Chapitre 4

Bases de données orientées objets

Master II STIC

Année universitaire 2025/2026

1. Concepts fondamentaux des bases de données orientées objets

1.1. Motivation

Les bases relationnelles montrent leurs limites lorsque les données comportent :des structures complexes (objets imbriqués, collections),un fort couplage avec un langage orienté objet (Java, C++, Python),des besoins de persistance transparente.

Les bases orientées objets ont été introduites pour réduire l'impédance sémantique entre le modèle objet et le modèle relationnel.

1. Concepts fondamentaux des bases de données orientées objets

1.2. Caractéristiques principales

Encapsulation Les données et les méthodes sont stockées ensemble.

Identité d'objet (OID) Chaque instance possède un identifiant unique, indépendant de sa valeur.

Héritage Une classe peut hériter de propriétés d'une autre.

Polymorphisme Une même méthode peut avoir plusieurs implémentations selon le type de l'objet.

Relations (références directes) Contrairement aux clés étrangères, les BD orientées objets utilisent des références d'objet.

Collections Support direct des listes, ensembles, tableaux, dictionnaires.

1. Concepts fondamentaux des bases de données orientées objets

3. Modèle de données d'une BD orientée objet Une base orientée objet stocke les mêmes concepts qu'un langage OO :

3.1. Classes Définitions de structures et comportements.

Exemple :

```
class Personne {
   String nom;
   Date naissance;
   void afficher();
}
```

1. Concepts fondamentaux des bases de données orientées objets

3.2. Objets Instances persistantes de classes, identifiées par des OID.

3.3. Types complexes

- objets imbriqués
- collections
- types définis par l'utilisateur

3.4. Méthodes persistantes Certaines BD exécutent directement la méthode stockée (Gemstone, db4o).D'autres ne stockent que les données (ObjectDB).

1. Concepts fondamentaux des bases de données orientées objets

4. Modèle ODL (Object Definition Language)

ODL est un langage de définition d'objets proposé par l'OMG dans ODMG-93.

4.1. Objectifs

- décrire les types persistants ;
- définir relations, attributs, contraintes ;
- base pour OQL (Object Query Language).

1. Concepts fondamentaux des bases de données orientées objets

4.2. Exemple ODL

```
interface Personne {
attribute string nom;
attribute Date naissance;
relationship Adresse adresse
        inverse Adresse::proprietaire;};
```

1. Concepts fondamentaux des bases de données orientées objets

4.3. Éléments du modèle

- interfaces / classes
- Attributs
- Relations (unidirectionnelles ou bidirectionnelles)
- Collections (set, bag, list, array)
- Constructeurs et méthodes

1. Concepts fondamentaux des bases de données orientées objets

5. Modèle OQL (Object Query Language)

Langage de requêtes compatible objet, similaire à SQL mais manipulant objets et collections.

5.1. Caractéristiques

- Navigation via références d'objets.
- Résultats typés (objets, attributs, collections).
- Fonctions d'agrégation intégrées.
- Compatible avec les types définis en ODL.

1. Concepts fondamentaux des bases de données orientées objets

5.2. Exemple de requêtes

Sélection :

```
select p.nom
from Personne p
where p.naissance.year() < 1990;
```

1. Concepts fondamentaux des bases de données orientées objets

Navigation dans les relations :

```
select p.adresse.ville
from Personne p;
```

Agrégation :

```
select avg(p.salaire)
from Employe p;
```

1. Concepts fondamentaux des bases de données orientées objets

Requêtes imbriquées :

```
select p
from Personne p
where p.adresse.ville in
        (select a.ville from Adresse a where a.codePostal = "16000");
```

Modèle ODL (Object Definition Language)

ODL est un langage de définition d'objets standardisé par l'ODMG (Object Data Management Group).

Il permet de décrire :

- les classes,
- les attributs,
- les relations,
- les collections,
- les types complexes,
- et les contraintes.

