Standards Objets

OMG - ODMG

ODL - OQL

Tirés des documents de Y. Pigneur, S. Rey et D Donsez

Plan

- Introduction à l'ODMG:
  - Contenu de la proposition;
  - Architecture d'un SGBDO;
- Object Definition Language (ODL)
- Object Query Language (OQ)

Object Management Group

- Consortium industriel à but non lucratif fondé en 1989
- Objectif : gérer les normes relatives aux objets.
- Composition : plus de 400 membres dont Sun microsystems, Borland, Hitachi, Unisys, Oracle etc.
- Contrairement à l'ISO, ANSI et IEEE, OMG n'est pas une organisation de normalisation reconnue
- Son but : développer des recommandations sous formes de standards de fait, qui pourrait éventuellement déboucher sur des normes.

ODMG (*Object Database Management Group*)

www.odmg.org

- Objectifs de l'ODMG:
  - Réaliser l'équivalent de la norme SQL pour les bases de données objets.
  - Permettre l'utilisation directe des types des langages objet.
  - Définir un modèle abstrait de définition de bases de données objet, mis en oeuvre par un langage appelé ODL (*Object Definition Language*).
  - Adapter le modèle à un langage objet particulier:
    - C++;
    - Smalltalk;
    - Java.
  - Proposer un langage d'interrogation: OQL (*Object Query Language*).

Contenu de la proposition

- **ODL - *Object Definition Language***

  Langage de définition de schéma des bases de données objet proposé par l'ODMG. (Equivalent des *DDL - Data Definition Language* des SGBD.)

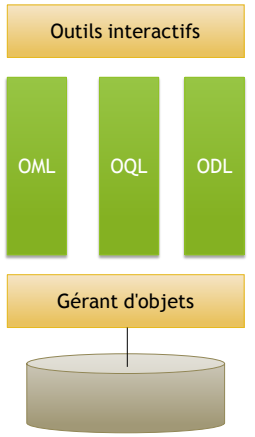
- **OQL - *Object Query Language***:

  Langage d'interrogation de bases de données objet proposé par l'ODMG, basé sur des requêtes SELECT proches de celles de SQL.

- **OML - *Object Manipulation Language***:

  Langage de manipulation intégré à un langage de programmation objet permettant la navigation, l'intérrogation (OQL) et la mise à jour de collections d'objets persistants, dont l'OMG propose trois variantes: OML C++, OML Smalltalk, et OML Java.

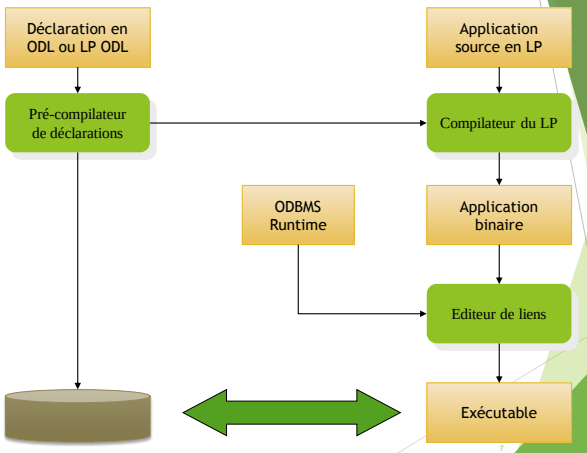
Architecture d'un SGBDO conforme à l'ODMG



- **Editeur de classes**
- **Manipulateur d'objets**
- **Bibliothèques graphiques**
- **Débogueur, éditeur**

- **Persistance**
- **Identification**
- **Accès**
- **Concurrence**
- **Fiabilité**
- **Sécurité**

Lien avec les langages de programmation



* LP - Langage de Programmation

Object Definition Language

Object Definition Language

- ODL est un langage pour décrire le schéma des bases de données objet.
- ODL définit les types d'objet que l'on peut implémenter dans de nombreux langages de programmation:
  - ODL n'est pas lié, ni à la syntaxe, ni à la sémantique d'un langage de programmation.
- ODL est basé sur IDL, le *Interface Definition Langage* de l'OMG:
  - www.omg.org pour plus d'information.

