

*République Algérienne Démocratique Et Populaire
Ministre De L'enseignement Supérieur Et La
Recherche Scientifique
Université De Bechar*



Institut : Sciences exactes

Département : Informatique

Filières : 5 Informatique N/6

Module : info 40

SGBD-RO

- *Présenté Par :*
- *ELALAOUI ABD ERRAHMANNE*
- *KHAOUID ALI*

Charge module :
Mr.Hocieni .y

 **2008-2009**

SOMMAIRE

I. Introduction

II. Les Bases de Données Objet (BDO)

II.1. Concepts de l'orienté objet :

II.1.1. objet

II.1.2. Classe d'objets

II.1.3. Abstraction, encapsulation et communication entre objets

II.1.4. Relations entre classes et liens entre objets

II.1.5. Identité et persistance d'un objet :

II.2. Conception d'une base de données objet (BDO)

II.3. Caractéristiques des BDO :

III. Les SGBD objets (SGBDO)

III.1. L'approche BD relationnel-objet.

III.2. Implémentation d'une base de données objet.

III.2.1. Structures de données dans les modèles orientés objets

III.2.2. Définition d'une classe d'objets.

III.2.3. Définition des relations entre classes d'objets.

IV. Avantages et limites des SGBD objets

IV.1. Avantages

IV.2. Limites

V. Conclusion

VI. Glossaire

VII. Bibliographie

I. Introduction

Les SGBDOO n'ont pas pu faire leur place sur le marché des SGBD. Mais malgré le succès des SGBDR et leur performance, ceux-ci ne sont pas sans insuffisances.

Parmi ces insuffisances, le caractère atomique des attributs et l'absence de représentation de la relation de généralisation/spécialisation. Pour pallier ces problèmes, et bénéficier des avancées apportées par l'approche orientée objet, les développeurs des SGDBR ont commencé, dès les années 90, à étendre leurs systèmes pour supporter le paradigme objet. Ceci permet de bénéficier des atouts des deux modèles. Dans ces systèmes, les objets sont stockés dans des tables relationnelles, ce qui permet d'en indexer les accès. L'avantage de ces systèmes par rapport aux SGBDOO est la rapidité d'accès et la compatibilité ascendante avec les SGBDR. De plus, leur langage de requêtes est basé sur la norme SQL3 qui est une extension du langage SQL à des concepts objet.

II. Les Bases de Données Objet (BDO)

Avant de présenter les BDO, nous allons expliciter les concepts clés du paradigme objet.

II.1. Concepts de l'orienté objet :

II.1.1. Objet :

Un objet est un élément d'un domaine d'étude que l'on peut distinguer par son identité, son état et son comportement.

Objet = identité + comportement + état

Identité (oid) : unique et invariant servant de référence à l'objet de manière non ambiguë.

Comportement : définit l'ensemble des opérations (méthodes) applicables à l'objet.

Etat : c'est l'ensemble des valeurs des attributs caractérisant l'objet, ces valeurs peuvent être des littéraux ou des références d'objets.

Exemple :

Voici deux exemples d'objets, un objet de type *Etudiant* décrit par un matricule, un nom, un prénom, une date de naissance, une année d'étude et une filière, et un objet de type *Module* décrit par un code, un libellé et un coefficient.

<i>Etudiant</i>
Mat : 520032569 Nom:"Abdi" Prénom:"mohamed" Datjais: 2/05/1986 An-etude :2 Filière: "info"

<i>Module</i>
cod_mod:"ALGO" Libelle_mod:" algorithmique" Coef:5

Les valeurs (520032569, "Abdi", "Mohamed", 12/05/1986, 2, "info") décrivent l'état de l'objet de type *Etudiant*.

De même, les valeurs ("ALGO", "algorithmique", 5) décrivent l'état de l'objet de type *Module*.

II.1.2. Classe d'objets :

Une classe factorise la structure et le comportement commun à une collection d'objets (les instances de la classe), elle est caractérisée par :

Classe - opérations + attributs + instanciation

Attributs (appelées aussi variables d'instances) ayant un nom et un type (simple ou complexe).

