Dr. Rim Samia Kaabi

Bases de Données Avancées
Le modèle objet: standard de l'ODMG - ODL

1

Diversité des modèles

• Norme ODMG
mais de nombreux SGBDO ne la suivent pas.

•Ce cours définit :
- •les principes communs aux SGBD OO
- •les alternatives importantes

•Ce cours emploie une syntaxe tirée de celle d'ODMG

Principaux concepts

Monde réel	BD OO
objet	objet, classe d'objets
propriété	attribut
	méthode
lien	lien de composition
	binaire
	sans attribut
	orienté
représentation multiple	hiérarchie de généralisation/ spécialisation, héritage

ODL : Aperçu

- Une déclaration de classe inclue:
  1. Un nom.
  2. déclaration(s) optionelle(s) des clés.
  3. Déclaration d'*Extent* = noms de l'ensemble des objets issus de la classe.
  4. Déclarations des éléments. Un *élément* est soit un attribut, une relation ou une méthode.

4

Déclaration d'une Classe

```
class <name> {
   <liste de la déclaration des éléments
   séparés par des ;>
}
```

5

Attributs et Relations

- Les Attributs sont (généralement) des éléments avec un type "primitif".

  attribute <type> <name>;

- Relations connectent un objet à un ou plusieurs objets

  relationship<type> <name> inverse <relations>;

6

Inverse Relations

- Supposant une classe *C* ayant une relation *R* avec une classe *D*.
- Alors la classe *D* doit avoir une relation S avec *C*.
- *R* et *S* doivent être des réciproques.
  - Si un objet *d* est en relation avec un objet *c* par *R*, alors *c* doit être reliée à *d* par *S*.

7

Exemple: Attributs et Relation

```
class Train {
    attribute string code;
    attribute number capacite;
    relationship Set<Ville> desserve inverse
    Ville::desserviePar;
}

class Ville {
    attribute string nom;
    attribute string CP;
    relationship Set<Train> desserviePar inverse
    Train::desserve;
}
```

8

Types des Relations

Le type d'une relation est soit

1. Une classe, comme train. Dans ce cas, un objet avec cette relation ne peut être connecté qu'a un seul objet train.
2. Set<Ville> {non ordonné}: l'objet est connecté à un ensemble d'objets villes.
3. Bag<Ville>{avec répétition} , List<Ville> {ordonnée}, Array<Ville> (une dimension): l'objet est connecté à un bag, une liste, ou un tableau d'objets villes.

9

Multiplicité des Relationships

- Toutes les relations en ODL sont binaires.
- Les relations M-M doivent être des Set<…>
- Les relations 1-M sont des Set<…> dans la relation M et une classe dans la classe 1
- Les relations 1-1 sont des classes de part et d'autre

10

Exemple: Multiplicité

```
class Train { …
    relationship Set<wagon> composede inverse
    Wagon::compose;
    relationship <Wagon> Moteurde inverse
    Wagon::tracte;
}
class Wagon { …
    relationship Set<Train> compose inverse
    Train::composede;
    relationship <Train> tracte inverse Train::Moteurde;
}
```

11

Exemple 2

```
class Personne {
    attribute … ;
    relationship Personne pere inverse
    Personne::fils;
    relationship Personne fils inverse
    Personne::pere;
    relationship Set<Personne> freres
        inverse Personne::freres;
}
```

12

Simuler une relation n-aire

- Supposons que nous voulons connecter les classes X,Y et Z par une relation R
- On construit une classe *C*, constituée du triplet d'objets (*x*, *y*, *z*) des classes *X*, *Y*, et *Z*.
- Créer 3 m-1 relations de (*x*, *y*, *z*) à *x*, *y*, et *z*.

13

Exemple

- Soient les classes hôtel et chambre.
- Nous voulons représenter le prix de la chambre dans chaque hôtel
  - Une relation m-n entre hôtel et chambre ne peut pas être porteuse de données (pas comme E/R)
  - Une solution: créer une classe prix et une classe de connexion PCH pour représenter le prix d'une chambre dans un hôtel.

14

Exemple --- Suite

- Puisque les objets de la classe Prix sont juste des nombres, une meilleur solution serait de :
  1. Ajouter aux objets PCH un attribut Prix.
  2. Utiliser 2 relations m-1 entre PCH, chambre et hôtel.

15

Exemple---- fin

```
class PCH {
   attribute prix:real;
   relationship Chambre laChambre inverse
   chambre::toPCH;
   relationship Hotel lhotel inverse Hotel::toPCH;
}
```

- Chambre et hôtel doivent être modifiées pour inclure les relations toPCH, qui sont de type Set<PCH>.

16

Structs et Enums

- Les attributs peuvent avoir des structures ou être des énumérations.
- Déclaration

attribute [Struct ou Enum] <nom de la struct ou enum> { <détails> }
<nom de l'attibut>;

- Détails sont des noms de champs et de types pour les Struct(s), une liste de constantes pour les Enum(s).

17

Exemple: Struct and Enum

```
class Hotel {
   attribute string name;
   attribute Struct Addr
        {string street, string city, int zip} address;
   attribute Enum cat
        { '1*', '2*', '3*', '4*', '5*', Palace } Categorie;
   relationship …
}
```

Nom de la structure et de l'énumeration

Noms des attributs

18

Attention!!!

