**Praktikum Datenbanken / DB2**
**Woche 5: SQL als Datenmanipulationssprache**

Raum: LF 230 Bearbeitung: 23.-30. Mai 2005

| | |
|---|---|
| **Datum** | |
| **Gruppe** | |
| | |
| | |
| **Vorbereitung** | |
| **Präsenz** | |

Aktuelle Informationen unter:
`http://www.is.informatik.uni-duisburg.de/courses/dbp_ss05/`

# Füllen von Tabellen

Unter der Instanz `dbprak` auf dem Datenbank–Server `salz` existiert neben der Datenbank `sample` auch noch eine zweite Datenbank `almanach`. Diese ist mit geographischen und politischen Beispieldaten aus der Mondial–Datensammlung gefüllt. Wir wollen diese im Praktikum benutzen. Dazu müssen die Daten allerdings zunächst einmal in Eure eigenen Datenbanken überführt werden.

## Beispieldatenbank

Das Schema der Beispiel-Datenbank ist wie folgt (nicht alle Daten werden unbedingt benötigt):

- **Land:** Name, Code (Landeskürzel), Hauptstadt, Provinz (der Hauptstadt), Gebiet, Bevölkerung
- **Wirtschaft:** Land (Landeskürzel), BSP (Bruttosozialprodukt), Landwirtschaft, Dienstleistung, Industrie (jeweils Prozentanteil am BSP), Inflation
- **Bevoelkerung:** Land (Landeskürzel), Wachstum, Kindersterblichkeit
- **Politik:** Land (Landeskürzel), Unabhängigkeit, Regierung (Regierungsform)
- **Sprachen:** Land (Landeskürzel), Name, Prozentanteil

- **Religion:** Land (Landeskürzel), Name, Prozentanteil
- **Grenze:** Land1, Land2 (jeweils Landeskürzel), Laenge
- **Kontinent:** Name, Gebiet
- **Landmasse:** Land (Landeskürzel), Kontinent, Prozent
- **Stadt:** Name, Land (Landeskürzel), Provinz, Bevoelkerung, Laengengrad, Breitengrad
- **Provinz:** Name, Land (Landeskürzel), Gebiet, Bevoelkerung, Hauptstadt (der Provinz), ProvHS (Provinzname der Provinzhauptstadt)
- **Organisation:** Name, Abkuerzung, Stadt, Land, Provinz (des Sitzes der Organisation), Gruendung
- **Mitglied:** Organisation (Abkürzung), Land (Landeskürzel), Art
- **See:** Name, Gebiet
- **Fluss:** Name, Fluss (mündet in...), See (mündet in...), Meer (mündet in...), Laenge
- **Meer:** Name, Tiefe
- **Berg:** Name, Hoehe, Laengengrad, Breitengrad
- **Insel:** Name, Inseln (Inselgruppe), Gebiet, Laengengrad, Breitengrad
- **Wueste:** Name, Gebiet
- **Lage_Berg:** Berg, Land, Provinz (analog: *Lage_Insel, Lage_Wueste, Lage_Fluss, Lage_See, Lage_Meer*)

## 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 laender.ixf OF ixf
  SELECT name,code,bevoelkerung
  FROM land
```

```
  WHERE name like 'A%'
```

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 laender.ixf OF ixf
  INSERT INTO meinelaender
```

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 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.

Angenommen, es existiert eine Tabelle `laender` mit den Spalten `name VARCHAR(100)`, `gegruendet DATE` und `regierungsform VARCHAR(100)`, sowie eine Textdatei mit Namen, Länderkürzel, Regierungsform und Unabhängigkeitsdatum getrennt durch ein „|". Dann würde das folgende LOAD-Statement aus jeder Zeile der Textdatei das 2., 4. und 3. Element auswählen und mit diesen Daten die Tabelle füllen:

```
LOAD FROM "textdatei" OF DEL MODIFIED BY COLDEL|
     METHOD P (2,4,3)
     MESSAGES "logfile"
     INSERT INTO laender
       (name, gegruendet, regierungsform)
```

Es werden also bei diesem Import nur jeweils drei der vier 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.

