TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Michael Jungmair

***Database System Concepts for Non-Computer Scientist* - WiSe 25/26**

http://db.in.tum.de/teaching/ws2526/DBSandere/?lang=en

**Sheet 05**

## Exercise 1

Write a query that determines the kind of degree a student is pursuing. In our database, we assume that this can be deduced from the student's semester in the following way: A student who has not reached her 7th semester yet is still considered a "bachelor student". Once in the 7th semester, she should be categorized as a "master student". Starting in the 11th semester, we label her as a "phd student".

**Solution:**

```
select s.studNr, s.name,
   case
      when s.semester < 7 then 'bachelor student'
      when s.semester < 11 then 'master student'
      else 'phd student'
   end) as degree
from Students s
```

## Exercise 2

Answer the following questions on our university database using SQL:

a) Calculate how many lectures each student is attending. Students who do not attend any lecture should be included in the result as well ($attend\_count = 0$) (use outer joins).

b) Figure out how many students each professor knows: A professor knows students from one of their lectures or via a test they have supervised. Include professors not knowing any students and use outer joins. Hint: [1]

[1] Remember that SQL has set operations.

**Solution:**

a)
```
select s.studNr, s.name, count(a.studNr)
from Students s left outer join attend a on s.studnr = a.studnr
group by s.studNr, s.name
```

b)
```
select p.persNr, p.name, count(p.studNr)
from
((
  select p.persNr, p.name, t.studNr
  from Professors p
    left outer join test t on p.persNr = t.persNr
)
union
(
  select p.persNr, p.name, a.studNr
  from Professors p
    left outer join Lectures l on p.persNr = l.given_by
    left outer join attend a on l.lectureNr = a.lectureNr
)) p
group by p.persNr, p.name
```

Uncorrelated subqueries can be easily transformed into with-statements to make the query more readable:

```
with known_from_tests as (
   select p.persNr, p.name, t.studNr
   from Professors p
      left outer join test t on p.persNr = t.persnr
),
known_from_lectures as (
   select p.persNr, p.name, a.studNr
   from Professors p
      left outer join Lectures l on p.persNr = l.given_by
      left outer join attend a on l.lectureNr = a.lectureNr
),
known as (
   select * from known_from_tests
      union
   select * from known_from_lectures
)
select persNr, name, count(distinct studNr)
from known
group by persNr, name
```

## Exercise 3

How many *Assistants* does each *Professor* have? Print out a list with the name of the professor and the number of her assistants. Think about why professors that don't have any assistants don't show up. How could you include them in the list. Hint: [2]

[2] Using union, we can combine the result of two queries with the same schema: select a.name from (select 'Kant' as name union select 'Sokrates' as name) a;

**Solution:** The intension of this exercise was to play around with `group by`. We can write a rather short query to get the number of assistants for each professor. However, the professors without any assistants are missing, because these tuples are dropped during the join, because they don't find a join partner.

```
select p.persNr, p.name, count(*) as numberOfAssistants
from Professors p, Assistants a
where p.persNr = a.boss
group by p.persNr, p.name
```

To solve this we could use union to add the missing tuples:

```
select p.persNr, p.name, count(*) as numberOfAssistants
from Professors p, Assistants a
where p.persNr = a.boss
group by p.persNr, p.name

union

select p.persNr, p.name, 0 as numberOfAssistants
from Professors p
where not exists (select *
                  from Assistants a
                  where a.boss = p.persNr)
```

However, the simplest way to solve this query would be by using a sub query:

```
select p.persNr, p.name, (select count(*)
                          from Assistants a
                          where a.boss = p.persNr)
from Professors p
```