Übung zur Vorlesung „Datenbanksysteme für Hörer anderer Fachrichtungen“

Richard Kuntschke (richard.kuntschke@in.tum.de)

Lösungen zu Blatt 3

Aufgabe 1

Formulierung in SQL:

Wir ermitteln in einem ersten Schritt die transitive Hülle der *verbindet*-Relation. Der reine SQL92-Standard bietet keine Rekursion, wie sie hierfür notwendig ist, so dass die transitive Hülle dann iterativ ermittelt werden muss. Der SQL-Dialekt von IBM DB2 hingegen ermöglicht Rekursionen, die wir im Folgenden beispielhaft verwenden:

```
with verbindetTrans(
        ZugNr, VonBahnhof, NachBahnhof, Abfahrt, Ankunft)
as( select * from verbindet
    union all
    select t.ZugNr, t.VonBahnhof, n.NachBahnhof,
           t.Abfahrt, n.Ankunft
    from  verbindetTrans t, verbindet n
    where t.ZugNr = n.ZugNr
     and  t.Abfahrt < n.Ankunft
     and  t.NachBahnhof = n.VonBahnhof)
```

Darauf aufbauend sehen die drei Anfragen zum Beispiel wie folgt aus:

Anfrage 1:

```
select v.ZugNr, v.Abfahrt, v.Ankunft
from verbindetTrans v, Bahnhöfe von, Bahnhöfe nach
where von.SName = 'Passau' and nach.SName = 'Karlsruhe'
 and  v.VonBahnhof = von.BName and v.NachBahnhof = nach.BName
```

Anfrage 2:

```
select v1.ZugNr, v2.ZugNr, v1.Abfahrt, v2.Ankunft
from verbindetTrans v1, verbindetTrans v2,
          Bahnhöfe von, Bahnhöfe nach
where von.SName = 'Passau' and nach.SName = 'Aachen'
 and  v1.VonBahnhof = von.BName and v2.NachBahnhof = nach.BName
 and  v1.NachBahnhof = v2.VonBahnhof and v1.Ankunft < v2.Abfahrt
 and  YEAR(v1.Ankunft) = YEAR(v2.Abfahrt)
 and  MONTH(v1.Ankunft) = MONTH(v2.Abfahrt)
 and  DAY(v1.Ankunft) = DAY(v2.Abfahrt) and v1.ZugNr <> v2.ZugNr
```

Anfrage 3:

Wir setzen die Anfrage aus den Teilergebnissen für keinmal, einmal, zweimal und dreimal umsteigen zusammen:

```
select v1.Abfahrt, v1.Ankunft
from verbindetTrans v1, Bahnhöfe von, Bahnhöfe nach
where von.SName = 'Passau' and nach.SName = 'Westerland'
 and  v1.VonBahnhof = von.BName and v1.NachBahnhof = nach.BName
union
select v1.Abfahrt, v2.Ankunft
from verbindetTrans v1, verbindetTrans v2,
          Bahnhöfe von, Bahnhöfe nach
where von.SName = 'Passau' and nach.SName = 'Westerland'
 and  v1.VonBahnhof = von.BName and v2.NachBahnhof = nach.BName
 and  v1.NachBahnhof = v2.VonBahnhof and v1.Ankunft < v2.Abfahrt
 and  v1.ZugNr <> v2.ZugNr
union
select v1.Abfahrt, v3.Ankunft
from verbindetTrans v1, verbindetTrans v2, verbindetTrans v3,
          Bahnhöfe von, Bahnhöfe nach
where von.SName = 'Passau' and nach.SName = 'Westerland'
 and  v1.VonBahnhof = von.BName and v3.NachBahnhof = nach.BName
 and  v1.NachBahnhof = v2.VonBahnhof and v1.Ankunft < v2.Abfahrt
 and  v1.ZugNr <> v2.ZugNr
 and  v2.NachBahnhof = v3.VonBahnhof and v2.Ankunft < v3.Abfahrt
 and  v2.ZugNr <> v3.ZugNr
union
select v1.Abfahrt, v4.Ankunft
from verbindetTrans v1, verbindetTrans v2, verbindetTrans v3,
          verbindetTrans v4, Bahnhöfe von, Bahnhöfe nach
where von.SName = 'Passau' and nach.SName = 'Westerland'
 and  v1.VonBahnhof = von.BName and v4.NachBahnhof = nach.BName
 and  v1.NachBahnhof = v2.VonBahnhof and v1.Ankunft < v2.Abfahrt
 and  v1.ZugNr <> v2.ZugNr
 and  v2.NachBahnhof = v3.VonBahnhof and v2.Ankunft < v3.Abfahrt
 and  v2.ZugNr <> v3.ZugNr
 and  v3.NachBahnhof = v4.VonBahnhof and v3.Ankunft < v4.Abfahrt
 and  v3.ZugNr <> v4.ZugNr
```

