt werden. Sie können beispielsweise die Zahl 41250 durch die Verwendung von PAY(41250) umsetzen.

Benutzerdefinierte Funktionen

Wie bereits unter „Verwenden von Funktionen” auf Seite 34 beschrieben, bietet DB2 Universal Database integrierte und benutzerdefinierte Funktionen (UDF). Diese Gruppe von Funktionen kann jedoch nicht alle Anforderungen erfüllen. Häufig müssen für bestimmte Tasks angepaßte Funktionen erstellt werden. Mit benutzerdefinierten Funktionen können Sie angepaßte Funktionen erstellen.

Es gibt vier Arten von benutzerdefinierten Funktionen: *Quellenfunktionen* (oder Schablonen), *externe Skalarfunktionen*, *externe Tabellenfunktionen* und *externe Tabellenfunktionen aus OLE DB*.

Dieser Abschnitt behandelt Quellenfunktionen und externe Skalarfunktionen. Weitere Informationen zu externen Tabellenfunktionen und externen Tabellenfunktionen aus OLE DB enthält das Handbuch *SQL Reference*.

Benutzerdefinierte Quellenfunktionen ermöglichen benutzerdefinierten Datentypen einen selektiven Verweis auf eine andere integrierte oder benutzerdefinierte Funktion, die der Datenbank bereits bekannt ist. Sie können sowohl Skalar- als auch Spaltenfunktionen verwenden.

Im nächsten Beispiel wird eine benutzerdefinierte Funktion namens MAX erstellt, die auf der integrierten Spaltenfunktion MAX basiert, welche wiederum einen Datentyp DECIMAL als Eingabe verwendet. Die benutzerdefinierte Funktion MAX verwendet einen Datentyp PAY als Eingabe und gibt einen Datentyp PAY als Ausgabe zurück.

```
CREATE FUNCTION MAX(PAY) RETURNS PAY
   SOURCE MAX(DECIMAL)
```

Externe benutzerdefinierte Funktionen werden von Benutzern in einer Programmiersprache geschrieben. Es handelt sich hierbei um *externe Skalarfunktionen* und *externe Tabellenfunktionen*. Beide werden im Handbuch *SQL Reference* erläutert.

Beim folgenden Beispiel wird davon ausgegangen, daß bereits eine Funktion geschrieben wurde, die die Wörter in einer Zeichenfolge zählt. Diese Funktion kann durch die Anweisung CREATE FUNCTION mit dem Namen WORDCOUNT für die Datenbank registriert werden. Anschließend kann sie in SQL-Anweisungen verwendet werden.

Die folgende Anweisung gibt die Personalnummern und die Anzahl der Wörter in den Lebensläufen der Mitarbeiter im ASCII-Format zurück. WORDCOUNT ist eine externe Skalarfunktion, die durch den Benutzer für die Datenbank registriert wurde und nun in der Anweisung verwendet wird.

```
SELECT EMPNO, WORDCOUNT(RESUME)
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
```

Genauere Informationen zum Schreiben von benutzerdefinierten Funktionen finden Sie im Handbuch *Application Development Guide*.

Große Objekte (LOB)

Der Begriff *großes Objekt* (Large Object) und sein Akronym *LOB* werden zur Bezeichnung von drei Datentypen verwendet: BLOB, CLOB oder DBCLOB. Diese Datentypen können eine große Menge von Daten enthalten und werden für Objekte wie beispielsweise Audioobjekte, Fotos und Dokumente verwendet. Ein *großes Binärobjekt* (Binary Large OBject - BLOB) ist eine in Byte gemessene Zeichenfolge mit variabler Länge, die bis zu 2 Gigabyte lang sein kann. Ein großes Binärobjekt wird vorwiegend für nicht konventionelle Daten wie beispielsweise Bilder, Sprachdaten und gemischte Daten verwendet.

Ein *großes Zeichenobjekt* (Character Large OBject - CLOB) ist eine in Byte gemessene Zeichenfolge mit variabler Länge, die bis zu 2 Gigabyte lang sein kann. In einem großen Zeichenobjekt werden umfangreiche Daten im Einzelbytezeichensatz wie beispielsweise Dokumente gespeichert. Ein großes Zeichenobjekt wird als Zeichenfolge betrachtet.

Ein *großes Doppelbytezeichenobjekt* (Double-Byte Character Large OBject - DBCLOB) ist eine Zeichenfolge aus Doppelbytezeichen mit variabler Länge, die bis zu 2 Gigabyte lang sein kann (1 073 741 823 Doppelbytezeichen). In einem großen Doppelbytezeichenobjekt werden umfangreiche Daten im Doppelbytezeichensatz wie beispielsweise Dokumente gespeichert. Ein großes Doppelbytezeichenobjekt wird als Grafikzeichenfolge betrachtet.