Motivations

- ODMG
  - Normalisation des langages d 'interface des SGBD-OO
- Principe : indépendance entre le LDD et les LMD
  - LDD : ODL
  - LMD : C++, SmallTalk, Java
- ODL étend la syntaxe de l 'IDL de OMG/CORBA

ODMG ODL

- **interface** – Noms et declaration des objets
- **extent** – Noms attribués aux ensembles d'objets
- **key[s]** – Déclaration des Clés
- **persistent** | **transient** – extent persistant ou temporaire
- **attribute** – Déclarer les attributs
- **readonly** – Attribut à lecture seule.
- **Set** | **List** | **Bag** | **Array** – Type collection (ensemble, liste, sac, tableau)
- **relationship** – déclarer les relations
- **inverse** – Déclarer les relations inverses

Types ODL

- Distinction entre Litéral et Objet
- Litéral
  - Types atomiques
    - int, real, string
  - Constructeurs de type
    - énumération enum
    - structures struct, union
    - collections génériques : set, bag, list, array
  - *chaque litéral est « caractérisé » par sa valeur*
    - *L1 et L2 sont égaux si leurs valeurs sont égales*
- Objet
  - définition de l 'interface
  - *chaque objet est identifié par son identité*
  - *O1 et O2 sont égaux si leurs identifiants sont égaux*

Définition d'une interface

- **interface = spécification d 'un type**
  - super (Héritages simple et multiple)
  - extent, clés candidates
  - attributs : attribute <type> <nomattr>;
  - associations et associations inverses
    - relationship <type> <nomasso> inverse <nom-d-interface>::<nomasso>;
  - **méthodes**
    - <type-retourné> <nommeth> (<type-paramêtre> : <type>, …)
    - raise (<type-d-exception>)
    - <type-paramêtre> : in, out, inout
  - **classe**
    - interface + une implantation particulière du type
    - dans un des LMD disponibles

Exemple

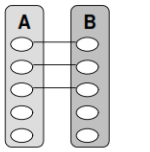
Dept interface **Employee** {

attribute int numemp; attribute string name;
attribute float basesalary; attribute Date birthday;
attribute Struct Addr { string street, string city, int zip } address;
attribute enum TypeEmp { worker,manager} typeEmp;
attribute Set<string> phones;
relationship Département Dept inverse Dept::members;
float salary();
}

interface **Département** {
attribute string name;
attribute Struct Employee::Addr address;
relationship Set<Employee> members inverse Employee::dept ;
float totsalary() ;

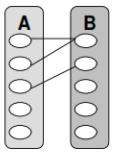
}

Cardinalités des Assocations



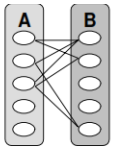
One-to-one
1-1

```
interface A {
 relationship B rb
      inverse B::ra;
}
interface B {
 relationship A ra
      inverse A::rb;
}
```



Many-to-one
N-1

```
interface A {
 relationship B rb
      inverse B::ra;
}
interface B {
 relationship Set<A> ra
      inverse A::rb;
}
```



Many-to-many
N-M

```
interface A {
 relationship Set<B> rb
      inverse B::ra;
}
interface B {
 relationship Set<A> ra
      inverse A::rb;
}
```


Rôles

- Nom des associations
- Rôles asymétriques

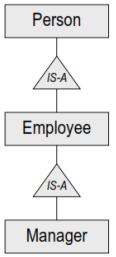
interface Person { …
relationship Person ***epoux*** inverse Person::***epouse***;
relationship Person ***epouse*** inverse ***epous***;
relationship Dept ***dept*** inverse Dept::***members***;

