Datenbanken
Wintersemester 2018/19
Prof. Dr. W. May

3. Übungsblatt: SQL

Besprechung voraussichtlich am 12./19.12.2016

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.

Das Ziel dieser Aufgabe besteht aus mehreren Ebenen:

- vordergründig Datenbanken: Übersetzung von Ausdrücken der relationalen Algebra nach SQL,
- Prinzipien der Informatik: Strukturelle Induktion, Verwendung von Formalismen.

Zu zeigen ist, dass es für jeden Ausdruck e einen äquivalenten Ausdruck in SQL gibt; d.h., sei $\mathbf{R}$ ein relationales Schema und $\mathcal{S}$ ein Datenbankzustand über $\mathbf{R}$, dann gibt es eine SQL-Anfrage $q(e)$, so dass das Ergebnis von $q(e)$ (mit den üblichen "Semantik-Klammern" als $[\![q(e)]\!]_{\mathbf{R}}$ bezeichnet) gleich $\mathcal{S}(e)$ ist.

Um dies für *alle* Algebra-Ausdrücke zu beweisen, folgt man der Definition der Algebra-Ausdrücke durch *strukturelle Induktion*:

Induktionsanfang: die Basisfälle:

- Sei e ein Relationsname r. Dann ist `SELECT * FROM` r die entsprechende SQL-Anfrage $q(e)$.
- Sei $e = cname : \{x\}$, d.h. die "atomare Tabelle"

cname
x

  Dann ist (in Oracle) `SELECT 'x' AS cname FROM DUAL` die gesuchte Anfrage.
  Im allgemeinen werden Konstanten aber innerhalb von anderen Anfrage einfach eingesetzt, indem man z.B. `SELECT name, 'Fluss' AS gewaessertyp FROM River` schreibt.

Induktionschritt: Sei e ein zusammengesetzter Ausdruck, d.h. die Anwendung eines Algebraoperators auf einen oder zwei einfachere Ausdrücke e_1 und ggf. e_2. Dann existiert nach Induktionsvoraussetzung eine zu e_i äquivalente SQL-Anfrage $q(e_i)$, die man jeweils verwendet.
Fallunterscheidung nach dem jeweiligen Operator (zu betrachten sind nur die Basisoperatoren $\cup$, $\setminus$, $\sigma[...]$, $\pi[...]$, $\rho[...]$ und $(...) \bowtie (...)$ aus der Definition der relationalen Algebra):
Für die ersten 5 Fälle sei das Format von e_1, $q(e_1)$ und ggf. e_2 und $q(e_2)$ jeweils $[A_1, \ldots, A_n]$.

- $e = e_1 \cup e_2$:
  `(SELECT` $A_1, \ldots, A_n$ `FROM` $q(e_1)$`)`
  `UNION`
  `(SELECT` $A_1, \ldots, A_n$ `FROM` $q(e_2)$`)`
- $e = e_1 \setminus e_2$:
  `(SELECT` $A_1, \ldots, A_n$ `FROM` $q(e_1)$`)`
  `EXCEPT`
  `(SELECT` $A_1, \ldots, A_n$ `FROM` $q(e_2)$`)`
- $\sigma[cond](e_1)$:
  `SELECT` $A_1, \ldots, A_n$
  `FROM` $q(e_1)$
  `WHERE` $cond'$
  ($cond'$ ist der $cond$ entsprechende Ausdruck in SQL-Syntax)
-

- $\pi[A_{i_1}, \ldots, A_{i_k}](e_1)$ mit $1 \leq i_j \leq n$: `SELECT` $A_{i_1}, \ldots, A_{i_k}$ `FROM` $q(e_1)$
  Hinweis: in SQL sind außer den A_i auch geeignete Ausdrücke über den Attributnamen erlaubt: Arithmetik, Stringoperationen etc.; z.B.
  `SELECT name, population, area, population/area FROM Country.`
- $\rho[A_1 \to C_1, \ldots, A_n \to C_n](e_1)$: `SELECT` A_1 `AS` C_1 `,...,` A_n `AS` C_n `FROM` $q(e_1)$
- $e = e_1 \bowtie e_2$, wobei e_2 das Format $[A_{i_1}, \ldots, A_{i_k}, B_{k+1}, \ldots, B_m]$ hat, $\{i_1, \ldots, i_k\} \subseteq \{1 \ldots n\}$ und paarweise ungleich:
  `SELECT S.`A_1`, ..., S.`A_n`,` $B_{k+1}, \ldots B_m$
  `FROM` $q(e_1)$ `S ,` $q(e_2)$ `T;`
  `WHERE S.`A_{i_1} `= T.`A_{i_1}`, ..., S.`A_{i_k} `= T.`A_{i_k}`;`