Bearbeiten von großen Objekten (LOBs)

Da LOB-Werte sehr umfangreich sein können, ist ihre Übertragung vom Datenbank-Server an ein Client-Anwendungsprogramm unter Umständen zeitaufwendig. In der Regel werden LOB-Werte jedoch Stück für Stück und nicht als Ganzes verarbeitet. In den Fällen, in denen eine Anwendung nicht den vollständigen LOB-Wert im Anwendungsspeicher speichern muß (oder soll), kann auf diesen Wert über eine Variable, einen sogenannten *LOB-Querverweis*, verwiesen werden. Nachfolgende Anweisungen können dann die Querverweise zur Ausführung von Operationen für die Daten verwenden, ohne hierzu zwangsläufig das vollständige große Objekt abrufen zu müssen. Mit Querverweisvariablen kann der Speicherbedarf für die Anwendungen reduziert und die Leistung verbessert werden, da der Datenfluß zwischen dem Client und dem Server verringert wird.

Dateireferenzvariablen stellen einen weiteren Mechanismus dar. Mit ihrer Hilfe kann ein großes Objekt direkt in eine Datei abgerufen oder ein großes Objekt in einer Tabelle direkt aus einer Datei heraus aktualisiert werden. Mit Dateireferenzvariablen kann der Speicherbedarf der Anwendungen reduziert werden, da sie die Daten des großen Objekts nicht speichern müssen. Weitere Informationen enthalten die Handbücher *Application Development Guide* und *SQL Reference*.

Sonderregister

Ein *Sonderregister* ist ein Speicherbereich, der durch den Datenbankmanager für eine Verbindung definiert und zum Speichern von Informationen verwendet wird, auf die in SQL-Anweisungen verwiesen werden kann. Im folgenden werden nur einige Beispiele für die häufiger verwendeten Sonderregister dargestellt. Eine Liste aller Sonderregister und genauere Informationen finden Sie im Handbuch *SQL Reference*.

- CURRENT DATE: Enthält das anhand des Tageszeitgebers zum Ausführungszeitpunkt der SQL-Anweisung festgestellte Datum.
- CURRENT FUNCTION PATH: Enthält einen Wert für den Funktionspfad, der zum Auflösen von Funktions- und Datentypverweisen verwendet wird.
- CURRENT SERVER: Gibt den aktuellen Anwendungs-Server an.
- CURRENT TIME: Enthält die anhand des Tageszeitgebers zum Ausführungszeitpunkt der SQL-Anweisung festgestellte Uhrzeit.
- CURRENT TIMESTAMP: Gibt die anhand des Tageszeitgebers zum Ausführungszeitpunkt der SQL-Anweisung festgestellte Zeitmarke an.
- CURRENT TIMEZONE: Gibt den Unterschied zwischen der Weltzeit und der Ortszeit auf dem Anwendungs-Server an.
- USER: Gibt die Berechtigungs-ID für Laufzeit an.

Den Inhalt eines Sonderregisters können Sie mit der Anweisung VALUES anzeigen. Beispiel:

```
VALUES (CURRENT TIMESTAMP)
```

Zu diesem Zweck könnten Sie auch die folgende Anweisung verwenden:

```
SELECT CURRENT TIMESTAMP FROM ORG
```

Diese Anweisung gibt den Wert TIMESTAMP für jeden Zeileneintrag in der Tabelle zurück.

Einführung in Katalogsichten

DB2 erstellt und verwaltet für jede Datenbank eine umfangreiche Gruppe von Systemkatalogtabellen. Diese Tabellen enthalten Informationen zur logischen und physischen Struktur von Datenbankobjekten wie z. B. Tabellen, Sichten, Paketen, Beziehungen der referentiellen Integrität, Funktionen, einzigartigen Datentypen und Auslösern. Sie werden bei Erstellung der Datenbank erstellt und während des normalen Betriebs aktualisiert. Sie können nicht explizit erstellt oder gelöscht werden. Ihren Inhalt können Sie jedoch abfragen und anzeigen.

Weitere Informationen finden Sie im Handbuch *SQL Reference*.

Auswählen von Zeilen aus Systemkatalogen

Katalogsichten werden wie alle anderen Datenbanksichten verwendet. Genau wie bei jeder anderen Sicht auf dem System können Sie die Daten mit Hilfe von SQL-Anweisungen anzeigen.

Außerordentlich nützliche Informationen zur Tabellen finden Sie im Katalog SYSCAT.TABLES. Um die Namen vorhandener Tabellen zu ermitteln, die von Ihnen erstellt wurden, können Sie eine Anweisung ähnlich der folgenden verwenden:

```
SELECT TABNAME, TYPE, CREATE_TIME
   FROM SYSCAT.TABLES
   WHERE DEFINER = USER
```

Diese Anweisung erzeugt das folgende Ergebnis:

```
TABNAME            TYPE CREATE_TIME
------------------ ---- --------------------------
ORG                T    1999-07-21-13.42.55.128005
STAFF              T    1999-07-21-13.42.55.609001
DEPARTMENT         T    1999-07-21-13.42.56.069001
EMPLOYEE           T    1999-07-21-13.42.56.310001
EMP_ACT            T    1999-07-21-13.42.56.710001
PROJECT            T    1999-07-21-13.42.57.051001
EMP_PHOTO          T    1999-07-21-13.42.57.361001
EMP_RESUME         T    1999-07-21-13.42.59.154001
SALES              T    1999-07-21-13.42.59.855001
CL_SCHED           T    1999-07-21-13.43.00.025002
IN_TRAY            T    1999-07-21-13.43.00.055001
```

Die folgende Liste enthält Katalogsichten für die in diesem Handbuch behandelten Objekte und Funktionen. Es gibt viele weitere Katalogsichten. Ausführliche Listen finden Sie in den Handbüchern *SQL Reference* und *Systemverwaltung*.

