TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS19/20

Christoph Anneser, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws1920/grundlagen/

Blatt Nr. 07

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

Hausaufgabe 1

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 die Typen `TIME` für Uhrzeiten und `INTERVAL` für Zeitintervalle verwenden.

Schreiben Sie z.B. für 10:30 Uhr: `TIME '10:30:00'`.

Das 0-Intervall kann z.B. so konstruiert werden: `INTERVAL '00:00:00'`.

a) Geben Sie eine Anfrage an, welche für alle Stationen ermittelt, welche **anderen** Stationen erreicht werden können. Beachten Sie, dass nur tatsächlich mögliche Verbindungen ausgegeben werden sollen, d.h. die Abfahrt an einer Haltestelle darf nicht vor der Ankunft liegen.

b) Erweitern Sie ihre Anfrage aus Teilaufgabe a), sodass zusätzlich die summierte Fahrtzeit und Wartezeit sowie die gesamte Reisezeit ausgegeben wird. Die Fahrtzeit ist dabei nur die Zeit, in der man sich in einem Verkehrsmittel befindet. Die Wartezeit ist die Zeit, die bei einem Umstieg zwischen Ankunft des alten und Abfahrt des neuen Verkehrsmittels vergeht. Die Reisezeit ist die Zeit zwischen Abfahrt des ersten und Ankunft des letzen Verkehrsmittels.

c) Erweitern Sie ihre Anfrage aus Teilaufgabe a) oder b) nochmals und geben Sie die Anzahl der Umstiege für jede Verbindung aus.

d) Finden Sie die „guten“ Verbindungen, um von Fröttmaning pünktlich zur Vorlesung „Grundlagen: Datenbanken“ um 10:30 Uhr zu kommen. Verwenden Sie dazu Ihre Anfrage aus Teilaufgabe c). Eine Verbindung ist „gut“, wenn sie spätestens um 10:30 in „Garching, Forschungszentrum“ ist und es keine andere Verbindung gibt, die später abfährt aber noch rechtzeitig eintrifft, deren Reisezeit geringer ist und bei der man weniger Umstiege hat.

Lösung:

a) Geben Sie eine Anfrage an, welche für alle Stationen ermittelt, welche **anderen** Stationen erreicht werden können. Beachten Sie, dass nur tatsächlich mögliche Verbindungen ausgegeben werden sollen, d.h. die Abfahrt an einer Haltestelle darf nicht vor der Ankunft liegen.

Für diese Aufgabe werden *rekursive Sichten* mit `with recursive` benötigt. Die Relation `Fahrplan` bildet dort den Rekursionsanfang. Dann werden rekursiv mit `union all` weitere Verbindungen hinzugefügt. Eine neue Verbindung soll nur dann eingefügt werden, wenn die Ankunftshaltestelle gleich der Abfahrtshaltestelle der neuen Teilverbindung ist, wenn diese zeitlich erreichbar ist und wenn die neue Ankunftshaltestelle ungleich der ursprünglichen Abfahrtshaltestelle ist. Dadurch ergeben sich die `where`-Bedingungen.

```
with recursive fahrplan_rec as (
    select von, nach, abfahrt, ankunft from fahrplan
    union all
    select fr.von, f.nach, fr.abfahrt, f.ankunft
    from fahrplan_rec fr, fahrplan f
    where
        fr.nach = f.von and
        fr.ankunft <= f.abfahrt and
        fr.von != f.nach
)
select * from fahrplan_rec
```

b) Erweitern Sie ihre Anfrage aus Teilaufgabe a), sodass zusätzlich die summierte Fahrtzeit und Wartezeit sowie die gesamte Reisezeit ausgegeben wird. Die Fahrtzeit ist dabei nur die Zeit, in der man sich in einem Verkehrsmittel befindet. Die Wartezeit ist die Zeit, die bei einem Umstieg zwischen Ankunft des alten und Abfahrt des neuen Verkehrsmittels vergeht. Die Reisezeit ist die Zeit zwischen Abfahrt des ersten und Ankunft des letzen Verkehrsmittels.

