**Datenbanken**
**Wintersemester 06/07**
Prof. Dr. W. May

# 3. Übungsblatt: SQL

Besprechung am 29.11./6.12.

**Aufgabe 1 (SQL ist relational vollständig)** Zeigen Sie, dass SQL *relational vollständig* ist, d.h. zu jedem Ausdruck der relationalen Algebra gibt es einen äquivalenten Ausdruck in SQL.

---

Gegeben seien die Relationen $R(A_1, \ldots, A_n)$, $S(A_1, \ldots, A_n)$ und $T(A_{i_1}, \ldots, A_{i_k}, B_{k+1}, \ldots, B_m)$, $\{i_1, \ldots, i_k\} \subseteq \{1 \ldots n\}$ und paarweise ungleich.

$R \cup S$ :
```
SELECT A_1, ..., A_n
  FROM R
UNION SELECT A_1, ..., A_n
  FROM S;
```

$R \setminus S$ :
```
SELECT A_1, ..., A_n
  FROM R
EXCEPT SELECT A_1, ..., A_n
  FROM S;
```

$\sigma[F](R)$ :
```
SELECT A_1, ..., A_n
  FROM R
  WHERE F̄;
```
$\bar{F}$ ist der $F$ entsprechende Ausdruck in SQL-Syntax

$\pi[A_1, \ldots, A_i](R)$ :
```
SELECT A_1, ..., A_i
  FROM R;
```

$\rho[A_1 \to C_1, \ldots, A_n \to C_n](R)$ :
```
SELECT A_1 AS C_1, ..., A_n AS C_n
  FROM R
```

$S \bowtie T$ :
```
SELECT A_1, ..., A_n, B_{k+1}, ... B_m
  FROM S,T;
  WHERE S.A_{i_1} = T.A_1, ..., S.A_{i_k} = T.A_k;
```

---

**Aufgabe 2 (Gruppierung)**

- Die Frage nach der größten Landesfläche in der Mondial-Datenbank lautet

```
SELECT MAX(area)
  FROM Country;
```

Zusätzlich soll dazu der Landes-Code ausgegeben werden. Warum ist die folgende SQL-Anfrage fehlerhaft? Geben Sie eine entsprechend korrigierte SQL-Anfrage an.

```
SELECT MAX(area), code
  FROM Country;
```

- In der Vorlesung wurde für jedes Land die Bevölkerungszahl der größten Stadt ermittelt. Geben Sie eine Anfrage an, die zusätzlich auch den Namen dieser Stadt ausgibt.

---

- Man koennnte sich auf den ersten Blick als Ergebnis der ersten Anfrage `SELECT MAX(area), code FROM Country;` das Tupel (17075200, R) wünschen (Russland ist das groesste Land und hat 17075200 km$^2$ Fläche). Aber, was sollte dann `SELECT SUM(area), code FROM Country` ergeben? Die Summe der Flaeche ist 133287962.24, aber was soll für "code" ausgegeben werden?

  Die Aggregationsoperatoren MAX, MIN, SUM, AVG, COUNT nehmen (auf den ersten Blick) als Eingabe eine Menge von Tupeln und wenden den Aggregationsoperator auf die entsprechende Spalte an, und ergeben *einen* atomaren Ergebniswert.

  In der obigen falschen Anfrage wird also durch die Anwendung der Aggregatfunktion ein einzeiliges Ergebnis für die gesamte Tabelle erzeugt. Das Attribut `code` liefert jedoch nicht ein einzelnes Ergebnis, sondern für jedes Land eines (probieren Sie z.B. mal `SELECT MAX(area), MAX(code) FROM Country;` aus).

