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

Datenbanksysteme 1 — 6. Übungsblatt
Herbst-/Wintersemester 2016

Aufgabe 1

Gegeben sei das Schema der Terra-Datenbank. Formulieren Sie die folgende Anfrage in SQL.

Testen Sie bitte diese Anfrage *nicht* über die SQL-Schnittstelle auf unserer Webseite, da bei falscher Anfragenstellung die Ausführung der Anfrage nicht terminieren muss! Die Anfrage kann stattdessen lokal mit SQLite getestet werden.

Bestimmen Sie die direkten und indirekten Nachbarländer der Bundesrepublik Deutschland. Finden Sie einen Weg, die Rekursionstiefe zu beschränken.

Lösung

Wenn Rekursion mit union unterstützt wird (z.B. SQLite):

```
with benachbart2(l_id1, l_id2) as
(
select * from (
  select l_id1, l_id2 from benachbart
    union
  select l_id2, l_id1 from benachbart
        )
),
indirekteNachbarn(N1, N2) as
(
  select l_id1, l_id2 from benachbart2
  union
  select i.N1, b.l_id2 from
  indirekteNachbarn i, benachbart2 b
  where i.N2 = b.l_id1
        and i.N1 <> b.l_id2
)
select name from land where
l_id in
(select N1 from indirekteNachbarn where N2 in
        (select l_id from land where name = 'Bundesrepublik_Deutschland'));
```

Falls nur union all möglich ist, muss die Rekursion beschränkt werden (z.B. DB2):

```
with benachbart2(l_id1, l_id2) as
(
select * from (
  select l_id1, l_id2 from benachbart
```

```
    union
  select l_id2 , l_id1 from benachbart
        )
),
indirekteNachbarn(N1, N2, steps) as
(
  select l_id1 , l_id2 , 1 from benachbart2
  union all
  select i.N1, b.l_id2 , i.steps + 1 from
  indirekteNachbarn i, benachbart2 b
  where i.N2 = b.l_id1
        and i.N1 <> b.l_id2
        and i.steps <10
)
select count(*) from land where
(l_id in
(select N1 from indirekteNachbarn where N2 in
        (select l_id from land where name = 'Bundesrepublik_Deutschland')));
```

Aufgabe 2

Aufgabe 2 a)

Geben Sie für folgendes Datenbankschema die DDL-Anweisungen an.

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

Wählen Sie sinnvolle Datentypen für die Attribute aus. Definieren Sie auch Schlüssel- und Fremdschlüsselattribute.

Lösung

```
CREATE TABLE Zug (
     ZugNr AS INTEGER NOT NULL PRIMARY KEY,
     Zugtyp AS INTEGER NOT NULL)
)
CREATE TABLE Verbindung (
     ZugNr AS INTEGER NOT NULL REFERENCES Zug,
     Wochentag AS INTEGER NOT NULL check (Wochentag between 1 and 7),
     StartBhf AS VARCHARR(50) NOT NULL,
     ZielBhf AS VARCHAR(50) NOT NULL
     PRIMARY KEY(ZugNr,Wochentag)
```

```
)
```

Achtung: Züge mit existierenden Verbindungen dürfen nicht gelöscht werden.

```
CREATE TABLE Teilstrecke (
     ZugNr AS INTEGER NOT NULL,
     Wochentag AS INTEGER NOT NULL,
     vonBhf AS VARCHARR(50) NOT NULL,
     nachBhf AS VARCHAR(50) NOT NULL,
     AbfahrtZeit AS TIME NOT NULL,
     Ankunftzeit AS TIME NOT NULL,
     Preis AS DECIMAL(5,2) NOT NULL,
     Entfernung AS INTEGER NOT NULL,
     PRIMARY KEY(ZugNr,Wochentag,vonBhf),
     FOREIGN KEY(ZugNr, Wochentag) REFERENCES Verbindung ON DELETE CASCADE
)
```

Aufgabe 2 b)

Die Teilstrecke von "Niebüll" nach "Westerland" muss wegen des maroden Damms vom Festland nach Sylt stillgelegt werden. Es sollen alle Daten mit einer einzigen SQL-Anweisung gelöscht werden, die mit dieser Teilstrecke zusammenhängen. Welche Annahmen müssen Sie hierfür treffen?