Aufgabe 2 (Mondial (SQL)) Gegeben sei folgendes Datenbankschema (Auszug aus Mondial)

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

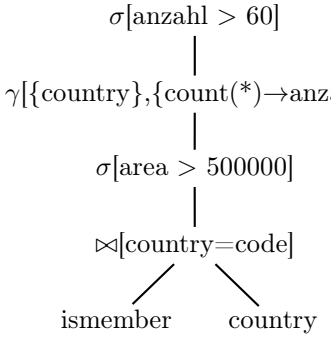
Hinweis: es wird in `ismember` für jedes Land eine Gruppe gebildet. Behalten werden diejenigen Gruppen, die mindestens 60 Einträge (Mitliedschaften des Landes in einer Organisation) enthalten). Für jede der Gruppen wird eine Zeile im Ergebnis generiert.

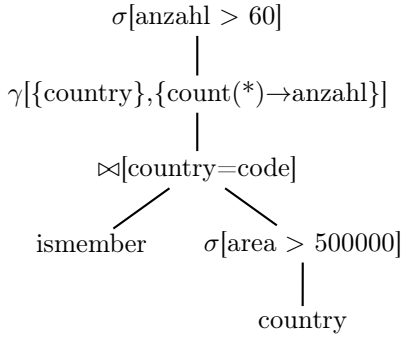
c) Welche Länder mit einer Fläche von mehr als 500000 km^2 sind Mitglied in mehr als 60 Organisationen?

- naheliegendste Formulierung

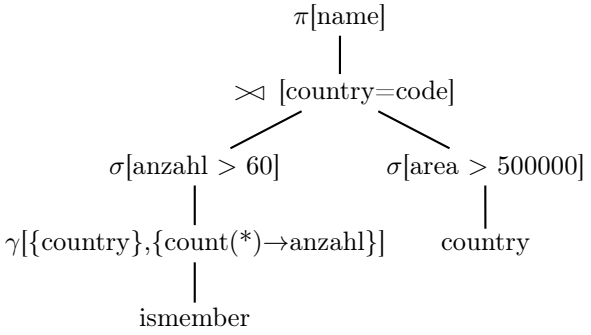
```
SELECT country
FROM ismember i, country c
WHERE i.country = c.code   -- join condition
  AND c.area > 500000      -- additional condition
GROUP BY i.country
HAVING count(*) > 60
```

- vgl. Aufgabe zu GROUP-BY-HAVING/Algebra:
  als Algebra-Baum:





  Auswertungsstrategien:
  - ismember (8000 Tupel), an jedes das passende Land (Foreign key → key) dranjoinen (immer noch 8000 Tupel), nach area auswählen (2300 Tupel), gruppieren (50 Gruppen/50 Tupel), Anzahl auswerten (bleiben 6)
  - Nur die Länder mit mehr als 500000 km^2 betrachten (51 Tupel), Mitgliedschaften dranjoinen (2300 Tupel), gruppieren (51 Gruppen), Anzahl auswerten.
  - Baum aus Teil b) nehmen: ismember nach country gruppieren, zählen, Anzahl auswerten (18 Tupel). Countries mit mehr als 500000 km^2 berechnen (50 Tupel). Semijoinen. Bleiben 6 Tupel.




```
SELECT country
FROM (SELECT country
      FROM ismember
      GROUP BY country
      HAVING count(*) > 60),
     (SELECT code FROM country WHERE area > 500000)
WHERE country = code;
```

  Letzteres ist sicher keine naheliegende SQL-Formulierung. Es ist aber egal, wie man die Anfrage formuliert, da sie von dem internen Optimierer vorverarbeitet wird.

- 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?

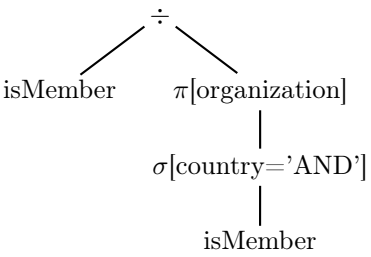
```
-- buggy in Oracle 12
SELECT name, code
FROM country c
WHERE NOT EXISTS
(SELECT * FROM organization o
```

```
  WHERE ('AND',o.abbreviation)  IN (SELECT country,organization FROM ismember)
    AND (c.code,o.abbreviation) NOT IN (SELECT country,organization FROM ismember))
```