Beschreibung	**Katalogsicht**
Prüfung auf Integritätsbedingung	SYSCAT.CHECKS
Spalten	SYSCAT.COLUMNS
Spalten, auf die durch Prüfungen auf Integritätsbedingung verwiesen wird	SYSCAT.COLCHECKS
In Schlüsseln verwendete Spalten	SYSCAT.KEYCOLUSE
Datentypen	SYSCAT.DATATYPES
Funktionsparameter oder Ergebnis einer Funktion	SYSCAT.FUNCPARMS
Referentielle Integritätsbedingungen	SYSCAT.REFERENCES
Schemata	SYSCAT.SCHEMATA

Beschreibung	Katalogsicht
Integritätsbedingungen in Tabellen	SYSCAT.TABCONST
Tabellen	SYSCAT.TABLES
Auslöser	SYSCAT.TRIGGERS
Benutzerdefinierte Funktionen	SYSCAT.FUNCTIONS
Sichten	SYSCAT.VIEWS

Anhang A. Tabellen der Beispieldatenbank

In diesem Anhang sind die Daten dargestellt, die in den Beispieltabellen der Beispieldatenbank SAMPLE enthalten sind. Außerdem wird beschrieben, wie sie erstellt und entfernt werden können.

DB2 Universal Database enthält weitere Datenbanken, die die Funktionen des Informationsmanagements veranschaulichen und im Lernprogramm für das Informationsmanagement verwendet werden. In diesem Anhang wird jedoch nur der Inhalt der Beispieldatenbank SAMPLE beschrieben. Weitere Informationen zu den Beispieldatenbanken für das Informationsmanagement können Sie dem Handbuch *Data Warehouse-Zentrale Verwaltung* entnehmen.

Die Beispieltabellen werden in den Beispielen verwendet, die im vorliegenden Handbuch und in anderen Handbüchern dieser Bibliothek dargestellt sind. Außerdem werden die Daten aufgeführt, die in den Beispieldateien mit den Datentypen BLOB und CLOB enthalten sind.

Der vorliegende Anhang enthält die folgenden Abschnitte:

In den Beispieltabellen wird ein Nullwert durch einen Bindestrich (-) dargestellt.

Beispieldatenbank

Die Beispiele in diesem Handbuch verwenden eine Beispieldatenbank. Zur Verwendung dieser Beispiele müssen Sie die Beispieldatenbank SAMPLE erstellen. Zur Verwendung der Beispieldatenbank muß der Datenbankmanager installiert sein.

Erstellen der Beispieldatenbank

Die Beispieldatenbank wird durch eine ausführbare Datei erstellt.[2] Zur Erstellung der Datenbank benötigen Sie die Berechtigung SYSADM.

- **Bei Verwendung von Plattformen auf UNIX-Basis**

  Wenn Sie die Eingabeaufforderung des Betriebssystems verwenden, geben Sie folgendes ein:

  ```
  sqllib/bin/db2sampl <pfad>
  ```

  Die Eingabe muß im Ausgangsverzeichnis für den Eigner des Datenbankmanagerexemplars erfolgen. Hierbei steht *pfad* für einen wahlfreien Parameter, der den Pfad angibt, in dem die Beispieldatenbank erstellt werden soll. Drücken Sie die Eingabetaste.[3] Das Schema für DB2SAMPL ist der Sonderregisterwert CURRENT SCHEMA.

- **Bei Verwendung von OS/2- oder Windows-Plattformen**

  Wenn Sie die Eingabeaufforderung des Betriebssystems verwenden, geben Sie folgendes ein:
  `db2sampl` *e*
  Hierbei steht *e* für einen wahlfreien Parameter, der das Laufwerk angibt, auf dem die Datenbank erstellt werden soll. Drücken Sie die Eingabetaste.[4] Wenn Sie nicht über die Benutzerprofilverwaltung an Ihrer Workstation angemeldet sind, werden Sie hierzu aufgefordert.

Löschen der Beispieldatenbank

Wenn Sie nicht auf die Beispieldatenbank zugreifen müssen, können Sie sie mit dem Befehl DROP DATABASE löschen:

```
db2 drop database sample
```

2. Informationen zu diesem Befehl finden Sie im Abschnitt über den Befehl DB2SAMPL des Handbuchs *Command Reference*.

3. Wird der Parameter für den Pfad nicht angegeben, wird die Beispieldatenbank in dem Standardpfad erstellt, der in der Konfigurationsdatei des Datenbankmanagers durch den Parameter DFTDBPATH angegeben ist.

4. Wird der Parameter für das Laufwerk nicht angegeben, wird die Beispieldatenbank auf demselben Laufwerk erstellt, auf dem DB2 installiert ist.

