Datenbanken
Wintersemester 09/10
Prof. Dr. W. May

3. Übungsblatt: SQL

Besprechung voraussichtlich am 9./16.12.2009

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 größte Land und hat 17075200 km^2 Fläche). Aber, was sollte dann `SELECT SUM(area), code FROM Country` ergeben? Die Summe der Fläche 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 (Relationale Division) Gegeben seien die Relationen $R(A, B)$ und $S(B)$. Prüfen Sie ob der folgende Ausdruck die relationale Division $R \div S$ korrekt abbildet.

```
SELECT A
FROM R
WHERE B IN ( SELECT B FROM S )
GROUP BY A
HAVING COUNT(*) = ( SELECT COUNT (*) FROM S );
```

Geben Sie ggf. die dazu notwendigen Bedingungen oder Korrekturen am Ausdruck an und diskutieren Sie seine Effizienz gegenüber den in der Vorlesung präsentierten Ausdrücken.

Der Ausdruck ist prinzipiell nicht schlecht. Er ist aber nicht korrekt, wenn Duplikate beteiligt sind (echte Duplikate in Tabellen sind selten, aber oft entsteht S oder T durch eine Subquery).

Die folgenden minimalen Gegenbeispiele zeigen das Problem:

- Wenn $R(A, B)$ Duplikate enthält, muss für dieses a nur ein B-Wert auch in $\pi[B](S)$ vorhanden sein, damit a im Ergebnis erscheint:

R	
A	**B**
a	1
a	1

S
B
1
2

- Wenn $S(B)$ Duplikate enthält, fehlen Ergebnisse (dieser Fall kommt häufig vor wenn B durch eine Subquery berechnet wird):

R	
A	**B**
a	1

S
B
1
1

  Die Definition der Semantik der Division ist rein mengenorientiert, so dass a hier im Ergebnis erscheinen müsste.

- Korrektur:

```
SELECT A
FROM R
WHERE B IN ( SELECT B FROM S )
GROUP BY A
HAVING COUNT(DISTINCT *) = ( SELECT COUNT DISTINCT (*) FROM S );
```

- Effizienz: Intern werden üblicherweise (ad-hoc)-Indexe verwendet (in diesem Fall über $\pi[A](R)$ zur Unterstützung des `GROUP BY` und des `COUNT` und $\pi[B](S)$). Damit kann diese Variante schneller sein als die normalerweise angegebene, wenn S nicht durch eine korrelierte Unterabfrage berechnet werden muss.

Aufgabe 4 (Zusätzliche Algebraoperatoren) Man kann weitere Algebra-operatoren definieren, die zusätzliche Funktionalität bereitstellen.

a) Definieren Sie zwei Operatoren group-by und top-k wie folgt:
   - top-k: Für eine gegebene Relation country(..., area, ...) enthält topk(5, area, desc)(country) die flächenmäßig größten 5 Länder.
     (die Bedeutung des top-k-Operators besteht in der Praxis darin, von einer komplexen Anfrage die k besten Ergebnisse möglichst schnell zurückzugeben, wobei nach Möglichkeit garnicht alle Ergebnisse wirklich berechnet werden müssen).
   - group-by soll die aus SQL bekannte Funktionalität von `GROUP BY` haben,

   Gehen Sie dabei wie bei der Definition der Basisoperatoren vor:
   - Welche Parameter müssen dem Operator mitgegeben werden?
   - Welche Signatur hat er?
   - Welche Signatur besitzt die Ergebnisrelation (in Abhängigkeit der Eingaberelation(en))?
   - Wie ist die erhaltene Tupelmenge definiert?

b) Wenn Sie Group-by definiert haben, wie behandeln Sie die HAVING-Klausel?

c) Geben Sie eine SQL-Anfrage sowie einen Algebra-Baum für die Anfrage "Welche Länder sind Mitglied in mehr als 60 Organisationen?" an.

