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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS16/17

Harald Lang, Linnea Passing (gdb@in.tum.de)
http://www-db.in.tum.de/teaching/ws1617/grundlagen/

Blatt Nr. 06

Tool zum Üben von SQL-Anfragen: `http://hyper-db.com/interface.html`.

Hausaufgabe 1

Gegeben die Relationen:

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

Wer wurde Weltmeister? Gegen 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 2

Was bringt der Vorlesungsbesuch? Finden Sie heraus, ob es für Prüfungen von Vorteil ist, die jeweiligen Vorlesungen auch gehört zu haben. Ermitteln Sie dazu die Durchschnittsnote der Prüfungen, zu denen die Studenten die Vorlesungen nicht gehört haben und die Durchschnittsnote der Prüfungen, zu denen sie die Vorlesungen gehört haben. - Formulieren Sie Ihre Antwort in SQL.

Lösung:

Diese Anfrage lässt sich auf zwei Arten beantworten. Zum einen kann ermittelt werden, wie das Verhältnis der Prüfungen für jede Vorlesung aussieht. Eine mögliche Formulierung hierfür ist folgende:

```
select ngehoert.VorlNr, ngehoert.ds, gehoert.ds
from (select p.VorlNr, avg(p.Note) as ds
      from pruefen p
      where not exists( select *
                        from hoeren h
                        where h.MatrNr = p.MatrNr
                        and h.VorlNr = p.VorlNr)
      group by p.VorlNr) ngehoert,
     (select p.VorlNr, avg(p.Note) as ds
      from pruefen p
      where p.VorlNr in (select h.VorlNr
                         from hoeren h
                         where h.MatrNr = p.MatrNr)
      group by p.VorlNr) gehoert
where ngehoert.VorlNr = gehoert.VorlNr;
```

Alternativ kann aber auch bestimmt werden, wie das Verhältnis der Prüfungsleistungen von gehörten zu nicht gehörten Vorlesungen sich insgesamt darstellt.

```
create view nichtgehoert as
  select avg(Note) as DnoteVLNichtGehoert
  from  pruefen p
  where not exists (select *
                    from hoeren h
                    where h.VorlNr = p.VorlNr
                      and h.MatrNr = p.MatrNr);

create view gehoert as
```

```
select avg(Note) as DnoteVLGehoert
from pruefen p
where exists (select *
              from hoeren h
              where h.VorlNr = p.VorlNr
                and h.MatrNr = p.MatrNr);
```

Da beide Views aus jeweils nur einem Wert bestehen, ergibt sich das Resultat der Anfrage als Kreuzprodukt:

```
select *
from nichtgehoert, gehoert;
```

Hausaufgabe 3

Welche Studenten haben alle Vorlesungen, die sie haben prüfen lassen, auch tatsächlich vorher gehört?

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 4

„Frühestes Semester": Formulieren Sie eine SQL-Anfrage, um das Semester zu ermitteln in dem die Vorlesung "Der Wiener Kreis" frühestens gehört werden kann. Testen Sie die Anfrage auch mit anderen Vorlesungen, insbesondere mit „Logik".

Lösung:

```
with recursive voraussetzenRek(vorlnr, counter) as (
(
  select v.vorlnr, 1 as counter
    from vorlesungen v
   where v.titel = 'Der Wiener Kreis'
)
union
(
  select vs.vorgaenger, vsr.counter + 1 as counter
    from voraussetzenRek vsr, voraussetzen vs
   where vsr.vorlnr=vs.nachfolger
)
)
select max(counter) as fruehestesSemester
```

```
from voraussetzenRek;
```

Hausaufgabe 5

Gegeben sei die Tabelle `ubahn`, die strukturell dem folgenden Beispiel gleicht:

Von	Nach	Dauer
Garching Forschungszentrum	Garching	2
Garching	Garching Hochbrück	2
Garching Hochbrück	Fröttmaning	4
Fröttmaning	Kieferngarten	2
Kieferngarten	Freimann	1
...	...	...
Odeonsplatz	Marienplatz	1
...	...	...
Haderner Stern	Klinikum Großhadern	1

a) Geben Sie eine Anfrage an, welche für eine gegebene Station, beispielsweise `Garching Forschungszentrum`, ermittelt, welche **anderen** Stationen von hier erreicht werden können. Geben Sie diese **duplikatfrei** aus.

b) Ermitteln Sie die Gesamtdauer, die eine Fahrt von einer gegeben zu jeder anderen Station benötigt.

c) Geben Sie eine SQL Anfrage an, welche alle von `Freimann` in beide Richtungen rekursiv erreichbaren Stationen ermittelt. Bilden Sie hierzu zunächst die Hülle in beide Richtungen. Was ist das Problem bei der Erstellung der einfachen Hülle auf der symmetrischen Basisrelation?

Lösung:

a) **Erreichbarkeit**

```
with huelle (von, nach) as (
  select von, nach from ubahn
  union all
  select h.von, u.nach
  from huelle h, ubahn u
  where h.nach = u.von
)
select distinct nach from huelle
 where von = 'Garching Forschungszentrum'
 order by von;
```

Die `ORDER BY` Klausel ist nicht nötig, macht das Ergebnis aber lesbarer.

b) **Dauer**

```
with huelle(von, nach, dauer) as (
  select von, nach, dauer from ubahn
  union all
  select u.von, h.nach, u.dauer + h.dauer
  from ubahn u, huelle h
  where u.nach = h.von
)
select * from huelle order by von;
```

Die Hülle kann alternativ wie oben "andersrum" aufgebaut werden.

c) **Doppelhülle**

```
with huelle(von, nach)  as (
  select von, nach from ubahn
  UNION ALL
  select u.von, h.nach from ubahn u, huelle h
  where u.nach = h.von
), doppelhuelle(von, nach) as (
  select nach, von from huelle
  UNION ALL
  select h.von, d.nach from huelle h, doppelhuelle d
  where h.nach = d.von
)
select distinct * from doppelhuelle order by von;
```

Bildet man die Hülle auf der symmetrischen Variante von `ubahn`, so terminiert die Anfrage nicht, da SQL Bag-Semantik nutzt und daher keine Duplikate eliminiert. Würde man Duplikate während der Rekursion ausschließen, wäre dies eine bessere alternative Lösung.