UNIVERSITÄT DES SAARLANDES
Fachrichtung 6.2 - Informatik
Lehrstuhl Prof. Weikum

Informationssysteme SS 2002

Übung 6 Beispiellösung

Aufgabe 1: DB-Entwurf mit ODL und Anfragen in OQL

a) Entwerfen Sie zur – bisher relational spezifizierten – Musikdatenbank aus Übung 4 ein entsprechendes ODL-Schema unter Ausnutzung der „relationship“-Klausel.

Disk	(DiskID, DiskTitel, Preis)
Musikstück	(DiskID, StückID, Titel, Länge)
Person	(PID, Name, Nationalität)
Interpret	(PID, DiskID, StückID, Funktion, Instrument)
Autor	(PID, DiskID, StückID, Tätigkeit)

```
class Disk {
   extent Disks;
   attribute String DiskTitel;
   attribute Float Preis;
   relationship list<Musikstück> hatStücke inverse Musikstück::aufDisk;
}

class Musikstück {
   extent Musikstücke;
   relationship Disk aufDisk inverse Disk::hatStücke;
   relationship set<Interpret> hatInterpreten inverse Interpret::musikstück;
   relationship set<Interpret> hatAutoren inverse Autor::musikstück;
   attribute String Titel;
   attribute Integer Länge;
}

class Person {
   extent Personen;
   attribute String Name;
   attribute String Nationalität;
   relationship set<Interpret> interpretIn inverse Interpret::person;
   relationship set<Autor> autorVon inverse Autor::person;
}

class Interpret {
   extent Interpreten;
   relationship Person person inverse Person::interpretIn;
   relationship Musikstück musikstück inverse Musikstück::hatInterpreten;
   attribute String Funktion;
   attribute String Instrument;
}

class Autor {
   extent Autoren;
   relationship Person person inverse Person::autorVon;
   relationship Musikstück musikstück inverse Musikstück::hatAutoren;
   attribute String Tätigkeit;
}
```

b) Erweitern Sie das ODL-Schema von a) um:

- Attribute zur Speicherung von Musikaufnahmen als digitales Audio (z.B. im MP3-Format), sowie zur Speicherung von Fotos von Musikern (z.B. im JPEG-Format), wobei die Audios und Fotos selbst einfach vom Typ Binary sein sollen (dies entspricht BLOB in SQL),

```
class Musikstück {
  extent Musikstücke;
  relationship Disk aufDisk inverse Disk::hatStücke;
  relationship set<Interpret> hatInterpreten inverse Interpret::musikstück;
  relationship set<Interpret> hatAutoren inverse Autor::musikstück;
  attribute String Titel;
  attribute Integer Länge;
  relationship Audio audio inverse Audio::zuHören;
}

class Audio {
  extent Audios;
  attribute Binary audio;
  relationship Musikstück zuHören inverse Musikstück::audio;
}

class Person {
  extent Personen;
  attribute String Name;
  attribute String Natinalität;
  relationship Bild bild inverse Bild::zeigtPerson;
}

class Bild {
  extent Bilder;
  attribute Binary bild;
  relationship Person zeigtPerson inverse Person::bild;
}
```

- Methoden für das Ermitteln
  - der Gesamtdauer einer CD
  - der verschiedenen Nationen, aus denen die Interpreten eines Musikstücks kommen, sowie
  - der verschiedenen Nationen, aus denen die Interpreten einer Disk kommen.

```
class Disk {
  ...
  Real Gesamtlänge();
  Set<String> verschiedeneNationen();
}

class Musikstück {
  ...
  set<String> verschiedeneNationen();
}

class Disk::Gesamtlänge {
  Integer l = 0;
  list<Ref<Musikstück>> DiskStücke = this->hatStücke;
  Iterator<Ref<Musikstück>> it = DiskStücke->create_iterator();
  Ref<Musikstück> m;
  while (m = it.next())
  {
    l += m->Länge;
  }
  return l;
}
```

```
class Musikstück::verschiedeneNationen {
   set<String> nationen;
   list<Ref<Interpret>> interpreten = this->hatInterpreten;
   Iterator<Ref<Interpret>> it_i = interpreten->create_iterator();
   Ref<Interpret> I;
   while (i = it_i.next())
   {
     nationen->insert_element(i->person->Nationalität);
   }
   return nationen;
}

class Disk::verschiedeneNationen {
   set<String> nationen;
   list<Ref<Musikstück>> diskStücke = this->hatStücke;
   Iterator<Ref<Musikstück>> it_m = diskStücke->create_iterator();
   Ref<Musikstück> m;
   While (m = it_m.next())
   {
     nationen->union(m->verschiedeneNationen());
   }
   return nationen;
}
```