  Eine korrekte SQL-Anfrage für die Problemstellung ist etwa:

```
SELECT area, code
FROM country
WHERE area =
  (SELECT MAX(area)
   FROM country);
```

- Im obigen Beispiel wird die Aggregatfunktion auf die gesamte aktuelle Relation angewendet. Dies kann durch `GROUP BY attrlist` geändert werden: es werden Mengen von Tupeln gebildet, die in attrlist übereinstimmen. Damit werden die Aggregationsoperatoren MAX, MIN, SUM, AVG, COUNT im allgemeinen Fall auf *eine Menge von Mengen von Tupeln* angewendet und werten innerhalb jeder Gruppe den Aggregationsoperator auf der entsprechenden Spalte aus und liefern für jede Gruppe einen atomaren Ergebniswert:

```
SELECT Name, Country, population
FROM City
WHERE (country, population) IN
  (SELECT country, MAX(population)
   FROM City
   GROUP BY Country);
```

---

**Aufgabe 3 (Mondial (SQL))** Gegeben sei folgendes Datenbankschema (Auszug aus Mondial)

```
Country(Name, Code, Capital, Province, Area, Population)
Organization(Name, Abbreviation, Established)
Is_member(Organization, Country, Type)
```

Formulieren Sie die folgenden Anfragen in SQL:
(in den Teilaufgaben a) - e) brauchen verschiedene Arten von Mitgliedschaften nicht berücksichtigt werden!)

a) Geben Sie von jeder Organisation die Summe der Einwohner aller Mitgliedsländer absteigend geordnet an.

b) Welche Länder sind Mitglied in mehr als 60 Organisationen?

c) Welche Länder mit einer Fläche von mehr als 500000 km$^2$ sind Mitglied in mehr als 60 Organisationen?

d) Welche Länder sind in mindestens einer Organisation Mitglied, in der auch Deutschland ('D') Mitglied ist?

e) Welche Länder sind in mindestens den Organisationen Mitglied, in denen auch Andorra ('AND') Mitglied ist?

f) Zeigen Sie, dass es in der Datenbank keine Organisation gibt, in der alle Länder Mitglied sind!

Diese Anfragen können mit der Web-Schnittstelle zur Mondial-DB getestet werden (siehe Vorlesungsseite).

---

a) Geben Sie von jeder Organisation die Summe der Einwohner aller Mitgliedsländer absteigend geordnet an.

```
SELECT is_Member.Organization, SUM(Population)
FROM   is_Member, Country
WHERE  is_Member.Country = Country.Code
GROUP BY is_Member.Organization
ORDER BY 2 DESC;
```

b) Welche Länder sind Mitglied in mehr als 60 Organisationen?

```
SELECT country
FROM is_member
GROUP BY country
HAVING count(*) > 60
```

Hinweis: es wird in `is_member` für jedes Land eine Gruppe gebildet. Behalten werden diejenigen Gruppen, die mindestens 60 Einträge (Mitliedschaften des Landes in einer Organisation) enthalten). Für jede der Gruppen wird eine Zeile im Ergebnis generiert.

c) Welche Länder mit einer Fläche von mehr als 500000 $km^2$ sind Mitglied in mehr als 60 Organisationen? See another exercise (Section optimization, indexes)

d) Welche Länder sind in mindestens einer Organisation Mitglied, in der auch Deutschland ('D') Mitglied ist?

Alternative 1: alles in einem breiten Join:

```
SELECT DISTINCT C.Name
FROM Country C, Is_Member M1, Is_Member M2
WHERE C.code = M1.country
  AND M1.organization = M2.organization
  AND M2.country = 'D' AND C.code <> 'D'
```

Alternative 2: "Welches (Land hat eine Mitgliedschaft in einer Organisation), in der auch (Deutschland Mitglied ist)?"

```
SELECT Name
FROM Country, Is_Member M1
WHERE C1.code = M1.country
  AND M1.organization IN (
        SELECT M2.organization
        FROM Is_Member M2
        WHERE M2.country = 'D')
  AND C.code <> 'D'
```