Opérations (appelées aussi méthodes) applicables aux objets de la classe, et permettent de changer l'état d'un objet.

Schématiquement, une classe est représentée comme suit :

<i>Nom de la classe</i>
Attr1 : type1 ; Attr2 : type2 ;
Méthode 1(); Methode2();

Trois compartiments sont distingués :

Le premier contient le nom de la classe,

Le second contient la liste des attributs décrivant les objets de la classe, chaque attribut possède un type qui peut être simple, structuré ou même une référence à un autre objet (à voir plus loin).

Enfin le 3^{ème} compartiment comporte les méthodes (opérations) qu'on peut effectuer sur les objets de la classe, suite auxquelles les objets changent d'états.

- *Instanciation :*

L'instanciation est un mécanisme de création des objets (c'est l'opération new dans les langages de programmation tels que C++ et java).

Les attributs et les opérations sont appelés propriétés de la classe. Une classe peut être vue comme :

Un type abstrait de données, Un regroupement d'objets de même type, ou Un générateur d'instances.

Exemple :

La classe Etudiant est un type qui décrit tous les étudiants, il peuvent être définies comme suit :

<i>Etudiant</i>
Matricule:string ; Nom : string ; Prénom:string ; Dat_nais:date ; An_étude:integer ; filière : string ;
AffieherO ; AjouterO ; SupprimerO ; Changer an_etude() ;

II.1.3. Abstraction, encapsulation et communication entre objets

a. Abstraction :

Représente l'examen sélectif de certains aspects d'un problème. Il s'agit d'isoler les aspects importants et de supprimer ceux qui ne le sont pas.

Une classe est une abstraction d'un type de données, donc à travers le nom de la classe seulement on a toutes les informations concernant la classe.

b. Encapsulation :

C'est le processus de dissimulation des détails d'un objet qui ne contribuent pas à ses caractéristiques essentielles. L'idée est d'interdire l'accès direct aux attributs en obligeant l'utilisateur de passer par les fonctions (méthodes) définies pour effectuer le traitement voulu. Cette limitation des possibilités permet d'assurer la compatibilité entre les données fournies et les données attendues.

Les règles de visibilité (protection) viennent compléter ou préciser la notion d'encapsulation selon trois niveaux de visibilité :

Public : l'attribut ou la méthode est visible globalement, i.e pour toutes les classes.

Protégé : l'attribut ou la méthode est visible pour les classes dérivées de la classe mère (la super-classe).

Privé : l'attribut ou la méthode est totalement opaque.

L'encapsulation présente un double avantage :

Les données sont protégées des accès intempestifs (garantir leur intégrité).

Les utilisateurs sont déchargés de connaître l'implémentation des objets.

c. Communication entre objets :

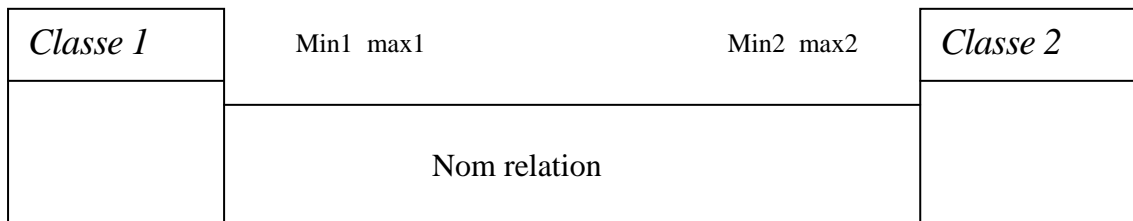
Les objets communiquent entre eux par envoi de messages. L'envoi d'un message correspond à l'appel d'une méthode qui s'exécute sur l'objet et permet ainsi d'accéder à la structure, de l'objet.

II.1.4. Relations entre classes et liens entre objets :

Les liens particuliers qui relient les objets peuvent être vus de manière abstraite dans le monde des classes : à chaque famille de liens entre objets correspond une relation entre les classes de ces mêmes objets. De même que les objets sont des instances de classes, les liens entre objets sont des instances des relations entre classes.

