

Praktikum Datenbanken / DB2
Woche 7: Noch mehr SQL (Fortgeschrittene Konzepte)

Raum: LF 230

Nächste Sitzung: 24./27. 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.

Index

Ein Index ist ein Datenbankobjekt, das den Zugriff über bestimmte Attribute einer Tabelle beschleunigen soll und dies in den allermeisten Fällen auch tut. Man kann auch (UNIQUE)-Indexe verwenden, um die Einhaltung von Schlüsselbedingungen sicherzustellen.

Werden bei der Tabellendefinition Primärschlüssel- (PRIMARY KEY) oder Sekundärschlüsselbedingungen (UNIQUE) angegeben, so legt DB2 selbständig Indexe an, um die Einhaltung dieser sicherzustellen. Ein einmal angelegter Index wird dann automatisch bei Änderungen an den Inhalten der entsprechenden Tabelle gepflegt. Dies ist natürlich mit einem erhöhten Verwaltungsaufwand verbunden.

Näheres zu Indexen und den zur Realisierung verwendeten Datenstrukturen in der Vorlesung **Datenbanken** oder in der begleitenden Literatur (siehe Blatt der ersten Woche).

Das Anlegen eines Index geht mit dem CREATE INDEX-Statement, das Löschen entsprechend über DROP INDEX. Man benötigt dazu mindestens das INDEX- oder das CONTROL-Recht für die Tabelle, sowie das CREATEIN-Recht im angegebenen Schema. Der Benutzer, der einen Index anlegt, erhält auf diesem das CONTROL-Recht.

*db2s2e81.pdf,
S. 268*

Über das Schlüsselwort UNIQUE kann man bei der Erstellung eines Index angeben, dass keine Tupel in allen Werten der angegebenen Attribute übereinstimmen dürfen. Sollten zum Zeitpunkt der Indexgenerierung Tupel in der Tabelle dagegen verstoßen, so wird eine Fehlermeldung ausgegeben und der Index nicht angelegt.

Es ist nicht möglich, zwei gleiche Indexe anzulegen. Dabei darf ein Index 16 Attribute umfassen und die Attribute dürfen zusammen nicht länger als 1024 Bytes sein. Die Ordnung ASC bzw. DESC gibt an, ob die Indexeinträge in aufsteigender bzw. absteigender Reihenfolge angelegt werden. Nur wenn UNIQUE angegeben wurde, können über die INCLUDE-Klausel weitere Attribute in den Index aufgenommen werden, für die aber keine Schlüsselbedingung gelten soll.

Beispiel:

```
CREATE INDEX studios  
ON produktion (studio)
```

Indexe werden nicht explizit aufgerufen, sondern vom Datenbank-Managementsystem implizit bei der Bearbeitung von Anfragen herangezogen.

Rekursive Anfragen

In einem vollen SELECT-Statement ist es auch möglich, rekursive Anfragen zu stellen. Rekursive Anfragen sind solche Anfragen, die sich in ihrer Definition auf sich selbst beziehen. Derart kann man etwa die *transitive Hülle* eines Tupels oder Stücklisten berechnen.

db2s1e81.pdf,
S. 601

Ein kurzer Ausflug, um das Konzept der Rekursion zu erklären:

Rekursion

aus Wikipedia, der freien Enzyklopädie

(<http://de.wikipedia.org/wiki/Rekursion>)

Rekursion bedeutet Selbstbezüglichkeit. Sie tritt immer dann auf, wenn etwas auf sich selbst verweist oder mehrere Dinge aufeinander, so dass merkwürdige Schleifen entstehen. So ist z.B. der Satz "Dieser Satz ist unwahr" rekursiv, da er von sich selber spricht. [...]

Dabei ist Rekursion ein allgemeines Prinzip zur Lösung von Problemen, das in vielen Fällen zu "eleganten" mathematischen Lösungen führt. Als Rekursion bezeichnet man den Aufruf bzw. die Definition einer Funktion durch sich selbst. Ohne geeignete Abbruchbedingung geraten solche rückbezüglichen Aufrufe in einen sog. infiniten Regress [Endlosrekursion].

[...] Die Definition von rekursiv festgelegten Funktionen ist eine grundsätzliche Vorgehensweise in der funktionalen Programmierung. Ausgehend von einigen gegebenen Funktionen (wie zum Beispiel unten die Summenfunktion) werden neue Funktionen definiert, mithilfe derer weitere Funktionen definiert werden können.