- Rôles symétriques

…
relationship Person ***conjoint*** inverse Person::***conjoint***;
}

Héritage

```
interface Person {
    attribute string name;
    attribute date_t birthday
    attribute Addr addr;
}
interface Employee : Person {
    attribute int basesalary;
    float salary();
}
interface Manager : Employee {
    attribute int bonus;
    float salary();
}
```



Identité d 'objets

- **OID : Identifiant d 'objet**
  - chaque objet est identifié de manière unique par son identifiant d 'objet OID
  - Propriétés de l 'OID
    - Unique
    - Invariable (i.e ne dépend pas de la valeur de l 'objet comme les clés de SQL)
    - N 'est pas accessible au développeur (contrairement à SQL3)

Clés

- Clé = groupe d 'attributs

```
interface Person ( key numemp){
    attribute string numemp;
    attribute string numss;
    attribute string name;
    attribute date_t birthday
    attribute Addr addr;
}
```

- Clés candidates

```
interface Person ( key numemp; (numss); (name,birthday) ) { … }

interface Enseignement( key (filiere,intitule);(salle,heure) ) { … }
```


Extent

- ensemble des objets instanciés pour l'interface

```
interface Person
  extent Persons
{
   attribute string name;
   …
}
```

```
interface Employee : Person
  extent Employees
{
   attribute int basesalary;
   …
}
```

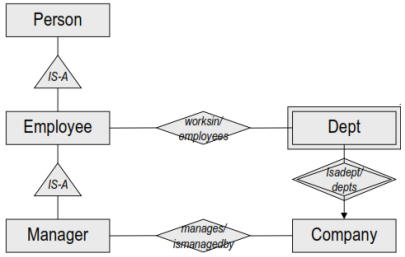
- utilisé par OQL (clause FROM)

Typedef, Module

- Typedef : Définition de types litéraux
  - typedef int Franc, Euro;
  - typedef struct { string street, string city, int zip } Addr ;
  - typedef enum Couleur {rouge, rose, blanc};
- Module
  - regroupement dans le même espace de nom

```
Module DRH {
typedef struct { ... } Addr ;
interface Person { …; Addr addr; ... };
};
module Marketing {
typedef struct { ... } Addr ;
interface Customer { …; Addr deliv_addr; Addr bill_addr; ...  };

};
```


Exemple

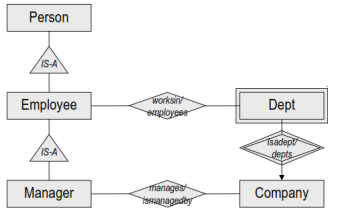



```
interface Person {
   attribute string name;
   attribute date_t birthday
   attribute Addr addr;

   int age();
}
```

```
interface Employee : Person {
   attribute int basesalary;
   relationship Dept worskins
      inverse Dept::employees;

   float salary();
}
```

```
interface Manager : Employee {
   attribute int bonus;
   relationship Dept manages
      inverse Dept::manager;
   float salary();
}
```

Suite




```
interface Company {
    attribute string name;
    attribute Addr fiscaddr;

    relationship Set<Dept> depts
        inverse Dept::company;

    int allsalaries();
}
```

```
interface Dept {
    attribute string name;
    attribute Addr postaddr;

    relationship Company company
        inverse Company::depts;
    relationship Set<Employee> employees
        inverse Employee::worksin
    relationship Manager manager
        inverse Manager::manages

    int allsalaries();
}
```


Object Query Language

OQL

- Permettre un accès facile à une base objet;
- offrir un accès non procédural pour permettre des optimisations automatiques (ordonnancement, index, ...);
- garder une syntaxe proche de SQL;
- rester conforme au modèle de l'ODMG;
- permettre de créer des résultats littéraux, objets, collections, ...;
- supporter des mises à jour limitées via les opérations sur objets, ce qui garantit le respect de l'encapsulation.