Praktikum Datenbanken / DB2
Woche 5: SQL als Datenmanipulationssprache

Raum: LF 230 Nächste Sitzung: 10./13. November 2003

Die Dokumentation zu DB2 steht online zur Verfügung. Eine lokale Installation der Dokumentation findet sich unter der Adresse `http://salz.is.informatik.uni-duisburg.de/db2doc/de_DE/index.htm`.
Die englischsprachigen Handbücher für IBM DB2 V8.1 liegen als PDF Dateien lokal im Verzeichnis `/usr/projects/db2doc/`. Diese Handbücher sind sehr umfangreich und sollten nur zum Betrachten am Bildschirm herangezogen und nicht ausgedruckt werden. Die Referenzen in diesem Arbeitsblatt beziehen sich auf diese Handbücher. Die gleichen Informationen sind jedoch auch in der Online-Dokumentation zu finden.

Füllen von Tabellen

Export und Import

Aus einer bestehenden Tabelle kann man Daten in eine Datei exportieren und später diese Daten in eine andere Tabelle wieder einfügen. Dabei kann man entweder das EXPORT-Statement benutzen, oder den Export-Wizard unter CC verwenden, der über das Kontextmenü einer Tabelle aufgerufen werden kann.

db2dme80.pdf, Kapitel 1

Beim Export kann die erzeugte Datei entweder eine Textdatei mit Begrenzern sein, aus denen in einer anderen Anwendung wieder eine Tabelle erzeugt werden kann, oder aber eine Datei im IXF (*integrated exchange format*). Letzteres Format kann zusätzliche Informationen über das Schema der Tabelle aufnehmen.

Wenn eine Tabelle exportiert wird, gibt man zusätzlich zur Exportdatei ein SELECT-Statement an (dazu mehr in der nächsten Sitzung). Das einfachste SELECT-Statement

```
SELECT * FROM schema.tabelle
```

exportiert eine vollständige Tabelle mit allen Zeilen und Spalten. Ein beispielhafter Export-Befehl sieht etwa so aus:

```
EXPORT TO comedies.ixf OF ixf
  SELECT title,year FROM genres
```

Entsprechend kann eine exportierte Datei auch wieder importiert werden. Am einfachsten geht dies für zuvor als IXF Dateien exportierte Tabellen. Man kann hierfür das IMPORT-Statement bzw. den Import-Wizard benutzen, der sich ebenfalls über das Kontextmenü einer Tabelle aufrufen läßt. Eine andere Möglichkeit ist das mächtigere und komfortablere LOAD-Statement, das im nächsten Abschnitt behandelt wird.

db2dme80.pdf, Kapitel 2

Load

Das Einspoolen von Daten in eine Tabelle aus einer beliebigen Textdatei kann entweder mit den Kommandos IMPORT oder LOAD geschehen oder über den Load-Wizard aus dem Tools-Menü des CC. Dabei ist der Dateiname anzugeben, der Dateityp (ASC = Textdatei ohne Begrenzer, DEL = Textdatei mit Begrenzer, IXF), die Logdatei und zusätzliche Optionen. Die genaue Syntax und die möglichen Optionen sind im Handbuch beschrieben.

db2dme80.pdf, Kapitel 3

Beim Einspoolen von Textdateien (nicht IXF-Dateien) kann man folgende Importmodi benutzen:

`INSERT`: Hinzufügen der neuen Tupel

`INSERT_UPDATE`: Hinzufügen der neuen Tupel, bzw. Ändern alter Tupel mit gleichem Primärschlüssel

`REPLACE`: Löschen aller alten Tupel, Einfügen der neuen Tupel

IXF-Dateien enthalten zusätzliche Schema-Informationen. Hier gibt es noch die Modi `REPLACE_CREATE` und `CREATE`, mit denen wenn nötig eine Tabelle angelegt wird. Zum Einspoolen von Daten in eine Tabelle müssen die nötigen Rechte gesetzt sein.

