Praktikum Datenbanken / DB2
Woche 5: Füllen von Tabellen und Pflege von Tabelleninhalten

Betreuer: Gudrun Fischer, Tobias Tuttas, Camille Pieume
Raum: LF 230
Bearbeitung: 08., 12. und 13. Juni 2006

Datum	
Team (Account)	
Vorbereitung	
Präsenz	

Aktuelle Informationen, Ansprechpartner und Material unter:
`http://www.is.informatik.uni-duisburg.de/courses/dbp_ss06/index.html`

Wochenziele

In dieser Woche füllen wir das erstellte Datenbankschema mit Testdaten zu fiktiven Anwendern und mit realen Daten über Filme und Serienfolgen aus der Internet Movie Database (`www.imdb.org`).

Außerdem wenden wir einige im System bereits vordefinierte Funktionen an, um E-Mail-Adressen auf syntaktische Korrektheit zu überprüfen.

Pflege von Tabelleninhalten

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 kategorie ('name')
VALUES
     ('Horror'), ('Komödie')

INSERT INTO produktion
    SELECT titel, drehjahr, 'F' AS art
    FROM imdbfilme
```

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 anwender
       WHERE nick='Hase'

DELETE FROM produktion
       WHERE titel NOT LIKE '%star%wars%'
         OR drehjahr>2000
```

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 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.

db2s2e80.pdf, Seite 737ff

```
UPDATE gespeichertauf
  SET sprache = UCASE(sprache)

UPDATE anwender
  SET adresse =
    (SELECT adresse FROM anwender
       WHERE name='Mouse' AND vorname='Minnie')
  WHERE name='Mouse' AND vorname='Mickey'
```

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 benutzt man das EXPORT-Statement.

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. Das einfachste SELECT-Statement

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

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

```
EXPORT TO filme.ixf OF ixf
  SELECT filmtitel,drehjahr
  FROM imdbfilme
  WHERE filmtitel like 'B%'
```

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 benutzen.

db2dme80.pdf, Kapitel 2

```
IMPORT FROM filme.ixf OF ixf
  INSERT INTO meinefilme
```

Eine andere Möglichkeit ist das mächtigere und komfortablere LOAD-Statement, das im nächsten Abschnitt behandelt wird.

Load

Das Einspoolen von Daten in eine Tabelle aus einer beliebigen Textdatei kann mit den Kommandos IMPORT oder LOAD geschehen. Dabei ist der Dateiname anzugeben, der Dateityp (ASC = Textdatei ohne Begrenzer, DEL = Textdatei

db2dme80.pdf, Kapitel 3

mit Begrenzer, IXF), die Logdatei und zusätzliche Optionen. Die genaue Syntax und die möglichen Optionen sind im Handbuch beschrieben.

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.

Angenommen, es existiert eine Tabelle `filme` mit den Spalten `titel VARCHAR(30)` und `drehjahr DATE`, sowie eine Textdatei mit Drehjahren, Produzenten und Titeln von Filmen, jeweils durch ein ‚|' getrennt. Dann würde das folgende LOAD-Statement aus jeder Zeile der Textdatei das 3. und 1. Element auswählen und mit diesen Daten die Tabelle füllen:

```
LOAD FROM "textdatei" OF DEL MODIFIED BY COLDEL|
     METHOD P (3,1)
     MESSAGES "logfile"
     INSERT INTO produktion
       (titel, drehjahr)
```

Es werden also bei diesem Import nur jeweils zwei der drei Werte jeder Zeile in vertauschter Reihenfolge hergenommen (um sie der Reihenfolge der Tabellenattribute anzupassen). Durch `INSERT` werden die Daten in die Tabelle hinzugefügt, statt die bestehenden Daten zu überschreiben.

Optionen für IDENTITY-Spalten

Falls in der Zieltabelle `IDENTITY`-Spalten definiert sind, dann kann man bei `IMPORT`- und `LOAD`-Befehlen explizit angeben, wie diese Spalten zu behandeln sind.

