Lehrstuhl für Praktische Informatik III
Prof. Dr. Guido Moerkotte
Email: moer@db.informatik.uni-mannheim.de

Datenbanksysteme 1 — 5. Übungsblatt
Herbst-/Wintersemester 2016

Aufgabe 1

Gegeben seinen die beiden folgenden Relationenausprägungen L und R.
Geben Sie für folgende Teilaufgaben jeweils die Anfrage in SQL, als auch die Ergebnisausprägung an!

L		
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

R		
c_1	d_1	e_1
c_1	d_1	e_2
c_3	d_3	e_3
c_3	d_3	e_3

Lösung

```
CREATE TABLE l (a character(2), b character(2), c character(2))
CREATE TABLE r (c character(2), d character(2), e character(2))
insert into l values ('a1', 'b1', 'c1'), ('a2', 'b2', 'c2'), ('a3', 'b3', 'c3')
insert into r values ('c1', 'd1', 'e1'), ('c1', 'd1', 'e2'), ('c3', 'd3', 'e3'),
('c3', 'd3', 'e3')
```

Aufgabe 1 a)

$L \ltimes_{L.c=R.c} R$ ohne Duplikateliminierung.

Lösung

```
select l.*
from l inner join r on l.c=r.c
```

a_1	b_1	c_1
a_1	b_1	c_1
a_3	b_3	c_3
a_3	b_3	c_3

Aufgabe 1 b)

$L \ltimes_{L.c=R.c} R$ ohne Duplikate.

Lösung

```
select distinct l.*
from l inner join r on l.c=r.c
```

a_1	b_1	c_1
a_3	b_3	c_3

Aufgabe 1 c)

Alle Tupel aus L, zu denen ein Tupel in R existiert (Gleichheit auf Attribut c) ohne Duplikateliminierung.

Lösung

```
select l.*
from l
where exists (select * from r where l.c=r.c)
```

a_1	b_1	c_1
a_3	b_3	c_3

Aufgabe 1 d)

Alle Tupel aus L, zu denen ein Tupel in R existiert (Gleichheit auf Attribut c) ohne Duplikate.

Lösung

```
select distinct l.*
from l
where exists (select * from r where l.c=r.c)
```

a_1	b_1	c_1
a_3	b_3	c_3

Aufgabe 1 e)

$R \ltimes_{L.c=R.c} L$ ohne Duplikateliminierung.

Lösung

```
select r.*
from l inner join r on l.c=r.c
```

c_1	d_1	e_1
c_1	d_1	e_2
c_3	d_3	e_3
c_3	d_3	e_3

Aufgabe 1 f)

$R \ltimes_{L.c=R.c} L$ ohne Duplikate.

Lösung

```
select distinct r.*
from l inner join r on l.c=r.c
```

c_1	d_1	e_1
c_1	d_1	e_2
c_3	d_3	e_3

Aufgabe 1 g)

Alle Tupel aus R, zu denen ein Tupel in L existiert (Gleichheit auf Attribut c) ohne Duplikateliminierung.

Lösung

```
select r.*
from r
where exists (select * from l where l.c=r.c)
```

c_1	d_1	e_1
c_1	d_1	e_2
c_3	d_3	e_3
c_3	d_3	e_3

Aufgabe 1 h)

Alle Tupel aus R, zu denen ein Tupel in L existiert (Gleichheit auf Attribut c) ohne Duplikate.

Lösung

```
select distinct r.*
from r
where exists (select * from l where l.c=r.c)
```

c_1	d_1	e_1
c_1	d_1	e_2
c_3	d_3	e_3

Aufgabe 1 i)

Lässt sich das Ergebnis der Aufgabe c) und g) auch ohne `exists` reproduzieren?

Lösung

```
select l.*
from l inner join (select distinct c from r) as r on l.c = r.c

select *
from l
where c in (select c from r)

select r.*
from r inner join (select distinct c from l) as l on l.c = r.c

select *
from r
where c in (select c from l)
```

Aufgabe 2

Bei der Anfrageoptimierung spielt die Überführung von relationalen Ausdrücken in andere logisch äquivalente Ausdrücke eine entscheidende Rolle. Welche der folgenden algebraischen Äquivalenzen sind korrekt?

Aufgabe 2 a)

$A \bowtie B = B \bowtie A$

Lösung

Korrekt.

Aufgabe 2 b)

$A \ltimes B = B \ltimes A$

Lösung

Falsch.

Aufgabe 2 c)

$A \ltimes B = \Pi_A(A \bowtie B)$

Lösung

Korrekt.

Aufgabe 2 d)

$\Pi_x(A \bowtie B) = \Pi_x(A) \bowtie B$

Lösung

Falsch (evtl. kein Join mehr möglich).