Lösung

```
DELETE FROM Verbindung
WHERE ZugNr IN
     (SELECT ZugNr FROM Teilstrecke
     WHERE vonBhf = 'Niebuell' AND nachBhf = 'Westerland')
```

Wenn die referenzielle Integität wie in Teilaufgabe a) definiert ist, dann werden alle Teilstrecken mit gelöscht.

Aufgabe 3

Implementieren Sie mit einer Programmiersprache Ihrer Wahl die folgenden Algorithmen.

Aufgabe 3 a)

Berechnen Sie für eine Menge von Attributen und funktionalen Abhängigkeiten die Hülle der Attributmenge, d.h. die Menge aller Attribute, die mit Hilfe der funktionalen Abhängigkeiten aus der Eingabemenge hergeleitet werden können.

Lösung

Siehe Webseite.

Aufgabe 3 b)

Implementieren Sie ein Programm, das die kanonische Überdeckung aus einer Menge von funktionalen Abhängigkeiten berechnet. *Hinweis: Verwenden Sie ihr Programm aus Aufgabenteil a).*

Lösung

Siehe Webseite.

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 oder mit SQLite.

Aufgabe 4 a)

Bestimmen Sie die Namen aller Wüsten, die in ihrem Namen an irgendeiner Stelle den String "Gross" enthalten. Sortieren Sie die Liste der Namen alphabetisch absteigend.

Lösung

```
select name
from wueste
where name like '%Gross%'
order by name desc
```

Aufgabe 4 b)

Bestimmen Sie die Namen aller Wüsten, die in Australien liegen.

Lösung

```
select distinct w.name
from Wueste w, geo_wueste gw, Landesteil lt, Land l
where w.w_id = gw.w_id
  and gw.lt_id = lt. lt_id
  and lt. l_id  = l. l_id
  and l.name = 'Australien'
```

Aufgabe 4 c)

Welche ist die nördlichste Stadt? Geben Sie den Namen der Stadt aus, die die größte Breite hat.

Lösung

```
select s.name
from stadt s
where breite = (select max(breite) from Stadt)
```

Aufgabe 4 d)

Bestimmen Sie für jeden Kontinent den Flächenanteil von Wüsten. Geben Sie den Namen des Kontinents und den Flächenanteil in der Reihenfolge aus. Sortieren Sie die Kontinentnamen absteigend nach dem Flächenanteil. Gehen Sie zur Vereinfachung davon aus, dass Länder und Wüsten immer vollständig zu Kontinenten gehören, d.h. ignorieren Sie das Attribut `Prozent` in Relation `umfasst`. Kontinente ohne Wüsten sollen nicht berücksichtigt werden.

Lösung

```
with
Wueste2Kontinent as
(select distinct gw.w_id, u.k_id
 from umfasst u, Landesteil lt, geo_Wueste gw
 where u.l_id = lt.l_id
   and gw.lt_id = lt.lt_id ),
Kontinent2Wuestenflaeche (k_id, flaeche) as
(select wk.k_id, sum(w.flaeche)
 from Wueste w, Wueste2Kontinent wk
 where wk.w_id = w.w_id
 group by wk.k_id
 having sum(w.flaeche) > 0
)
select k.name, w.flaeche/k.flaeche as anteil
from Kontinent k, Kontinent2Wuestenflaeche w
where w.k_id = k.k_id and
      k.flaeche > 0
order by anteil desc
```

Aufgabe 4 e)

Geben Sie die Namen aller Meere aus, die weniger als 200 Meter tief sind. Geben Sie für jedes dieser Meere die Anzahl der Seen aus, die tiefer sind als das Meer (in dieser Reihenfolge). Sortieren Sie die Meere absteigend nach der Anzahl der Seen und bei gleicher Anzahl nach dem Namen.