---

a) **topk:** Der top-k-Operator ist eng mit der Selektion verwandt. Allerdings ist das Selektionskriterium keine Bedingung an ein einzelnes Tupel, sondern eine Bedingung, die alle Tupel in der Eingaberelation R berücksichtigt: Parameter (Anwendung auf eine Relation $R(\bar{X})$):
   - eine Vergleichsfunktion, mit dem Tupel über $\bar{X}$ verglichen/geordnet werden können. Falls f nur die Auswahl eines Attributes ist, z.B. {Area}, ist die Semantik offensichtlich. f kann auch ein anwendungsabhängig definierter Funktionsterm sein, z.B. Population/Area, ähnlich zu den `MAP/ORDER FUNCTIONS` zu Objekttypen in PL/SQL.
     Beispiel: Sporttabelle nach Punkten/Tordifferenz/geschossene Tore ordnen.
   - k: wieviele Ergebnisse sollen ausgegeben werden?

   Eingabe: eine Relation $R(\bar{X})$.
   Ausgabe: auch eine Relation mit Format $[\bar{X}]$.

   Definition:

$$\begin{aligned} \mathsf{topk}[k, f(\bar{X}), <](r) &= \{t \in r : \ |\{t' \in R : f(t') > f(t)\}| < k\} \\ &= \{t \in r : \ |\{\sigma[f > f(t)](r)\}| < k\} \end{aligned}$$

   Hinweis: Wenn mehrere Tupel gleichgroße Werte von $\bar{A}$ haben, können auch $\ell > k$ Tupel ausgegeben werden; für jedes dieser Tupel gibt es aber maximal $k-1$ größere.
   Beispiel: die 5 größten Länder:

$$\begin{aligned} \mathsf{topk}[5, \mathsf{area}, <](r) &= \{t \in r : \ |\{t' \in R : \pi[\mathsf{area}](t') > \pi[\mathsf{area}](t)\}| < 5\} \\ &= \{t \in r : \ |\{\sigma[\mathsf{area} > \pi[\mathsf{area}](t)](r)\}| < 5\} \end{aligned}$$

   group-by: Parameter (Anwendung auf eine Relation $R(\bar{X})$):
   - eine Attributmenge $\bar{A} \subseteq \bar{X}$ nach der gruppiert wird,
   - eine Menge von Definitionen neuer Attributen. Jede solche Definition enthält den Namen z_i des neuen Attributs sowie dessen Definition als Funktionsausdruck $f_i(\bar{X})$ über $\bar{X}$. f_i darf dabei Aggregatoperationen (count, sum, max, min, avg) sowie Attribute aus $\bar{A}$ enthalten (z.B. $\mathsf{max}(x_3)$ oder $\mathsf{sum}(x_4)/\mathsf{count}(*)$).

   Eingabe: eine Relation $R(\bar{X})$.
   Ausgabe: eine Relation mit Format $[\bar{A} \cup \bar{Z}]$ wobei $\bar{Z} = \{z_1, \ldots, z_n\}$ die Menge der neuen Attributnamen ist.
   Definition:

$$\begin{aligned} &\mathsf{group\text{-}by}[\bar{A}, (z_1 := f_1(\bar{X}), \ldots, z_k := f_k(\bar{X}))](r) \\ &\qquad = \{t \in \mathsf{Tup}(\overline{AZ}) : t[\bar{A}] \in \pi[\bar{A}](R) \text{ und } t[z_i] = f_i(\sigma[\bar{A} = t[\bar{A}]](r))\} \end{aligned}$$

   wobei f_i über einer Menge von Tupeln über $\bar{X}$ (der jeweiligen Gruppe) rekursiv ausgewertet wird.

b) Das Ergebnis der Anwendung von Group-by ist eine ganz "normale" Relation. HAVING entspricht einer Selektion auf dieser.

c) SQL:

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

π[country](σ[number>60](group-by[{country}, {number := count(*)}](is_member)))

Aufgabe 5 (Duplikate) a) Überlegen Sie sich, welche Gründe es gibt, dass (i) die relationale Algebra keine Duplikate erlaubt, aber (ii) in SQL Duplikate erlaubt sind. (Es gibt jeweils mindestens 2 "gute" Gründe.)

b) wie kann man in SQL Duplikate aus einer Tabelle entfernen?

