Databases and Information Systems

5. Übungsblatt

(Testatwoche: 18. - 20. Mai 2010)

Einführung in Datenbanksysteme
Datenbanken für die Bioinformatik

Heinz Schweppe, Katharina Hahn

Aufgabe 1 (Kardinalitäten von Ergebnisrelationen) *6 Punkte*

Gegeben sind die Relationen *A*, *B* und *C* mit den Attributen *a1, a2*, … bzw. *b1, b2*, … bzw. *c1, c2* …. und den Kardinalitäten $|A|$, $|B|$ und $|C|$.
Die Schranken der Kardinalitäten werden im folgenden als (*min*, *max*) angegeben.

a) Zwischen welchen Schranken liegt die Kardinalität der folgenden Ergebnismengen. Geben Sie möglichst scharfe Schranken an.

(1) `SELECT * FROM A JOIN B ON (a1 = b1) JOIN C ON (b2 = c3)`
(0, $|A|\cdot|B|\cdot|C|$): Wenn eine der Relation leer ist, ist auch das Ergebnis leer – ansonsten besteht es aus den Kreuzprodukten.

(2) `SELECT a1, a2 FROM A UNION SELECT b2, b1 FROM B`
(max{$|A|$, $|B|$}, $|A|+|B|$): Im geringsten Fall sind alle Tupel (*a1*, *b2*) redundant mit allen Tupeln (*b2*, *b1*), wodurch die Vereinigung (nach Entfernen der Duplikate) nur noch die eindeutigen Tupel enthält – und das sind mindestens soviele wie in der kleineren der beiden Mengen (Duplikate **innerhalb** einer Menge werden ja nicht entfernt). Oder aber es sind alle Tupel paarweise unterschiedlich, dann besteht die Vereinigung genau aus der Summe der beiden Mengen.

(3) `SELECT a1 FROM A WHERE a1 NOT IN (SELECT b2 FROM B)`
(0, $|A|$): Im ersten Fall sind alle Attribute *a1* in den Attributen *b2* enthalten, im zweiten keines.

(4) `SELECT * FROM A, B, C`
(0, $|A|\cdot|B|\cdot|C|$): Wenn eine der Tabellen leer ist, ist auch das Kreuzprodukt leer.

b) Nehmen Sie an, dass *ai* Schlüssel von *A*, *bj* ein Fremdschlüssel auf *A* und *ck* Schlüssel von *C* ist. Zwischen welchen Schranken liegt die Kardinalität der folgenden Mengen. Geben Sie möglichst scharfe Schranken an.

(1) `SELECT * FROM A JOIN B ON (ai = b1)`
(0, $|B|$): *B* ist hier die limitierende Relation, da nur Elemente aus *A* ausgewählt werden, die sich in *B* wiederfinden. Im Extremfall existiert zwar nur ein einziges Element in *A* aber alle Elemente *b1* aus *B* verweisen darauf (*b1* muss nicht eindeutig sein), und das Ergebnis enthält ein Tupel für jedes aus *B*. Wenn umgekehrt nur ein Element in *B* existiert, dann steht im Ergebnis auch nur ein Tupel, egal, wie viele Tupel *A* enthält.

(2) `SELECT * FROM A JOIN B ON (ai = bj) JOIN C ON (b2 = ck)`
(0, $|B|$): Im ersten Verbund gilt dasselbe wie für (1). Im zweiten Verbund ist diese Beziehung genau umgekehrt, da *ck* ein Schlüssel von *C* und somit eindeutig ist. Wieder bestimmt also allein *B* die Anzahl der Ergebnistupel.

Aufgabe 2 (Advanced SQL) *14 Punkte*

Gegeben ist das Kinodatenbankenschema des letzten Übungsblatts.

KinoDB (
Orte(Kino, Adresse, Telefon),
Filme(Titel, Regie, Schauspieler),
Programm(Kino, Titel, Zeit)
)

Formulieren Sie folgende Anfragen in SQL.