Für unsere Beispielanwendung der Filme und Serien stehen aufbereitete Textdateien mit Datensätzen unter /usr/projects/db2doc/data/ zur Verfügung. Jeder Datensatz steht in einer Zeile, Trennzeichen zwischen den Werten eines Datensatzes ist der senkrechte Strich '|'.

actors-1950+: alle Schauspieler mit den Filmen und Serien in denen sie seit 1950 mitgespielt haben (1.565.093 Datensätze); jeder Datensatz besteht aus den Werten 'Künstlername', 'Filmtitel', 'Drehjahr', 'Rolle', 'Rang in den Credits' (letztere beiden können auch leer sein)

actresses-1950+: alle Schauspieler mit den Filmen und Serien in denen sie seit 1950 mitgespielt haben (825.590 Datensätze); jeder Datensatz besteht aus den Werten 'Künstlername', 'Filmtitel', 'Drehjahr', 'Rolle', 'Rang in den Credits' (letztere beiden können auch leer sein)

biographies: biographische Daten zu Personen (135.087 Datensätze); jeder Datensatz besteht aus den Werten 'Künstlername', 'Geburtsname', 'Geburtsdatum', 'Todesdatum' (dabei können alle Werte außer Künstlername auch leer sein); das Datumsformat ist YYYY-MM-DD

business: wirtschaftliche Daten zu Produktionen (6.976 Datensätze); jeder Datensatz besteht aus den Werten 'Produktionstitel', 'Drehjahr', 'Budget', 'Studio' (letztere beiden können auch leer sein); das Budget ist als Zeichenkette mit Währungskürzel angegeben

composers, costume-designers, directors, writers: alle Komponisten, Designer und Regisseure mit den Produktionen an denen sie beteiligt waren (185.025, 52.242, 345.887 und 382.339 Datensätze); jeder Datensatz besteht aus den Werten 'Künstlername', 'Filmtitel', 'Drehjahr'

genres: alle Filme mit ihren Genre-Zuordnungen (353.404 Datensätze); für jedes Film/Genre-Paar enthält diese Datei einen Datensatz mit 'Filmtitel', 'Drehjahr' und 'Genre'

movie_links: alle Verknüpfungen zwischen zwei Produktionen (341.102 Datensätze); jeder Datensatz besteht aus den Werten 'Filmtitel' und 'Drehjahr' der ersten Produktion, 'Filmtitel' und 'Drehjahr' der zweiten Produktion, sowie 'Art der Verknüpfung'

movies: alle Filme (1.158.313 Datensätze); jeder Datensatz besteht aus den Werten 'Filmtitel', 'Drehjahr', 'Art', sowie der Information, ob der Film veröffentlich wurde ('released', 'unreleased', 'unfinished')

series: alle Serien (27.424 Datensätze); jeder Datensatz besteht aus den Werten 'Serientitel', 'Drehjahr', 'Anfangsjahr', 'Endjahr' und der Information, ob es eine Miniserie ist oder nicht ('true' oder 'false')

Angenommen, es existiert eine Tabelle SPIELT_IN mit den Spalten FILM VARCHAR(100), JAHR INTEGER und SCHAUSPIELER VARCHAR(100). Dann würde das folgende LOAD-Statement die Tabelle mit Daten füllen:

```
LOAD FROM "/usr/projects/db2doc/data/actors-1950+" OF DEL
        MODIFIED BY COLDEL|
     METHOD P (2,3,1)
     MESSAGES /tmp/"actors.log"
     INSERT INTO imdb.spielt_in
       (film, jahr, schauspieler)
```

Dabei verwendet der Import nur die jeweils ersten drei Werte jedes Datensatzes, vertauscht die Reihenfolge, um sie der Reihenfolge der Tabelle anzupassen und fügt die Datensätze hinzu, statt die Tabelle zu überschreiben.