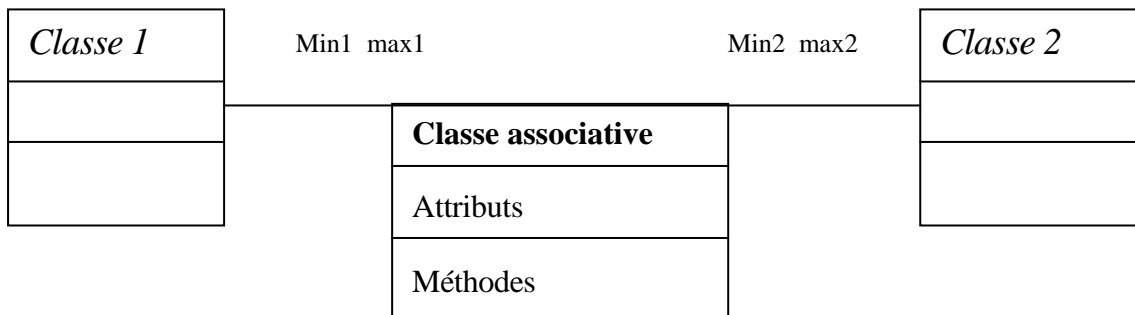
a. Association :

Spécifie une relation entre deux classes, elle est définie par le nom de la relation ainsi que les multiplicités désignant la participation des objets de chacune des classes dans la relation. Dans certains cas, l'association peut avoir des propriétés propres à elle, elle génère alors une nouvelle classe appelée classe associative. Schématiquement, l'association se présente de la manière suivante :



Mini et Maxi désignent respectivement les cardinalités min et max des objets de la classe i dans la relation.

Dans le cas d'une classe associative, le schéma sera le suivant :

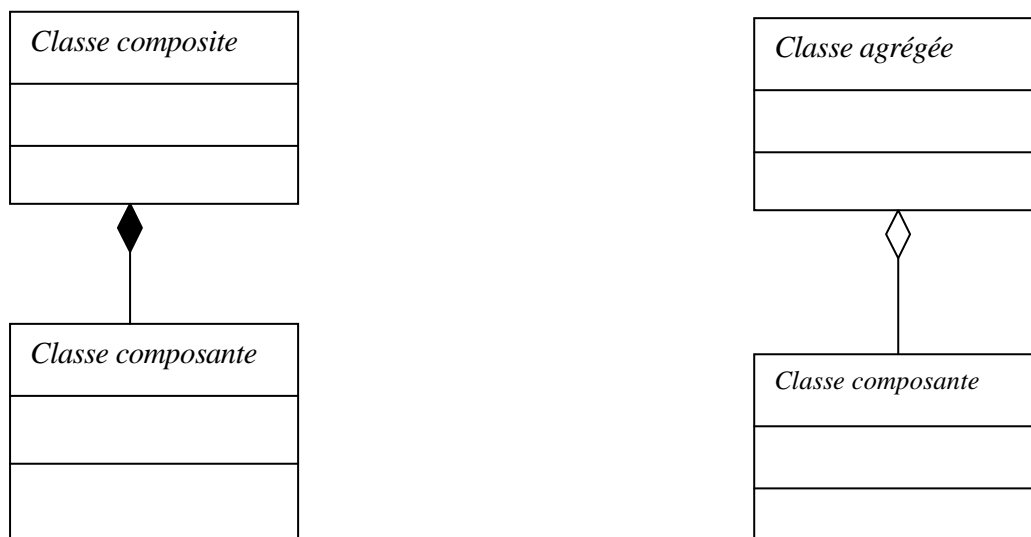


b. Composition/agrégation :

C'est une relation entre deux classes spécifiant que les objets de la classe cible sont des composants de la classe source. La composition permet de représenter des associations de type maître/esclave, tout/parties composé/composants.