Tabelle CL_SCHED

Name:	CLASS_CODE	DAY	STARTING	ENDING
Typ:	char(7)	smallint	time	time
Beschreibung:	Klassencode (raum:lehrer)	Tages-Nr. im Viertagesplan	Klassenstartzeit	Klassenendezeit

Tabelle DEPARTMENT

Name:	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION
Typ:	char(3) not null	varchar(29) not null	char(6)	char(3) not null	char(16)
Beschreibung:	Abteilungsnummer	Name, der den allgemeinen Aufgabenbereich der Abteilung angibt	Personalnummer (EMPNO) des Abteilungsleiters	Übergeordnete Abteilung (DEPTNO)	Name des fernen Standorts
Werte:	A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00	-
	B01	PLANNING	000020	A00	-
	C01	INFORMATION CENTER	000030	A00	-
	D01	DEVELOPMENT CENTER	-	A00	-
	D11	MANUFACTURING SYSTEMS	000060	D01	-
	D21	ADMINISTRATION SYSTEMS	000070	D01	-
	E01	SUPPORT SERVICES	000050	A00	-
	E11	OPERATIONS	000090	E01	-
	E21	SOFTWARE SUPPORT	000100	E01	-

Tabelle EMPLOYEE

Namen:	EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE
Typ:	char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date
Beschreibung:	Personalnummer	Vorname	Initiale zweiter Vorname	Nachname	Abteilung (DEPTNO) des Mitarbeiters	Rufnummer	Einstellungsdatum

JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
Tätigkeitsbezeichnung	Dauer der ordentlichen Schulbildung in Jahren	Geschlecht (M=männlich, F=weiblich)	Geburtsdatum	Jahresgehalt	Jährliche Bonuszahlung	Jährliche Provision

Die Werte in der Tabelle EMPLOYEE finden Sie auf der nächsten Seite.

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date	char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

Tabelle EMP_ACT

Name:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
Typ:	char(6) not null	char(6) not null	smallint not null	dec(5,2)	date	date
Beschreibung:	Personal-nummer	Projektnummer	Aktivitäts-nummer	Anteil der auf das Projekt verwendeten Arbeitszeit	Beginn der Aktivität	Ende der Aktivität
Werte:	000010	AD3100	10	.50	1982-01-01	1982-07-01
	000070	AD3110	10	1.00	1982-01-01	1983-02-01
	000230	AD3111	60	1.00	1982-01-01	1982-03-15
	000230	AD3111	60	.50	1982-03-15	1982-04-15
	000230	AD3111	70	.50	1982-03-15	1982-10-15
	000230	AD3111	80	.50	1982-04-15	1982-10-15
	000230	AD3111	180	1.00	1982-10-15	1983-01-01
	000240	AD3111	70	1.00	1982-02-15	1982-09-15
	000240	AD3111	80	1.00	1982-09-15	1983-01-01
	000250	AD3112	60	1.00	1982-01-01	1982-02-01
	000250	AD3112	60	.50	1982-02-01	1982-03-15
	000250	AD3112	60	.50	1982-12-01	1983-01-01
	000250	AD3112	60	1.00	1983-01-01	1983-02-01
	000250	AD3112	70	.50	1982-02-01	1982-03-15
	000250	AD3112	70	1.00	1982-03-15	1982-08-15
	000250	AD3112	70	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.25	1982-08-15	1982-10-15
	000250	AD3112	80	.50	1982-10-15	1982-12-01
	000250	AD3112	180	.50	1982-08-15	1983-01-01
	000260	AD3113	70	.50	1982-06-15	1982-07-01
	000260	AD3113	70	1.00	1982-07-01	1983-02-01
	000260	AD3113	80	1.00	1982-01-01	1982-03-01
	000260	AD3113	80	.50	1982-03-01	1982-04-15
	000260	AD3113	180	.50	1982-03-01	1982-04-15
	000260	AD3113	180	1.00	1982-04-15	1982-06-01
	000260	AD3113	180	.50	1982-06-01	1982-07-01
	000270	AD3113	60	.50	1982-03-01	1982-04-01
	000270	AD3113	60	1.00	1982-04-01	1982-09-01
	000270	AD3113	60	.25	1982-09-01	1982-10-15
	000270	AD3113	70	.75	1982-09-01	1982-10-15
	000270	AD3113	70	1.00	1982-10-15	1983-02-01