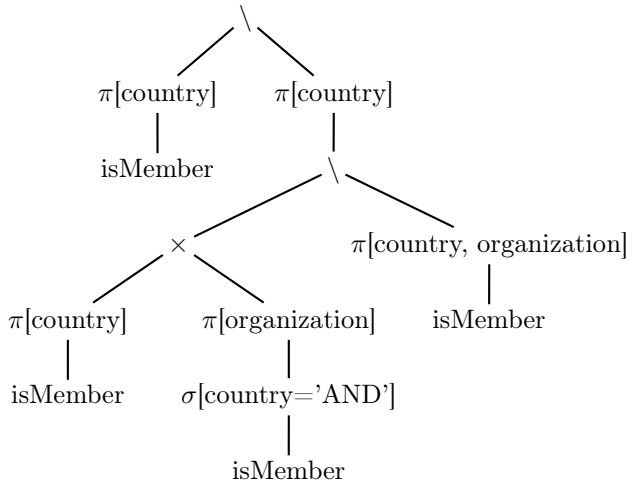
```
SELECT name, code
FROM country c
WHERE NOT EXISTS
 (SELECT organization
  FROM ismember i1
  WHERE country = 'AND'
    AND NOT EXISTS
    (SELECT DISTINCT organization
     FROM ismember i2
     WHERE i1.organization = i2.organization
       AND country = c.country ))
```

```
SELECT name, code
FROM country c
WHERE NOT EXISTS
 ((SELECT organization
   FROM ismember
   WHERE country = 'AND')
  MINUS
  (SELECT DISTINCT organization
   FROM ismember
   WHERE country = c.code
  ))
```

Als Algebra-Ausdruck ist es eine relationale Division:



Die Umschreibung ohne ÷ zeigt die Ähnlichkeit mit dem Pattern $r \div s = \pi[Z](r) - \pi[Z]((\pi[Z](r) \times s) - r)$ für die Division aus der Vorlesung:
Das kartesische Produkt aller Länder mit den Organisationen, in denen Andorra Mitglied ist, bezeichnet die "Soll-Menge", davon werden die tatsächlichen Mitgliedschaften abgezogen, die betreffenden Länder (bei denen Mitgliedschaften "fehlen") durch Projektion ausgewählt, und von der Menge aller Länder abgezogen:



Etwas komplizierter: Mengengleichheit *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

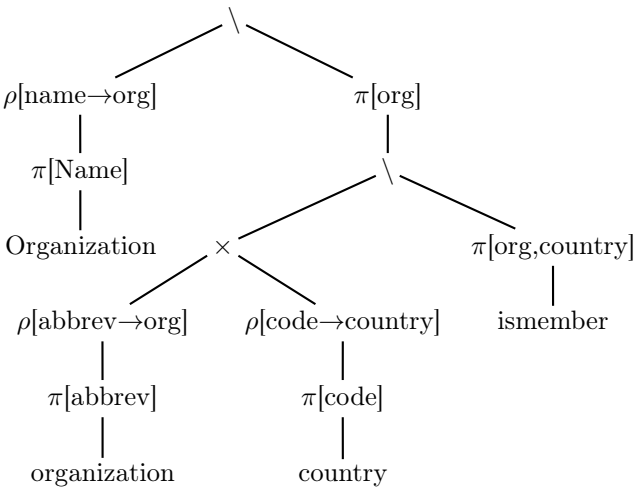
```
SELECT O.Name
FROM Organization O
WHERE NOT EXISTS
   (SELECT name
    FROM country C
    WHERE NOT EXISTS
       (SELECT organization
        FROM ismember I
        WHERE C.code = I.country
          AND I.organization = O.abbreviation))
```

... relationale Division:

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

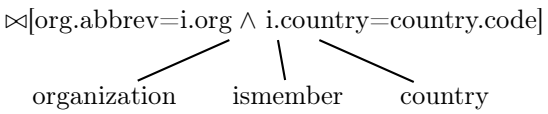
entsprechend der Zerlegung der Division in algebraische Grundoperationen (unter Benutzung

von π[country](ismember)=π[code]country und π[org](ismember)=π[abbrev](org):



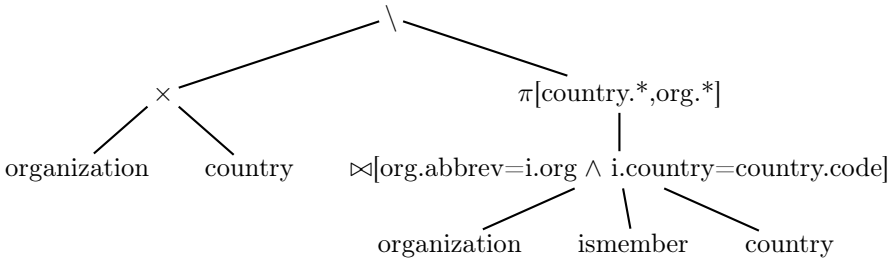
Man kann auch das NOT EXISTS entsprechend der Vorlesung in die Algebra übersetzen:

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

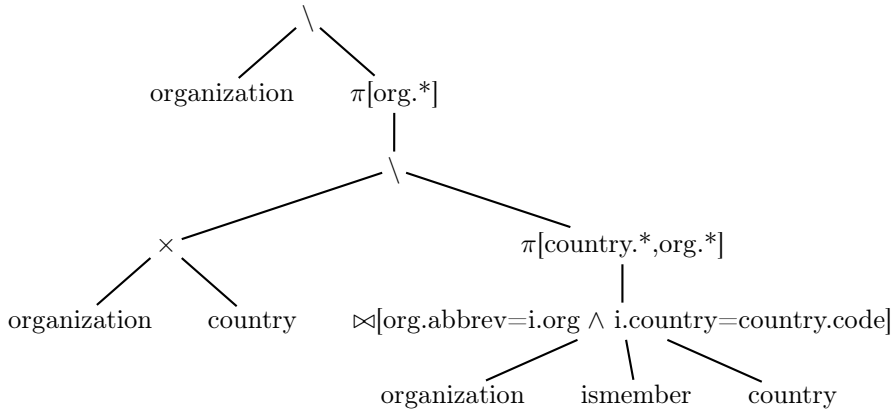
  Das Ergebnis dieses Ausdruckes wird auf die Schlüsselattribute der korrelierenden Relationen *organization* und *country* projiziert (ergibt diejenigen Paare von Organisationen und Länder, für die eine Mitgliedschaft existiert).

- 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));
```

Aufgabe 3 (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 4 (SQL und Algebra: HAVING) Diese Aufgabe behandelt die `GROUP BY` und `HAVING`-Klauseln von SQL-Anfragen.

a) Zeigen Sie: Alle SQL-Anfragen können auch ohne Verwendung von `HAVING` ausgedrückt werden. Geben Sie die SQL-Anfragen mit und ohne HAVING für die Anfrage "Welche Länder sind Mitglied in mehr als 60 Organisationen (mit Angabe der Anzahl der Mitgliedschaften)?" an.

b) Algebra: Definieren Sie einen Operator group-by, der die aus SQL bekannte Funktionalität von `GROUP BY` hat.

Betrachten Sie dabei für die Aggregatfunktionen nur einfache Anwendungen auf Attribute, wie z.B. max(population), nicht aber komplexere Ausdrücke wie max(population/area).

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?

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

---

a) Die `HAVING`-Bedingung wird auf die (virtuelle) Relation nach der Gruppierung (und Auswertung der Aggregatoperationen) angewendet. Damit kann man die virtuelle Relation in eine Subquery schachteln:

`SELECT` *attrs*, $op_1(attr_1)$, $op_n(attr_n)$
`FROM` *tables*
`WHERE` *where-cond*

```
GROUP BY group-attrs
HAVING having-cond
```

```
SELECT attrs, x_1, ..., x_n
FROM (SELECT attrs, op_1(attr_1) as x_1, op_n(attr_n) as x_n
      FROM tables
      WHERE where-cond
      GROUP BY group-attrs)
WHERE having-cond mit aliasen x_1, ..., x_n
```

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

SELECT country, num
FROM ( SELECT country, count(*) as num
       FROM ismember
       GROUP BY country)
WHERE num > 60;
```

b) **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 Attribute. Jede solche Definition enthält den Namen z_i des neuen Attributs sowie dessen Definition. Im angenommenen einfache Fall ist dies ein Paar (Aggregationsoperator, Attribut). Im allgemeinen Fall kann es ein Funktionsausdruck $f_i(\bar{X})$ über $\bar{X}$ sein, wobei f_i Aggregatoperationen (count, sum, max, min, avg) sowie Attribute aus $\bar{A}$ enthalten (z.B. $\mathsf{max}(x_3)$, $\mathsf{sum}(x_4)/\mathsf{count}(*)$, oder $\mathsf{max}(x_1/x_3)$).

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:

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

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

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

c) π[country, num](σ[num>60](group-by[{country}, {num := count(*)}](ismember)))

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
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';

oder als zweiten Teil:
DROP TABLE R;
RENAME R' TO R;
```

In der Praxis geht letzteres nur, wenn man das Recht hat, Tabellen zu erstellen, und (fremde) Tabellen zu löschen. Der Owner von R ist nachher derjenige, der die obigen Kommandos ausführt.

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 ...