Wenn die definierte Tabelle weniger oder genau die Daten enthält, die in einer der vorgegebenen Dateien enthalten sind, dann kann man das LOAD-Statement wie beschrieben benutzen. Was aber, wenn die Tabelle Daten aus zwei oder noch mehr Dateien zusammenfasst? Dann müssen die Daten zunächst in temporäre Tabellen gespoolt werden, und die Datensätze in der Datenbank zusammengeführt werden. Wer die Daten anderweitig vorbehandeln möchte, kann sich der mächtigen Kommandozeilen-Werkzeuge unter Linux bedienen:

cat NAME gibt eine Datei NAME auf die Standardausgabe aus (normalerweise der Bildschirm), mit | kann man die Ausgabe als Eingabe an ein anderes Programm übergeben, mit > kann man die Ausgabe in eine Datei umleiten, mit » wird die Ausgabe an eine bestehende Datei angehängt

sed EXPRESSION kann zum Ersetzen von Ausdrücken in einem Text benutzt werden

grep EXPRESSION filtert Texte nach bestimmten Ausdrücken, **grep -v** ist ein reverser Filter

uniq eliminiert Duplikate in einer sortierten Datei

sort sortiert eine Datei

column formatiert Eingabe in Spalten

awk ist ein Werkzeug zur Manipulation von Texten

Die folgende Kombination von Befehlen filtert aus der Datei `genres` alle Einträge für das Genre Comedy, separiert die Einträge am '|' und extrahiert die dritte Spalte. Die Ausgabe wird sortiert und Duplikate eliminiert, bevor das Ergebnis in eine Datei geschrieben wird. Als Resultat erhält man eine Liste fast aller Genres (außer Comedy). Mit `man BEFEHL` kann man sich die Hilfeseiten zu diesen Werkzeugen ansehen.

```
cat genres | grep -v Comedy | awk -vFS='|' '{ print $3}; ' | \
    sort | uniq > nearly_all_genres
```

Verändern von Tabellendefinitionen

Zum Ändern einer Tabellendefinition sind das ALTER- oder CONTROL-Recht an der zu ändernden Tabelle, das ALTERIN-Recht für das Schema der entsprechenden Tabelle oder ein übergeordnetes Recht nötig. Um eine Fremdschlüsselbeziehung einzuführen oder zu entfernen benötigt man für die referenzierte Tabelle zusätzlich noch das

REFERENCES-, das CONTROL-, das DBADM- oder das SYSADM-Recht. Entsprechend muß beim Löschen einer Schlüsselbedingung für alle die zugehörigen Attribute referenzierenden Tabellen die anfangs genannten Rechte bestehen.

Geändert wird eine Tabellendefinition entweder durch ein ALTER TABLE-Statement oder über das CC:

db2s2e80.pdf, Seite 41ff

```
ALTER TABLE imdb.movies
    ALTER title SET DATA TYPE VARCHAR (100)
    ADD released CHAR(1)
    DROP CONSTRAINT validyear
```

Das Beispiel demonstriert die drei weesentlichen Änderungsmöglichkeiten in einem ALTER TABLE-Statement für die Tabelle MOVIES im Schema IMDB:

- eine Attributdefinition wird geändert
  Über die ALTER-Klausel können VARCHAR Attribute vergrößert (aber nicht verkleinert) werden, außerdem kann durch diese Klausel der Audruck zur Erzeugung von automatisch generierten Attributwerten verändert werden.
- ein neues Attribut wird hinzugefügt
  Bei der Verwendung der ADD-Klausel gilt das gleiche wie beim Erstellen einer neuen Tabelle. Wird eine neue Schlüsselbedingung hinzugefügt, und existiert noch kein Index hierfür, so wird ein neuer Index angelegt. Die ADD COLUMN-Klausel wird stets zuerst ausgeführt.
- ein benannter CONSTRAINT wird aus der Tabelle gelöscht
  Man kann keine Attribute und Constrants löschen, die direkt bei einer Attributdefinition angegeben wurden.