Ausgehend von der Lösung von Teilaufgabe a) muss hier als Rekursionsanfang zusätzlich die Fahrzeit und die Wartezeit gesetzt werden. Bei einer Direktverbindung ergibt sich die Fahrtzeit aus der Differenz zwischen Ankunft und Abfahrt, die Wartezeit ist am Anfang 0. Im Rekursionsschritt muss nun die Fahrtzeit der neuen Teilverbindung der aktuellen Fahrtzeit hinzuaddiert werden. Ähnlich muss zur Wartezeit die Dauer zwischen der Ankunft der aktuellen Verbindung und der Abfahrt der neuen hinzuaddiert werden. Da sich die Reisezeit aus der Summe der Fahrtzeit und der Wartezeit ergibt, muss diese nicht notwendigerweise schon während der Rekursion berechnet werden, sondern kann durch eine weitere Sicht berechnet werden.

```
with recursive fahrplan_rec_linie as (
    select
        von,
        nach,
        abfahrt,
        ankunft,
        ankunft - abfahrt as fahrtzeit,
```

```
        INTERVAL '00:00:00' as wartezeit
    from fahrplan
    union all
    select
        fr.von,
        f.nach,
        fr.abfahrt,
        f.ankunft,
        fr.fahrtzeit + (f.ankunft - f.abfahrt),
        fr.wartezeit + (f.abfahrt - fr.ankunft)
    from fahrplan_rec_linie fr, fahrplan f
    where
        fr.nach = f.von and
        fr.ankunft <= f.abfahrt and
        fr.von != f.nach
),
fahrplan_rec as (
    select
        von,
        nach,
        abfahrt,
        ankunft,
        fahrtzeit,
        wartezeit,
        fahrtzeit + wartezeit as reisezeit
    from fahrplan_rec_linie
)
select * from fahrplan_rec
```

c) Erweitern Sie ihre Anfrage aus Aufgabe a) oder b) nochmals und geben Sie die Anzahl der Umstiege für jede Verbindung aus.

Man muss immer genau dann umsteigen, wenn durch den Rekursionsschritt eine neue Teilverbindung hinzukommt, die zu einer anderen Linie gehört als der aktuellen, oder wenn die Ankunftszeit der bisherigen Verbindung kleiner als die Abfahrtszeit der neuen Teilverbindung ist. Um ersteres zu erkennen, muss in der rekursiven Sicht immer die aktuelle Linie mitgeführt werden. Diese wird im Rekursionsanfang auf die Linie der Teilverbindung gesetzt. Im Rekursionsschritt wird dann mithilfe von `case` ermittelt, ob ein Umstieg stattfindet oder nicht. Im Rekursionsschritt muss auch die aktuelle Linie aktualisiert werden, damit die Erkennung von Umstiegen weiterhin korrekt funktioniert.

```
with recursive fahrplan_rec_linie as (
    select
        von,
        nach,
        abfahrt,
        ankunft,
        linie as aktuelle_linie,
        0 as umstiege,
```

```
        ankunft - abfahrt as fahrtzeit,
        INTERVAL '00:00:00' as wartezeit
    from fahrplan
    union all
    select
        fr.von,
        f.nach,
        fr.abfahrt,
        f.ankunft,
        f.linie,
        fr.umstiege + case
            when
                f.linie != fr.aktuelle_linie or
                f.abfahrt > fr.ankunft
            then 1
            else 0
        end,
        fr.fahrtzeit + (f.ankunft - f.abfahrt),
        fr.wartezeit + (f.abfahrt - fr.ankunft)
    from fahrplan_rec_linie fr, fahrplan f
    where
        fr.nach = f.von and
        fr.ankunft <= f.abfahrt and
        fr.von != f.nach
),
fahrplan_rec as (
    select
        von,
        nach,
        abfahrt,
        ankunft,
        umstiege,
        fahrtzeit,
        wartezeit,
        fahrtzeit + wartezeit as reisezeit
    from fahrplan_rec_linie
)
select * from fahrplan_rec
```

d) Finden Sie die „guten“ Verbindungen, um von Fröttmaning pünktlich zur Vorlesung „Grundlagen: Datenbanken“ um 10:30 Uhr zu kommen. Verwenden Sie dazu Ihre Anfrage aus Teilaufgabe c). Eine Verbindung ist „gut“, wenn sie spätestens um 10:30 in „Garching, Forschungszentrum“ ist und es keine andere Verbindung gibt, die später abfährt aber noch rechtzeitig eintrifft, deren Reisezeit geringer ist und bei der man weniger Umstiege hat.

Ausgehend von der Lösung von Teilaufgabe c) kann diese Aufgabe mit `not exists` gelöst werden:

```
select * from fahrplan_rec fr
```

```
where
    fr.von = 'Fröttmaning' and
    fr.nach = 'Garching,␣Forschungszentrum' and
    fr.ankunft <= TIME '10:30:00' and
    not exists (
        select * from fahrplan_rec fr2
        where
            fr2.von = fr.von and
            fr2.nach = fr.nach and
            fr2.ankunft <= TIME '10:30:00' and
            fr2.abfahrt > fr.abfahrt and
            fr2.reisezeit < fr.reisezeit and
            fr2.umstiege < fr.umstiege
    )
```

Hausaufgabe 2

Gegeben sei eine Relation

$$R : \{[A : \text{integer}, B : \text{integer}, C : \text{integer}, D : \text{integer}, E : \text{integer}]\},$$

die schon sehr viele Daten enthält (Millionen Tupel). Sie „vermuten“, dass folgendes gilt:

a) AB ist ein Superschlüssel der Relation

b) $DE \to B$

Formulieren Sie SQL-Anfragen, die Ihre Vermutungen bestätigen oder widerlegen.