Lösung

```
select g.name, count(distinct s.g_id) as Anzahl
from meer m, gewaesser g, see s
where m.g_id = g.g_id and m.tiefe < 200 and s.tiefe > m.tiefe
group by g.name
order by Anzahl desc, g.name asc
```

Aufgabe 4 f)

Finden Sie die Namen aller Städte, die laut Datenbank mehr Einwohner haben, als alle Städte in der `Volksrepublik_China`. Berücksichtigen Sie nur Städte, für die eine Einwohnerzahl bekannt ist.

Lösung

```
with StadtLand(sname,einwohner,lname) as
(select distinct s.name, s.einwohner, l.name
 from land l, landesteil lt, gehoert_lt glt,
      stadt s, umfasst u, kontinent k
 where l.l_id = lt.l_id
   and lt.lt_id = glt.lt_id
   and s.s_id = glt.s_id)

select distinct sl.sname
from StadtLand sl
where sl.einwohner > all
          (select sl1.einwohner
           from StadtLand sl1
           where sl1.lname = 'Volksrepublik_China'
             and sl1.einwohner > 0)

select name
from stadt
where einwohner >
(select max(s.einwohner)
 from stadt s, gehoert_lt glt, landesteil lt, land l
 where s.s_id = glt.s_id and glt.lt_id = lt.lt_id and
 lt.l_id = l.l_id and l.name = 'Volksrepublik_China')
```

Aufgabe 4 g)

Finden Sie die Namen aller Landesteile, die an mindestens einer Wüste, mindestens einer Insel und mindestens einem Gewässer liegen. Sortieren Sie die Namen alphabetisch aufsteigend.

Lösung

```
select name
from landesteil
where lt_id = any (select lt_id from geo_wueste)
  and lt_id = any (select lt_id from geo_gewaesser)
  and lt_id = any (select lt_id from geo_insel)
order by name
```

Alternativ mit `IN`:

```
select name
from landesteil
where lt_id in (select lt_id from geo_wueste)
  and lt_id in (select lt_id from geo_gewaesser)
  and lt_id in (select lt_id from geo_insel)
order by name
```

Alternativ:

```
select name
from landesteil lt
where exists (select 1 from geo_wueste gw where gw.lt_id = lt.lt_id)
  and lt_id in (select lt_id from geo_gewaesser)
  and lt_id = any (select lt_id from geo_insel)
order by name
```

Aufgabe 4 h)

Geben Sie die Namen aller Hauptstädte an, deren Länder höchstens in zweimal mehr Organisationen Mitglied sind, als die Hauptstadt Sitz einer Organisationen ist.

Lösung

```
with
StadtOrganisation as
(select s.s_id, count(o.name) as socount
 from stadt s, hat_sitz_in hsi, organisation o, land l
 where s.s_id = hsi.s_id
   and hsi.o_id = o.o_id
   and l.hauptstadt = s.s_id
 group by s.s_id),
LandMitglied as
(select l.l_id, count(o.name) as locount
 from land l, ist_mitglied m, organisation o
 where l.l_id = m.l_id
   and m.o_id = o.o_id
 group by l.l_id)
select s.name
```

```
from LandMitglied lm, StadtOrganisation so, land l, stadt s
where lm.l_id = l.l_id
  and l.hauptstadt = so.s_id
  and locount <= 2*socount
  and s.s_id = so.s_id
```

Aufgabe 4 i)

Erstellen Sie eine Liste der fünf größten Wüsten. Die Wüste mit der größten Fläche habe den kleinsten Rang (also 1). Geben Sie den Namen der Wüste und den Rang (in dieser Reihenfolge) an. Die Fläche der Wüste soll nicht mit ausgegeben werden. Wüsten ohne Flächenangaben sollen nicht mit in der Liste aufgeführt werden. Sortieren Sie die Wüsten aufsteigend nach dem Rang.

Lösung

```
select w1.name, count(*) as rang
from wueste w1, wueste w2
where (w1.flaeche < w2.flaeche or w1.w_id = w2.w_id)
  and w1.flaeche > 0
group by w1.name
having count(*) <= 5
order by rang
```

alternativ mit SQL-Funktion Rank

```
select t.name, t.rang
from (select name, rank() over (order by flaeche desc) as rang
      from wueste
      where flaeche > 0
      order by rang asc) as t
where t.rang <= 5
```