Name:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000270	AD3113	80	1.00	1982-01-01	1982-03-01
	000270	AD3113	80	.50	1982-03-01	1982-04-01
	000030	IF1000	10	.50	1982-06-01	1983-01-01
	000130	IF1000	90	1.00	1982-01-01	1982-10-01
	000130	IF1000	100	.50	1982-10-01	1983-01-01
	000140	IF1000	90	.50	1982-10-01	1983-01-01
	000030	IF2000	10	.50	1982-01-01	1983-01-01
	000140	IF2000	100	1.00	1982-01-01	1982-03-01
	000140	IF2000	100	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-03-01	1982-07-01
	000140	IF2000	110	.50	1982-10-01	1983-01-01
	000010	MA2100	10	.50	1982-01-01	1982-11-01
	000110	MA2100	20	1.00	1982-01-01	1982-03-01
	000010	MA2110	10	1.00	1982-01-01	1983-02-01
	000200	MA2111	50	1.00	1982-01-01	1982-06-15
	000200	MA2111	60	1.00	1982-06-15	1983-02-01
	000220	MA2111	40	1.00	1982-01-01	1983-02-01
	000150	MA2112	60	1.00	1982-01-01	1982-07-15
	000150	MA2112	180	1.00	1982-07-15	1983-02-01
	000170	MA2112	60	1.00	1982-01-01	1983-06-01
	000170	MA2112	70	1.00	1982-06-01	1983-02-01
	000190	MA2112	70	1.00	1982-02-01	1982-10-01
	000190	MA2112	80	1.00	1982-10-01	1983-10-01
	000160	MA2113	60	1.00	1982-07-15	1983-02-01
	000170	MA2113	80	1.00	1982-01-01	1983-02-01
	000180	MA2113	70	1.00	1982-04-01	1982-06-15
	000210	MA2113	80	.50	1982-10-01	1983-02-01
	000210	MA2113	180	.50	1982-10-01	1983-02-01
	000050	OP1000	10	.25	1982-01-01	1983-02-01
	000090	OP1010	10	1.00	1982-01-01	1983-02-01
	000280	OP1010	130	1.00	1982-01-01	1983-02-01
	000290	OP1010	130	1.00	1982-01-01	1983-02-01
	000300	OP1010	130	1.00	1982-01-01	1983-02-01
	000310	OP1010	130	1.00	1982-01-01	1983-02-01
	000050	OP2010	10	.75	1982-01-01	1983-02-01
	000100	OP2010	10	1.00	1982-01-01	1983-02-01
	000320	OP2011	140	.75	1982-01-01	1983-02-01

Tabellen der Beispieldatenbank

Name:	EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
	000320	OP2011	150	.25	1982-01-01	1983-02-01
	000330	OP2012	140	.25	1982-01-01	1983-02-01
	000330	OP2012	160	.75	1982-01-01	1983-02-01
	000340	OP2013	140	.50	1982-01-01	1983-02-01
	000340	OP2013	170	.50	1982-01-01	1983-02-01
	000020	PL2100	30	1.00	1982-01-01	1982-09-15

Tabelle EMP_PHOTO

Name:	EMPNO	PHOTO_FORMAT	PICTURE
Typ:	char(6) not null	varchar(10) not null	blob(100k)
Beschreibung:	Personalnummer	Fotoformat	Foto des Mitarbeiters
Werte:	000130	bitmap	db200130.bmp
	000130	gif	db200130.gif
	000130	xwd	db200130.xwd
	000140	bitmap	db200140.bmp
	000140	gif	db200140.gif
	000140	xwd	db200140.xwd
	000150	bitmap	db200150.bmp
	000150	gif	db200150.gif
	000150	xwd	db200150.xwd
	000190	bitmap	db200190.bmp
	000190	gif	db200190.gif
	000190	xwd	db200190.xwd

- Das Foto der Mitarbeiterin Delores Quintana sehen Sie unter „Quintana - Foto” auf Seite 99.
- Das Foto der Mitarbeiterin Heather Nicholls sehen Sie unter „Nicholls - Foto” auf Seite 100.
- Das Foto des Mitarbeiters Bruce Adamson sehen Sie unter „Adamson - Foto” auf Seite 102.
- Das Foto des Mitarbeiters James Walker sehen Sie unter „Walker - Foto” auf Seite 103.

Tabelle EMP_RESUME

Name:	EMPNO	RESUME_FORMAT	RESUME
Typ:	char(6) not null	varchar(10) not null	clob(5k)
Beschreibung:	Personalnummer	Format des Lebenslaufs	Lebenslauf des Mitarbeiters
Werte:	000130	ascii	db200130.asc
	000130	script	db200130.scr
	000140	ascii	db200140.asc
	000140	script	db200140.scr
	000150	ascii	db200150.asc
	000150	script	db200150.scr
	000190	ascii	db200190.asc
	000190	script	db200190.scr

- Den Lebenslauf der Mitarbeiterin Delores Quintana finden Sie unter „Quintana - Lebenslauf” auf Seite 99.
- Den Lebenslauf der Mitarbeiterin Heather Nicholls finden Sie unter „Nicholls - Lebenslauf” auf Seite 100.
- Den Lebenslauf des Mitarbeiters Bruce Adamson finden Sie unter „Adamson - Lebenslauf” auf Seite 102.
- Den Lebenslauf des Mitarbeiters James Walker finden Sie unter „Walker - Lebenslauf” auf Seite 103.

Tabelle IN_TRAY

Name:	RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
Typ:	timestamp	char(8)	char(64)	varchar(3000)
Beschreibung:	Eingangsdatum und -uhrzeit	Benutzer-ID des Memo-Absenders	Kurzbeschreibung	Memo

Tabelle ORG

Name:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
Typ:	smallint not null	varchar(14)	smallint	varchar(10)	varchar(13)
Beschreibung:	Abteilungs-nummer	Abteilungsname	Nummer des Abteilungsleiters	Unternehmens-bereich	Stadt
Werte:	10	Head Office	160	Corporate	New York
	15	New England	50	Eastern	Boston
	20	Mid Atlantic	10	Eastern	Washington
	38	South Atlantic	30	Eastern	Atlanta
	42	Great Lakes	100	Midwest	Chicago
	51	Plains	140	Midwest	Dallas

Tabellen der Beispieldatenbank

Name:	DEPTNUMB	DEPTNAME	MANAGER	DIVISION	LOCATION
	66	Pacific	270	Western	San Francisco
	84	Mountain	290	Western	Denver

