



5. ÜZ Musterlösung

Einführung in Datenbanksysteme
Datenbanken für die Bioinformatik

Heinz Schweppe, Jürgen Broß, Manuel Scholz

Übungsaufgaben

1. Aufgabe (SQL DML)

Gegeben sei das von Übungszettel 3 bereits bekannte BigL-Szenario. Die Datenbank enthalte eine Mitarbeitertabelle 'M' (PK: personalnummer, vorname, nachname, ...), eine Observationstabelle 'O' (PK: observationsID, stichwort, konfidenz), sowie eine Zuordnungstabelle 'B' (PK/FK: personalnummer, PK/FK: observationsID), die angibt auf welche Mitarbeiter sich eine Observation bezieht.

Formulieren Sie in SQL die folgenden Anfragen an die BigL-Datenbank:

- Finden Sie alle Mitarbeiter, die bereits abgehört wurden!

```
SELECT * FROM M m WHERE EXISTS (SELECT b.personalnummer FROM B b
WHERE b.personalnummer = m.personalnummer);
```
- Finden Sie alle Mitarbeiter, die noch nie abgehört wurden!

```
SELECT * FROM M m WHERE NOT EXISTS (SELECT b.personalnummer FROM B b
WHERE b.personalnummer = m.personalnummer);
```
- Finden Sie für alle Mitarbeiter die Häufigkeit mit der sie abgehört wurden! (Die Häufigkeit kann auch 0 betragen!)

```
SELECT m.personalnummer, count(*) AS anzahl FROM M m JOIN B b ON
(m.personalnummer = b.personalnummer) GROUP BY 1 UNION SELECT
m.personalnummer, 0 FROM M m WHERE NOT EXISTS (SELECT
b.personalnummer FROM B b WHERE b.personalnummer = m.personalnummer);
```
- Finden Sie alle Mitarbeiter bei denen das Stichwort „Pause“ mindestens zehnmal mit einer Konfidenz von mindestens 0,8 abgehört wurde und ordnen Sie diese Liste nach Häufigkeit des Auftretens des gegebenen Stichworts (häufigste zuerst)!

```
SELECT m.personalnummer, count(*) AS anzahl FROM M m JOIN B b ON
(m.personalnummer = b.personalnummer) JOIN O o ON (b.observationsID =
o.observationsID) WHERE o.stichwort = 'Pause' AND o.konfidenz >= 0.8
GROUP BY m.personalnummer HAVING count(*) >= 10 ORDER BY 2 DESC;
```
- Finden Sie alle Mitarbeiter, die mit mehr als einem anderen Mitarbeiter eine Liebesbeziehung haben! Mitarbeiter haben eine Liebesbeziehung miteinander, wenn in der Tabelle 'B' eine Observation mit den Stichwörtern „Techtelmachtel“ oder „Liebele“ existiert, die sich auf beide Mitarbeiter bezieht.

```
SELECT m.personalnummer, count(DISTINCT partner.personalnummer) AS
polygamie_count FROM M m JOIN B b ON (m.personalnummer =
b.personalnummer) JOIN O o ON (b.observationsID = o.observationsID)
JOIN B partner ON (partner.observationsID = b.observationsID AND
partner.personalnummer != b.personalnummer) WHERE o.stichwort =
'Techtelmachtel' OR o.stichwort = 'Liebele' GROUP BY
m.personalnummer HAVING count(DISTINCT partner.personalnummer) >= 2;
```

2. Aufgabe (SQL DML)

Gegeben sei die geographische Datenbank Terra-DB (Beschreibung siehe Zusatzblatt.) Formulieren Sie die folgenden Anfragen in SQL. Testen Sie ihre Anfragen auf der Terra-Datenbank. Accounts für die Nutzung der Datenbanken werden in den Tutorien vergeben. Technische Informationen zum Zugriff auf die Datenbank finden Sie unter dieser URL: <http://www.inf.fu-berlin.de/lehre/SS08/DBS-Intro/software.html>

- a) Finden Sie alle Landesteile, durch die der Mekong fließt. Geben Sie die Landesteile und das jeweilige Land namentlich an.

```
break on Land
select distinct l.name as Land, t.name as Landesteil
from terrauser.land l, terrauser.landesteil t, terrauser.geo_fluss f
where t.L_ID = f.L_ID
and t.LT_ID = f.LT_ID
and t.L_ID = l.L_ID
and f.fluss = 'Mekong'
order by l.name, t.name;
```

- b) Finden Sie alle Städte, die nicht an Meeren liegen. Geben Sie den Stadtnamen und den Landesnamen aus, geordnet nach Ländern.

```
1. Schritt: Alle Städte am Meer
SELECT DISTINCT c.*
FROM City c JOIN LOCATED_AT la ON la.city=c.name
WHERE la.sea IS NOT NULL
```

```
2. Schritt: Städte nicht am Meer
SELECT c.* FROM City c
EXCEPT
SELECT DISTINCT c.*
FROM City c JOIN LOCATED_AT la ON la.city=c.name
WHERE la.sea IS NOT NULL
```

```
3. Schritt: Land und Stadt nicht am Meer
SELECT c.name AS Stadt, cou.name AS Land
FROM Country cou JOIN (
SELECT c.* FROM City c
EXCEPT
SELECT DISTINCT c.*
FROM City c JOIN LOCATED_AT la ON la.city=c.name
WHERE la.sea IS NOT NULL
) c ON cou.c_id=c.c_id
ORDER BY Land
```