Une autre forme de composition est l'agrégation qui signifie composition faible. La différence entre les deux est la suivante :

Dans la composition, la destruction de l'objet composite implique la destruction de ses composants, alors que dans l'agrégation l'objet agrégé (composite) peut être détruit sans la destruction de ses composants. Ces deux relations sont schématisées comme suit :



c. Héritage :

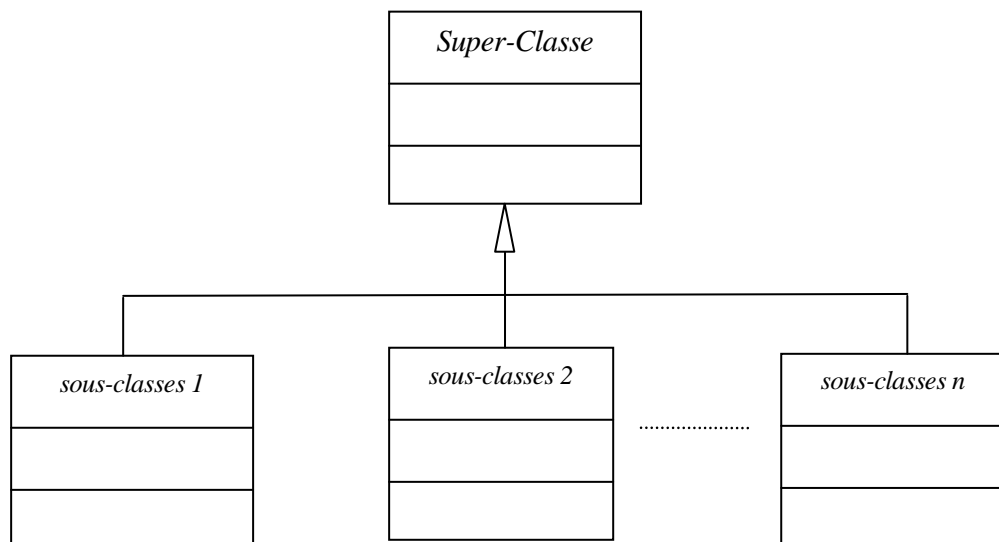
C'est un mécanisme de transmission des propriétés d'une classe vers sous-classes, chacune des sous-classes possède ses propres propriétés et les propriétés de ses super-classes. Une sous-classe peut hériter de plusieurs autres, c'est l'héritage multiple. L'intérêt majeur de l'héritage est de pouvoir ajouter de nouvelles propriétés pour les classes dérivées et de bénéficier des propriétés existantes, donc éviter la redondance des définitions des attributs et des méthodes communes.

d. Généralisation/Spécialisation :

En se basant sur la relation d'héritage entre les classes, on peut définir deux relations :

- *Généralisation* : fonction qui fait correspondre à une sous-classe, une classe plus générale (appelée super-classe, classe mère ou classe générique).
- *Spécialisation* : c'est la fonction inverse, donc elle fait correspondre à une classe toutes ses sous-classes.

Ceci permet de construire un graphe qu'on appelle graphe de classes ou graphe d'héritage où la relation de correspondance (est un) est représentée par une flèche de la sous-classe vers la classe supérieur. L'intérêt est la classification des objets en fonction de leurs points communs (généraux) et leurs spécificités.



e. Polymorphisme :

C'est la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes. Le polymorphisme désigne la possibilité de déclencher des opérations différentes en réponse à un même message (appel d'une méthode). Chaque sous-classe a la possibilité de surcharger (redéfinir) une méthode héritée afin de tenir compte des propriétés locales (non héritées) de la sous-classe.

L'intérêt du polymorphisme est double :

- Il minimise le coût de l'évolution du logiciel. Ainsi, il permet le choix dynamique d'un traitement en fonction de l'objet auquel v est appliqué.

- Le polymorphisme rend possible le choix automatique de la bonne méthode à adopter en fonction du type des données passées e paramètre. On dit que la méthode est polymorphe.

II.1.5. Identité et persistance d'un objet :

a. Identité d'objet :

L'identité d'un objet permet de le distinguer des autres, en effet même si deux objets ont la même valeur, ils sont distincts par leurs identités.

