Object Database Standards, Languages, and Design
Part II: Object Query language (OQL)
Douglas S. Kerr
CIS 671
Elmasri & Navathe Ch 12
Corrected 4/24/01

Object Query Language (OQL)
• “Bring the best of SQL to the object world.”
• Syntax similar to SQL
• Plus additional features
• Works with programming languages where
  ODMG has defined bindings
  – Java, C++, Smalltalk
  – Returns object matching type system of that language
  – May implement class operations in these languages
• SQL3 – “Bring the best of the object-oriented world to the relational world.”

Movies Example: Movies, Stars, Studios

Class Star
(extent Stars key name)
{
attribute string name;
attribute struct Addr;
{string street, string city} address;
relationship set<Movie> starredIn
inverse Movie :: stars;
};

Class Studio
(extent Studios key name)
{
attribute string name;
relationship set<Movie> owns
inverse Movie :: ownedBy;
};

Movies Example, contd.

Class Movie
(extent Movies key (title, year))
{
attribute string title;
attribute integer year;
attribute integer length; /* in minutes */
attribute enumeration (color, blackAndWhite) filmType;
relationship set<Star> stars
inverse Star :: starredIn;
relationship Studio ownedBy
inverse Studio :: owns;
float lengthInHours() raises(noLengthFound);
starNames(out set<Star>);
otherMovies(in Star, out set<Movie>) raises(noSuchStar);
};

Specifying data from ODB
• Path expression
  – Dot notation similar to structure in programming language,
  – e.g., o.a attribute a of object o.
• Select-From-Where expression
  – Similar to SQL.
  – References are to data classes, not relations.
• Integrated directly into host language.
Path Expressions
Assume myMovie a host-language variable, value a Movie object.

- myMovie.length - length of the movie.
- myMovie.lengthInHours() - real number, computed as length in minutes.
- myMovie.stars – set of Star objects related to myMovie by relationship stars.
- myMovie.starNames(myStars) – returns no value, sets output variable myStars of method starNames to set of strings with names of the stars of the movie.
- myMovie.ownedBy.name – string that is name of studio owning myMovie.

Select-From-Where Expressions, I
- Year of the movie Gone with the Wind.
  - select m.year
    - from m in Movie
      - where m.title = "Gone with the Wind";
- Bag containing names of stars of Casablanca.
  - select s.name
    - from m in Movies, s in m.stars
      - where m.title = "Casablanca";
- Set containing names of stars of Disney movies.
  - select distinct s.name
    - from m in Movies, s in m.stars
      - where m.ownedBy.name = "Disney";

Select-From-Where Expressions, II
- Pairs of stars living at the same address.
  - select distinct struct(s1, s2)
    - from s1 in Stars, s2 in Stars
      - where s1.addr = s2.addr and s1.name < s2.name;
- Set containing names of stars of Disney movies, using subquery.
  - Select distinct s.name
    - from m in Movies
      - where m.ownedBy.name = "Disney" and s in dm.stars
        - Bag of all Disney movies.
      - Stars in those movies.
    - from s in Stars
      - where m.ownedBy.name = "Disney";
    - Select s.name
      - from s in Stars
      - where m in Movies
        - order by m.length, m.title;

Select-From-Where Expressions, III
- Set containing names of stars of Disney movies, using existential quantifier.
  - select distinct s.name
    - from s in Stars
    - where exists m in s.starredIn : m.ownedBy.name = "Disney";
- Set containing names of stars that have appeared only in Disney movies, using universal quantifier.
  - select distinct s.name
    - from s in Stars
    - where for all m in s.starredIn : m.ownedBy.name = "Disney";

Select-From-Where Expressions, IV
- Average length of all movies.
  - W a n t b a g of lengths, not set.
    - Avg(select m.length from Movies m);
- Table of lengths of movies for each studio for each year.
  - select std, yr, sum(length) from select p.m.length
    - from p in partition
      - from m in Movies
      - group by std: m.studio, yr: m.year;
- Table of lengths of movies for each studio for each year where studio produced at least one movie of over 120 minutes in that year.
  - select std, yr, sum(length) from select p.m.length
    - from p in partition
      - from m in Movies
      - group by std: m.studio, yr: m.year
        - having max(select p.m.length from partition p) > 120;

Views
- View returning Set containing movies by studio studio.
  - define moviesFromStudio(studio) as
    - select m
      - from m in Movies
      - where m.ownedBy.name = studio;
  - Do not need "distinct m", since Movies is already a set.
Returning a *single* element

- **Bag** or **Set** (if distinct included) normally returned.
- May want *single* element.
- Return *Movie* with name *Casablanca*.

```
Element (select m
    from m in Movies
    where m.title = "Casablanca");
```

---

Find *names* of those who *starred* in all *Disney* movies:

**First attempt.**

1. Using “if p then q”:
   
   ```
   select s.name
   from s in Star
   where for all m in Movie :
      (if m.ownedBy.name = 'Disney'
      then exists s1 in m.stars :
         s1.name = s.name);
   ```

2. Converting “if p then q” to “¬p or q”:
   
   ```
   select s.name
   from s in Star
   where for all m in Movie :
      (m.ownedBy.name ≠ 'Disney'
      or
      exists s1 in m.stars :
         s1.name = s.name);
   ```

---

Find *names* of those who *starred* in all *Disney* movies:

**Using a view of all *Disney* movies.**

**View** returning **Set** of *Disney* movies.

```sql
define DisneyMovies as
    select m
      from m in Movies
      where m.ownedBy.name = 'Disney';
```

3. Query for stars using the DisneyMovies view.
   ```
   select s.name
   from s in Star
   where for all dm in DisneyMovie :
      (exists s1 in dm.stars :
         s1.name = s.name);
   ```

---

Find *names* of those who *starred* in all *Disney* movies:

**Using the Movie.starNames function.**

**Change**

- starNames(out set<String>) to a function
- set<String> starNames()

4. Query for stars using the DisneyMovies view and starNames() function.
   ```
   select s.name
   from s in Star
   where for all dm in DisneyMovie :
      s.name in dm.starNames();
   ```