---

a) Algebra

- E. Codd entwarf die Relationale Algebra 1970 auf Basis der mathematischen Mengentheorie. Mengen enthalten keine Duplikate. Soweit also einfach eine naheliegende Entscheidung.
- Es gibt in der relationalen Algebra keine einfache Möglichkeit, Duplikate festzustellen!
  - kein count-Operator,
  - keine Tupel-IDs (SQL hat implizite ROWIDs, man kann also schauen, ob eine Tabelle das gleiche Tupel mit zwei ROWIDs enthält).
- die relationale Algebra war zuerst da. Es gab also keinen Grund, bereits einen Vergleich mit SQL anzustellen und die Entscheidung anzugreifen.

a) SQL

- In der Realität gibt es Duplikate (insbesondere z.B. nach Projektion und vor Anwendung von Aggregationen). Datenbanken sollen die Realität abbilden können.
- Duplikatentfernung ist relativ aufwändig ($n \cdot \log n$). Dies auf Verdacht jedes Mal durchzuführen wäre unnötiger Aufwand.

b) Es gibt in SQL keinen Operator, um Duplikate aus einer *Tabelle* zu entfernen, sondern nur eine Duplikatentfernung während der relationalen Auswertung (Mengenoperationen, DISTINCT).

Will man Duplikate entfernen, kann man dies über ROWID machen (Laufzeit bis zu $O(n^2)$)

```
DELETE FROM R R1, R R2
WHERE R1.A = R2.A AND ... AND R1.ROWID < R2.ROWID;

DELETE FROM R
WHERE EXISTS
(SELECT * FROM R R2
 WHERE R.A = R2.A AND ... AND R.ROWID < R2.ROWID)
```

oder über eine Hilfstabelle (Laufzeit $O((n \cdot \log n) + n)$):

```
CREATE TABLE R' (mit selbem Schema wie R);
INSERT INTO R' (SELECT DISTINCT * FROM R);
DELETE FROM R;
INSERT INTO R (SELECT * FROM R');
DROP TABLE R';
```

Aufgabe 6 (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 ismember.Organization, SUM(Population)
FROM   ismember, Country
WHERE  ismember.Country = Country.Code
GROUP BY ismember.Organization
ORDER BY 2 DESC;
```

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

```
SELECT country
FROM ismember
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, ismember M1, ismember 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, ismember M1
WHERE C1.code = M1.country
  AND M1.organization IN (
        SELECT M2.organization
        FROM ismember 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 ismember M1
  WHERE M1.country = Country.code
    AND M1.organization IN
        (SELECT M2.organization
         FROM ismember 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 ismember)
WHERE NOT EXISTS
 (SELECT *
  FROM ismember M2
  WHERE M2.country='D'
    AND M2.organization IN
      (SELECT organization
       FROM ismember
       WHERE country = CC));
```

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

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

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

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

