

Prof. Dr. Guido Moerkotte

Email: moer@db.informatik.uni-mannheim.de

Alexander Böhm

B6, 29, Raum C0.08

68131 Mannheim

Telefon: (0621) 181-2585

Email: alex@pi3.informatik.uni-mannheim.de

Datenbanksysteme 1

Frühjahrs-/Sommersemester 2008

4. Übungsblatt

02. April 2008

---

**Aufgabe 1**

---

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. Wählen sie die Datenbank *terra2* aus; das Schema ist auch dort zu finden.

**Aufgabe 1 a)**

Bestimmen Sie alle direkte 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
where l.lid in
(select b.lid1 from benachbart b, land l2 where b.lid2=l2.lid and l2.name='USA');
```

**Aufgabe 1 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.lid = lt.lid
and glt.lt_id = lt.lt_id and glt.s_id = s.s_id;
```

**Aufgabe 1 c)**

Geben Sie für jedes Land dessen Namens, 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;
```

-- ohne Duplikate (danke Daniel) --

```
with bergeProLand as
(
  select distinct l.name, b.b_id, b.hoehe
  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
)

select b.name, count(*) as anzahlBerge, max(b.hoehe) as hoehe
from bergeProLand b
group by b.name
order by hoehe desc;
```

Aufgabe 1 d)

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

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
group by g.name, s.name;
```

Aufgabe 1 e)

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

Lösung

```

select l.name
from land l, landesteil lt
where lt.lt_id = l.lt_id and not exists(select 1 from geo_wueste g where g.lt_id=lt.lt_id);

```

Aufgabe 1 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;

```

-----alternativ-----

```

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 1 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 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)

$$A \times B = B \times A$$

Lösung

Falsch.

Aufgabe 2 c)

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

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)

$$A \bowtie B = A \times B$$

Lösung

Falsch.

Aufgabe 2 g)

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

Lösung

Falsch (Selektion auf rechter Seite evtl. ohne Auswirkungen).

---

### Aufgabe 3

---

Eine Firma verwaltet ihre Mitarbeiter und Produktionsdaten mit Hilfe eines Datenbanksystems. Aus dem ER-Modell (siehe Übungsblatt 2) wurde das folgende, relationale Schema erstellt:

- Standort(Name, geleitetVon)
- Mitarbeiter(PersNr, Name, arbeitetAnStandort)
- Zeitarbeiter(PersNr, Firma)
- Festangestellter(PersNr, EinstellungsDatum)
- Maschine(InventarNr, Hersteller)
- Schicht(Tag, Startzeit, geleitetVon, Bericht)
- Charge(Tag, Startzeit, Nummer, Stück)
- arbeitetAnMaschine(PersNr, Tag, StartZeit, InventarNr)

Aufgabe 3 a)

Geben Sie die SQL-Anweisungen an mit denen die zugehörigen Relationen in einem Datenbanksystem erstellt werden (`create table...`) können. Achten Sie hierbei insbesondere darauf das die *referentielle Integrität* zwischen den einzelnen Relationen gewahrt bleibt. Diskutieren Sie eventuelle Umsetzungsalternativen.

Lösung

```

create table mitarbeiter(PersNr int primary key, Name varchar(50),
    arbeitetAnStandort varchar(50) references standort on delete set null);
create table festangestellter (PersNr int not null references mitarbeiter on delete cascade,
    Einstellungsdatum date not null);
create table zeitarbeiter(PersNr int not null references mitarbeiter on delete cascade,
    Firma varchar(50) not null);
create table standort(name varchar(50) primary key,
    geleitetVon int references mitarbeiter on delete set null);
create table maschine(InventarNr int primary key,
    Hersteller varchar(50) check (Hersteller in ('MAN','Thyssen','SAAB')));
create table schicht(tag date, Startzeit int check(Startzeit between 0 and 23),
    geleitetVon int references mitarbeiter on update set null,
    Bericht varchar(1000) not null, primary key(tag,startzeit));
create table charge(tag date, startzeit int, nummer int, stueck int,
foreign key (tag,startzeit) references schicht(tag, startzeit ),
primary key(tag,startzeit,nummer));
create table arbeitetAnMaschine(PersNr int references mitarbeiter on update set null,
    tag date, startzeit int check(Startzeit between 0 and 23),
    InventarNr int references maschine on update set null,
primary key (PersNr, tag, startzeit),
foreign key (tag,startzeit) references schicht(tag, startzeit ));

```

Zusätzliche Alternative: Einsatz von Triggern zur Sicherstellung von referentieller Integrität (Vorsicht im Umgang mit Triggern!).

