Cours(1MSIR): Les bases de données et l'objet

Dr. Seddiki Noureddine

Université Tahri Mohammed Bechar

14 novembre 2021

Sommaire

Introduction

Aujourd'hui, en programmation, il existe deux principaux modèles de représentation du monde :

- Le modèle fonctionnel
- Le modèle objet

Dans une approche fonctionnelle, les programmes sont composés d'une série de fonctions, qui assurent ensemble certains services. Il s'agit d'une approche logique, cohérente et intuitive de la programmation. Cette approche a un avantage certain appelé la factorisation des comportements (c'est à dire que pour créer une fonction d'une application, rien ne vous empêche d'utiliser un autre ensemble de fonctions (qui sont donc déjà écrites)).

Mais, l'approche fonctionnelle a aussi ses défauts, comme par exemple une maintenance complexe en cas d'évolution dŠune application (une simple mise à jour de l'application à un point donné peut impacter en cascade sur d'autres fonctions de lŠapplication). L'application sera alors retouchée dans sa globalité.

L'approche fonctionnelle n'est pas adaptée au développement d'applications qui évoluent sans cesse et dont la complexité croit continuellement (plusieurs dizaines de milliers de lignes de code).

Introduction

L'approche objet possède donc plusieurs avantages :

• La modélisation des objets de l'application (consiste à modéliser informatiquement un ensemble d'éléments d'une partie du monde réel en un ensemble d'entités informatiques. Ces entités informatiques sont appelées objet).

• La modularité (la programmation modulaire permet la réutilisation du code, via l'écriture de librairies).

• La réutilisabilité.

• L 'extensibilité (pour une meilleure productivité des développeurs et une plus grande qualité des applications).

• L'approche objet a été donc inventée pour faciliter l'évolution d'applications complexes. Elle apporte l'indépendance entre les programmes, les données et les procédures parce que les programmes peuvent partager les mêmes objets sans avoir à se connaître comme avec le mécanisme d'import/export.

Introduction

• L'approche objet a introduit indépendamment de tout langage de programmation à l'objet trois concepts de base : objet, classe et héritage entre classes. Les concepts objet et classe sont interdépendants, c'est-à-dire qu'un objet est une instance d'une classe et la classe décrit la structure et le comportement communs d'objets (ses instances).

Atouts du relationnel

Atouts du relationnel

- Très bon support théorique
- Simplicité des concepts basés sur de ensembles
- Modèle de données (externe, logique, physique)
- LDD et LMD

Atouts d'un SGBD(R)

- Persistance : les données survivent aux programmes.
- Taille : accès efficace aux grandes quantités de données : gestion des tampons, index, optimisation des requêtes.
- Performances sur le transactionnel (OLTP).
- Standard industrie :

a. SQL2 : indépendance des applications

b. Architecture client-serveur

c. Optimisation de requêtes

Limite du relationnel

- Modèle plat : pas de données complexes
  - ◇ Difficile d'exprimer des choses structurées
    - ✓ 1FN
  - ◇ nbre relations
  - ◇Difficile de naviguer dans la structure
    - ✓ Jointure
    - ✓ inadaptés aux application navigationnelles (réseaux, CAO)
  - ◇ Pas de types nouveaux
  - ◇ Volumineux ou/et multimédia
  - ◇ Non intégration des opérations
- SQL déclaratif, simple
  - ◇ Langage non complet
    - ✓ interfaçage (conixion) avec un langage de programmation

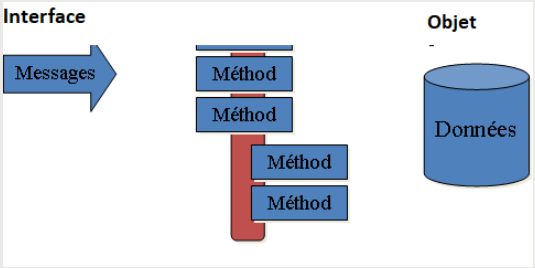
Définition d'un objet

Définition d'un objet

Entité informatique complète identifiable caractérisée pas son état et un ensemble de méthodes.

Associe traitements (méthodes) et données dans une même entité.

L'interface est le seul moyen d'accéder à l'objet, par l'envoi de messages.



Abstraction informatique d'une entité du monde réel caractérisée par une identité, un état et un comportement. Un objet est l'instance d'une classe, et une classe, est un type de données abstrait, caractérisé par des propriétés (ses attributs et ses méthodes) communes à des objets, qui permet de créer des objets possédant ces propriétés.