Alternative 3: "Für welches Land gibt es eine Mitgliedschaft in einer Organisation, die auch in der Menge der Organisationen, in denen D Mitglied ist, enthalten ist"?

```
SELECT Country.Name
FROM Country
WHERE EXISTS
 (SELECT * from is_member M1
```

```
  WHERE M1.country = Country.code
    AND M1.organization IN
        (SELECT M2.organization
         FROM Is_Member M2
         WHERE M2.country = 'D'))
```

Gegenfrage: welche Länder (die überhaupt irgendwo Mitglied sind) sind nicht in einer Organisation Mitglied, in der Deutschland Mitglied ist?

```
SELECT *
FROM
 (SELECT DISTINCT country AS CC
  FROM is_member)
WHERE NOT EXISTS
 (SELECT *
  FROM is_member M2
  WHERE M2.country='D'
    AND M2.organization IN
      (SELECT organization
       FROM is_member
       WHERE country = CC));
```

e) Welche Länder sind in mindestens den Organisationen Mitglied, in denen auch Andorra ('AND') Mitglied ist?

```
SELECT DISTINCT Country
FROM is_member M
        WHERE NOT EXISTS
        ((SELECT organization
          FROM is_member
          WHERE country = 'AND')
         MINUS
         (SELECT DISTINCT organization
          FROM is_member
          WHERE country = M.country
        ))
```

*Welche Länder sind in genau den Organisationen Mitglied, in denen auch Andorra ('AND') Mitglied ist?*

```
SELECT DISTINCT Country
FROM is_member M
        WHERE NOT EXISTS
                ((SELECT DISTINCT organization
                  FROM is_member
                  WHERE country = 'AND')
                 MINUS
                 (SELECT DISTINCT organization
                  FROM is_member
                  WHERE country = M.country))
          AND NOT EXISTS
                ((SELECT DISTINCT organization
                  FROM is_member
                  WHERE country = M.country)
                 MINUS
                 (SELECT DISTINCT organization
                  FROM is_member
```

```
                    WHERE country = 'AND'))
```

f) Zeigen Sie, dass es in der Datenbank keine Organisation gibt, in der alle Länder Mitglied sind!

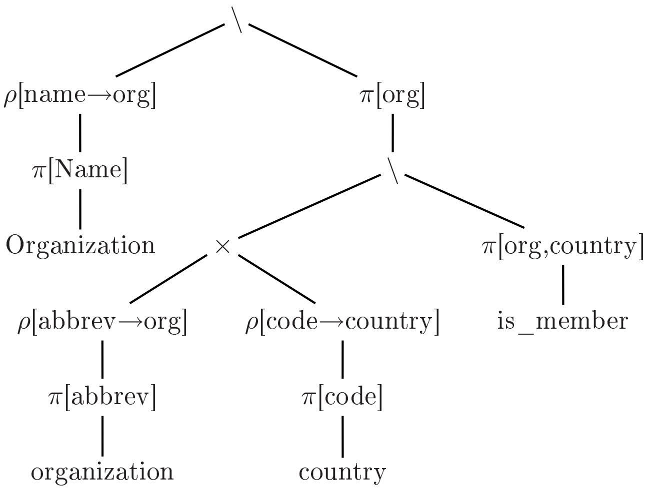
*Alle Organisationen für die es kein Land gibt, das nicht in dieser Organisation ist*

```
SELECT O.Name
FROM Organization O
WHERE NOT EXISTS
   (SELECT name
    FROM country C
    WHERE NOT EXISTS
       (SELECT organization
        FROM is_member I
        WHERE C.code = I.country
          AND I.organization = O.abbreviation))
```

... relationale Division:

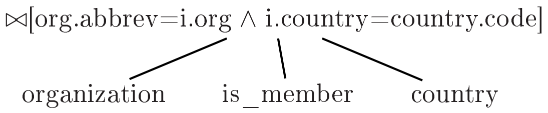
$$\pi[\text{org,country}](\text{is\_member}) \div \rho[\text{code}\rightarrow\text{country}](\pi[\text{code}](\text{country}))$$

entsprechend der Zerlegung der Division in algebraische Grundoperationen (unter Benutzung von $\pi$[country](is_member)$=\pi$[code]country und $\pi$[org](is_member)$=\pi$[abbrev](org):



