Chapitre 1 : Introduction

Maîtrise informatique
F. Bannay

October 26, 2000

Contents

Bibliographie

Gardarin "Bases de données objet & relationnel"
Adiba, Collet "Objets et bases de données le SGBD O_2"
Soutou "Objet-relationnel sous Oracle 8"
/usr/local/O2/manuals/MAINMENU.pdf (sous vega)
http://doc.etud.univangers.fr/databases/Oracle8/server.804

1 Définition d'un SGBD

BD : années 70, grandes quantités de données fichiers (Fortran, Cobol, ...) Besoin de structurer les données et d'avoir plusieurs utilisateurs

Définition 1.1 *Une* Base de données *est une* organisation cohérente *de* données permanentes *accessibles à des* utilisateurs concurrents*.*

Un système de gestion de base de données (SGBD) doit permettre à l'utilisateur d'interagir avec la base de données, c'est-à-dire de pouvoir :

- définir
- consulter,
- mettre à jour

les données.

2 Les limites du modèle relationnel

Années 80, (2ème génération de SGBD (après les systèmes navigationnels)) Objectifs : optimisation de l'accès aux données (indépendance données/schémas, langage déclaratif d'interrogation, fondement théorique : algèbre relationnelle) et client/serveur
Limites du modèle relationnel : les données sont de type simple (attributs atomiques : 1ère FN) (Des nouvelles applications (CAO, bureautique...) nécessitent : structure de données complexe (hiérarchie, graphe de dépendance))
Limites pour le programmeur

- pb intégration des requêtes dans un langage de programmation : pb de conversions de types, perte de performances
- réutilisation, estensions des programmes

Besoins administrateur : modif des schémas, plusieurs concepteurs.
En résumé : les besoins actuels en BD

- persistance des données, gestion de la concurrence d'accès, tolérance aux pannes (SGBD classique)
- données complexes et liens entre objets
- indépendance données/programmes
- support théorique solide (comme l'AR)
- réutilisation.

3 La philosophie Objet

La structure de donnée est cachée seule l'interface (i.e. les propriétés externes de l'objet et les opérations que l'on peut faire dessus) est visible. (introduit dans les années 60 avec Simula, puis 70 les types abstraits, puis Smaltalk, puis Frames (Minsky 75) et réseaux sémantiques (Winston 84).
Actuellement, la technologie objet est très répandue : il est présent dans tous les outils IHM clients/serveur, dans 80% entreprises (conception objet + implémentation classique).

Définition 3.1 *Un* objet *est un triplet* $<$ *OID, classe, état* $>$*, l'OID est l'identifiant (object identifier) de l'objet, il est unique et invariant au cours du programme, contrairement à son état (valeur des attributs de l'objet) qui peut varier. Sa structure et la façon dont il varie (son comportement) sont définis par les attributs et les opérations de sa classe.*

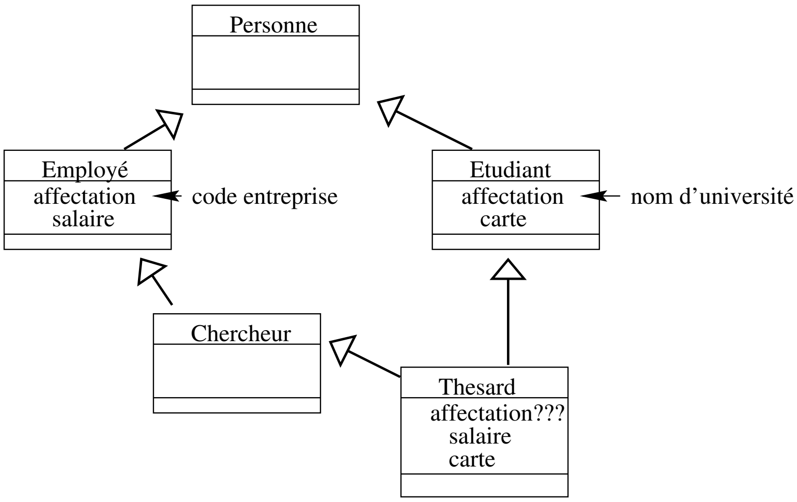
Définition 3.2 *Une* classe *est un type de données abstrait permettant de définir les propriétés d'un ensemble d'objets regroupées en deux catégories : les attributs et les opérations. Une classe fournit un mécanisme de création d'objet : l'instanciation qui permet de créer un nouvel identifiant et une structure pour un objet de cette classe. L'ensemble des instances d'une classe est appelée extension.*

