Object Databases

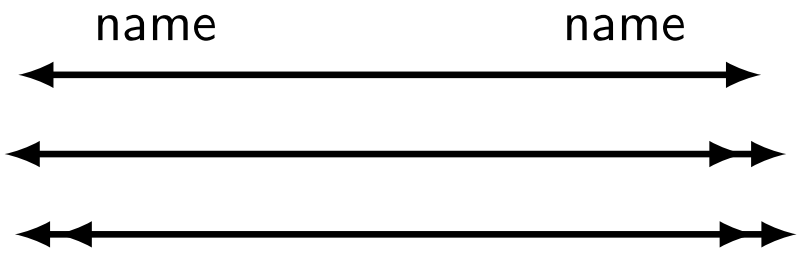
INFO-H-415

Université Libre de Bruxelles

April 13, 2011

ODMG Schemas Notations

- Classes and Inheritance
  - Similar to UML
- Relations
  - 1—1
  - 1—N
  - M—N



ODL Class Definition

- All object-object relations are binary
- M — N relations are modelled by **collections**:
  - Unordered:
    - Bag
    - Set (Without duplicate)
  - Ordered:
    - List (Array)

ODL Class Definition

```
class Person (extent Persons){
    attribute string name;
    attribute struct Address{
        unsigned short number,
        string street
    } address;
    relationship Person spouse
        inverse Person::spouse;
    relationship set<Person> children
        inverse Person::parents;
    relationship list <Person> parents
        inverse Person::children;
}
```

OQL Queries

- Select the employees named *Pat*.

```
select p
from p in Persons
where p.name = "Pat"
```

- Essentially: SQL with the object dot notation
- Select objects from the **extent** of a class (Persons) not from the class itself (Person).
- The return type of this query is bag<Person>

Selecting literals

- Select the name of the employees.

```
select p.name
from p in Persons
```

- The return type of this query is bag<string>

Distinct and Set

- Select the name of the employees (distinct).

```
select distinct p.name
from p in Persons
```

- The return type of this query is `set<string>`

Struct

- Select the name and address of the employees.

```
select p.name, livesin: p.address
from p in Persons
```

- Alternatively:

```
select struct(
    name: p.name,
    livesin: p.address)
from p in Persons
```

- The return type of these queries is
  `bag<struct(address: string, livesin: Address)>`
- Works with distinct too!

Joins

- Pair employees with their children.

```
select p.name, c.name
from p in Persons,
     c in p.children
```

- Automatically performs the appropriate joins (i.e. return one parent with one of his/her children at a time)

Navigation

- For 1–1 relations, the `.` operator resolves the appropriate object.
  - `s.spouse.address.city.name`
- This does **not** work with multivalued attributes and relations.
- For m–n relations, use joins and/or subqueries.

Functions

- distinct(1,2,2,1) returns (1,2)
- count(1,2,3,4) returns 4
- not(true) returns false
- max, sum
- flatten((1,2,3), (1,2)) returns (1,2,3,1,2)

Group By

- Classify the employees according to their earnings.

```
select *
from Employees e
group by low : salary < 1000,
         medium : salary >= 1000 and
                  salary < 10000,
         high : salary >= 10000
```

- The return type of this query is

```
bag<struct(
     low:boolean,
     medium:boolean,
     high:boolean,
     struct(e: Employee))>}
```

Group By

- Select the distinct salaries, along with the name of the employees who earn that salary.

```
select struct(
    salary: sal,
    names: (select p.e.name
            from partition p)
)
from e in Persons
group by sal: salary
```