# 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 durch ein ALTER TABLE-Statement: *db2s2e80.pdf, Seite 41ff*

```
ALTER TABLE dbprak.staff
    ALTER name SET DATA TYPE VARCHAR (100)
    ADD hasemail CHAR(1)
    DROP CONSTRAINT validbirthdate
```

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

- eine Attributdefinition wird geändert

  Über die ALTER-Klausel können VARCHAR Attribute vergrößert (aber nicht verkleinert) werden, außerdem kann durch diese Klausel der Ausdruck 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 löschen und keine Constrants oder Checks, die direkt und ohne Bezeichner 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 land
       WHERE bevölkerung < 1000000

DELETE FROM politik
       WHERE regierung NOT LIKE '%republic%'
         AND regierung NOT LIKE '%democracy%'
```

### 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 land
  SET title = UCASE(name)

UPDATE mitglied
  SET art = (SELECT art FROM mitglied WHERE land='D')
  WHERE land='F'
```

### 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 land ('name')
VALUES
     ('Phantasia'), ('Atlantis')

INSERT INTO provinz
    SELECT name, bevoelkerung, gebiet, hauptstadt
    FROM land
```

## 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

*db2s1e80.pdf, Kapitel 3*

sich in zwei Arten:

**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

*db2s1e80.pdf, S. 187ff*

Wert), - (Differenz, negativer Wert), * (Produkt), / (Quotient) und || (Stringkonkatenation).

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) > 2005 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:

```
genre 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 |
| alter table *name* | ändere die Definiation einer existierenden Tabelle |
| 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

## Zur Vorbereitung

(a) Formuliert die notwendigen Export-Statements, um die zum Füllen Eurer Datenbank nötigen Daten aus den Tabellen der Beispieldatenbank zu exportieren.

(b) Angenommen, die Text- oder IXF-Dateien mit den exportierten Daten liegen bereits vor. Formuliert die notwendigen Import-Statements, um Eure Datenbank aus diesen Dateien aufzufüllen.

(c) Formuliert die INSERT, UPDATE und DELETE-Statements für Präsenzaufgabe (c).

## Präsenzaufgaben

(a) Wie oben beschrieben existiert bereits eine Beispieldatenbank `almanach` mit den geographischen Daten unter der Instanz `dbprak` auf `salz`. Verbindet Euch zunächst lokal mit der Beispieldatenbank `almanach`.

Ein lokaler Knoten für den Zugriff auf die Instanz sollte auf früherer Übung bereits existieren. Eventuell muß man an diesem Knoten noch die Datenbank `almanach` unter einem Alias (z.B. `vorgabe`) lokal katalogisieren, falls noch nicht geschehen.

Wenn Ihr den Vorgaben der Aufgabenblätter gefolgt seid, sollten nun folgende Knoten im *node directory* existieren:

- `salz` – Zugriff auf die Instanzt `dbprak`
- `salzXX` – Zugriff auf Eure eigene Instanz `dbps05XX`

Außerdem sollten an diesen Knoten die folgenden Datenbanken katalogisiert sein (falls Ihr die Namen wie vorgegeben gewählt habt):

- `almanach` – Eure selbsterstellte Datenbank lokal auf dem Rechner `salz.is.informatik.uni-duisburg.de`
- `almanXX` – Zugriff auf Eure Datenbank `almanach` über den Knoten `salzXX`
- `vorgabe` – Zugriff auf die vorgegebene Datenbank `almanach` über den Knoten `salz`
- `sampleXX` – Zugriff auf die Beispiel-Datenbank `sample` über den Knoten `salz`

(b) Benutzt die Befehle EXPORT, IMPORT und LOAD, um Eure in der letzten Sitzung angelegte und katalogisierte Datenbank **almanXX** mit Daten zu füllen. Betrachtet vorher die Tabellendefinitionen der vorgegebenen Datenbank, ob Ihr beim Importieren der Daten eventuell Datentypkonversionen durchführen müsst.

Falls Ihr feststellt, dass Eure existierenden Tabellen unzureichend definiert sind, verwendet nach Möglichkeit das ALTER TABLE-Statement.

(c) Nehmen wir an, dass die Integration der Europäischen Union in ein paar Jahrzehnten große Fortschritte gemacht hat. Aus dem Staatenbund (eine überstaatliche Organisation) wird ein Bundesstaat, der als ein Land in die Tabelle aufgenommen werden soll.

($i$) Großbritannien geht diese Entwicklung zu weit, daher tritt es aus der „Europäischen Union" aus. Schreibt ein SQL–Statement, das diese Änderung bewirkt und führe es aus.

(*ii*) Schreibt ein oder mehrere SQL–Statement(s), um das neue Land „Vereinigte Staaten von Europa" in Eure Datenbank einzufügen und führt es oder sie aus. Die VSE sollen als Hauptstadt den alten Sitz der „Europäischen Union" erhalten. Die übrigen Werte sollen zunächst ungesetzt bleoben.

(*iii*) Jahr der Unabhängigkeit der VSE soll das Gründungsjahr der „Europäischen Union" sein. Staatsform die Staatsform der alten Bundesrepublik Deutschland. Schreibt ein oder mehrere SQL–Statement(s), um diese Information in die Datenbank einzufügen und führt es oder sie aus.

(*iv*) Für eine Übergangszeit bleibt die Europäische Union (nun Vereinigte Staaten von Europa) noch als Organisation bestehen. Ersetze die Mitgliedschaften in der Europäischen Union durch Mitgliedschaften in den VSE.