Mögliche Optionen sind:

- `IDENTITYMISSING` – Diese Option besagt, dass die Quelldaten keine eigene `IDENTITY`-Spalte enthalten. Das System generiert also beim Einfügen automatisch passende, neue Werte.

  ```
  IMPORT FROM filme.ixf OF ixf
    MODIFIED BY IDENTITYMISSING
    INSERT INTO meinefilme
  ```

- `IDENTITYIGNORE` – Bei dieser Option sind zwar in den Quelldaten IDs enthalten, diese werden jedoch ignoriert und durch neue, vom System generierte ersetzt.

- `IDENTITYOVERRIDE` – Hier werden die IDs aus den Quelldaten übernommen, selbst wenn sie in der Zieltabelle schon vorhanden sind. Da so unter Umständen vorhandene Daten ersetzt oder modifiziert werden können, ist diese Option nur beim `LOAD`-Befehl verfügbar.

```
LOAD FROM "textdatei" OF DEL
  MODIFIED BY IDENTITYOVERRIDE
    INSERT INTO Produktion
       (ProduktionsID,Titel,Drehjahr)
```

Weitere Informationen zu `IDENTITY`-Spalten und ihren Zusammenhang mit `IMPORT`, `EXPORT` und `LOAD` findet Ihr im Netz unter:

`http://www.ibm.com/developerworks/db2/library/techarticle/0205pilaka/0205pilaka2.html`

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 politik
WHERE YEAR(unabhängigkeit) > 1980
```

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 art
    WHEN 'applicant' THEN 'Anwärter'
    WHEN 'member' THEN 'Vollmitglied'
    WHEN 'associate member' THEN 'Assoziiertes Mitglied'
    WHEN 'observer' THEN 'Beobachter'
END

CASE
    WHEN YEAR(unabhängigkeit) > 1945 THEN 'Nach WK II'
    WHEN YEAR(unabhängigkeit) > 2006 THEN 'Huch!?'
ELSE 'Vor dem Ende des 2. WK'
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:

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

('1871,'federal republic') = ANY (SELECT year(unabhängigkeit),
                                         regierung
                                  FROM politik)

EXISTS (SELECT *
        FROM land
        WHERE bevölkerung > 1000000000)

1000000000 > ALL (SELECT bevölkerung FROM land)
```

Übersicht über die neuen Befehle

export to *file* of *type*	exportiere Daten in eine Datei
import from *file* of *type*	importiere Daten aus einer Datei
load from *file* of *type*	importiere Daten mit Festlegung der Importmethode
delete from *name* where *condition*	lösche Tupel aus einer Tabelle auf die *Condition* zutrifft
update *name* set *assignment* where *condition*	ändere den Inhalt von Tupeln auf die *Condition* zutrifft
insert into *name* values	füge direkt neue Werte in eine existierende Tabelle ein
insert into *name statement*	füge das Ergebnis eines *Statements* als neue Werte in eine existierende Tabelle ein

Aufgaben

Beispieldatenbank

Unter der Instanz `dbprak` auf dem Datenbank–Server `salz` existiert (*ab dem 08. Juni*) neben der Datenbank `sample` auch noch eine zweite Datenbank `filme`. Diese ist mit Beispieldaten aus der Internet Movie Database (`http://www.imdb.org`) gefüllt.

Das Schema der Beispiel-Datenbank ist wie folgt:

- **Film:** Drehjahr, Titel
- **Serienfolge:** FolgenTitel, FolgenNr, StaffelNr, Drehjahr, SerienTitel
- **Schauspieler:** ID, Name, Vorname
- **spieltInFilm:** SchauspielerID (verweist auf `ID` in `Schauspieler`), Drehjahr, Titel, Rolle
- **spieltInFolge:** SchauspielerID (verweist auf `ID` in `Schauspieler`), Drehjahr, SerienTitel, StaffelNr, FolgenNr, Rolle

Wir wollen diese Daten mit unserer Video- und DVD-Datenbank zusammenführen. Dazu müssen die IMDB-Daten in Eure eigenen Datenbanken überführt werden. In dieser Woche sollen jedoch nur *Filme* und *Serienfolgen* betrachtet werden.