Tabelle PROJECT

Name:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
Typ:	char(6) not null	varchar(24) not null	char(3) not null	char(6) not null	dec(5,2)	date	date	char(6)
Beschreibung:	Projekt-nummer	Projektname	Verantwortliche Abteilung	Verantwortliche Mitarbeiter	Geschätzter durchschnittlicher Personalauf-wand	Geschätztes Startdatum	Geschätztes Enddatum	Hauptprojekt (bei Unter-projekt)
Werte:	AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	-
	AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
	AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
	AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
	AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
	IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	-
	IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	-
	MA2100	WELD LINE AUTOMAT-ION	D01	000010	12	1982-01-01	1983-02-01	-
	MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
	MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
	MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
	MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
	OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	-
	OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
	OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	-
	OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000

Name:	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
	OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
	OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
	OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
	PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

Tabelle SALES

Name:	SALES_DATE	SALES_PERSON	REGION	SALES
Typ:	date	varchar(15)	varchar(15)	int
Beschreibung:	Verkaufsdatum	Nachname des Mitarbeiters	Verkaufsregion	Anzahl Verkäufe
Werte:	12/31/1995	LUCCHESSI	Ontario-South	1
	12/31/1995	LEE	Ontario-South	3
	12/31/1995	LEE	Quebec	1
	12/31/1995	LEE	Manitoba	2
	12/31/1995	GOUNOT	Quebec	1
	03/29/1996	LUCCHESSI	Ontario-South	3
	03/29/1996	LUCCHESSI	Quebec	1
	03/29/1996	LEE	Ontario-South	2
	03/29/1996	LEE	Ontario-North	2
	03/29/1996	LEE	Quebec	3
	03/29/1996	LEE	Manitoba	5
	03/29/1996	GOUNOT	Ontario-South	3
	03/29/1996	GOUNOT	Quebec	1
	03/29/1996	GOUNOT	Manitoba	7
	03/30/1996	LUCCHESSI	Ontario-South	1
	03/30/1996	LUCCHESSI	Quebec	2
	03/30/1996	LUCCHESSI	Manitoba	1
	03/30/1996	LEE	Ontario-South	7
	03/30/1996	LEE	Ontario-North	3
	03/30/1996	LEE	Quebec	7
	03/30/1996	LEE	Manitoba	4
	03/30/1996	GOUNOT	Ontario-South	2
	03/30/1996	GOUNOT	Quebec	18
	03/30/1996	GOUNOT	Manitoba	1
	03/31/1996	LUCCHESSI	Manitoba	1
	03/31/1996	LEE	Ontario-South	14
	03/31/1996	LEE	Ontario-North	3
	03/31/1996	LEE	Quebec	7

Tabellen der Beispieldatenbank

Name:	SALES_DATE	SALES_PERSON	REGION	SALES
	03/31/1996	LEE	Manitoba	3
	03/31/1996	GOUNOT	Ontario-South	2
	03/31/1996	GOUNOT	Quebec	1
	04/01/1996	LUCCHESSI	Ontario-South	3
	04/01/1996	LUCCHESSI	Manitoba	1
	04/01/1996	LEE	Ontario-South	8
	04/01/1996	LEE	Ontario-North	-
	04/01/1996	LEE	Quebec	8
	04/01/1996	LEE	Manitoba	9
	04/01/1996	GOUNOT	Ontario-South	3
	04/01/1996	GOUNOT	Ontario-North	1
	04/01/1996	GOUNOT	Quebec	3
	04/01/1996	GOUNOT	Manitoba	7

Tabelle STAFF

Name:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
Typ:	smallint not null	varchar(9)	smallint	char(5)	smallint	dec(7,2)	dec(7,2)
Beschreibung:	Personal-nummer	Mitarbeiter-name	Abteilungs-nummer	Tätigkeits-bezeichnung	Betriebs-zugehörigkeit in Jahren	Aktuelles Gehalt	Provision
Werte:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50

Name:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

Tabelle STAFFG

Anmerkung: Die Tabelle STAFFG wird nur für Doppelbyte-Codepages erstellt.

Name:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
Typ:	smallint not null	vargraphic(9)	smallint	graphic(5)	smallint	dec(9,0)	dec(9,0)
Beschreibung:	Personal-nummer	Mitarbeiter-name	Abteilungs-nummer	Tätigkeits-bezeichnung	Betriebs-zugehörigkeit in Jahren	Aktuelles Gehalt	Provision
Werte:	10	Sanders	20	Mgr	7	18357.50	-
	20	Pernal	20	Sales	8	18171.25	612.45
	30	Marenghi	38	Mgr	5	17506.75	-
	40	O'Brien	38	Sales	6	18006.00	846.55
	50	Hanes	15	Mgr	10	20659.80	-
	60	Quigley	38	Sales	-	16808.30	650.25
	70	Rothman	15	Sales	7	16502.83	1152.00
	80	James	20	Clerk	-	13504.60	128.20
	90	Koonitz	42	Sales	6	18001.75	1386.70
	100	Plotz	42	Mgr	7	18352.80	-