Aufgabe 2 e)

$\sigma_{A.x=B.y}(A \times B) = A \bowtie_{A.x=B.y} B$

Lösung

Korrekt.

Aufgabe 2 f)

$A \bar{\ltimes} B = A \times B$

Lösung

Falsch.

Aufgabe 2 g)

$\Pi_x(\sigma_y(A \bowtie B)) = \sigma_y(\Pi_x(A \bowtie B))$

Lösung

Falsch (evtl. keine Selektion mehr möglich).

Aufgabe 3

Die folgenden SQL-Anfragen beziehen sich auf ein relationales Schema eines Streckennetzes einer Bahngesellschaft

- Zug(ZugNr, Zugtyp)
- Verbindung(ZugNr, Wochentag, StartBhf, ZielBhf)
- Teilstrecke(ZugNr, Wochentag, vonBhf, nachBhf, AbfahrtZeit, AnkunftZeit, Preis, Entfernung)

Drei der Anfragen liefern ein identisches Ergebnis, eine Anfrage liefert ein davon abweichendes Ergebnis. Welche Anfrage liefert das abweichende Ergebnis?

1. ```
   select distinct Zugtyp
   from   Zug
   where  ZugNr in (select ZugNr
                    from   Verbindung
                    where  StartBhf = 'Mannheim Hbf'
                    and    ZielBhf = 'Frankfurt Hbf');
   ```

2. ```
   select distinct Z.Zugtyp
   from   Zug Z, Verbindung V
   where  Z.ZugNr = V.ZugNr
   and    StartBhf = 'Mannheim Hbf'
   and    ZielBhf = 'Frankfurt Hbf';
   ```

3. ```
   select distinct Zugtyp
   from   Zug
   where  exists ((select ZugNr
                   from   Verbindung
                   where  StartBhf = 'Mannheim Hbf'
                   and    ZielBhf = 'Frankfurt Hbf')
                  intersect
                   (select ZugNr
                   from   Zug
                  ));
   ```

4. ```
   select Z.Zugtyp
   from   Zug Z
   where  exists (select *
                  from   Verbindung V
                  where  V.StartBhf = 'Mannheim Hbf'
                  and    V.ZielBhf = 'Frankfurt Hbf'
                  and    Z.ZugNr = V.ZugNr)
   group by Z.Zugtyp;
   ```

Lösung

3. liefert nicht das richtige Ergebnis, weil die beiden in `exists` geschachtelten Anfragen immer das selbe Ergebnis liefern, unabhängig von dem aktuellen Tupel der äußeren Anfrage. D.h. das Anfrageergebnis ist entweder immer leer, oder es enthaält immer alle Zugtypen unabhängig von der Strecke.

Aufgabe 4

Zum üben zu Hause.

Gegeben sei das Schema der Terra-Datenbank. Formulieren Sie die folgenden Anfragen in SQL. Testen Sie Ihre Anfragen in der SQL-Schnittstelle auf unserer Webseite.

Alternativ können Sie die Anfragen auch auf Ihrem Rechner in *SQLite* testen. Sie können *SQLite* hier herunterladen: https://sqlite.org. Auf der Lehrstuhlseite befindet sich eine Datei namens *terra2.db*, die mit dem Befehl `./sqlite3 terra2.db` (für unix-Systeme) geladen werden kann. Anschließend können Anfragen eingegeben werden. Weitere Informationen gibt es auf der SQLite-Webseite unter *Documentation*.

Aufgabe 4 a)

Bestimmen Sie alle direkten Nachbarländer, die von den 'USA' auf dem Landweg zu erreichen sind. Ein Nachbarland ist erreichbar, wenn es direkt zu den USA benachbart ist.

Lösung

```
SELECT l.name
FROM land l, land l2, benachbart b
where l2.name = 'USA' and
((l.l_id = b.l_id1 and b.l_id2 = l2.l_id) or
(l.l_id = b.l_id2 and b.l_id1 = l2.l_id))
```

```
select l.name
from land l
where l.l_id in
(select b.l_id1 from benachbart b, land l2 where b.l_id2=l2.l_id and l2.name='USA'
UNION
select b.l_id2 from benachbart b, land l2 where b.l_id1=l2.l_id and l2.name='USA')
```

Aufgabe 4 b)

Geben Sie für jede Organisation deren Namen sowie die Stadt und das Land an, in dem die Organisation ihren Sitz hat. Organisationen ohne Sitz müssen nicht in der Ergebnismenge auftauchen.