Pflege von Tabelleninhalten

Löschen von Tupeln

Das Löschen von Tupeln geschieht mit dem DELETE-Statement. Dieses löscht alle Tupel aus der aktuellen Sicht, welche die Suchbedingung erfüllen. Man benötigt hierzu das DELETE-Recht oder das CONTROL-Recht auf der Tabelle/Sicht, das DBADM- oder das SYSADM-Recht. Tritt bei der Löschung eines Tupels ein Fehler aus, so bleiben **alle** Löschungen dieses Statements unwirksam.

db2s2e80.pdf, Seite 497ff

Wird keine Suchbedingung angegeben, so werden **alle** Tupel gelöscht.

```
DELETE FROM movies
       WHERE release_year < 1990

DELETE FROM persons
       WHERE name LIKE 'Schwarzenegger%'
          OR name LIKE 'Keitel%'
```

Ändern von Tupeln

Das Ändern von Tupeln geschieht mit dem UPDATE-Statement. Analog zum DELETE-Statement wird über eine Suchbedingung angegeben, welche Tupel der angegebenen Sicht/Tabelle verändert werden sollen. Wiederum müssen die nötigen

db2s2e80.pdf, Seite 737ff

Rechte gegeben sein: das UPDATE-Recht auf allen zu verändernden Attributen, das UPDATE-Recht auf der Tabelle/Sicht, das CONTROL-Recht auf der Tabelle/Sicht, das DBADM-Recht oder das SYSADM-Recht.

Das folgende UPDATE-Statement ändert alle `title` Attribute in der Tabelle `movies` auf Großschreibung:

```
UPDATE movies
  SET title=UCASE(title)
```

Einfügen von Tupeln

Das Einfügen von neuen Tupeln in eine bestehende Tabelle geschieht mit dem INSERT-Statement. Die Werte der einzufügenden Tupel müssen in Bezug auf Datentyp und Länge zu den jeweiligen Attributen passen. Außerdem müssen die einzufügenden Tupel alle Constraints (Schlüsselbedingungen, Fremdschlüssel und Checks) erfüllen. Wird in eine Sicht eingefügt, so muß diese das zulassen.

db2s2e80.pdf, Seite 603ff

Wird beim INSERT-Statement für ein Attribut kein Wert eingetragen, so wird (falls vorhanden) der Defaultwert oder der NULL-Wert eigetragen. Für jedes Attribut, das keinen Defaultwert besitzt und keine NULL-Werte zulässt, muß ein Wert angegeben werden.

```
INSERT INTO series (titel, drehjahr, start, ende, mini)
VALUES
     ('Buffy the Vampire Slayer', 1997, 1997, 2003, 'N'),
     ('Scrubs', 2001, 2001, NULL, 'N')

INSERT INTO persons (name, realname, dateofbirth, dateofdeath)
    SELECT alias, name, birth, death
    FROM actors
```

Vordefinierte Funktionen

Bei der Arbeit mit IBM DB2-SQL stehen eine Reihe von vordefinierten Funktionen zur Verfügung, mit denen Werte manipuliert werden können. Diese teilen sich in zwei Arten:

db2s1e80.pdf, Kapitel 3

Skalare Funktionen: Auswertung einer Liste von skalaren Parametern mit Rückgabe eines skalaren Wertes; werden in Ausdrücken benutzt

Aggregatfunktionen: Anwendung auf Spalten einer Gruppe bzw. einer Relation mit Rückgabe eines skalaren Wertes; werden z.B. in Anfragen benutzt

Man kann auch benutzerdefinierte Funktionen der beiden genannten Arten, sowie Tabellenfunktionen, die ganze Relationen zurückgeben, erstellen. Diese können dann genau wie die Systemfunktionen benutzt werden. Bei allen Funktionen ist darauf zu achten, dass die von ihnen zurückgegebenen Datentypen von den Parametertypen abhängen. Viele der Funktionen sind überladen, d.h. für unterschiedliche Datentypen definiert.

Skalare Funktionen

Die wichtigsten skalaren Funktionen seien hier genannt, zur Definition und ausführlichen Beschreibung der Funktionen sei auf das Handbuch oder die Online-Dokumentation verwiesen.