Tabellen der Beispieldatenbank

Name:	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
	110	Ngan	15	Clerk	5	12508.20	206.60
	120	Naughton	38	Clerk	-	12954.75	180.00
	130	Yamaguchi	42	Clerk	6	10505.90	75.60
	140	Fraye	51	Mgr	6	21150.00	-
	150	Williams	51	Sales	6	19456.50	637.65
	160	Molinare	10	Mgr	7	22959.20	-
	170	Kermisch	15	Clerk	4	12258.50	110.10
	180	Abrahams	38	Clerk	3	12009.75	236.50
	190	Sneider	20	Clerk	8	14252.75	126.50
	200	Scoutten	42	Clerk	-	11508.60	84.20
	210	Lu	10	Mgr	10	20010.00	-
	220	Smith	51	Sales	7	17654.50	992.80
	230	Lundquist	51	Clerk	3	13369.80	189.65
	240	Daniels	10	Mgr	5	19260.25	-
	250	Wheeler	51	Clerk	6	14460.00	513.30
	260	Jones	10	Mgr	12	21234.00	-
	270	Lea	66	Mgr	9	18555.50	-
	280	Wilson	66	Sales	9	18674.50	811.50
	290	Quill	84	Mgr	10	19818.00	-
	300	Davis	84	Sales	5	15454.50	806.10
	310	Graham	66	Sales	13	21000.00	200.30
	320	Gonzales	66	Sales	4	16858.20	844.00
	330	Burke	66	Clerk	1	10988.00	55.50
	340	Edwards	84	Sales	7	17844.00	1285.00
	350	Gafney	84	Clerk	5	13030.50	188.00

Beispieldateien mit Datentypen BLOB und CLOB

In diesem Abschnitt werden die Daten dargestellt, die sich in den Dateien EMP_PHOTO (Fotos der Mitarbeiter) und EMP_RESUME (Lebensläufe der Mitarbeiter) befinden.

Quintana - Foto

Abbildung 7. Delores M. Quintana

Quintana - Lebenslauf

Der folgende Text befindet sich in den Dateien db200130.asc und db200130.scr.

Resume: Delores M. Quintana

Personal Information

Address:	1150 Eglinton Ave Mellonville, Idaho 83725
Phone:	(208) 555-9933
Birthdate:	September 15, 1925
Sex:	Female
Marital Status:	Married
Height:	5’2″
Weight:	120 lbs.

Department Information

Employee Number:	000130
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-4578
Hire Date:	1971-07-28

Education

1965	Math and English, B.A. Adelphi University

1960	Dental Technician Florida Institute of Technology
Work History	
10/91 - present	Advisory Systems Analyst Producing documentation tools for engineering department.
12/85 - 9/91	Technical Writer Writer, text programmer, and planner.
1/79 - 11/85	COBOL Payroll Programmer Writing payroll programs for a diesel fuel company.

Interests
- Cooking
- Reading
- Sewing
- Remodeling

Nicholls - Foto

Abbildung 8. Heather A. Nicholls

Nicholls - Lebenslauf

Der folgende Text befindet sich in den Dateien db200140.asc und db200140.scr.

Resume: Heather A. Nicholls

Personal Information

Address:	844 Don Mills Ave Mellonville, Idaho 83734
Phone:	(208) 555-2310
Birthdate:	January 19, 1946
Sex:	Female
Marital Status:	Single
Height:	5'8"
Weight:	130 lbs.

Department Information

Employee Number:	000140
Dept Number:	C01
Manager:	Sally Kwan
Position:	Analyst
Phone:	(208) 555-1793
Hire Date:	1976-12-15

Education

1972	Computer Engineering, Ph.D. University of Washington
1969	Music and Physics, M.A. Vassar College

Work History

2/83 - present	Architect, OCR Development Designing the architecture of OCR products.
12/76 - 1/83	Text Programmer Optical character recognition (OCR) programming in PL/I.
9/72 - 11/76	Punch Card Quality Analyst Checking punch cards met quality specifications.

Interests

- Model railroading
- Interior decorating
- Embroidery
- Knitting

Adamson - Foto

Abbildung 9. Bruce Adamson

Adamson - Lebenslauf

Der folgende Text befindet sich in den Dateien db200150.asc und db200150.scr.

Resume: Bruce Adamson

Personal Information	
Address:	3600 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-4489
Birthdate:	May 17, 1947
Sex:	Male
Marital Status:	Married
Height:	6'0″
Weight:	175 lbs.
Department Information	
Employee Number:	000150
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-4510
Hire Date:	1972-02-12
Education	
1971	Environmental Engineering, M.Sc. Johns Hopkins University
1968	American History, B.A. Northwestern University

Work History

8/79 - present	Neural Network Design Developing neural networks for machine intelligence products.
2/72 - 7/79	Robot Vision Development Developing rule-based systems to emulate sight.
9/71 - 1/72	Numerical Integration Specialist Helping bank systems communicate with each other.

Interests

- Racing motorcycles
- Building loudspeakers
- Assembling personal computers
- Sketching

Walker - Foto

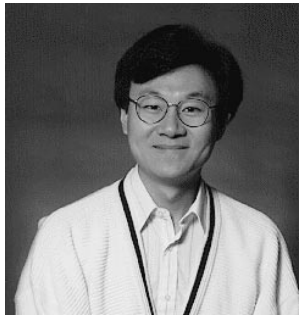


Abbildung 10. James H. Walker

Walker - Lebenslauf

Der folgende Text befindet sich in den Dateien db200190.asc und db200190.scr.