Définition d'un objet

Définition d'un objet

Objet = identité + état (attributs) + comportement (méthodes)

- Chaque objet a une identité indépendante de sa valeur.
- On peut mettre à jour la valeur sans changer l'identité.
- Identité et égalité (surface ou profondeur ?) sont deux concepts différents.
- Les objets peuvent être partagés.
- Les objets peuvent être périodiques.
- Les identificateurs ne sont pas accessibles aux utilisateurs

Les concepts de base d'un objet

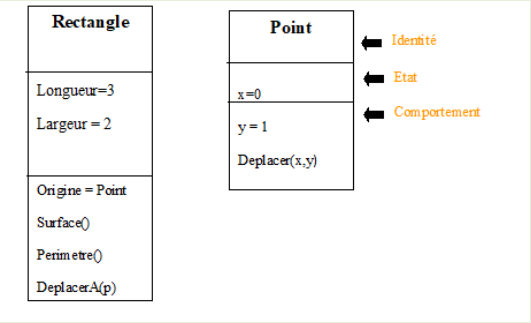
• **L'identité** : L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état. On construit généralement cette identité grâce à un identifiant découlant naturellement du problème (par exemple un produit pourra être repéré par un code, une voiture par un numéro de série, ...)

•**Les attributs** : Il s'agit des données caractérisant l'objet. Ce sont des variables stockant des informations d'état de l'objet

•**Les méthodes (appelées parfois fonctions membres)** : Les méthodes d'un objet caractérisent son comportement, c'est-à-dire l'ensemble des actions (appelées opérations) que l'objet est à même de réaliser. Ces opérations permettent de faire réagir l"objet aux sollicitations extérieures (ou d'agir sur les autres objets). De plus, les opérations sont étroitement liées aux attributs, car leurs actions peuvent dépendre des valeurs des attributs, ou bien les modifier.

Les concepts de base d'un objet

Exemple



Modèle objet pur

- Premier SGBD objet : 1983 (Gemstone)
- Approche : étendre le langage de programmation objet aux fonctions de SGBD
- Persistance
  - ◇Orthogonale au type
  - ◇ langages : C++, smalltalk, Java/OQL
- Produits
- O2, ObjectStore, Ontos, Objectivity, Jasmine, Versant
- Niches technologiques
  - ◇ réseau, CAO, SIG, Gestion de données Techniques
  - ◇ pas de transactionnel lourd

Règles d'or d'un SGBD Objet

Pour être qualifié d'objet, un SGBD doit respecter treize règles :

1. Données persistantes
   - Les données persistantes et non persistantes (temporaires) sont manipulées de la même façon par un programme.
2. Grande quantité de données
   - Techniques de regroupement physique, d'indexation, d'optimisation de requêtes et de gestion de cache.
3. Fiabilité des données
   - Transactions et privilèges
4. Partages de données
   - multi-utilisateur $\longrightarrow$ mécanismes de verrous
5. Facilité d'interrogation
   - Langage de requêtes.
   - Le résultat d'une requête n'est pas forcément un objet d'une classe existante.

Règles d'or d'un SGBD Objet

- Le langage offre en outre la puissance d'un langage de programmation Encapsulation
- Les données sont accessibles par des méthodes à différents degrés (visibilité)

1. Objets composites
   - Les structures de données à définir et à manipuler peuvent être complexes
2. Identificateur d'objet
   - Accès aux objets directement ou par des liens inter-objets via les OID
   - Objets partagés
3. Classes, types et méthodes
   - Types abstraits de données
   - Concepts de classes et de méthodes de l'approche objet
4. . Héritage

Règles d'or d'un SGBD Objet

1. Surcharge et liaison dynamique

   •Les méthodes peuvent être surchargées. La liaison dynamique est la capacité d'établir la correspondance entre le nom d'une méthode et son implantation lors de l'exécution et non lors de la compilation

2. Langage de programmation complet

   • Le système dispose d'un langage de programmation qui ne nécessite pas d'opérateur externe pour écrire une application

3. Extensibilité

   • Le système autorise l'ajout dynamique de nouveaux types abstraits de donnée, de nouvelles classes, de nouvelles méthodes, etc.