Aufgabe 2

Entsprechende Einträge müssen, soweit sie noch nicht vorhanden sind, in die Relation **hoeren** eingefügt werden.

```
insert into hoeren
  (select s.MatrNr, v.VorlNr
   from Studenten s, Vorlesungen v, Professoren p
   where p.Name = 'Sokrates' and p.PersNr = v.gelesenVon
    and (s.MatrNr, v.VorlNr) not in (select * from hoeren))
```

Aufgabe 3
Zu beachten ist, dass ein- und derselbe Student einen Professor nur einmal kennen kann. D.h. auch wenn er bei ihm eine Vorlesung hört und bei ihm geprüft wurde, kennt er ihn trotzdem nur einmal. Dies wird im Folgenden durch die Duplikatelimination bei **union** sichergestellt.

```
create view kenntprof as
  select h.Matrnr, v.gelesenVon as PersNr
  from   hoeren h, Vorlesungen v
  where  v.VorlNr = h.VorlNr
  union
  select p.MatrNr, p.PersNr
  from   pruefen p;

select k.PersNr, count(*) as grad
from   kenntprof k
group by PersNr;
```

Aufgabe 4
Annahme: Sokrates (PersNr = 2125) hat nur einen Assistenten dessen Fachgebiet noch nicht gesetzt ist (ist also null).

Wir betrachten nun folgende Anfrage:

select * **from** Vorlesungen **where** Titel not in
(**select** Fachgebiet **from** Assistenten **where** Boss = 2125);

Für jeden Titel t des äußeren **selects** führt das **not in** zu folgender Verknüpfung mit dem Ergebnis RI der geschachtelten Anfrage:

$$\text{Tupel mit Titel } t \text{ qualifiziert sich} \Leftrightarrow \bigwedge_{r \in RI} (r \neq t)$$

Die geschachtelte Anfrage berechnet als Ergebnis RI die Menge $\{[\text{null}]\}$. Es ergibt sich somit:

$$\text{Tupel mit Titel } t \text{ qualifiziert sich} \Leftrightarrow (\text{null} \neq t)$$

Das Prädikat auf der rechten Seite wird aber zu **unknown** evaluiert. Somit qualifiziert sich kein Tupel aus Vorlesungen und die Ergebnismenge ist leer.

Wir betrachten nun die Anfrage:

select * **from** Vorlesungen **where** not exists
(**select** * **from** Assistenten **where** Boss = 2125 **and** Fachgebiet = Titel);

Für jeden Titel t des äußeren **selects** wird die geschachtelte Anfrage ausgeführt und das letzte Prädikat im `where`-Teil der geschachtelten Anfrage wird zu:

$$\text{null} = t$$

Dies wird stets zu **unknown** evaluiert, womit der **where**-Teil lautet:

$$\text{Boss} = 2125 \wedge \textbf{unknown}$$

Dies bedeutet der **where**-Teil wird stets entweder zu **unknown** oder zu **false** ausgewertet und das Ergebnis der inneren Anfrage ist stets leer.
$\Rightarrow$ alle Vorlesungen qualifizieren sich in der kompletten Anfrage.