Resume: James H. Walker

Personal Information

Address:	3500 Steeles Ave Mellonville, Idaho 83757
Phone:	(208) 555-7325
Birthdate:	June 25, 1952
Sex:	Male
Marital Status:	Single
Height:	5’11″
Weight:	166 lbs.

Department Information

Employee Number:	000190
Dept Number:	D11
Manager:	Irving Stern
Position:	Designer
Phone:	(208) 555-2986
Hire Date:	1974-07-26

Education

1974	Computer Studies, B.Sc. University of Massachusetts
1972	Linguistic Anthropology, B.A. University of Toronto

Work History

6/87 - present	Microcode Design Optimizing algorithms for mathematical functions.
4/77 - 5/87	Printer Technical Support Installing and supporting laser printers.
9/74 - 3/77	Maintenance Programming Patching assembly language compiler for mainframes.

Interests

- Wine tasting
- Skiing
- Swimming
- Dancing

Anhang B. 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 124. 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 für 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 119. 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 statt dessen 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 123
- „Suchen nach Online-Informationen” auf Seite 128
- „Bestellen der gedruckten Handbücher” auf Seite 119
- „Drucken der PDF-Handbücher” auf Seite 118

Tabelle 1. 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 zu 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 SC12-2877 db2d2g70 SC12-2878 db2d3g70	db2d0

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
Konnektivität Ergänzung	Dieses Handbuch enthält Konfigurations- und Referenzinfor mationen 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 Form-nummer 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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
DB2 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
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.	SC27-0787 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 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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
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

Tabelle 1. Informationen zu DB2 (Forts.)

Name	Beschreibung	IBM Form PDF-Dateiname	HTML-Verzeichnis
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
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 Dateinamen `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 1 auf Seite 107.

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, *%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 2. 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` Datei 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 *DB2 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 *DB2 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 124.

Anhang C. 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. Anstelle 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 Web-Sites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Web-Sites dar. Das über diese Web-Sites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Web-Sites 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:

Neue deutsche Rechtschreibung

Durch die Einführung der neuen deutschen Rechtschreibung bei IBM zum 1. September 1999 kann es vorkommen, dass in dem vorliegenden Handbuch bestimmte Wörter sowohl nach der alten als auch nach der neuen Schreibweise verwendet werden, und zwar immer dann, wenn auf existierende Handbuchkapitel und/oder Programmteile zurückgegriffen wird.

Änderungen in der IBM Terminologie

Die ständige Weiterentwicklung der deutschen Sprache nimmt auch Einfluss auf die IBM Terminologie. Durch die daraus resultierende Umstellung der IBM Terminologie kann es u. U. vorkommen, dass in diesem Handbuch sowohl alte als auch neue Termini gleichbedeutend verwendet werden. Dies ist der Fall, wenn auf ältere existierende Handbuchkapitel und/oder Programmteile zurückgegriffen wird.

Aufgrund kurzfristiger Änderungen der Software, die in die Dokumentation nicht mehr aufgenommen werden konnten, entsprechen die in den Handbüchern aufgeführten Programmelemente möglicherweise nicht den im eigentlichen Programm angezeigten Elementen.

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 Extenders
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

E

F

G

H

I

K

L

N

V

W

Z

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 über folgende Nummern:

- Unter 0180 3/313 233 erreichen Sie Hallo IBM, wo Sie Antworten zu allgemeinen Fragen erhalten.
- Unter 0190/772 243 erreichen Sie die DB2 Helpline, wo Sie Antworten zu DB2-spezifischen Problemen erhalten.

Produktinformationen

Telefonische Unterstützung erhalten Sie über folgende 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.

Antwort

IBM DB2 Universal Database
SQL Erste Schritte
Version 7

IBM Form SC12-2882-00

Anregungen zur Verbesserung und Ergänzung dieser Veröffentlichung nehmen wir gerne entgegen. Bitte informieren Sie uns über Fehler, ungenaue Darstellungen oder andere Mängel.

Zur Klärung technischer Fragen sowie zu Liefermöglichkeiten und Preisen wenden Sie sich bitte entweder an Ihre IBM Geschäftsstelle, Ihren IBM Geschäftspartner oder Ihren Händler.

Unsere Telefonauskunft ″HALLO IBM″ (Telefonnr.: 01803/31 32 33) steht Ihnen ebenfalls zur Klärung allgemeiner Fragen zur Verfügung.

Kommentare:

Danke für Ihre Bemühungen.

Sie können ihre Kommentare betr. dieser Veröffentlichung wie folgt senden:
- Als Brief an die Postanschrift auf der Rückseite dieses Formulars
- Als E-Mail an die folgende Adresse: comment@tcvm.vnet.ibm.com

Name

Adresse

Firma oder Organisation

Rufnummer

E-Mail-Adresse

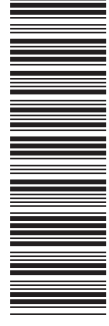
Antwort
SC12-2882-00

IBM Deutschland Informationssysteme GmbH
SW NLS Center

70548 Stuttgart

Teilenummer: CT7YHDE

Printed in Ireland



CT7YHDE

SC12-2882-00