Hier ein Beispiel für eine Funktion $sum : N_0 \rightarrow N_0$, die die Summe der ersten n Zahlen berechnet:

$$sum(n) = \begin{cases} 0, & \text{falls } n = 0 \text{ (Rekursionsbasis)} \\ sum(n-1) + n, & \text{falls } n \neq 0 \text{ (Rekursionsschritt)} \end{cases}$$

Bei einer rekursiven Anfrage (RA) muß man folgende Regeln beachten:

- Jedes Fullselect, das Teil einer RA ist, muß mit SELECT beginnen (aber nicht mit SELECT DISTINCT). Es darf nicht geschachtelt sein. Als Vereinigung **muß** UNION ALL benutzt werden.
- Im WITH-Teil müssen Spaltennamen explizit angegeben werden.
- Das erste Fullselect in der ersten Vereinigung darf sich nicht auf die zu definierende Tabelle beziehen. Es bildet die Rekursionsbasis.
- Die Datentypen und Längen der Spalten der zu definierenden Tabelle werden durch die SELECT-Klausel der Rekursionsbasis festgelegt.
- Kein Fullselect in der rekursiven Definition darf eine Aggregatfunktion, eine GROUP BY- oder eine HAVING-Klausel enthalten. Die FROM-Klauseln in der rekursiven Definition dürfen höchstens eine Referenz auf die zu definierende Tabelle besitzen.
- Es dürfen keine Subqueries verwendet werden.

Wenn die Daten Zyklen enthalten können, dann ist es möglich, dass durch eine RA eine Endlosrekursion erzeugt wird. Dann ist es wichtig, eine Abbruchbedingung zu definieren:

- Die zu definierende Tabelle muß ein Attribut enthalten, das durch eine INTEGER-Konstante initialisiert wird, und regelmäßig erhöht wird.
- In jeder WHERE-Klausel muß eine Prüfbedingung auftreten.

Die Auswertung einer rekursiven Anfrage läuft wie folgt ab: Zuerst werden die Tupel der Rekursionsbasis berechnet, dann werden ausgehend von diesen Tupeln gemäß des zweiten Teils des Fullselects (nach dem UNION ALL) weitere Tupel berechnet. Dabei werden jedoch nur diejenigen Tupel aus der rekursiv definierten Relation verwendet, die beim letzten Berechnungsschritt neu hinzugekommen sind (beginnend also mit den Tupeln der Rekursionsbasis). Kamen in einem Berechnungsschritt keine neuen Tupel hinzu, so endet die Berechnung.

Beispiel:

Gegeben sei eine Tabelle mit Bauteilen, in denen die Komponenten für Werkstücke festgehalten werden. Dabei können diese Komponenten selber wiederum aus Einzelteilen zusammengesetzt sein.

```
CREATE TABLE bauteile (  
    stueck      VARCHAR(8),  
    komponente VARCHAR(8),  
    menge      INTEGER);
```

Um nun zu ermitteln, welche Basiskomponenten insgesamt nötig sind, um ein Beispiel-Bauteil herzustellen, kann man eine rekursive Anfrage benutzen. In der folgenden RA ist die im ersten Teil der WITH-Klausel definierte Query **rek** die Rekursionsbasis, im zweiten Teil der WITH-Klausel folgt dann die Definition der Rekursion. Die so rekursiv definierte Ergebnisrelation wird dann im SELECT DISTINCT-Teil benutzt.

```
WITH rek (stueck, komponente, menge) AS (  
    SELECT r.stueck, r.komponente, r.menge  
    FROM bauteile r  
    WHERE r.stueck = 'Beispiel-Bauteil'  
UNION ALL  
    SELECT kind.stueck, kind.komponente, kind.menge  
    FROM rek vater,  
         bauteile kind  
    WHERE vater.komponente = kind.stueck  
)  
SELECT DISTINCT stueck, komponente, menge  
FROM rek  
ORDER BY stueck, komponente, menge
```

Trigger

Als Trigger (deutsch Auslöser) bezeichnet man in SQL eine Anweisungsfolge (eine Abfolge von Aktionen), die ausgeführt wird, wenn eine verändernde Anweisung auf einer bestimmten Tabelle ausgeführt werden soll. Wir erinnern uns aus Woche 5, dass verändernde Anweisungen für Tabellen DELETE, INSERT und UPDATE sind.

Man kann Trigger zum Beispiel nutzen, um

- neu eingefügte oder zu verändernde Tupel auf ihre Gültigkeit bezüglich vorgegebener Bedingungen zu überprüfen,
- Werte zu generieren,


```
VALUES ('Hancock, Herb', 'Ein toller Film', 2003);
```

```
SELECT anzahl  
FROM fps  
WHERE name='Hancock, Herb';
```

```
ANZAHL
```

```
-----
```

```
1
```

Noch ein Beispiel:

```
CREATE TRIGGER check_year  
NO CASCADE BEFORE INSERT ON spielt_in  
REFERENCING NEW AS n  
FOR EACH ROW MODE DB2SQL  
WHEN n.jahr > (year(current date) +1)  
SIGNAL SQLSTATE '75000'  
('Kein Einfuegen von Filmen aus uebernaechstem Jahr ')
```