ODL joue le rôle du DDL dans les bases relationnelles.

Modèle ODL (Object Definition Language)

1. Structure générale d'une définition ODL

Une définition ODL possède généralement la structure suivante :

```
interface NomClasse (super ClasseParent) {
attribute Type attribut1;
attribute Type attribut2;
relationship TypeRelation nomRelation
        [inverse ClasseOpposee::nomInverse];
// méthodes optionnelles
void methode(in Type param);};
```

Modèle ODL (Object Definition Language)

2. Éléments essentiels du modèle ODL

2.1. Interfaces / Classes

ODL utilise le mot-clé interface, mais il s'agit bien de classes de stockage (persistantes).

```
interface Personne {
 attribute string nom;};
```

Modèle ODL (Object Definition Language)

2. Éléments essentiels du modèle ODL

2.2. Héritage (single inheritance)

```
interface Employe : Personne {
attribute double salaire;
};
```

Modèle ODL (Object Definition Language)

2. Éléments essentiels du modèle ODL

2.3. Attributs

Les attributs peuvent être :

- simples : string, integer, boolean, float, date, …
- complexes : struct, enum, array, …

Exemple struct :

struct Coordonnees {

float x;

float y;};

Modèle ODL (Object Definition Language)

2. Éléments essentiels du modèle ODL

2.4. Collections

Types :

- set<T> : ensemble sans doublons
- bag<T> : multi ensemble
- list<T> : liste ordonnée
- array<T, n> : tableau fixe

Exemple :

```
interface Cours {
attribute set<Etudiant> inscrits;
};
```

Modèle ODL (Object Definition Language)

2. Éléments essentiels du modèle ODL

2.5. Relations (associations)

ODL permet de définir des relations bidirectionnelles ou unidirectionnelles.

relationship Adresse adresse

inverse Adresse::proprietaire;

Relation inverse :

interface Adresse {

relationship Personne proprietaire

inverse Personne::adresse;};

Modèle ODL (Object Definition Language)

2. Éléments essentiels du modèle ODL

2.6. Contraintes

ODL permet de préciser :

- Unicité
- Cardinalité
- types de relations (1-1, 1-n, n-n)

Exemple cardinalité :

relationship set<Etudiant> etudiants

inverse Etudiant::cours;

2. Langage OQL (Object Query Language)

OQL est l’équivalent objet de SQL.

Basé sur :

- navigation par références d'objet (pas de clés étrangères),
- requêtes manipulant objets complets,
- fonctions, méthodes, collections.

2. Langage OQL (Object Query Language)

2.1. Structure générale d'une requête OQL

select <expression>

from <variable> in <collection>

where <condition>

group by <expression>

having <condition>

order by <expression>

2. Langage OQL (Object Query Language)

2.2. Types d'expressions OQL

2.2.1. Accès aux attributs

```
select p.nom from Personne p;
```

2.2.2. Navigation d'objets

```
select p.adresse.ville from Personne p;
```

2.2.3. Sélections

```
select pfrom Personne p where p.age > 30;
```

2. Langage OQL (Object Query Language)

2.3. Collections et opérations intégrées

OQL supporte :

- size(x)
- avg(x)
- sum(x)
- max(x)
- min(x)

Exemple :

```
select avg(e.salaire)
from Employe e;
```

2. Langage OQL (Object Query Language)

2.4. Requêtes imbriquées

```
select p from Personne p
where p.adresse.ville in
        (select a.ville from Adresse a
                where a.codePostal = "16000");
```

2. Langage OQL (Object Query Language)

2.5. Jointures (implicites via navigation)

Pas de JOIN explicite comme en SQL :

```
select e.nom, e.departement.nom
from Employe e;
```

2. Langage OQL (Object Query Language)

2.6. Projection d'objets complets

```
select struct(nom: p.nom, age: p.age)
from Personne p;
```