[4131658] Jan Germann
[4054962] Christian Ratz
Übungsgruppe 1

Relationale Datenbanken
23. Februar 2013

Aufgabenblatt 8

8.1 Aufgabe

a. Name of all persons who played a role in a »parody« of any movie.

```
SELECT p.name
FROM Movie m, Person p, actor a, ``connection co''
WHERE p.id = a.person
AND a.movie = m.id
AND m.id = ``co''.movie
AND ``co.type'' = 'parody';
```

b. Names of all persons who were born in July.

```
SELECT name
FROM Person
WHERE MONTH(birthday)='7';
```

c. Titles of all movies that have been directed by more than one person.

```
SELECT m.title
FROM Movie m
JOIN director d
ON (m.id = d.movie)
GROUP BY m.id;
```

Nur gruppieren reicht nicht!

d. Id, name and number of movies the person acted in for each person. The result should also include those persons who never played a role in a movie yet.

```
SELECT p.id, p.name, COUNT(*) AS MovieCount
FROM Person p
JOIN actor a ON (p.id = a.person)
JOIN Movies m ON (m.id = a.movie)

UNION
SELECT p.id, p.name, '0'
FROM Person p, actor a
WHERE a.person <> p.id;
```

Left Join tut genau das gewollte!

e. Titles of all movies in which »Ted Codd« played a role but »Ray Boyce« did not.

```
SELECT m.title
FROM Movie m, Person p, actor a
WHERE a.person = p.id
AND a.movie = m.id
AND p.name = 'Ted Codd'
EXCEPT (SELECT m.title
FROM Movie m, Person p, actor a
WHERE a.person = p.id
AND a.movie = m.id
AND p.name = 'Ray Boyce');
```

f. The names of all persons, who acted and directed in the same movie for at least two times.

```
SELECT p.name
FROM person p
JOIN actor a ON (p.id = a.person)
JOIN director d ON (a.person = d.person) AND (a.movie = d.movie)
WHERE COUNT(p.id) > 1;
```

Subquery fehlt!

g. Considering the average number of movies persons played roles in. Return the names of all persons who played roles in more movies than the average.

```
WITH
  Avrg AS (
    SELECT AVG(COUNT(actor.person)) as average FROM actor
  )
SELECT p.name,
COUNT(a.person) AS played
FROM Person p, actor a, Avrg av
WHERE  played > av.average;
```

8.2 Aufgabe

a. What is the difference between the WHERE and the HAVING clause?

Der erste Unterschied wäre, dass die WHERE-Klausel vor der **GROUP-BY**-Klausel verwendet wird, die HAVING-Klausel immer danach benutzt wird. Bei der Verwendung von **HAVING** wird erst eine komplette Abfrage ausgeführt, dann gruppiert und am Ende der Bedingungen an die Abfrage angepasst. Bei **WHERE** werden erst die entsprechenden Zeilen herausgesucht und am Ende gruppiert zurückgegeben.

b. Would SQL have the same expressiveness if the HAVING clause would not exist?
Ja.

c. What is the difference between correlated and uncorrelated subqueries?

Falls die WHERE-Klausel der inneren Abfrage (Subquery) ein Attribut aus einer Relation verwendet, das in der äußeren Abfrage deklariert wurde, sind beide Abfragen korreliert. D.h die innere Abfrage muss für jedes Tupel der äußeren Abfrage neu bewertet werden. Da dies ineffizient ist, sollte man Unterabfragen (Subqueries) vermeiden und falls möglich, auf JOINS zurückgreifen.

d. What is the difference between a set and a bag? How can you enforce the DBMS to return a set instead of a bag?

Bags sind multiple Mengen von Zeilen. Duplikate sind standardmäßig erlaubt, können jedoch beim Anfragen eliminiert werden. Jeder einzelne Wert ist eine Zeilenmenge. Bags werden oft auch Ergebnismenge genannt. Letztlich ist es dasselbe, bis auf die Tatsache, dass sich Ergebnismengen durch die ORDER BY - Operation zu einer Listen machen lassen. Sets erlauben keine Duplikate.