```
 FROM ismember
 WHERE country = M.country)
MINUS
(SELECT DISTINCT organization
 FROM ismember
 WHERE country = 'AND'))
```

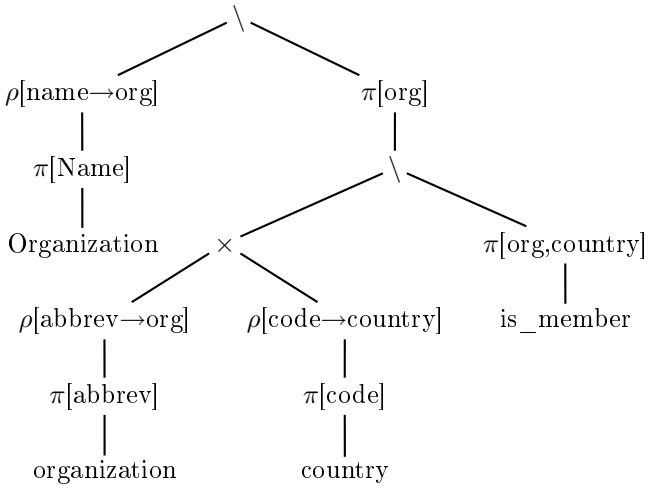
f) Zeigen Sie, dass es in der Datenbank keine Organisation gibt, in der alle Länder Mitglied sind!
Alle Organisationen für die es kein Land gibt, das nicht in dieser Organisation ist

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

... relationale Division:

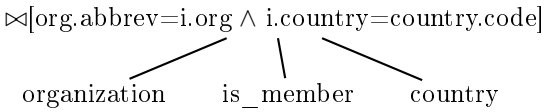
$$\pi[\text{org,country}](\text{is\_member}) \div \rho[\text{code}\rightarrow\text{country}](\pi[\text{code}](\text{country}))$$

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



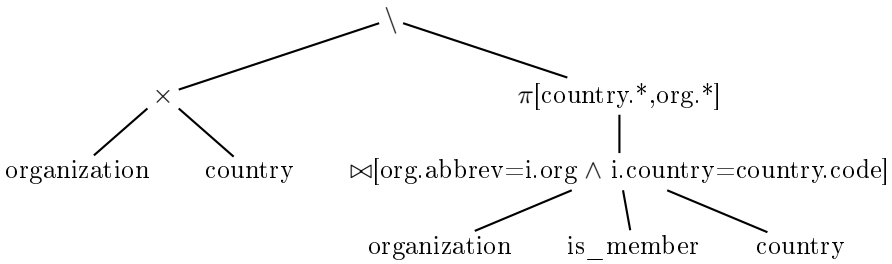
Man kann auch das NOT EXISTS entsprechend der Vorlesung in die Algebra übersetzen:

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

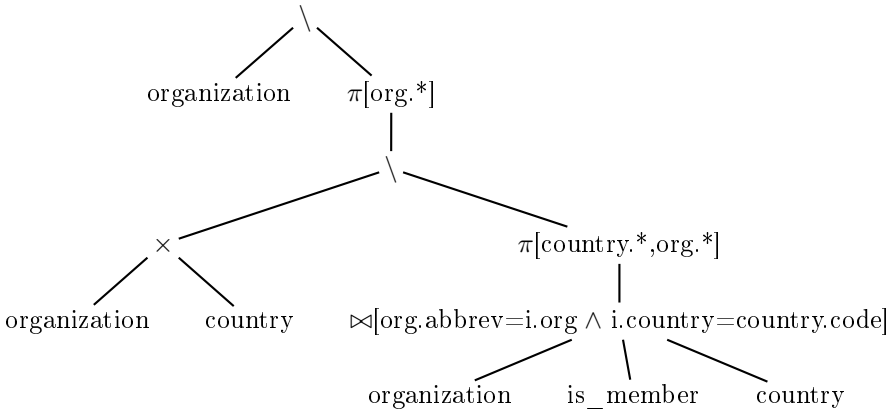
  Das Ergebnis dieses Ausdruckes wird auf die Schlüsselattribute der korrelierenden Relationen *organization* und *country* projiziert (ergibt diejenigen Paare von Organisationen und Länder, für die eine Mitgliedschaft existiert).
- Die Grundmenge der mittleren Subquery ist deren FROM zusammen mit der 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 ismember
     WHERE ismember.organization=organization.abbreviation
       AND ismember.country=CC));
```

Ausserdem:

- Formulieren Sie die Anfragen vom vorigen Blatt auch in SQL, und formulieren Sie die Anfragen an Mondial von diesem Blatt 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 ...