SGBDO ++

- Héritage multiple
- Gestion des versions
- CI
- Vues
- Evolution de schéma
- Persistance dynamique

Atouts de la technologie objet dans les bases de données

Atouts de la technologie objet dans les bases de données

- Origine dans les langages de programmation objet
- Objectif principal de l'approche objet : augmenter le niveau d'abstraction.
- Objet dans les SGBD : combiner les avantages
  - ◇ fichiers (simplicité et rapidité d'accès),
  - ◇bases hiérarchiques ou réseaux (navigation entre objets)
  - ◇bases relationnelles (langage de requêtes)
- Réduction des différences
  - ◇ entre le langage de programmation et la BD,
  - ◇ entre le monde à modéliser et le modèle de données d'autre part

Limites de la technologie objet dans les bases de données

Limites de la technologie objet dans les bases de données

- Manque de théorie dans la conception d'un schéma. Modèle de données
  - ◇ complexe
  - ◇ manque de philosophie

SGBD objet-relationnel

- Un SGBD objet-relationnel doit prendre en compte les quatre mécanismes suivants :
  - ◇ L'extension des types de données
  - ◇ Les objets complexes (en terme de structures de données)
  - ◇ L'héritage
  - ◇ Les systèmes de règles

Propriétés RICE

- Réutilisation
  - ◇ finalité du paradigme objet
  - ◇ héritage, généralité, composition, polymorphisme
- Identité
  - ◇ identifier un objet de manière unique
- Complexité
  - ◇ définition d'objets complexes et/ou fortement structurés
- Encapsulation
  - ◇ boîte noire avec des méthodes de manipulation

RICE/BDO/BDOR

	BDO	BDOR
Réutilisation	Héritage multiple	Héritage ? Polymorphisme (surcharge) et TEMPLATE(généricité)
Identité	OID	ROW ID
Complexité	Collections (SET, BAG, LIST, ARRAY) Pointeurs REF et INVERSE	ADT Collections (SET, LIST, MULTISET) Opérateurs VALUE, REF et DEREF
Encapsulation	Attributs SET_VALUE et GET_VALUE Méthodes	FUNCTION et PROCEDURE associées à ADT Niveau d'encapsulation (public, protected, private)

Les messages ou la communication entre objets(1)

Les messages ou la communication entre objets(1)

1. Par rapport aux messages on distingue trois catégories d'objets :

☐ les acteurs (qui envoient des messages)

☐ les serveurs (qui attendent des messages)

☐ les agents (qui savent envoyer et recevoir des messages)

2. Il y a 5 types de messages :

☐ **les constructeurs** : qui spécifient comment créer un objet

☐ **les destructeurs** : qui spécifient ce qui se passe quand un objet disparait.

☐**les sélecteurs** : qui renvoient tout ou partie de l'état d'un objet

☐ **les modificateurs** : qui changent l'état de l'objet .

☐**les itérateurs** : qui appliquent un traitement sur une liste dŠattributs.

3. On distingue aussi les messages asynchrones, synchrones, retardés et minutés.

Il est important de noter qu'on distingue deux types de comportement d'objet en réponse à un message :

Les messages ou la communication entre objets(1)

Les messages ou la communication entre objets(1)

☐ **statique** (si le traitement du message ne dépend pas d'autres paramètres) .

☐**dynamique** (si le traitement du message dépend de paramètres externes ou internes à l'objet).

La plupart des objets ont un comportement mixte :

Statiques pour certains messages, dynamiques pour d'autres.

6. Les classes

On appelle classe la structure d'un objet, c'est-à-dire la déclaration de l'ensemble des entités qui composeront un objet. Les objets de même nature ont en général la même structure et le même comportement. La classe factorise les caractéristiques communes de ces objets et permet de les classifier. Un objet est donc, une instanciation d'une classe, c'est la raison pour laquelle on pourra parler indifféremment d'objet ou dŠinstance (éventuellement d'occurrence). Toutes les instances d'une classe constituent l'extension de la classe.

La classe est un type abstrait de données caractérisée par des propriétés (attributs et opérations) communes à ses objets, et un mécanisme permettant de créer des objets ayant ces propriétés.

Les messages ou la communication entre objets(1)

classe = instanciation + attributs (variables d'instances)+ opérations

☐L'instanciation : L'objet possède une identité, qui permet de le distinguer des autres objets, indépendamment de son état. L'instanciation représente la relation entre un objet et sa classe d'appartenance qui a permis de le créer.

☐ Les attributs (appelés aussi variables d'instances) : Ils ont un nom et soit un type de base (simple ou construit) soit une classe (l'attribut référence un objet de la même ou une autre classe).