a) Welche Filme (Titel) laufen nicht mehr aktuell im Kino?

```
SELECT DISTINCT Titel
FROM Filme
WHERE Titel NOT IN (SELECT Titel FROM Programm);
```

b) Wieviele Filme laufen aktuell, bei denen *Sofia Coppola* Regie geführt hat?

```
SELECT COUNT(DISTINCT Titel)
FROM Filme NATURAL JOIN Programm
WHERE Regie = 'Sofia Coppola';
```

c) Welche Filme (Titel) laufen nur zu einer Uhrzeit.

```
SELECT Titel
FROM Programm
GROUP BY Titel HAVING COUNT(DISTINCT Zeit) = 1;
```

d) In wie vielen Kinos läuft *A Single Man* nach 18h?

```
SELECT COUNT(DISTINCT Kino)
FROM Programm
WHERE Titel = 'A Single Man' AND Zeit > '18:00';
```

e) In welchen laufenden Filmen spielen **ausschließlich** Schauspieler mit, die schon mal mit *Steven Spielberg* gearbeitet haben?

```
SELECT Titel
FROM Programm
EXCEPT
-- Filme mit Schauspielern, die noch nicht mit Steven Spielberg
-- gearbeitet haben.
SELECT Titel
FROM Filme f
WHERE NOT EXISTS ((
    SELECT Schauspieler
    FROM Filme
    WHERE f.Schauspieler = Schauspieler)
  INTERSECT (
    -- Alle Mitarbeiter von Steven Spielberg.
    SELECT Schauspieler
    FROM Filme
    WHERE Regie = 'Steven Spielberg'));
```

Aufgabe 3 (Advanced SQL, Mondial Datenbank) *15 Punkte*

Formulieren Sie folgende Anfragen in SQL. Einige der Anfragen lassen sich unter zu Hilfenahme der Mengenquantoren `ALL`, `ANY` und `EXISTS` formulieren.
Geben Sie jeweils **gültige** SQL-Statements an.

a) Welche Länder liegen nicht an Meeren, wohl aber alle ihre Nachbarländer?

```
--Lösungsmöglichkeit:
--ohneKueste enthaelt alle Laender, die selbst nicht am Meer liegen
--nachbarlaender enthaelt alle Laender ohne Kueste mit ihren Nachbarn
   WITH ohneKueste AS (
            SELECT c.code AS okCode, c.name AS okName
            FROM mondial.country c
            WHERE c.code NOT IN (
                  SELECT country
                  FROM mondial.geo_sea)
        ), nachbarlaender AS (
```

```
            SELECT c.code AS nbId, b.country2 AS nbId2
            FROM mondial.country c, mondial.borders b
            WHERE c.code = b.country1
                  AND c.code NOT IN (
                        SELECT country
                        FROM mondial.geo_sea)
      )

--selektiere alle Laender aus ohneKueste, die selbst nicht Nachbar zu
einem Land ohne Kueste sind.  Ist ein Land L1 in ohneKueste Nachbar
zu einem Land L2 in nachbarlaender, das selbst keine Kueste besitzt,
entspricht L1 nicht mehr den Suchkriterien (es hat ein Nachbarland
ohne Küste).

      SELECT DISTINCT okName
      FROM ohneKueste
      WHERE NOT EXISTS (SELECT *
                        FROM nachbarlaender
                       WHERE nachbarlaender.nbId2 = ohneKueste.okCode)
      ORDER BY okName;
```

b) Welche Länder (Name) mit mehr als 10 Millionen Einwohnern haben eine höhere Bevölkerungsdichte (Einwohner/Fläche) als alle ihre Nachbarländer?

```
      SELECT c.name, c.population, c.population/c.area AS density
      FROM country c
      WHERE c.population > 10000000
            AND c.population/c.area > ALL (
                  SELECT c2.population/c2.area
                  FROM country c2,  borders b
                  WHERE (b.country1 = c.code
                        AND b.country2 = c2.code)
                      OR (b.country2 = c.code
                        AND b.country1 = c.code))
```

c) Geben Sie alle Flüsse an, die in ein Meer münden. Das Meer soll dabei nur an eines der Landesteile grenzen, durch die der Fluss fließt. Das Meer kann aber an andere Landesteile grenzen, durch die der Fluss nicht fließt.

```
      SELECT r.name
      FROM river r, geo_river gr, geo_sea gs
      WHERE r.name = gr.river
        AND r.sea <> ''
        AND gr.province = gs.province
        AND gr.country = gs.country
        AND r.sea = gs.sea
      GROUP BY r.name
      HAVING COUNT (gs.sea)=1
      ORDER BY r.name;
```