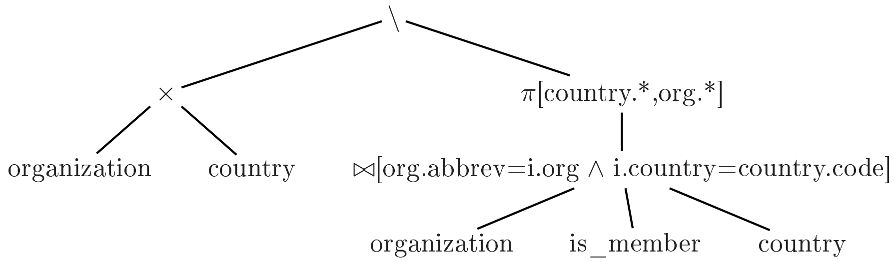

Man kann auch das NOT EXISTS entsprechend der Vorlesung in die Algebra übersetzen:

- Innere Subquery: importiert werden *organization* und *country*; also Join mit nachfolgendem von diesen beidem mit is_member:

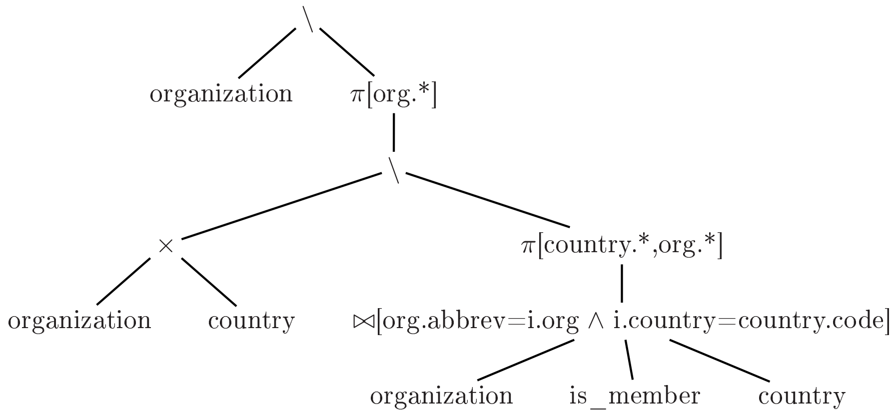
  Das Ergebnis dieses Ausdruckes wird auf die Schlüsselattribute der korrelierenden Relationen *organization* und *country* projiziert (ergibt diejenigen Paare von Organisationen und Länder, für die eine Mitgliedschaft existiert).

- Die Grundmenge der mittleren Subquery ist deren FROM zusammen mit der importierten Relation *organization*. Von diesem kartesischen Produkt zieht man das obige Ergebnis ab (NOT EXISTS):




Das Ergebnis dieses Ausdruckes wird sofort auf die Schlüsselattribute der korrelierenden Relation *organization* projiziert (gibt diejenigen Organisationen $O$, für die ein Land mit einer Nichtmitgliedschaft in $O$ existiert).

- Das wird jetzt noch von der Menge aller Organisationen abgezogen:




Man kann die Anfrage noch dahingehend verfeinern, dass man nur wissen will, ob alle Länder eines Kontinents Mitglied in einer Organisation sind:

```
SELECT abbreviation
FROM organization
WHERE NOT EXISTS
  (SELECT *
   FROM
    (SELECT country CC
     FROM encompasses
     WHERE continent ='Asia'
    )
   WHERE NOT EXISTS
    (SELECT *
     FROM is_member
     WHERE is_member.organization=organization.abbreviation
       AND is_member.country=CC));
```

---

**Aufgabe 4 (Optimierung, Aufwandsabschätzung (SQL))** Gegeben sei folgendes Datenbankschema (Auszug aus Mondial)

```
Country(Name, Code, Capital, Province, Area, Population)
Organization(Name, Abbreviation, Established)
Is_member(Organization, Country, Type)
```

Formulieren Sie die folgenden Anfragen in SQL:

a) Welche Länder sind Mitglied in mehr als 60 Organisationen?
b) Welche Länder mit einer Fläche von mehr als 500000 km$^2$ sind Mitglied in mehr als 60 Organisationen?

