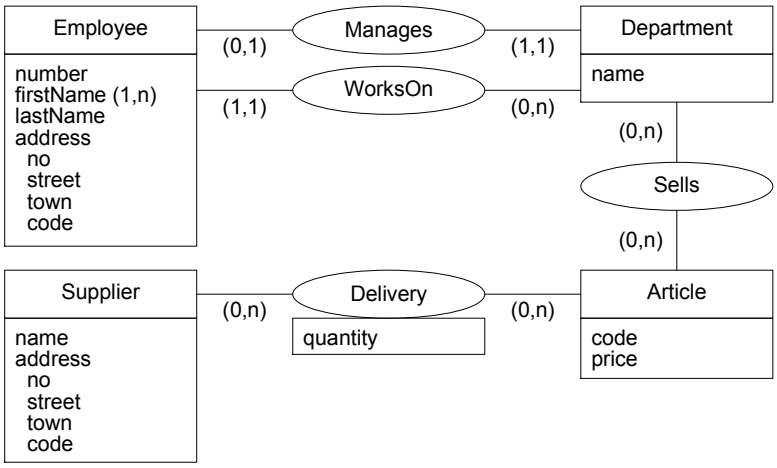
Object Databases
Exercice

Application Modeling

Give an ODMG schema corresponding to the ER schema below. Give a graphical and a textual representation



OQL Queries

Consider the following classes

```
class Address
(extent Addresses) {
    attribute integer number;
    attribute string street;
}

class Person
(extent Persons) {
    attribute string name;
    attribute date birthdate;
    relationship Person spouse inverse Person::spouse;
    relationship list<Person> children inverse Person::parents;
    relationship list<Person> parents inverse Person::children;
    relationship Apartment livesIn inverse Apartment::isUsedBy;
}

class Employee extends Person
(extent Employees) {
    attribute real salary;
}

class Building
(extent Buildings) {
    attribute Address address;
    relationship list(Apartments) apartments inverse Apartment::building;
```

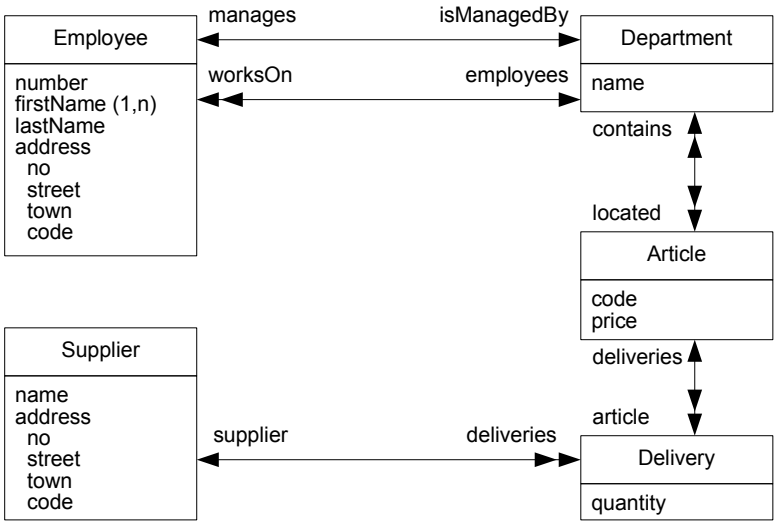
```
}

class Apartment
(extent Apartments) {
    attribute integer number;
    attribute integer price;
    relationship Building building inverse Building::apartments;
    relationship set<Person> isUsedBy inverse Person::livesIn;
}
```

Write in OQL the following queries.

(1) The name of persons
(2) The year of birth of persons
(3) The addresses of children
(4) The name of persons born after 1966
(5) The name and address of children without duplicates
(6) The name and address of children that has one parent living in "Rue Saint- Honoré"
(7) The name and address of children who do not live in the same apartment of one of their parents
(8) The name and address of children who do not live in the same apartment of both of their parents
(9) The name and salary of employees who earn at least 10 times the cost of the apartment where they live
(10) Sort the apartments according to their number of inhabitants
(11) The buildings with the number of inhabitants that live in
(12) The street of buildings with the number of inhabitants that live in
(13) Partition the persons in 3 sets: a set of persons born after 1980, a set of persons born between 1970 and 1980, and a set of persons born before 1970
(14) Display the names of members of the 3 preceding sets
(15) Display the number of persons contained in each of the 3 preceding sets