Définition 3.3 *Un* attribut *est défini par son nom et son type, une opération est une fonction qui permet de modifier l'état d'un objet ou de renvoyer une valeur.*

Définition 3.4 *On appelle encapsulation, le fait que des propriétés de l'objet ne soient accessibles que par ses opérations visibles et que la façon dont les données sont stockées, leur structure soit cachée. Cela permet une indépendance des programmes par rapport à la représentation interne des données (puisque si l'on modifie la représentation interne d'un objet on modifiera les opérations de mise à jour de cet objet, et tous les programmes qui manipulaient ces objets n'auront pas à être modifiés car ils sont définis sur les opérations et pas sur les données elles mêmes).*

Définition 3.5 *L'héritage est un mécanisme de transmissions des propriétés d'une classe vers une sous-classe.*

L'héritage est simple si la propriété est héritée d'une seule sur-classe. Il est multiple lorsque la propriété est présente dans plusieurs sur-classes.



Définition 3.6 *On appelle* surcharge *le fait de redéfinir dans une sous-classe une propriété qui serait héritée (d'une sur-classe).*

Définition 3.7 *On appelle* polymorphisme *le fait de disposer d'opérations ayant un même nom mais des paramètres différents en nombre ou en types. On appelle liaison dynamique le mécanisme de sélection du code correspondant à l'opération concernée en fonction des paramètres.*

Ex : Si on dispose des classes animal avec l'attribut rang (qui renvoie le rang dans le règne animal de l'animal) et de la classe caractère avec l'attribut ascii (qui renvoie le code ascii du caractère). Dans la classe entier, on peut définir la fonction supérieur de différentes façons :

- supérieur (x : entier) {return(self > x)}
- supérieur(x : caractère) {return (self > x.ascii)}
- supérieur(x : animal) {return (self > x.rang) }

Avantages de la philosophie objet :

- la modularité permet l'extensibilité
- meilleures production, qualité, maintenance
- privilégie la conception
- héritage permet la réutilisation du code, redondance réduite
- UML : méthodologie de conception

Désaventages :

- implémentation des compilateurs objets est plus difficile : liaison dynamique, ramasse-miettes
- nécéssité de concepteurs de qualité

4 Définitions pour les Bases de données objet

La différence essentielle entre les objets manipulés dans la programmation objet et ceux des BDO est la *notion de persistance* : la durée de vie de l'objet dans les BDO ne doit pas être limité à celle du programme qui le crée. D'autre part, si les objets sont gérés par l'intermédiaire de fichiers cela donne des *programmes coûteux* et ne permet pas les *accès concurrents*. Alors que dans le domaine des BD, la gestion de l'accès aux données est bien rodée.

Définition 4.1 *Une base de données objet est une organisation cohérente* d'objets *persistants partagés par des utilisateurs concurrents.*

Le rôle d'un SGBDO est d'assurer

- l'intégrité des données malgré les pannes ainsi que leur permanence, de contrôler la concurrence des transactions,
- d'assurer l'indépendance entre les schémas et les objets physiques,
- de permettre l'exécution de requêtes (dans un langage déclaratif et pas procédural (on dit ce qu'on veut mais pas comment on va le chercher)),
- de permettre l'évolution des schémas (modifier un attribut : ajout, suppression, modif du nom ou du type, modifier une opération : ajout, suppression, modification du nom ou du code, modifier l'héritage : modifier graphe de classes (ajout, suppression d'une classe dans la liste des sur-classes d'une classe ou ajout, suppression d'une classe, modification du nom d'une classe).

Il existe 2 types de SGBDO :

- le SGBDOO, système basé sur un langage objet auquel on ajoute le moyen de gérer la persistance d'objet. ex : ObjectStore (C++), Gemstone (Smalltalk) et O_2 (C, C++)
- le SGBDRO, SGBD auquel on ajoute les possibilités de gérer des objets, d'hériter et éventuellement le polymorphisme et l'héritage multiple. ex : Oracle 8 (SQL3) et UniSQL