Dieser Trigger verbietet das Einfügen neuer Tupel in `spielt_in`, bei denen das Drehjahr mehr als ein Jahr in der Zukunft liegt. Der Versuch führt zu einem Fehler und das Statement wird nicht ausgeführt:

```
INSERT INTO spielt_in (name, produktion, jahr)  
VALUES ('Testactor', 'Testfilm', 2007);
```

```
SQL0438N Application raised error with diagnostic text:  
"Kein Einfuegen von Filmen aus uebernaechstem Jahr ".  
SQLSTATE=75000
```

```
SELECT jahr  
FROM spielt_in  
WHERE name='Testactor';
```

```
JAHR
```

```
-----
```

```
0 record(s) selected.
```

Vorüberlegungen

Macht Euch mit Hilfe der angegebenen Handbuchabschnitte, bzw. der Online-Dokumentation oder anderer Literatur mit der Syntax des `WITH...SELECT`-Statements, des `CREATE INDEX`-Statements und des `CREATE TRIGGER`-Statements vertraut.

Beantwortet die folgenden Fragen:

- Was bedeutet die Benutzung von `UNIQUE` bei der Erstellung eines Index?
- Welche drei Triggerarten gibt es und was bedeuten sie? Als Reaktion auf welche Aktionen können Trigger ausgelöst werden?
- Nenne zwei andere Nutzen für einen in der `WITH`-Klausel definierten Tabellenausdruck außer der Verwendung in rekursiven Anfragen.

Aufgaben

Wie in der letzten Woche dürft Ihr wahlweise mit der von Euch erstellten oder der Beispieldatenbank `dbpw0323.IMDB23` arbeiten. Über den CLP kann man sich mit folgenden Befehlen mit dieser DB verbinden und herausfinden, was dort für Daten gespeichert sind:

```
db2 => catalog tcpip node NODENAME
      => remote salz.is.informatik.uni-duisburg.de
      => server 50023
      => remote_instance dbpw0323
      => system salz
      => ostype linux;
db2 => catalog database imdb23 as ALIAS
      => at node NODENAME;
db2 => connect to ALIAS user LOGINNAME;
db2 => list tables for schema dbpw0323;
db2 => describe table TABLENAME;
```

- (a) Findet heraus, welche Indexe bereits auf den Tabellen der Beispieldatenbank existieren. Der Befehl dazu lautet:

```
describe indexes for table TABELLE show detail
```

- (b) Definiert einen neuen Index für Genres, damit Suchaktionen nach Produktionen eines bestimmten Genres schneller gehen.
- (c) Nehmt Euch das erste Beispiel für Trigger als Vorlage und schreibt einen entsprechenden Trigger für ein DELETE auf der Tabelle `schauspieler`, der die Filmzahl wieder reduziert.
- (d) Im Falle eines UPDATE auf der Tabelle `schauspieler` sieht die Lage ein wenig komplexer aus. Schreibt einen Trigger, der die verschiedenen Fälle berücksichtigt und `fps` korrekt ändert.
- (e) Die Serie “Buffy the Vampire Slayer” lief sieben Jahre lang und ging in die Popkultur ein. In anderen Filmen und Serien tauchten immer wieder mal Referenzen auf die Serie auf:

```
SELECT produktion2
FROM beziehung
WHERE produktion1 = 'Buffy the Vampire Slayer'
      AND art = 'referenced in'
```

Diese Filme wurden wiederum in anderen Filmen referenziert und so fort. Das wollen wir als indirekte Referenzen bezeichnet. Schreibt ein rekursives SELECT-Statement, das alle (direkten und indirekten) Referenzen auf die Produktion ‘Buffy the Vampire Slayer’ ermittelt.

- (f) **Zusatzaufgabe:** Das Spiel “Six Degrees of Kevin Bacon” geht so (Beschreibung von <http://www.cs.virginia.edu/oracle>):

The object of the game is to start with any actor or actress who has been in a movie and connect them to Kevin Bacon in the smallest number of links possible. Two people are linked if they've been in a movie together. We do not consider links through television shows, made-for-tv movies, writers, producers, directors, etc. For example, you might wonder how Alfred Hitchcock can be connected to Kevin Bacon. One answer is that:

Alfred Hitchcock was in Show Business at War (1943) with Orson Welles, and Orson Welles was in A Safe Place (1971) with Jack Nicholson, and Jack Nicholson was in A Few Good Men (1992) with Kevin Bacon!
Then we can count how many links were necessary and assign the actor or actress a **Bacon number**.

Findet alle Schauspieler und Schauspielerinnen mit Bacon-Nummer 2. Als Gedankenübung schreibt ein rekursives SQL-Statement, dass die Schauspieler und Schauspielerinnen für beliebige Bacon-Nummern findet.