TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Database System Concepts for Non-Computer Scientist – **WiSe 20/21**

Alexander van Renen (renen@in.tum.de)
http://db.in.tum.de/teaching/ws2021/DBSandere/?lang=en

Sheet 09

Exercise 1

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]

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
```

Exercise 2

Create SQL DML statements for the following tasks:

a) “Professor meeting”: Move all professors to room 419.

b) “Lazy students”: Remove all students from the database who have ever failed a test (grade worse than 4.0).

[1] Remember that SQL has set operations.

Solution:

a) "Professor meeting": Move all professors to room 419.

```
update Professors set room = 419;
```

b) "Lazy students": Remove all students from the database who have ever failed a test (grade worse than 4.0).

```
delete from students s
where exists (select *
   from test t
   where t.grade > 4.0
   and t.studNr = s.studNr);
```

Exercise 3

Find those students who have attended all lectures that they wrote a test in.

Solution:

The requirement that students in the query result should have attended all lectures that they were tested in, can be rephrased as follow: "For a given student, there should be no test/exam, that has no entry in *attend*".

This can then be translated into sql easily.

```
select s.*
from Studenten s
where not exists(select * from pruefen p
                 where s.MatrNr = p.MatrNr
                   and not exists
                       (select *
                        from hoeren h
                        where h.MatrNr = s.MatrNr
                          and h.VorlNr = p.VorlNr)
                            );
```

This query is an example of a "for all query" where the counting-based technique can not be applied. The reason is that we can not simply count the number of attended lectures, because we need to make sure that the attended lectures match the ones that were tested.

Exercise 4

Considering the following table definitions:

1) `create table A(a int primary key);`
   `create table B(b int);`

2) `create table A(a int primary key);`
   `create table B(b int references A(a));`

Assuming the cardinalities (number of tuples) of the relation A and B are $|A|$ and $|B|$, respectively. How many tuples are produced by the following queries. If no exact estimate is possible, give an range. Alternatively you can use mathematical set operations.

a) `select * from A, B;`

b) `select * from A join B on A.a = B.b;`

c) `select * from A left outer join B on A.a = B.b;`

d) `select * from A right outer join B on A.a = B.b;`

e) `select * from A full outer join B on A.a = B.b;`

Solution:

As ranges:

	1)	2)
a)	exactly: $\|A\| \cdot \|B\|$	exactly: $\|A\| \cdot \|B\|$
b)	between 0 and $\|B\|$	exactly: $\|B\|$
c)	between $\|A\|$ and $\|A\| + \|B\| - 1$	between $\|A\|$ and $\|A\| + \|B\| - 1$
d)	exactly: $\|B\|$	exactly: $\|B\|$
e)	between $\max(\|A\|, \|B\|)$ and $\|A\| + \|B\|$	between $\max(\|A\|, \|B\|)$ and $\|A\| + \|B\| - 1$