Typkonvertierung: BIGINT, BLOB, CHAR, CLOB, DATE, DBCLOB, DECIMAL, DREF, DOUBLE, FLOAT, GRAPHIC, INTEGER, LONG, LONG_VARCHAR, LONG_VARGRAPHIC, REAL, SMALLINT, TIME, TIMESTAMP, VARCHAR, VARGRAPHIC

Mathematik: ABS, ACOS, ASIN, ATAN, CEIL, COS, COT, DEGREES, EXP, FLOOR, LN, LOG, LOG10, MOD, POWER, RADIANS, RAND, ROUND, SIGN, SIN, SQRT, TAN, TRUNC

Stringmanipulation: ASCII, CHR, CONCAT, DIFFERENCE, DIGITS, HEX, INSERT, LCASE, LEFT, LENGTH, LOCATE, LTRIM, POSSTR, REPEAT, REPLACE, RIGHT, RTRIM, SOUNDEX, SPACE, SUBSTR, UCASE, TRANSLATE

Datumsmanipulation: DAY, DAYNAME, DAYOFWEEK, DAYOFYEAR, DAYS, HOUR, JULIAN_DAY, MICROSECOND, MIDNIGHT_SECONDS, MINUTE, MONTH, MONTHNAME, QUARTER, SECOND, TIMESTAMP_ISO, TIMESTAMPDIFF, WEEK, YEAR

System: EVENT_MON_STATE, NODENUMBER, PARTITION, RAISE_ERROR, TABLE_NAME, TABLE_SCHEMA, TYPE_ID, TYPE_NAME, TYPE_SCHEMA

Sonstige: COALESCE, GENERATE_UNIQUE, NULLIF, VALUE

Ausdrücke

Ausdrücke werden in SELECT-Klauseln und in Suchbedingungen benutzt, um Werte darzustellen. Sie setzen sich aus einem oder mehreren *Operanden*, Klammern und unären oder binären *Operatoren* zusammen: + (Summe, positiver Wert), - (Differenz, negativer Wert), * (Produkt), / (Quotient) und || (Stringkonkatenation).

db2s1e80.pdf, S. 187ff

Als Operanden sind erlaubt:

Attributnamen: möglicherweise qualifiziert durch Tabellennamen oder Tupelvariablen

Konstanten: ganze Zahlen (12, -10), Dezimalzahlen (2.7, -3.24), Fließkommazahlen (-12E3, 3.2E-12), Zeichenketten ('hello'), Datumswerte ('12/25/1998','25.12.1998','1998-12-25'), Zeitwerte ('13.50.00', '13:50', 1:50 PM'), Zeitstempel ('2001-01-05-12.00.00.000000')

Funktionen

Zeitdauerangaben: bei arithmethischen Operationen mit Datums-, Zeit- oder Zeitstempelwerten können Zeitdauern benutzt werden, z.B. 5 DAYS, 1 HOUR (MONTH/S, DAY/S, HOUR/S, SECOND/S, MICROSECOND/S)

Registerwerte: Hierunter fallen

CURRENT DATE: aktuelles Datum

CURRENT TIME: aktuelle Uhrzeit

CURRENT TIMESTAMP: aktueller Zeitstempel

CURRENT TIMEZONE: aktuelle Zeitzone
CURRENT NODE: aktuelle Nodegruppennummer
CURRENT SCHEMA: aktuelles Schema
CURRENT SERVER: Servername
USER: Benutzername

```
SELECT name
FROM stars
WHERE DAYOFYEAR(birthdate) = DAYOFYEAR(current date)
```

Typumwandlungen: Explizite Umwandlung in einen bestimmten Typ, z.B. CAST (NULL AS VARCHAR(20)) oder CAST(Gehalt*1.2345 AS DECIMAL(9,2))

Subqueries: liefert eine Anfrage immer einen skalaren Wert zurück, dann kann man diese als Operand benutzen

Fallunterscheidungen: z.B.

```
CASE movietype
    WHEN 'M' THEN 'Kinofilm'
    WHEN 'V' THEN 'Videofilm'
    WHEN 'G' THEN 'Videospiel'
    WHEN 'T' THEN 'Fernsehspiel'
END

CASE
    WHEN YEAR(release_date) < 1945 THEN 'B/W'
    WHEN YEAR(release_date) = 1984 THEN 'Big Brother Is Watching'
ELSE 'Colour'
END
```

Die Bedingungen werden der Reihe nach ausgewertet. Ergebnis ist der erste als wahr evaluierende Fall, sonst der Wert in der ELSE-Klausel. Gibt es keine ELSE-Klausel so wird als Defaultwert NULL genommen. Der Ergebnisausdruck muß die Bestimmung des Datentyps des Gesamtausdrucks zulassen, insbesondere müssen die Ergebnistypen der einzelnen Ausdrücke kompatibel und dürfen nicht alle NULL sein.