### Aufgabe 3 b)

Inwieweit werden bei Ihrer Umsetzung die auf Blatt 2, Aufgabenteil 1b) besprochenen Probleme vermieden? Diskutieren Sie eventuelle Alternativen mit denen diese Probleme vermieden werden können.

### Lösung

Die Daten der Festangestellten und Zeitarbeiter sind immernoch auf zwei verschiedene Relationen verteilt. Die referentielle Integrität wird mit Hilfe von zugehörigen Einschränkungen sichergestellt, jedoch müssen Anfragen immernoch auf jeweils zwei Relationen zugreifen. Abhilfe können hier entsprechende Views schaffen (die allerdings nur die Abfrage, jedoch nicht das Einfügen neuer Daten erleichtern)

```

create view festangestellte_komplett as (
    select m.persnr, m.name, m.arbeitetAnStandort, f.einstellungsdatum
    from mitarbeiter m, festangestellter f
    where m.persnr = f.persnr
);
create view zeitarbeiter_komplett as (
    select m.persnr, m.name, m.arbeitetAnStandort, z.firma
    from mitarbeiter m, zeitarbeiter z
    where m.persnr = z.persnr

```

);

---

## Aufgabe 4

---

In der Vorlesung wurden mit Embedded SQL und Dynamic SQL zwei unterschiedliche Verfahren zur Einbettung von SQL Anweisungen in eine Host-Programmiersprache (wie Java oder C++) vorgestellt. Diskutieren Sie die Vor- und Nachteile der beiden unterschiedlichen Verfahren (z.B. im Hinblick auf Implementierungskomfort, Fehleranfälligkeit, Performance, Flexibilität, Sicherheit usw.).

### Lösung

#### **Vorteile Embedded SQL**

- Überprüfung der SQL Statements auf Korrektheit während der Übersetzung (Compilezeit) vermeidet Laufzeitfehler
- Übersetzung der Statements in Anfragepläne während der Compilezeit vermeidet Laufzeitoverhead (prepare)
- Programmiersprachenunabhängig

#### **Nachteile Embedded SQL**

- Erneute Übersetzung des Programmes notwendig bei Änderung des Datenbankschemas
- Evtl. zusätzlicher Präprozessor notwendig zur Übersetzung
- Keine/geringe Anpassungsmöglichkeiten von SQL-Anfragen zur Laufzeit

#### **Vorteile Dynamic SQL**

- Hohe Flexibilität, Konstruktion und Ausführung beliebiger Anfragen zur Laufzeit
- Kein Präprozessor benötigt

#### **Nachteile Dynamic SQL**

- Anfragen werden als Strings repräsentiert, keine Syntax-, Typ- oder Semantikprüfung zur Compilezeit  $\Rightarrow$  potentiell mehr Laufzeitfehler
- (Evtl. mehrfache) Erzeugung von Auswertungsplänen zur Laufzeit
- Sicherheitsprobleme bei schlechter Programmierung (SQL Injection usw)

---

## Freiwillige Zusatzaufgabe 5

---

Installieren Sie ein relationales Datenbankmanagementsystem Ihrer Wahl auf Ihrem Computer (z.B. PostgreSQL).

### Aufgabe 5 a)

Erzeugen Sie mit Hilfe der SQL-Anweisungen aus Aufgabenteil 3a) eine neue Datenbankinstanz. Inwieweit unterstützt das von Ihnen verwendete System Konstrukte zur Sicherstellung der referentiellen Integrität?