d) Welche Länder, außer dem bevölkerungsreichsten Land haben mehr als 100 Millionen Einwohner? Lassen Sie Ländername, Bevölkerungszahl und Gesamtfläche ausgeben.

```
      SELECT c.name, c.population, c.area
      FROM country c
      WHERE c.population > 100000000
            AND c.population < ANY (SELECT c2.population
                                    FROM country c2
                                    WHERE c2.population > 100000000 )
      ORDER BY c.name
```

e) Welche Länder haben eine mindestens fünfmal größere Gesamtfläche als jeder der deutschen Nachbarstaaten. Lassen Sie Ländername und Gesamtfläche ausgeben.

```
SELECT c.Name, c.area
FROM country c
WHERE c.area/5 > ALL (
      SELECT c2.area
      FROM country c2, borders b
      WHERE (b.country1 = 'D'
            AND b.country2 = c2.code)
            OR
            (b.country2 = 'D'
            AND b.country1 = c2.code))
ORDER BY c.name
```

Aufgabe 4 (SQL Gruppenprädikate) *4 Punkte*

Kann man – wenn ja, wie; wenn nein, warum nicht – das *HAVING*-Prädikat für Gruppen durch Zeilenprädikate ausdrücken?

Ja! Bezieht sich das HAVING-Prädikat nur auf einzelne Spalten, kann die Bedingung in die WHERE-Klausel aufgenommen werden. Bezieht sich die Bedingung des HAVING-Prädikats auf die Aggregatfunktion der Gruppen, so kann man die Tabelle vorher in einzelne Tabellen zerlegen, in denen jeweils nur Tupel einer Gruppe auftauchen, durch Anwendung der Aggregatfunktion bekommt man pro Tabelle dann ein Tupel, auf das man mit Hilfe der WHERE-Klausel dann die Bedingung anwenden kann. Durch eine Vereinigung der einzelnen Tupel bekommt man dann das gewünschte Ergebnis.

Aufgabe 5 (SQL DATE und TIMESTAMP)

12 Punkte

Machen Sie sich mit den Datum- und Zeitstempelformaten von Postgres vertraut, z.B. unter http://developer.postgresql.org/pgdocs/postgres/functions-datetime.html für die Version 9. Betrachten Sie nun folgende Tabelle:

```
CREATE TABLE users (
  username VARCHAR(15) PRIMARY KEY,
  firstname VARCHAR(15),
  lastname VARCHAR(19),
  birthday DATE,
  registered TIMESTAMP
);
```

Beantworten Sie folgende Anfragen in SQL. Geben Sie jeweils **gültige** SQL Statements an.

a) Geben Sie den ältesten Benutzer mit Namen, Vornamen und Geburtsdatum aus. Falls es mehrere gibt, sollen alle sortiert nach Namen ausgegeben werden.

```
SELECT firstname, lastname, birthday
FROM users
WHERE birthday <= ALL (
                      SELECT birthday
                      FROM users)
ORDER BY lastname
```

b) Geben Sie alle Benutzer, die älter als 18 Jahre sind, unter Verwendung
(1) von CURRENT_DATE,

```
SELECT *
FROM users
WHERE (CURRENT_DATE - birthday >= 18*365)
```

(2) der AGE-Funktion aus.

```
SELECT *
FROM users
WHERE EXTRACT(YEAR from AGE(birthday)) >= 18
```

c) Geben Sie alle Benutzer aus, die beigetreten (Attribut *registered*) sind, bevor andere Benutzer geboren wurden.

```
SELECT *
FROM users
WHERE DATE(registered) < ANY (SELECT birthday FROM users)
```

d) Geben Sie alle Benutzer aus, die nach 16h beigetreten sind.

```
SELECT *
FROM users
WHERE EXTRACT(HOUR from registered) >= 16
```

e) Geben Sie die Paare von Benutzern aus, die am gleichen Tag Geburtstag haben (beachten Sie, dass diese nicht im gleichen Jahr geboren sein müssen). Geben Sie jeweils die Nachnamen, Vornamen und Geburtsdaten der Benutzer aus. Achten Sie darauf, dass ein solches Paar von Benutzers (u1, u2) nicht zusätzlich als (u2, u1) ausgegeben wird.

```
SELECT u1.username, u2.username, u1.birthday, u2.birthday
FROM users u1, users u2
WHERE u1.username < u2.username
      AND EXTRACT(DAY from u1.birthday)
            = EXTRACT(DAY from u2.birthday)
      AND EXTRACT(MONTH from u1.birthday)
            = EXTRACT(MONTH from u2.birthday)
```