Object Databases
Answers

Application Modeling



The relationship Delivery is translated into a class due to the attribute quantity.

```
Struct Address {
    string no;
    string street;
    string town;
    string code;
}

Class Employee
(extent employees) {
    attribute string number;
    attribute string lastName;
    attribute list<string> firstName;
    attribute Address address;
    relationship Department worksOn inverse Department::employees;
    relationship Department manages inverse Department::isManagedBy;
}

Class Department
(extent departments) {
    attribute string name;
    relationship set<Employees> employees inverse Employee::worksOn;
    relationship Employee isManagedBy inverse Employee::manages;
    relationship set<Article> contains inverse Article::located;
}

Class Article
(extent articles) {
    attribute string code;
    attribute real price;
```

```
    relationship set<Delivery> deliveries inverse Delivery::articles;
    relationship set<Department> located inverse Department::contains;
}

Class Delivery
(extent deliveries) {
    attribute integer quantity;
    relationship article inverse Article::deliveries;
    relationship supplier inverse Supplier::deliveries;
}

Class Supplier
(extent suppliers) {
    attribute string name;
    attribute Address address;
    relationship set<Delivery> deliveries inverse Delivery::supplier;
}
```

OQL Queries

(1) The name of persons

```
select e.name from e in Persons
```

(2) The year of birth of persons

```
select e.birthDate.year from e in Persons
```

(3) The addresses of children

```
select c.livesIn.building.address from p in Persons, c in p.children
```

Another answer

```
select p.livesIn.building.address from p in Persons where not(empty(p.parents))
```

(4) The name of persons born after 1966

```
select e.name from e in Persons where e.birthDate.year > 1966
```

(5) The name and address of children without duplicates

```
select distinct struct(name: c.name, address: c.livesIn.building.address)
from p in Persons, c in p.children
```

(6) The name and address of children that has one parent living in "Rue Saint-Honoré"

```
select distinct struct(name: c.name, address: c.livesIn.building.address)
from p in Persons, c in p.children
where p.livesIn.building.address.street = "Rue Saint-Honoré"
```

(7) The name and address of children who do not live in the same apartment of one of their parents

```
select distinct struct(name: c.name, address: c.livesIn.building.address)
from p in Persons, c in p.children
where p.livesIn != c.livesIn
```

(8) The name and address of children who do not live in the same apartment of both of their parents

```
select distinct struct(name: c.name, address: c.livesIn.building.address)
from p1 in Persons, c in p1.children, p2 in c.parents
where p1 != p2 and p1.livesIn != c.livesIn and p2.livesIn != c.livesIn
```

Another answer

```
select distinct struct(name: c.name, address: c.livesIn.building.address)
from c in Persons
where not ( c.livesIn in (select p.livesIn from p in c.parents) )
```

(9) The name and salary of employees who earn at least 10 times the cost of the apartment where they live

```
select struct(name: e.name, salary:e.salary)
from e in Employees
where e.salary >= (10*e.livesIn.price)
```

(10) Sort the apartments according to their number of inhabitants

```
select a from a in Apartments
order by count(a.isUsedBy)
```

(11) The buildings with the number of inhabitants that live in

```
select struct( building: b, nbInhabitants:
     sum ( select count(a.isUsedBy) from a in b.Apartments )
from b in Buildings
```

Another answer if the schema is changed and a person may live in several appartments

```
select struct( building: b, nbInhabitants:
     count ( distinct ( flatten (
          select a.isUsedBy from a in b.Apartments ) ) ) )
from b in Buildings
```

(12) The street of buildings with the number of inhabitants that live in

```
select struct( street: partition.address.street, nbInhabitants:
     ( select sum ( count ( a.isUsedBy ) )
       from b in partition, a in b.Apartments ) )
from b in Buildings
group by street: b.address.street
```

(13) Partition the persons in 3 sets: a set of persons born after 1980, a set of persons born between 1970 and 1980, and a set of persons born before 1970

```
select *
from p in Persons
group by young: p.birthDate.year > 80,
         middle: p.birthDate.year <= 80 and p.birthDate.year > 70,
         old: p.birthDate.year <= 70
```

(14) Display the names of members of the 3 preceding sets

```
select name: (select x.name from x in partition)
from p in Persons
group by  young: p.birthDate.year > 80,
          middle: p.birthDate.year <= 80 and p.birthDate.year > 70,
          old: p.birthDate.year <= 70
```

(15) Display the number of persons contained in each of the 3 preceding sets

```
select number: count (select x from x in partition)
from p in Persons
group by  young: p.birthDate.year > 80,
          middle: p.birthDate.year <= 80 and p.birthDate.year > 70,
          old: p.birthDate.year <= 70
```