Lösung:

a) Durch Gruppierung nach A und B kann anhand der Anzahl der Tupel ermittelt werden, ob hier eine Verletzung der Schlüsseleigenschaft vorliegt. Werden also mindestens zwei Tupel mit den gleichen Werten für A und B als Ergebnis ausgegeben, so bildet AB keinen Schlüssel der Relation, ist das Ergebnis der Anfrage jedoch leer, so ist AB ein Superschlüssel.

```
select A, B
from R
group by A, B
  having count(*) > 1;
```

b) In diesem Fall muss nur gelten, dass für alle Tupel, die gleiche Werte in D und E besitzen, auch die Werte für das Attribut B gleich sind. D.h. wenn nach D und E gruppiert wird, muss die Anzahl der verschiedenen Werte für B kleiner oder gleich 1 sein. Es gilt wieder, dass das Ergebnis der Anfrage alle Tupel enthält, die die Vermutung verletzen. Ist das Ergebnis leer, so gilt $DE \to B$.

```
select D, E
from R
group by D, E
  having count(distinct B) > 1;
```

Hausaufgabe 3

Betrachten Sie das Relationenschema

PunkteListe: {Name, Aufgabe, Max, Erzielt, KlausurSumme, KNote, Bonus, GNote}

mit der folgenden beispielhaften Ausprägung:

PunkteListe							
Name	Aufgabe	Max	Erzielt	KlausurSumme	KNote	Bonus	GNote
Bond	1	10	4	18	2	ja	1.7
Bond	2	10	10	18	2	ja	1.7
Bond	3	11	4	18	2	ja	1.7
Maier	1	10	4	9	4	nein	4
Maier	2	10	2	9	4	nein	4
Maier	3	11	3	9	4	nein	4

1. Bestimmen Sie die geltenden FDs.
2. Bestimmen Sie die Kandidatenschlüssel.

Lösung:

1. Im Relationenschema gelten die folgenden funktionalen Abhängigkeiten:
   - {KNote, Bonus} → {GNote}
   - {Aufgabe} → {Max}
   - {KlausurSumme} → {KNote}
   - {Name, Aufgabe} → {Erzielt}
   - {Name} → {KlausurSumme, Bonus}

   Natürlich gelten auch alle anderen funktionalen Abhängigkeiten, die mit Hilfe der Armstrong-Axiome daraus hergeleitet werden können.
2. Der Kandidatenschlüssel ist {Name, Aufgabe}. Aus {Name} können die Attribute {KlausurSumme, Bonus}, aus {KlausurSumme} wiederum {KNote}, und aus {KNote, Bonus} dann {GNote} abgeleitet werden. Aus {Aufgabe} kann {Max} abgeleitet werden, und aus {Name, Aufgabe} noch das verbleibende Attribut {Erzielt}.

Hausaufgabe 4

Geben Sie für jede der Normalformen 1NF, 2NF, 3NF, BCNF, 4NF jeweils eine Relation mit FDs an, sodass die Relation in der gewünschten Normalform ist (und in keiner höheren).

Lösung:

Für alle Normalformen betrachten wie die Relation $\mathcal{R} = \{A, B, C, D\}$.

- 1.NF:

  FDs:
  - $AB \to C$
  - $B \to D$

  Die Relation ist nicht mengenwertig, daher 1. NF. D ist lediglich von B abhängig, der Kandidatenschlüssel ist aber AB, weswegen D nicht voll funktional vom Kandidatenschlüssel abhängig ist, daher keine 2. NF.

- 2. NF:

  FDs:

  - $AB \to C$
  - $C \to D$

  Jedes Attribut der Relation ist voll funktional abhängig vom Kandidatenschlüssel AB, daher 2.NF. Das Attribut D ist transitiv und nicht direkt vom Kandidatenschlüssel abhängig, darum nicht 3. NF.

- 3. NF:

  FDs:

  - $AB \to CD$
  - $BC \to AD$
  - $D \to C$

  Für alle FDs gilt entweder, dass sie trivial ist, dass die linke Seite Superschlüssel ist oder dass die rechte Seite in einem Kandidatenschlüssel enthalten ist, daher 3. NF. Bei der BCNF fällt die dritte erlaubte Art von FD weg, daher FDs müssen trivial sein oder ihre linke Seite Superschlüssel. Da die dritte FD des Beispiels dies verletzt ist die Relation nicht in BCNF und daher genau in 3. NF.

- BCNF:

  FDs:

  - $AB \to CD$
  - $BC \to AD$
  - $D \to\!\!\!\!\to C$

  BCNF, da die BCNF verletzende FD aus dem Beispiel für 3. NF entfernt wurde. Nicht 4. NF weil eine nicht trivial MVD gilt, deren linke Seite nicht Superschlüssel ist.

- 4. NF

  FDs:

  - $AB \to CD$
  - $BC \to AD$

  Nach Entfernung der nicht trivialen MVD dann auch 4. NF.