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

# Übung zur Vorlesung *Grundlagen: Datenbanken* im WS22/23

Michael Jungmair, Stefan Lehner, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2223/grundlagen/

**Blatt Nr. 06**

## Hausaufgabe 1

Welche Studenten haben alle Vorlesungen, die sie haben prüfen lassen, auch tatsächlich vorher gehört? Formulieren Sie eine SQL-Anfrage, welche diese Studenten ausgibt.

**Lösung:**
Die Anforderung, dass die Studenten im Anfrage-Ergebnis alle Vorlesungen, die sie haben prüfen lassen auch tatsächlich gehört haben, lässt sich umschreiben zu: „Es darf keine Vorlesung geben, die geprüft wurde, zu der es aber keinen Eintrag in *hoeren* gibt."

```
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));
```

## Hausaufgabe 2

Gegeben die Relationen:

- $Spieler = \{\underline{SpielerID}, Name, Alter, Team\}$
- $Herkunft = \{\underline{Team}, Kontinent\}$
- $Einsatz = \{\underline{SpielerID, Datum, Ort}, Tore\}$

Wer wurde Weltmeister? Geben Sie ein SQL Statement an, welches den Namen des Teams bestimmt.

Hinweis: Das Finalspiel ist das einzige Spiel am letzten Tag der WM. Aggregatfunktionen wie z.B. MIN, MAX und COUNT sind auch für Datumswerte definiert.

Beispielausprägungen:

```
with Spieler (SpielerID, Name, Alter, Team) as (
values
  (1, 'Neuer', 99, 'Deutschland'),
  (2, 'Ronaldo', 99, 'Brasilien'),
  (3, 'Messi', 99, 'Argentinien')
),
Herkunft (Team, Kontinent) as (
values
  ('Deutschland', 'Europa'),
  ('Brasilien', 'Suedamerika'),
```

```
  ('Argentinien', 'Suedamerika')
),
Einsatz (SpielerID, Datum, Ort, Tore) as (
values
  (1, '2014-06-20', 'Egal', 5),
  (2, '2014-06-12', 'Egal', 0),
  (3, '2014-06-20', 'Egal', 0)
)
```

**Lösung:**

```
WITH Finale AS (
  SELECT s.Team, SUM(e.Tore) as tore
  FROM Spieler s, Einsatz e
  WHERE e.Datum = (SELECT MAX(e2.Datum) FROM Einsatz e2)
  AND s.SpielerID = e.SpielerID
  GROUP BY s.Team
)

SELECT f.Team
FROM Finale f
WHERE f.Tore = (SELECT MAX(f2.Tore) FROM Finale f2)
```

## Hausaufgabe 3

„Fleißige Studenten“: Formulieren Sie eine SQL-Anfrage, um die Studenten zu ermitteln, die mehr SWS belegt haben als der Durchschnitt. Berücksichtigen Sie dabei auch Totalverweigerer, die gar keine Vorlesungen hören.

**Lösung:**

Folgende SQL-Anfrage ermittelt die fleißigen Studenten:

```
select s.*
from Studenten s
where s.MatrNr in
  (select h.MatrNr
   from hoeren h join Vorlesungen v on h.VorlNr = v.VorlNr
   group by h.MatrNr
   having sum(SWS) >
      (select sum(cast(SWS as decimal(5,2)))/count(distinct(s2.MatrNr))
       from Studenten s2
          left outer join hoeren h2 on h2.MatrNr = s2.MatrNr
          left outer join Vorlesungen v2 on v2.VorlNr = h2.VorlNr));
```

Durch die Verwendung von **with** und **case** wird die Anfrage übersichtlicher:

```
with GesamtSWS as (
  select sum(cast(SWS as decimal(5,2))) as AnzSWS
  from hoeren h2, Vorlesungen v2
  where v2.VorlNr = h2.VorlNr
),
GesamtStudenten as (
  select count(MatrNr) as AnzStudenten
  from Studenten
)
select s.*
```

```
from Studenten s
where s.MatrNr in (
  select h.MatrNr
  from hoeren h join Vorlesungen v on h.VorlNr = v.VorlNr
  group by h.MatrNr
  having sum(SWS) > (select AnzSWS / AnzStudenten
                     from GesamtSWS, GesamtStudenten));
```

Alternativ:

```
with SWSProStudent as (
select s.MatrNr,
  cast((case when sum(v.SWS) is null
             then 0 else sum(v.SWS)
        end) as real) as AnzSWS
 from Studenten s
   left outer join hoeren h on s.MatrNr = h.MatrNr
   left outer join Vorlesungen v on h.VorlNr = v.VorlNr
 group by s.MatrNr
)

select s.*
from Studenten s
where s.MatrNr in (select sws.MatrNr
                   from SWSProStudent sws
                   where sws.AnzSWS > (select avg(AnzSWS)
                                       from SWSProStudent));
```

## Hausaufgabe 4

*Klausuraufgabe aus dem WiSe 2016/17:*

Gegeben sei das bekannte Uni-Schema. Formulieren Sie in SQL-92: Finden Sie alle Vorlesungen **(VorlNr und Titel duplikatfrei ausgeben)**, die nicht vor dem dritten Semester gehört werden sollten. – Ein Beispiel hierfür ist die Vorlesung Bioethik (5216), da diese die Vorlesung Ethik (5041) voraussetzt, welche wiederum die Vorlesung Grundzüge (5001) als Voraussetzung hat.

**Lösung:**

Ohne Rekursion:

```
SELECT DISTINCT v.VorlNr, v.Titel
FROM Vorlesungen v, voraussetzen v1, voraussetzen v2
WHERE v.VorlNr = v1.Nachfolger
  AND v1.Vorgaenger = v2.Nachfolger
```

Mit Rekursion:

```
WITH recursive vor as (
  SELECT *, 1 as cnt FROM voraussetzen
  UNION
```

```
  SELECT v1.vorgaenger, v2.nachfolger, v1.cnt + 1
  FROM vor v1, voraussetzen v2
  WHERE v1.nachfolger = v2.vorgaenger
  )
SELECT DISTINCT nachfolger, v.Titel
FROM vor, vorlesungen v
WHERE vor.nachfolger = v.VorlNr AND vor.cnt >= 2
```