- CLASS Etudiant

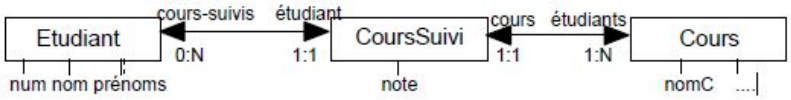
{ attribute int num ;
attribute string nom ;
attribute list string prénoms ;
attribute set struct {cours : Cours ;note : FLOAT } cours-suivis}
→ Impossible en ODMG

19

Solution!!!

CLASS Etudiant
{attribute int num ;
attribute string nom ;
attribute LIST STRING prénoms ;
attribute SET CoursSuivi cours-suivis }

CLASS CoursSuivi
{attribute Cours cours ;
attribute float note;
Relationship SET<Etudiant> cours-suivis INVERSE Etudiant::étudiant ;
}



20

Héritage en ODL

- Indique une super-classe avec ses attributs
- La sous-classe liste seulement les propriétés spécifiques et hérite les propriétés de la super-classe

21

interface

- Une interface: classe prédéfinie, source d'héritage par toutes les classes
- Non instanciable: sert à définir des opérations qui sont héritées par les classrs
- Interface nom-interface

  {

  }

22

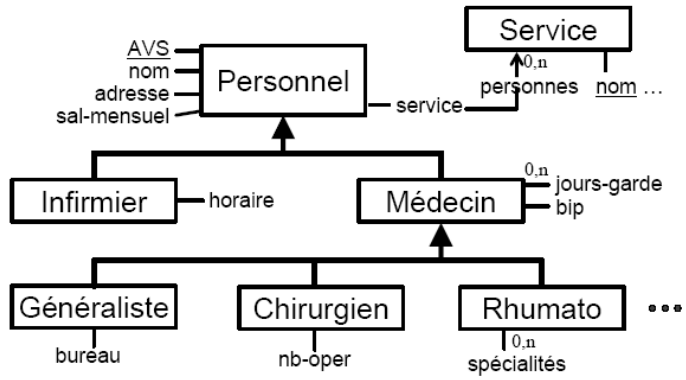
Exemple

- Commercial est une sous classe d'employé:

```
class commercial:employé {
  attribute real prime;
}
```

23

Exemple: le personnel d'un hôpital



Attention : 2 types de flèches : flèches minces : composition
flèches épaisses : is-a

Les méthodes

- Représentent le comportement du type
- Propriétés:
  - nom de l'opération
  - nom et type des arguments (in)
  - nom et type des paramètres retournés (out)
  - nom des conditions d'erreurs (raises)

Exemple de Méthode

```
void retirer (in float)
   raises(plusquelesolde);
```

1. la méthode retirer ne retourne rien
2. Retirer prend un argument (float).
3. retirer soulève l'exception `plusquelesolde`.

26

Méthode

- Une opération est appelée par la notation pointée:
- Pour comparer un objet o avec un objet p: o.same_as(p)
- Pour copier un objet: p=o.copy();

27

Le système de type d'ODL

- Types basiques : int, real/float, string, enumerated types, et classes.
- Type constructeurs:
  - Struct pour les structures.
  - *Types collection* : Set, Bag, List, Array
  - Types relations peut être seulement une classe ou une collection.

28

ODL : clés

- On peut déclarer un nombre infini de clés dans une classe
- Après le nom de la classe, on ajoute:

  (key <list of keys>)
- Une clé composée de plusieurs attributs doit être entre ().

29

Exemple: clés

```
class Hotel (key nom) { …
```

→ nom est la clé de la classe hôtel.

```
class Cours (key
  (dept,num),(salle, heure)){
```

→ dept et num forment une clé;

→ Salle et heure forment une deuxième clé.

30

Extents

- Pour chaque classe il y a un *extent* : c'est l'ensemble (set) des objets existants de cette classe.
  - Extent (relation entre la classe et ses objets.
- Indiquer l'extent après le nom de la classe et avec les clés:

  (extent <extent name> …)

31

Exemple: Extents

```
class Personne
  (extent LesPersonnes key
  numPersonne) { …
}
```

- Convention : utiliser le singulier pour le nom de la classe et le pluriel pour le nom de l'extent.

32

Population et persistance

- Objectifs :
  - BD : gérer des ensembles d'objets permanents : "populations"
  - LPOO : permettre aux utilisateurs de manipuler de la même façon des objets temporaires et des objets permanents

- SGBD classiques :
  - Relation
    1) définition de la structure des occurrences potentielles
    2) récipient contenant toutes les occurrences existantes, permanentes par définition
- LPOO :
  - Classe = 1) usine pour fabriquer des objets de même type
  - Les objets sont temporaires
    durée de vie = celle de leur programme (sauf s'ils sont stockés dans un fichier)

Conclusion

- Objectifs atteints
  - u meilleure représentation du monde réel
  - réutilisation
  - efficacité pour les applications nouvelles
- MAIS
  - Les SGBDO ne sont pas adaptés à tout type d'application !
  - Méthodologies de conception incomplètes
    normalisation de la structure, conception des méthodes
  - Compétition entre standards
  - Absence de théorie, formalisation
  - Vues
  - Evolution du schéma
  - Versions, temps
  - Migration difficile des SGBD classiques aux SGBDO