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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS18/19

Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws1819/grundlagen/

Blatt Nr. 05

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

Hausaufgabe 1

Formulieren Sie die folgenden Anfragen auf dem bekannten Universitätsschema in SQL:

a) Finden Sie alle Studenten, die drei oder mehr Vorlesungen hören

b) Finden Sie alle Grundlagenvorlesungen. Eine Vorlesung ist eine Grundlagenvorlesung, wenn sie mindestens 4 SWS hat und Voraussetzung von mindestens zwei anderen Vorlesungen ist.

c) Bestimmen Sie das durchschnittliche Semester der Studenten, die mindestens eine Vorlesung bei Sokrates hören.

d) Bestimmen Sie, wie viele Vorlesungen im Schnitt pro Student gehört werden. Beachten Sie, dass Studenten, die keine Vorlesung hören, in das Ergebnis einfließen müssen.

Lösung:

a) Finden Sie alle Studenten, die drei oder mehr Vorlesungen hören

```
select s.matrnr, s.name
from studenten s, hoeren h
where s.matrnr = h.matrnr
group by s.matrnr, s.name
having count(*) >= 3
```

b) Finden Sie alle Grundlagenvorlesungen. Eine Vorlesung ist eine Grundlagenvorlesung, wenn sie mindestens 4 SWS hat und Voraussetzung von mindestens zwei anderen Vorlesungen ist.

```
select v.vorlnr, v.titel
from vorlesungen v, voraussetzen vor
where
    v.vorlnr = vor.vorgaenger and
    v.sws >= 4
group by v.vorlnr, v.titel
having count(*) >= 2
```

c) Bestimmen Sie das durchschnittliche Semester der Studenten, die mindestens eine Vorlesung bei Sokrates hören.

```
with
vorlesungen_von_sokrates as (
  select *
  from vorlesungen v, professoren p
  where v.gelesenVon = p.persnr and p.name = 'Sokrates'
```

```
),
studenten_von_sokrates as (
  select *
  from studenten s
  where exists (
    select *
    from hoeren h, vorlesungen_von_sokrates v
    where h.matrnr = s.matrnr and v.vorlnr = h.vorlnr)
)
select avg(semester) from studenten_von_sokrates
```

Man beachte, dass die Formulierung mittels `WHERE EXISTS` für die Elimination von Duplikaten sorgt, d.h. ein Student, der 3 Vorlesungen von Sokrates hört kommt nur einmal in `studenten_von_sokrates vor`, was gewünscht ist. Alternativ kann man `studenten_von_sokrates` formulieren als:

```
select DISTINCT s.*
from studenten s, hoeren h, vorlesungen_von_sokrates v
where h.matrnr = s.matrnr and v.vorlnr = h.vorlnr
```

d) Bestimmen Sie, wie viele Vorlesungen im Schnitt pro Student gehört werden. Beachten Sie, dass Studenten, die keine Vorlesung hören, in das Ergebnis einfließen müssen.

Eine Möglichkeit, Studenten, die keine Vorlesung hören, in das Ergebnis einzubeziehen, ist ein äußerer Join. Dabei muss im ersten `count` ein beliebiges Attribut aus der Relation hören stehen, damit `NULL`-Werte nicht mitgezählt werden:

```
select count(h.vorlnr) * 1.000 / count(distinct s.matrnr)
from
    studenten s left outer join
    hoeren h on s.matrnr = h.matrnr
```

Andererseits ergibt sich die Lösung auch aus der Anzahl der Tupel in hören geteilt durch die Anzahl der Studenten:

```
select hcount/(scount*1.000)
from (select count(*) as hcount from hoeren) h,
     (select count(*) as scount from studenten) s
```

Hausaufgabe 2

Gegeben sei die Relation `Fahrplan`, die strukturell dem folgenden Beispiel gleicht:

Von	Nach	Linie	Abfahrt	Ankunft
Garching, Forschungszentrum	Garching	U6	09:06	09:09
Garching	Garching-Hochbrück	U6	09:09	09:11
Garching-Hochbrück	Fröttmaning	U6	09:11	09:15
...	...			
Fröttmaning	Garching-Hochbrück	U6	09:00	09:04
Garching-Hochbrück	Garching	U6	09:04	09:06
Garching	Garching, Forschungszentrum	U6	09:06	09:09
...	...			
Garching, Forschungszentrum	Technische Universität	690	17:56	17:57

Formulieren Sie die folgenden Anfragen auf diese Relation in SQL. Sie können dabei den Wert `CURRENT_TIME` für die aktuelle Zeit und die Funktion `EXTRACT(MINUTE FROM x)`, die aus einem Zeit- oder Zeitdifferenzwert die Minuten extrahiert, verwenden.

a) Geben Sie alle Einträge des Fahrplans aus, deren Abfahrtshaltestelle das Wort „Garching“ enthält.

b) Geben Sie alle Einträge des Fahrplans mit dem zusätzlichen Attribut „Verkehrsmittel“ aus. Alle Linien, die mit „U“ anfangen, haben das Verkehrsmittel „U-Bahn“, alle die mit „S“ anfangen „S-Bahn“, und alle restlichen „Bus/Tram“.

c) Finden Sie alle Abfahrten ab „Garching, Forschungszentrum“, die Sie heute noch erreichen können. Sortieren Sie das Ergebnis aufsteigend nach Abfahrtszeit.

d) Finden Sie alle Fahrten zwischen zwei Haltestellen (nicht-transitiv), die mindestens drei und höchstens fünf Minuten dauern.