L'identité des objets est implémentée par un identifiant système géré par le SGBDOO. Dès la création d'un objet, un identifiant système lui est attaché. La valeur de cet identifiant n'est ni affichable, ni imprimable ni modifiable. Cet identifiant est appelé référence de l'objet ou son oid.

b. Persistance d'un objet :

Il peut être souhaitable que certains objets créés par une application survivent à l'exécution de cette application, c'est à dire soient persistants. Cette caractéristique est particulièrement requise dans les situations suivantes :

- Les informations mémorisées par les objets sont partagées entre plusieurs applications.
- La même application peut s'exécuter ultérieurement en prenant comme état initial un état mémorisé par les objets persistants
- Les résultats de traitements effectués par l'application doivent être mémorisés en vue de leur archivage.

Le SGBDO doit fournir des opérations primitives pour créer, lire, écrire et détruire les objets persistants, ceci à partir d'un programme dans un langage à objets. Un objet persistant est créé avec un identifiant unique et permanent qui doit permettre de le retrouver rapidement sur disque.

II.2. Conception d'une base de données objet (BDO)

Pour concevoir une BDO, il suffit de suivre les étapes ci-dessous :

- Construire le dictionnaire des données du domaine d'étud considéré : il s'agit de déterminer les différents attributs manipulés dans l'étude, donner la signification et le type de chacun d'eux.
- Déterminer les différentes classes d'objet, en spécifiant pour chacune d'elles l'ensemble des attributs rattachés ainsi que les méthodes s'exécutant sur les objets de la classe.
- Déterminer les relations (associations, compositions, héritage) entre les classes d'objets.
- Construire le diagramme de classes en adoptant un formalisme graphique orienté objet (diagramme de classes d'UML, ou d'OMT...).

II.3. Caractéristiques des BDO :

Comme les BDO héritent de toutes les propriétés du paradigme objet elles sont caractérisées par quatre points essentiels :

- Un modèle de données qui permet de représenter des structures de données complexes;
- Les données et les traitements ne sont plus séparés. La dynamique (les méthodes) fait partie de la déclaration des objets;
- L'héritage ;
- Tout objet possède une identité qui le distingue de tout autre objet, même s'ils ont la même valeur.

D'autres critères sont souvent ajoutés à cette liste, notamment le suivant:

- il n'y a plus d'incompatibilité entre le langage de programmation et le langage de manipulation des données.

III. Les SGBD objets (SGBDO)

Un SGBD objet peut être défini par l'équation suivante :

$$SGBDO = SGBD + objets + héritage + polymorphisme$$

L'implémentation d'un SGBDO s'appuie sur deux technologies qui sont les bases de données et les langages à objets. Plusieurs approches sont possibles en combinant ces deux technologies, on pourrait tout simplement les caractériser en fonction du modèle supporté qui peut être soit un modèle *relationnel-objet* ou un modèle *orienté objet*.

III.1. L'approche BD relationnel-objet :

C'est une extension du modèle relationnel et de son langage SQL avec des concepts objets, une extension de SQL a donné naissance à ESQL2, SQL2 ensuite SQL3 en 1996, elle peut être vue comme une couche objet traduite en relationnel. Les SGBD de cette approche sont qualifiés de Relationnel-Objets (SGBDRO), UniSQL a été le premier SGBDRO intégrant les concepts objets dans SQL2, PostGres industrialisé dans le produit Illustra en est un autre exemple.

Les extensions de SQL ont été proposées par les grands constructeurs de SGBD relationnels tels que IBM, ORACLE, Sybase, Informix etc. Dans SQL3, la définition des types d'objets est confondue avec la définition de tables en utilisant la commande " *create type* " servant à la création de types simples ou structurés. SQL3 assimile la notion de classe à un type ayant des opérations associées pouvant être définies en utilisant le mot clé " *function* " .

SQL3 supporte la définition de collections, en effet la table n'est plus le constructeur de collection, le type générique SQL-table est spécialisé en plusieurs types de collections telles que Set, List, Bag et Array. Enfin, les clés étrangères dans les tables sont remplacées par des références d'objets.