☐ Les opérations (appelées parfois méthodes) : Elles sont les opérations applicables à un objet de la classe. Elles peuvent modifier tout ou en partie l'état d'un objet et retourner des valeurs calculées à partir de cet état.

6.1 L'héritage entre classes

L'héritage est un principe propre à la programmation orientée objet, permettant de créer une nouvelle classe à partir dŠune classe existante. Le nom d'héritage" (ou parfois dérivation de classe) provient du fait que la classe dérivée (la classe nouvellement créée) contient les attributs et les méthodes de sa superclasse (la classe dont elle dérive).

Les messages ou la communication entre objets(1)

Les messages ou la communication entre objets(1)

L'intérêt majeur de l'héritage est de pouvoir définir de nouveaux attributs et de nouvelles méthodes pour la classe dérivée, qui viennent s'ajouter à ceux et celles héritées. Par ce moyen on crée une hiérarchie de classes de plus en plus spécialisées.

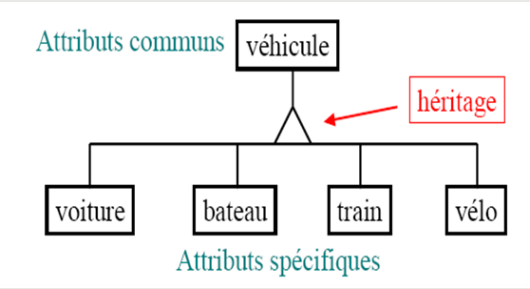
L'héritage est un mécanisme de transmission des propriétés d'une classe vers une sous-classe.

L'hiérarchie des classes ou l'association (relationship) est un concept essentiel pour la modélisation des données. On peut représenter sous forme de hiérarchie de classes, parfois appelée arborescence de classes, la relation de parenté qui existe entre les différentes classes.

L'hiérarchie commence par une classe générale appelée superclasse (classe de base, classe parent, classe ancêtre, classe mère ou classe père). Puis les classes dérivées (classe fille ou sous-classe) deviennent de plus en plus spécialisée.

Exemple d'héritage

Exemple d'héritage



Notion de polymorphisme

Notion de polymorphisme

Le nom de polymorphisme vient du grec et signifie qui peut prendre plusieurs formes. Cette caractéristique essentielle de la programmation orientée objet caractérise la possibilité de définir plusieurs fonctions de même nom mais possédant des paramètres différents (en nombre et/ou en type), si bien que la bonne fonction sera choisie en fonction de ses paramètres lors de l'appel. Le polymorphisme représente la faculté d'une opération de s'appliquer à des objets de classes différentes.

Le polymorphisme rend possible le choix automatique de la bonne méthode à adopter en fonction du type de donnée passée en paramètre (par exemple l'utilisation de la méthode addition() :)

- int addition(int, int) pourra retourner la somme de deux entiers.
- float addition(float, float) pourra retourner la somme de deux flottants.
- char addition(char, char) pourra définit au gré de lŠauteur la somme de deux caractères.

Le polymorphisme nécessite une liaison dynamique des messages à des méthodes. La liaison dynamique se produit lorsque la décision concernant la méthode à exécuter se prend lors de l'exécution d'un programme. Elle est nécessaire lors que :

Notion de polymorphisme

- Une variable réfère à un objet dont la classe dŠappartenance fait partie d'un arbre d'héritage
- Et lorsque plusieurs méthodes existent pour le même message dans l'arbre d'héritage

8. Encapsulation

L'encapsulation (ou encapsulage) en général est la notion de mettre une chose dans une autre. En imageant, on peut voir que cette chose est mise dans une capsule. En particulier, on peut retrouver ce terme dans plusieurs domaines en programmation, l'encapsulation de données est l'idée de cacher l'information.

En programmation, l'encapsulation désigne le principe de regrouper des données brutes avec un ensemble de routines permettant de les lire ou de les manipuler. Ce principe est souvent accompagné du masquage de ces données brutes afin de sŠassurer que l'utilisateur ne contourne pas l'interface qui lui est destinée. L'ensemble se considère alors comme une boîte noire ayant un comportement et des propriétés spécifiés.

L'encapsulation est un pilier de la programmation orientée objet, où chaque classe définit des méthodes ou des propriétés pour interagir avec les données membres, mais ce principe peut se rencontrer dans d'autres styles de programmation (par exemple la programmation modulaire).