Untersuchen Sie bei der 2. Teilaufgabe den Aufwand (Anzahl Hintergrundspeicherzugriffe, Vergleiche zur Ausführung des Join, Vergleiche zum Test der Fläche). An welchen Stellen wäre ein (Baum- oder anderer) Index nützlich? Untersuchen Sie auch für diesen Fall den Aufwand.

Können Sie Ihre Anfrage optimieren, um damit den Aufwand weiter zu reduzieren?

---

a) Welche Länder sind Mitglied in mehr als 60 Organisationen?

```
SELECT country
FROM is_member
GROUP BY country
HAVING count(*) > 60
```

b) Welche Länder mit einer Fläche von mehr als 500000 km$^2$ sind Mitglied in mehr als 60 Organisationen?

   Naheliegendste Lösung: WHERE-Klausel, um vor der Gruppierung zu selektieren.

```
SELECT country
FROM is_member
WHERE (select area
       from country
       where code=is_member.country) > 500000
GROUP BY country
HAVING count(*) > 60
```

   Aufwandsbetrachtung: Für jede der 7000 Mitgliedschaften wird überprüft, ob das betreffende Land grösser als 500000 km$^2$ ist.

   Fall 1: kein Index auf country. Also wird linear die gesamte Tabelle country durchsucht. Aufwand: $7000 \cdot 200 = 1400000$ Vergleiche von `code=is_member.country` und 7000 Flächentests.

   Fall 2: Baum-Index auf country.code [Tafelbild]. Dann wird für jede der Mitgliedschaften über den Baum zielsicher auf das entsprechende country-Tupel zugegriffen und die Fläche getestet (7000 Tests) (Dasselbe gilt auch für einen Hash-Index).

   Optimierung: der Flächentest wird z.B. für Deutschland 68 mal durchgeführt - für jede Mitgliedschaft einmal. Es wäre effizienter, doch erst nach "country" zu gruppieren, und den Test dann nur für jede Gruppe einmal durchzuführen:

```
SELECT country
FROM is_member
GROUP BY country
HAVING count(*) > 60
 and (select area
      from country
      where code=is_member.country) > 500000
```

   Damit: 200 Gruppen, bei nested-loop-Join also noch $200 \cdot 200 = 40000$ Vergleiche von `code=is_member.country` und 200 Flächentests, bei Baumindex nur noch 200 Flächentests.

   Weitere interne algorithmische Optimierung:
   Bei der Auswertung von GROUP BY wird der Inhalt der is_member-Tabelle sowieso sortiert. Wenn man nun noch einen Index über country.code hat, kann der area-Test auch als Merge ablaufen, wobei parallel die sortierten Gruppen und die Blätter des Indexbaumes (was allerdings bei einem Hash-Index nicht gehen würde!) durchlaufen werden

Hinweise:

- Üblicherweise wird automatisch ein Index über alle Schlüssel sowie Fremdschlüssel erzeugt, da diese häufig für Joins und Selektionen benötigt werden.
- Bei der Auswertung kann ein DBS auch temporär einen Index erstellen.

**Ausserdem:**

- Formulieren Sie die Anfragen vom vorigen Blatt auch in SQL, und formulieren Sie die Anfragen von Aufgabe 3 soweit möglich auch in der relationalen Algebra.
- Weitere Aufgaben finden Sie auf dem ersten Übungsblatt des SQL-Praktikums (`http://dbis.informatik.uni-goettingen.de/Teaching/DBP/`)
  Dort finden Sie auch detaillierte Folien sowie ein Skript zu SQL ...