Prädikate

Prädikate sind logische Tests, die in Suchbedingungen verwendet werden können. Sie können als Werte *true*, *false* und *unknown* annehmen. Die bekannten Prädikate sind:

db2s1e80.pdf, S. 225ff

- Vergleichsprädikate: =, <>, <, >, <=, >= (Vergleiche mit NULL-Werten ergeben *unknown*)
- NOT als Verneinung eines Prädikates
- BETWEEN, z.B. runtime BETWEEN 1 AND 3
- NULL, z.B. dateofdeath IS NOT NULL
- LIKE, Zeichenkettenvergleich mit Platzhaltern _ für ein Zeichen oder % für beliebig viele: name LIKE 'Schr_der%' würde z.B. auf 'Schröder', 'Schrader' oder 'Schrederbein' passen
- EXISTS, Test auf leere Menge

- IN, Test auf Vorkommen in einer Kollektion von Werten

Es ist möglich, Vergleichsprädikate durch SOME, ANY oder ALL zu quantifizieren und dadurch ein einzelnes Tupel mit einer Menge von Tupeln zu vergleichen. Beispiele:

```
genre IN ('Fantasy','Horror','Comedy','Drama')

('1991','Fantasy') = ANY (SELECT release_year, genre
                          FROM movie_genres)

EXISTS (SELECT *
        FROM movies
        WHERE budget > 100000000)

100000000 > ALL (SELECT budget FROM movies)
```

Aufgaben

(a) Benutzt die Befehle EXPORT, IMPORT und LOAD, um Eure in der letzten Sitzung angelegte Datenbank **IMDBxx** mit Daten zu füllen. Falls Ihr feststellt, dass die existierenden Tabellen unzureichend definiert sind, verwendet nach Möglichkeit das ALTER TABLE-Statement.

(b) Legt eine neue Tabelle für Schauspieler an, die Attribute `Name` und `Geschlecht` besitzt, oder erweitert Eure existierte Tabelle entsprechend. Das `Geschlecht` kann entweder 'männlich' oder 'weiblich' sein.

Die Dateien `actors-1950+` und `actresses-1950+` enthalten Daten zu Schauspielern, respektive Schauspielerinnen. Fügt diese Datensätze in die neue (oder geänderte Tabelle) ein, setzt dabei das Attribut `Geschlecht` auf den korrekten Wert.

(c) Schreibt die folgenden SQL-Statements und führt die Änderung auf Eurer Datenbank durch:

(*i*) Legt eine neue Tabelle für die Daten der Datei `costume-designer` an. Die nächsten Aufgaben beziehen sich auf diese Tabelle.

(*ii*) Führt ein ALTER TABLE-Statement aus, um ein neues Attribut `Aufgabe` einzuführen. Benutzt UPDATE, um bei *jedem* Kostümdesigner die `Aufgabe` 'Designer' einzutragen.

(*iii*) Wenn zwei Personen den gleichen Namen haben, wurde dieser in der Datenbasis durch Nachstellung einer römischen Zahl in Klammern eindeutig gemacht. Schreibt ein UPDATE-Statement, das die römischen Zahlen durch arabische Zahlen ersetzt:

I = 1, II = 2, III = 3, IV = 4, V = 5, VI = 6, VII = 7, VIII = 8, IX = 9, X = 10, XI = 11, XII = 12, XIII = 13, XIV = 14, XV = 15

(*iv*) Löscht alle Tupel aus der Tabelle, die sich auf Filme beziehen, die nach dem **aktuellen Datum** fertiggestellt werden. Hinweis: Benutzt für den Vergleich einen Registerwert.

(d) Formuliert drei eigene Aufgaben zu INSERT, UPDATE und DELETE (jeweils eine Aufgabe pro Statement) und gebt dazu die SQL-Statements an. **Dies ist eine Heimaufgabe. Bitte bereitet sie zur nächsten Praktikumssitzung vor!**