Lösung

```
select o.name, s.name, l.name
from organisation o, hat_sitz_in h, stadt s, land l, landesteil lt, gehoert_LT glt
where o.o_id = h.o_id and h.s_id = s.s_id and
l.l_id = lt.l_id and glt.lt_id = lt.lt_id and glt.s_id = s.s_id
```

Aufgabe 4 c)

Geben Sie für jedes Land dessen Namen, die Anzahl der Berge sowie die Höhe des höchsten Berges aus. Sortieren Sie die Ergebnisse so, dass das Land mit dem höchsten Berg als erstes in der Liste auftaucht.

Lösung

```
select l.name, count(*) as anzahlBerge, max(b.hoehe) as hoechsterBerg
from land l, landesteil lt, berg b, geo_berg gb
where lt.l_id=l.l_id and b.b_id=gb.b_id and gb.lt_id = lt.lt_id
group by l.name
order by max(b.hoehe) desc
```

Aufgabe 4 d)

Welche Hauptstädte liegen an einem Fluss? Geben Sie den Namen des Flusses sowie der Hauptstädte an. Die Ergebnisse sollen gruppiert nach Flussnamen ausgegeben werten.

Lösung

```
select g.name, s.name
from fluss f, gewaesser g, stadt s, liegt_an la, land l
where la.s_id = s.s_id and g.g_id = f.g_id and la.g_id = g.g_id
and s.s_id=l.hauptstadt
```

Aufgabe 4 e)

Bestimmen Sie die Namen aller Länder, in denen sich keine Wüsten befinden.

Lösung

```
select l.name
from land l
where not exists
(select 1
from Landesteil lt, geo_wueste g
where lt.L_ID = l.L_ID and g.LT_ID = lt.LT_ID)
```

Aufgabe 4 f)

Bestimmen Sie die Namen und die Mitgliederzahl aller Organisationen, die mehr als 20 Mitglieder haben. Die Resultate sollen absteigend nach der Mitgliederzahl sortiert sein.

Lösung

```
select o.name, count(*) as mitglieder
from organisation o, ist_Mitglied i
where i.o_id = o.o_id
group by o.name
having count(*) >20
order by count(*) desc
```

```
with anzahlMitglieder as (
select o.name, count(*) as mitglieder
from organisation o, ist_Mitglied i
where i.o_id = o.o_id
group by o.name
)
```

```
select * from anzahlMitglieder m
where m.mitglieder > 20
order by m.mitglieder desc
```

Aufgabe 4 g)

Bestimmen Sie den Namen und die Mitgliederzahl der Organisation mit den meisten Mitgliedstaaten.

Lösung

```
with anzahlMitglieder as (
select o.name, count(*) as mitglieder
from organisation o, ist_Mitglied i
where i.o_id = o.o_id
group by o.name
)


select * from anzahlMitglieder m
where m.mitglieder = (select max(mitglieder) from anzahlMitglieder)
```

Aufgabe 4 h)

Geben Sie die Namen aller Inselstaaten an und zählen Sie, aus wievielen Inseln diese Staaten jeweils bestehen. (Hinweis: Inselstaaten sind diejenigen Staaten, die an keinen anderen Staat grenzen.)

Lösung

```
select l.name, count(distinct g.i_id)
from   land l, landesteil lt, geo_insel g
where  l.l_id = lt.l_id
and    lt.lt_id = g.lt_id
and    l.l_id not in
   ((select l_id1
    from benachbart)
   union
    (select l_id2
     from benachbart))
group by l.l_id, l.name
order by l.name
```

```
select l.name, count(distinct g.i_id)
from   land l, landesteil lt, geo_insel g
where  l.l_id = lt.l_id
and    lt.lt_id = g.lt_id
and    not exists
   (select * from benachbart b
    where b.l_id1 = l.l_id or b.l_id2 = l.l_id)
group by l.l_id, l.name
order by l.name
```

Aufgabe 4 i)

Geben Sie eine Rangliste (d.h. durchnumeriert mit dem jeweiligen Rang) der Länder in Europa sortiert nach der Fläche an.

Lösung

```
with Europa(l_id, name, flaeche) as
(select l.l_id, l.name, l.flaeche
  from   land l, umfasst u, kontinent k
  where   l.l_id = u.l_id
  and    u.k_id = k.k_id
  and    k.name = 'Europa')
select count(*) as Rang, A.Name, A.Flaeche
from   Europa A, Europa B
where   (A.Flaeche < B.Flaeche
or       A.L_ID = B.L_ID)
group by A.L_ID, A.Name, A.Flaeche
order by Rang
```

Mit window function:

```
select l.name, rank() over (order by l.flaeche desc) as Rang
from   land l, umfasst u, kontinent k
where   l.l_id = u.l_id
and    u.k_id = k.k_id
and    k.name = 'Europa')
```