III.2. Implémentation d'une base de données objet :

Les SGBD objets (SGBDO) sont encore un domaine en évolution, de ce fait il n'y a pas de consensus sur le modèle de BD objet ou sur le langage de manipulation de données objets.

III.2.1. Structures de données dans les modèles orientés objets :

La structure des données dans les modèles orientés objets repose sur le seul concept qui est l'objet.

a. Définition d'une structure complexe :

La création d'une structure complexe se fait à travers le constructeur **STRUCT**, de la manière suivante :

STRUCT(Attribut1 : type1 ; Attribut2 : type2 ;...)

type1, type2.... peuvent être des types simples usuels (integer, float,boolean,...), des types structurés (définis à l'aide du mot clé STRUCT) ou même des références à des objet.

b. Constructeur de collection :

Pour la création d'un ensemble d'éléments, on utilise le constructeur **SET**, en écrivant :

SET type ;

LIST, **BAG** et **ARRAY** sont d'autres constructeurs usuels par les structures de liste, le multi-ensemble, et le tableau respectivement.

III.2.2. Définition d'une classe d'objets :

Pour la définition d'une classe, il faut spécifier le nom de la classe, la liste des attributs avec le type de chacun d'eux ainsi que les méthodes s'exécutant sur les objets de la classe. La syntaxe est la suivante :

```

CLASS    nom de la classe
  { /* liste des attributs */
    Att 1 : type1 ;
    Att2 : type2 ;
    .....
    Attn : typen ;
  
```

Methods

```

/*liste de méthodes*/
    methodel() ;
    méthode 2 ( ) ;...
}

```

Les attributs peuvent être de types simples, complexes (*STRUCT*, *SET*, *LIST*, ...) ou même des références à d'autres objets.

Exemple :

```

CLASS Module
    {Cod_mod : string ;
      Libelle_mod : string
      Coef : integer ;
      /*méthodes */
    }
Methods (afficher() ;
  ajouter () ;
  supprimer() ;
  modifier!) ;
}

```

III.2.3. Définition des relations entre classes d'objets :

a. L'association :

L'association entre deux classes class1 et class2 est exprimée en injectant dans l'une des classes (selon les cas) des attributs qui sont des pointeurs vers des objets de l'autre classe, ces attributs sont appelés *attributs-références* dont le domaine est celui de la classe qui les définit.

Exemple :

L'association entre les classes Etudiant et Module est exprimée en ajoutant dans la classe Module, un attribut référence appelé *Modules_suivis* représentant l'ensemble des modules suivis par un étudiant avec les notes correspondantes.

```

CLASS Etudiant
    {Matricule : integer ;
      Nom : str i n g ;
      Prénom : string ;
      Date na i ss : date ;
      An étude : integer ;
      Statut : string;
Modules_suivis : Set STRUCT{mod:Module
                          Note: float; }
    }
    / *méthodes*/
Methods

```

```
{afficherO ;
  ajouter O;
  suppr i mer ( ) ;
  changer an étude ( );
  changer statut();
  changer note ( );
}
```

Modules _suivis est un attribut référence dont le domaine est celui de la classe Module.

b. La composition/agrégation :

Cette relation est exprimée en faisant toujours appel aux attributs référence. En effet, il suffit d'injecter dans les classes composantes un attribut référence dont le domaine est celui de la classe composite.

Exemple : Pour exprimer l'agrégation entre la classe Etudiant et la classe Section, il suffit de créer dans la classe Etudiant, un nouvel attribut référence appelé **Sect** dont le domaine est celui de la classe Section. On aura alors :

```
CLASS Etudiant
{Matricule : integer ;
  Nom:string ; Prénom:string ;
  Date_naiss :date ;
  An_étude :integer ;
  Statut : string;
  Modules_suivis : SET STRUCT
  {mod:Module ; Note :float;}
  Sect : Section ;}
/*methodes*/
Methods
  {afficher ( ) ;
    ajouter ( ) ;
    supprimer ( ) ;
    changer an étude ( ) ;
    changer statut ( ) ;
    changer note ( ) ;
    changer section ( ) ;
  }
```

c. L'héritage :