c) Formulieren Sie die folgenden Anfragen in OQL:

- Welche Stücke hat F. Chopin komponiert?

```
SELECT   DISTINCT (M.Titel)
FROM     Musikstücke M
WHERE    (EXISTS A IN M.hatAutoren :  A.person.Name= "F.Chopin" AND
                                      A.Tätigkeit = "Komponist")
```

alternativ:

```
SELECT   DISTINCT (A.musikstück.Titel)
FROM     Autoren A
WHERE    A.person.Name="F.Chopin"
AND      A.Tätigkeit="Komponist"
```

aber <u>nicht</u>:

```
SELECT   DISTINCT (M.Titel)
FROM     Musikstücke M
WHERE    M.hatAutoren.person.Name="F.Chopin"
AND      M.hatAutoren.Tätigkeit="Komponist"
```

- Welche Disks enthalten kein Stück, das länger als 60 (Sekunden) dauert?

```
SELECT   DISTINCT (D.DiskTitel)
FROM     Disks D
WHERE    (FOR ALL M IN D.hatStücke: M.Länge ≤ 60)
```

- Welchen Durchschnittspreis haben Disks, auf denen Interpreten aus über drei Nationen zu hören sind und mindestens ein Interpret aus Deutschland kommt?

```
SELECT   AVG(D.Preis)
FROM     Disks D
WHERE    D.verschiedeneNationen()->cardinality ≥ 3
```

AND D.verschiedeneNationen()->contains_element(„Deutschland“)

- Auf welchen Disks sind Trompeten zu hören?

```
SELECT  I.musikstück.aufDisk.diskTitel
FROM    Interpreten I
WHERE   I.Instrument=“Trompete“
```

- Auf welchen Disks gibt es ein Musikstück mit einer Dauer von weniger als 10 (Sekunden)?

```
SELECT  D.DiskTitel
FROM    Disks D
WHERE   (EXISTS M IN D.musikstücke: M.Länge < 10)
```

oder:

```
SELECT  D.DiskTitel
FROM    Disks D
WHERE   D.musikstücke.Länge > 10
```

- Wieviele verschiedene Nationalitäten haben im Durchschnitt die Interpreten einer Disk?

```
SELECT  AVG(L.cardinality)
FROM    (SELECT D.verschiedeneNationen() AS L)
```

Aufgabe 2: OQL

Gegeben ist das folgende ODL-Schema einer extrem vereinfachten Klimadatenbank.

```
class Land {
   extent Laender;
   key Landesbez;
   attribute String Landesbez;
   relationship Set<Stadt> HatStaedte inverse Stadt::GehoertZuL;
   relationship Set<See> HatSeen inverse See::GehoertZuL;
}

class Stadt {
   extent Staedte;
   key Stadtname;
   attribute String Stadtname;
   attribute Integer Einwohnerzahl;
   relationship Land GehoertZuL inverse Land::HatStaedte;
   relationship Set<Fluss> LiegtAnF inverse Fluss::FliesstDurchS;
   attribute Zeitreihe Temperatur;
   attribute Zeitreihe Niederschlag;
}

class Fluss {
   extent Fluesse;
   key Flussbez;
   attribute String Flussbez;
   relationship Set<Stadt> FliesstDurchS inverse Stadt::LiegtAnF;
   relationship Set<See> HatSeen inverse See::GehoertZuF;
}

class See {
   extent See;
   key Seebez;
   attribute String Seebez;
   relationship Fluss GehoertZuF inverse Fluss::HatSeen;
   relationship Set<Land> GehoertZuL inverse Land::HatSeen;
```

```
  attribute Zeitreihe Pegelstand;
}

class Zeitreihe {
  attribute Array<Struct<Tagesdaten: Date; Messwerte: Real>>;
  Real Messwert (in Date);
  Real Durchschnittswert (in Date, in Date);
          // Durchschnitt von Datum1 bis Datum2
  Real Gleitdurchschnittswert (in Integer);
          // Gleitender Durchschnittswert für N aufeinander
          // folgende Tage
}
```

