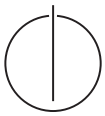

TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Michael Jungmair

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

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

**Sheet 06**

## Preparation

To make sure we can interactively work on real world data next week, please execute the following steps beforehand. Otherwise, e.g., the internet connection in the lecture hall could be the bottleneck.

(a) Install the DuckDB's Command Line Interface on your computer:
https://duckdb.org/install/

(b) Download and unpack the dataset we will be using:
https://db.in.tum.de/teaching/ws2526/DBSandere/dataset.zip

(c) Bring your computer to the lecture :)

## Exercise 1

Bob has programmed a course catalog for the university and published it online at `https://db.in.tum.de/teaching/ws2526/DBSandere/sql_catalog.html`.

To make searching easier, the number of SWS can be limited through a parameter. Find a specially crafted parameter which, when entered, outputs the list of students instead of the lectures. The database follows the well-known university schema.

Bob learns about the security vulnerability and suggests renaming the known tables once using random names, arguing that this way they cannot be found. Would this *security measure* help?

- Injection: `0 union all select name, matrnr, semester from studenten`
- No, because for example `0 union select tablename,1,1 from pg_tables` can output a list of the databases.

## Exercise 2

"Degree of popularity": Formulate an SQL query to determine the popularity of students. Assume that students know each other if they have attended the same lecture. Sort the result in descending order of popularity!

**Solution:**

First, we define a view that lists all acquaintances for each student. Afterwards, we count these acquaintances to determine the popularity of the students. To ensure that students without acquaintances also appear in the result, we use an outer join.

```
with Acquaintances as (
  select distinct a1.StudNr as Student, a2.StudNr as Acquaintance
  from attend a1, attend a2
```

```
  where a1.LectureNr = a2.LectureNr
    and a2.StudNr <> a1.StudNr
)
select s.StudNr, s.Name, count(ac.Acquaintance) as NumAcquaintances
from Students s left outer join Acquaintances ac
  on (s.StudNr = ac.Student)
group by s.StudNr, s.Name
order by NumAcquaintances desc
```

Without a view, the query becomes accordingly more complex:

```
select s.StudNr, s.Name, count(ac.Acquaintance) as NumAcquaintances
from Students s left outer join
  (select distinct a1.StudNr as Student, a2.StudNr as Acquaintance
   from attend a1, attend a2
   where a1.LectureNr = a2.LectureNr
     and a2.StudNr <> a1.StudNr
  ) ac on (s.StudNr = ac.Student)
group by s.StudNr, s.Name
order by NumAcquaintances desc
```