

Aufgabenblatt 8

8.1 Aufgabe

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

```
1 SELECT p.name
2 FROM Movie m, Person p, actor a, hasGenre hg
3 WHERE p.id = a.person
4 AND a.movie = m.id
5 AND m.id = hg.movie
6 AND hg.genre = 'parody';
```

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

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

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

```
1 SELECT m.title
2 FROM Movie m
3 JOIN director d
4 ON (m.id = d.movie)
5 GROUP BY m.id;
```

- 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.

```
1 SELECT p.id, p.name, COUNT(*) AS MovieCount
2 FROM Person p
3 JOIN actor a ON (p.id = a.person)
4 JOIN Movies m ON (m.id = a.movie)
5
6 UNION
7 SELECT p.id, p.name, '0'
8 FROM Person p, actor a
9 WHERE a.person <> p.id;
```

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

```

1 SELECT m.title
2 FROM Movie m, Person p, actor a
3 WHERE a.person = p.id
4 AND a.movie = m.id
5 AND p.name = 'Ted_Codd'
6 EXCEPT (SELECT m.title
7 FROM Movie m, Person p, actor a
8 WHERE a.person = p.id
9 AND a.movie = m.id
10 AND p.name = 'Ray_Boyce');

```

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

```

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

```

- 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.

```

1 WITH
2   Avrg AS (
3     SELECT AVG(COUNT(actor.person)) as average FROM actor
4   )
5 SELECT p.name,
6 COUNT(a.person) AS played
7 FROM Person p, actor a, Avrg av
8 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?

Nein. In der **WHERE**-Klausel werden Vorbedingungen definiert, durch die **GROUP-BY**-Klausel wird die Selektion gruppiert und optionale Nachbedingungen (z.B Funktionen) werden in der **HAVING**-Klausel beschrieben. Des Weiteren ist Aggregation in der **WHERE**-Klausel verboten. Beispielsweise will man alle Institute des Informatikzweigs der TU Braunschweig mit mehr als 10 Mitarbeitern die ein höheres Monatsgehalt als 5000 Euro bekommen.

```
1 SELECT Institut, COUNT(*) AS 'Anzahl_Mitarbeiter'
2 FROM Mitarbeiter
3 WHERE Monatsgehalt > 5000
4 GROUP BY Institut
5 HAVING COUNT(*) > 10;
```

Würde man die **HAVING**-Klausel weglassen und stattdessen die **WHERE**-Klausel folgendermaßen definieren:

```
WHERE Monatsgehalt > 5000 AND COUNT(*) > 10
```

Dann würde man einen Fehler wegen Aggregation bekommen, da in der **WHERE**-Klausel Aggregation verboten ist.

- 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.