Formulieren Sie die folgenden Anfragen in OQL:

a) Welche Seen wären von einer Wasserverschmutzung in einer Schweizer Stadt betroffen?

```
SELECT   DISTINCT (FLATTEN(L.HatStaedte.LiegtAnF.HatSeen))
FROM     Laender L
WHERE    L.Landesbez="Schweiz"
```

Bemerkung zu FLATTEN:
L.HatStaedte liefert eine Menge (Menge='set' ist Subklasse von ‚collection') von Städten, L.HatStaedte.LiegtAnF liefert dann eine Menge von Mengen von Flüssen und L.HatStaedte.LiegtAnF.HatSeen liefert schließlich eine Menge von Mengen von Mengen von Seen. Wir nehmen zur Vereinfachung an, daß die Funktion FLATTEN beliebig tief verschachtelte Collections von Collections von ... von Collections mit Instanzen vom Typ T in eine (flache) Collection mit Instanzen vom Typ T konvertieren kann. Laut ODGM 2.0 ist die Funktion nur auf eine Collection einer Collection vom Typ T anwendbar, so daß FLATTEN demnach eigentlich mehrfach angewendet werden müßte (siehe R.G.G. Cattel et al.: The Object Database Standard: ODMG 2.0, p 110f).

b) Welche Seen wären von einer Wasserverschmutzung in einer Schweizer Großstadt (d.h. mit mehr als 100.000 Einwohnern) betroffen?

```
SELECT DISTINCT(FLATTEN(S.LiegtAnF.HatSeen))
FROM   Staedte S
WHERE  S.GehoertZuL.Landesbez="Schweiz"
AND    S.Einwohnerzahl > 100.000
```

c) Wie hoch waren am 11. November 1999 die Pegelstände der Seen in Bangladesh?

```
SELECT S.Pegelstand.Messwert(11.Nov 1999)
FROM   Seen S
WHERE  S.GehoertZuL.Landesbez="Bangladesh"
```

oder

```
SELECT S.Pegelstand.Messwert(11.Nov 1999)
FROM   Seen S
WHERE  FLATTEN(S.GehoertZuL.Landesbez)->contains_element(„Bangladesh")
```

oder

```
SELECT S.Pegelstand.Messwert(11.Nov 1999)
FROM   Seen S
WHERE  (EXISTS L IN S.GehoertZuL: L.Landesbez="Bangladesh")
```

d) In welchen anderen Ländern könnte sich ein starker Regen in Bangladesh potentiell auswirken?

```
SELECT DISTINCT (L.Landesbez)
FROM   Laender L
WHERE  L.HatStaedte.LiegtAnF.FliesstDurchS.GehoertZuL.Landesbez="Bangladesh"
OR     L.HatSeen.GehoertZuL.Landesbez="Bangladesh"
```

Alternativen gemäß c)

e) Wie hoch ist der 7-Tages-Durchschnitt der Temperatur in den am Strom Sambesi liegenden Städte?

```
SELECT S.Stadtname, S.Temperatur.Gleitdurchschnittswert(7)
FROM   Staedte S
WHERE  S.LiegtAnF.Flussbez="Sambesi"
```

Alternativen gemäß c)

f) Wie hoch war am 15. November 1999 die Durchschnittstemperatur der Großstädte (d.h. mit mehr als 100.000 Einwohnern) in China?

```
SELECT   AVG(S.Temperatur.Messwert(15.Nov 1999))
FROM     Staedte S
WHERE    S.Einwohnerzahl > 100.000
AND      S.GehoertZuL.Landesbez="China"
```