- c) Finden Sie alle Berge, die in Ländern mit dem Regierungssystem "Volksrepublik" liegen. Geben Sie jeweilig den Namen des Berges und den der Volksrepublik an.

```
select l.name as Land, b.berg as Berg
from terrauser.land l, terrauser.geo_berg b
where b.L_ID = l.L_ID and l.system = 'Volksrepublik';
```

- d) Finden Sie die Namen aller Berge, die in mehr als einem Land liegen. Geben Sie den Namen des Berges und die Namen der Länder an, aufsteigend geordnet nach Bergname.

```
select b.name, l.name
from Terrauser.Geo_Berg gb2, Terrauser.Geo_Berg gb1, Terrauser.Land
l, Terrauser.berg b
where gb1.L_ID=l.L_ID
and gb1.L_ID!=gb2.L_ID
and gb1.Berg=b.name
and gb2.Berg=b.name
```

```
order by b.name;
```

e) Finden sie alle Länder, die direkt an Schweden grenzen.

```
select l2.name
from Terrauser.Land l1,Terrauser.Land l2,Terrauser.benachbart b
where l1.name='Schweden'
and( ( b.Land1=l1.L_ID
and b.Land2=l2.L_ID)
or( b.Land2=l1.L_ID
and b.Land1=l2.L_ID)
);
```

f) Geben Sie alle Flüsse an, die in ein Meer münden. Das Meer soll dabei nur an eines der Landesteile grenzen durch das der Fluss fließt. Das Meer kann aber an andere Landesteile grenzen, durch das der Fluss nicht fließt.

```
select gf.fluss from terrauser.fluss f, terrauser.geo_fluss gf,
terrauser.geo_meer gm
where length(f.meer) > 0 and f.name = gf.fluss
and gf.L_ID = gm.L_ID and gf.LT_ID = gm.LT_ID
group by gf.fluss having count(gm.meer)=1;
```

g) Geben Sie alle Städte, die an mehreren Landesteilen liegen, zusammen mit den Landesteilen an.

```
select s1.name as Stadtname, t.name as Landesteil
from terrauser.stadt s1, terrauser.landesteil t
where s1.L_ID = t.L_ID
and exists (
select s2.name from terrauser.stadt s2
where s2.name = s1.name and s2.L_ID = s1.L_ID
and s1.LT_ID <> s2.LT_ID
) order by Stadtname;
```

h) Gesucht ist ein Meer (Name) mit der maximalen Anzahl der anliegenden Städte mit mehr als 1 Mio. Einwohner und mit der maximalen Anzahl der Flüsse, die in diesem Meer münden.

```
select meer.name
from meer
where
(select max(count(stadt.name))
from meer, stadt, liegt_an
where liegt_an.meer = meer.name
and liegt_an.stadt = stadt.name
and stadt.einwohner >= 1000000
group by meer.name)
=
(select count(stadt.name)
from stadt, liegt_an
where liegt_an.meer = meer.name
and liegt_an.stadt = stadt.name
and stadt.einwohner >= 1000000
group by meer.name)
and
(select max(count(fluss.name))
from meer, fluss, liegt_an
where liegt_an.meer = meer.name
and liegt_an.fluss = fluss.name
group by meer.name)
```

```
=  
(select count(fluss.name)  
  from fluss, liegt_an  
  where liegt_an.meer = meer.name  
    and liegt_an.fluss = fluss.name  
  group by meer.name)  
;
```

- i) In welche Länder (Ländernamen angeben) kann man laut unserer Datenbank von Deutschland (D) aus auf dem Landwege kommen,? Warum erschließen sich nicht alle Möglichkeiten?

Das Problem in dieser Aufgabe ist, dass in der Tabelle ‚BENACHBART‘ nur Paare der Form (A, B) vorkommen, nicht jedoch (B, A). Wäre das der Fall, dann könnten wir mit einer rekursiven Anfrage ausgehend von Deutschland alle erreichbaren Länder transitiv ermitteln.

Diese rekursiven Anfragen können jedoch nur in einer Richtung erfolgen, d.h. wenn wir das Paar (A, B) gefunden haben, dann wird in der nächsten Anfrage nach Paaren (B, ?) gesucht, also der vormalige Ergebniseintrag wird Anfrageeintrag des aktuellen Durchgangs.

Stellt man sich nun die Länder als Knoten in einem Graphen vor und die Nachbarschaftsrelation als Kanten, dann haben wir in unserer Tabelle nur gerichtete Kanten. Diese können wir nur in einer Richtung (je nachdem wie wir die rekursive Anfrage stellen) traversieren. Also werden wir wahrscheinlich nit alle Länder erreichen können.

Auch zwei rekursive Anfragen in beide Richtungen leisten nicht das gewünschte, da wir uns nach jedem Schritt in eine Richtung neu entscheiden können wollen, ob wir Kanten zu diesem Knoten vorwärts oder rückwärts durchlaufen.