Laden Sie zum Testen entweder die SQL-Datei von der Übungswebseite in ein lokal installiertes Datenbanksystem oder verwenden Sie die Webschnittstelle.

Lösung:

a) Geben Sie alle Einträge des Fahrplans aus, deren Abfahrtshaltestelle das Wort „Garching“ enthält.

```
select * from fahrplan where von like '%Garching%'
```

b) Geben Sie alle Einträge des Fahrplans mit dem zusätzlichen Attribut „Verkehrsmittel“ aus. Alle Linien, die mit „U“ anfangen, haben das Verkehrsmittel „U-Bahn“, alle die mit „S“ anfangen „S-Bahn“, und alle restlichen „Bus/Tram“.

```
select
    *,
    case
        when linie like 'U%' then 'U-Bahn'
        when linie like 'S%' then 'S-Bahn'
        else 'Bus/Tram'
    end as Verkehrsmittel
from fahrplan
```

c) Finden Sie alle Abfahrten ab „Garching, Forschungszentrum“, die Sie heute noch erreichen können. Sortieren Sie das Ergebnis aufsteigend nach Abfahrtszeit.

```
select *
from fahrplan
where
    abfahrt > current_time and
    von = 'Garching,␣Forschungszentrum'
order by abfahrt
```

d) Finden Sie alle Fahrten zwischen zwei Haltestellen (nicht-rekursiv), die mindestens drei und höchstens fünf Minuten dauern.

Wenn Fahrten, die länger als eine Stunde dauern oder die vor 00:00 Uhr abfahren und danach ankommen, ignoriert werden, kann `extract` direkt verwendet werden:

```
select *
from fahrplan
where extract(minute from ankunft - abfahrt) between 3 and 5
```

Ansonsten müssen beide Fälle abgefangen werden. Bei Fahrten über Mitternacht muss die Differenz zwischen Abfahrt und Ankunft korrigiert werden. Um Fahrten, die länger als eine Stunde dauern, nicht auszugeben, kann statt `extract` direkt ein Vergleich mit `INTERVAL`-Werten durchgeführt werden:

```
with fahrplan_dauer as (
    select
        *,
        case
            when ankunft < abfahrt
            then cast('24:00:00' as INTERVAL) + (ankunft - abfahrt)
            else ankunft - abfahrt
        end as dauer
    from fahrplan
)
select * from fahrplan_dauer
where
    dauer between
        cast('00:03:00' as INTERVAL) and
        cast('00:05:00' as INTERVAL)
```

Hausaufgabe 3

Gegeben sei die folgende Relation `ZehnkampfD` mit Athletennamen und den von ihnen erreichten Punkten in den jeweiligen Zehnkampfdisziplinen:

$$\text{ZehnkampfD} : \{\underline{\text{Name}, \text{Disziplin}}, \text{Punkte}\}$$

Name	Disziplin	Punkte
Eaton	100 m	450
Eaton	Speerwurf	420
...	...	...
Eaton	Weitsprung	420
Suarez	100 m	850
Suarez	Speerwurf	620
...	...	...

Finden Sie alle ZehnkämpferInnen, die in *allen* Disziplinen besser sind als der Athlet mit dem Namen *Bolt*. Formulieren Sie die Anfrage in SQL

a) mit korrelierter Unteranfrage

b) basierend auf Zählen

Sie dürfen davon ausgehen, dass jeder Sportler in jeder Disziplin angetreten ist.

Laden Sie zum Testen entweder die SQL-Datei von der Übungswebseite in ein lokal installiertes Datenbanksystem oder verwenden Sie die Webschnittstelle.

Lösung:

Mit korrelierter Unteranfrage:

```
select distinct a.Name
from ZehnkampfD as a
where not exists (
  select *
  from ZehnkampfD as a2
  where a2.Name = a.Name
    and exists (
      select *
      from ZehnkampfD as b
      where b.Disziplin = a2.Disziplin
        and b.Name = 'Bolt'
        and b.Punkte >= a2.Punkte
    )
)
```

Basierend auf Zählen

```
with besserAlsBolt(name, disziplin) as (
  select a.name, a.disziplin
  from zehnkampfd a, zehnkampfd b
  where b.name = 'Bolt'
    and a.disziplin = b.disziplin
    and a.punkte > b.punkte
),
disziplinen(anzahl) as (
  select count(distinct disziplin) as anzahl
  from zehnkampfd
)
select name
from besserAlsBolt
group by name
having count(*) = (select anzahl from disziplinen)
```