La relation d'héritage est exprimée en définissant d'abord la classe mère (super-classe), ensuite en spécifiant que la sous-classe est dérivée de cette super-classe de la façon suivante :

```
CLASS nom de:la sous-classe:nom de la super-classe
{ /* attributs de la sous-classe*/ }
  Methods
  { /*méthodes de la sous-classe*/ }
```

Exemple :

Essayons d'exprimer le lien d'héritage entre *Etudiant/Personne* et *enseignant/Personne* tel qu'il est spécifié sur le schéma conceptuel de la base de données concernant la gestion de la scolarité.

```
class   Personne
  {nom : string •
  prénom: List string;
  Dat nais : date;
  }
Methods
  {   afficher() ;   }
```

```
CLASS Enseignant : Personne
  {Num_Ens : string;

  grade : string;
  Ancienneté_pedag: iriteger;

  Modules_enseignés : SET Module ;

  /*méthodes*/
  Methods
  {afficher( ) ;
  ajouter ( ) ;
  supprimer ( ) ;
  Modifier ancienneté! ) ;
  changer grade ( ) ;
  changer module ( ) ;
  }
```

IV. Avantages et limites des SGBD objets

IV.1. Avantages

Les avantages du modèle objet sont essentiellement :

- Ceux de l'approche orientée objet : réutilisation, maintenabilité, ...
- Richesse du modèle de données
- Prise en compte de la dynamique grâce aux méthodes qui sont implantées dans les classes d'objets.
- Fusion entre le langage de manipulation de données et le langage programmation : les deux sont compatibles à un degré assez élevé.

IV.2. Limites

Les limites d'un modèle objet sont principalement :

- Les systèmes objet ne supportent pas en général, des contraintes d'intégrité déclaratives : de telles contraintes doivent être gérées par biais de code procédural (méthodes ou programme d'application).
- Il n'existe pas encore de consensus sur un standard pour un langage de manipulation de données déclaratif de type SQL : deux propositions rivales existent ODMG et SQL3.
- Les SGBD objet ne permettent pas la gestion des vues, des travaux sont encore en cours.

V. Conclusion

L'exposé Constitue une introduction aux bases de données objet, nous commençons d'abord par énoncer quelques limites du modèle relationnel afin de ressortir la nécessité de recourir à un modèle s'adaptant au mieux, exigences des nouvelles applications, de la est ne le modèle objet. Nous mettons en évidence, les deux approches sur lesquels sont bases le SGBD objets, à savoir l'approche «relationnel -objet» et l'approche «OO». nous présentons les concepts de base de l'approche objet, un modèle pour conception de BD objet.

Les SGBD objets (SGBDO) sont encore un domaine en évolution, de ce fait il n'y a pas de consensus sur le modèle de BD objet ou sur le langage de manipulation de données objets.

Glossaire

PL/SQL: Procedural Language/Structured Query Language

SET :Standard d'Echange et de Transfert

SGBD :Système de Gestion de Bases de Données

SGBDR: Système de Gestion de Bases de Données Relationnelles

SGBDRO :Système de Gestion de Bases de Données Relationnelles-Objet

SGBDOO: Système de Gestion de Bases de Données Orientes-Objet

BDO: Bases de Données Objet

SQL: Structured Query Language

STEP: STandard for Exchange of Product model data

Bibliographie

[01] M^r SELMOUNE Nazih & M^{elle} BOUKHEDOUMA Souad,

«Bases de données & SGBD Relationnels et Objet»

Edition PAGESBLEUES, octobre 2000.

[02] Chris J. DATE,

«Introduction aux Bases de données»

7^{eme} Edition Vuibert Informatique ,Décembre 2000.

[03] Serge Miranda,

«Architectures ,Modèles Relationnels et Objets, SQL3»

Edition 2002.

[04] Mourad EL HADJ MIMOUNE,

«contribution a la modélisations explicité et a la représentation des données de

composants industriels » . Thèse de doctorat ; Université de Poitiers , Juillet2004