Vorbereitung (Hausaufgaben zur Semesteraufgabe)

V1: Testdaten vorbereiten

Schreibt Euch die nötigen `INSERT`-Befehle auf, um folgende Testfakten in Eure Datenbank einzufügen:

1) Ein neuer Anwender trägt sich in das System ein. Er heißt Adam Boyce-Codd, wohnt in Duisburg und hat die Telefonnummern 0203 1234567 und 0173 7654321. Seine E-Mail-Adresse lautet „dbprak@is.informatik.uni-duisburg.de". Sein Passwort lautet „rumpelstilzchen". Als Nick wählt er „Adam".
   Eva Musterfrau wohnt in Oberhausen, hat kein Telefon, ist aber unter der E-Mail-Adresse „eva@nowhere-in-the.net" zu erreichen. Ihr Passwort lautet „rotkaeppchen". Ihr Nick lautet „Ophelia".
2) Adam Boyce-Codd besitzt den Film „Pirates of the Caribbean: The Curse of the Black Pearl" aus dem Jahr 2003 als „Special Edition" auf 2 DVDs. Die DVDs enthalten den Film in den Sprachen Deutsch und Englisch.
3) Eva Musterfrau hat von der Serie „Buffy" aus dem Jahr 1997 die ersten zwei Folgen auf Deutsch auf einer Videokassette aufgenommen. Außerdem enthält die Videokassette schon den Film „For the Birds" aus dem Jahr 2000 auf Englisch.

4) Adam Boyce-Codd leiht sich von Eva am 01.06.2006 die Videokassette mit den Buffy-Folgen 1 und 2 und gibt sie ihr am 03.06.2006 zurück.

Wenn Details dieser Daten noch nicht in Eurem Schema ausdrückbar sind, dann notiert Euch, welche Tabellen Ihr wie ändern müsst.

V2: EXPORT / IMPORT vorbereiten

Überlegt Euch für die Tabellen aus der Beispieldatenbank `Filme`, wie Ihr die darin enthaltenen Film- und Serienfolgen-Daten sinnvoll in Euer Datenbankschema integrieren könnt. Filme und Serienfolgen sollen auf jeden Fall in Eure jeweiligen *eigenen* Tabellen übernommen werden.

Schreibt Euch die notwendigen `EXPORT`-, `IMPORT`- und/oder `LOAD`-Befehle auf.

Hinweis:

Die Schauspieler und die entsprechenden Beziehungen werden wir erst in Woche 6 hinzufügen.

V3: Stringfunktionen

Schlagt die Stringfunktionen aus dem Abschnitt „Vordefinierte Funktionen" in der Online-Dokumentation von DB2 nach.

Wie kann man mit Hilfe dieser Funktionen überprüfen, ob eine E-Mail-Adresse (die ja erstmal ein String ist) *genau ein* ‚@'-Zeichen enthält?

Präsenz

P1: Testdaten einfügen

Fügt die Testdaten aus V1 in Eure Datenbank ein.

Hinweis:

Wenn Ihr zwischendurch Testdaten aus einer Tabelle löscht, die eine automatisch generierte ID enthält, so wird beim nächsten Einfügen die ID trotzdem einfach weiter hochgesetzt. Wenn die IDs wieder bei 1 anfangen sollen, könnt Ihr die ID-Generierungssequenz mit dem Befehl `ALTER TABLE` neu starten:

`ALTER TABLE` *Tabellenname* `ALTER COLUM` *ID-Spalte* `RESTART`

P2: Importieren von Fremddaten

Exportiert die Daten zu Filmen und Serienfolgen (nicht aber zu Schauspielern) aus der Beispieldatenbank `Filme` (Server `salz`, Instanz `dbprak`).

Importiert die Daten dann in Eure eigene Datenbank. Achtet dabei darauf, dass die vorhandenen Testdaten mit den neuen Daten sinnvoll kombiniert werden.

Löscht danach die Export-Dateien!

P3: Gültigkeitsprüfung für die E-Mail-Adresse

Fügt Eurer Tabelle `Anwender` einen benannten `CONSTRAINT` namens `validMail` hinzu, der überprüft, ob die E-Mail-Adresse genau ein ‚@'-Zeichen enthält (also mindestens eins, und gleichzeitig höchstens eins).

Zum Schluss ...
stoppt bitte Eure Datenbank-Manager-Instanz auf `salz`,
loggt Euch von `salz` und vom lokalen Rechner aus,
aber schaltet den Rechner nicht ab.