UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE
FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE
LABORATOIRE DES SYSTÈMES INFORMATIQUES

BASES DE DONNEES AVANCEES

Pr. Z. Alimazighi

Dr. Kamel Boukhalfa

Boukhalk@gmail.com
http://boukhalfa.jimdo.com

PLAN DU COURS

- **Chapitre 1 :** Introduction aux SGBD orientés Objet
- **Chapitre 2 :** SGBD orientés Objet : les concepts
- **Chapitre 3 :** SGBD OO : Standards et systèmes
- **Chapitre 4 :** Introduction aux BD distribuées
- **Chapitre 5 :** SGBD distribués : Concepts avancés

INTRODUCTION AUX SGBD ORIENTÉS OBJET

Evolution des SGBD

- Depuis les années '60, avec :
  - les **BD hiérarchiques** (ex : IMS, Information Management Systems)
  - **BD réseaux** ou **CODASYL** (Committee on Data Systems and Languages; ex : IDS, Integrated Data Store)
- **BD relationnelles** (E.F. Codd, 1970)
- **BD objets** (ex : o2, Versant, 1990)
- **BD hybrides objets-relationnel** (ex : Oracle V8 en 1998)
- **BD natives XML** (ex: Tamino de Software AG, 2000)

Limites des SGBDR

- Le modèle relationnel est basé sur **la logique de premier ordre**
- Largement utilisé dans les SGBD commerciaux (SGBDR)
- Supporté par un langage puissant : **SQL**
- **Limites**
  - Piètre représentation des entités du monde réel
    - Conséquence de la normalisation
  - Structure de données trop homogène
    - Tuples avec un nombre fixe d'attributs
    - Pas de représentation d'entités complexes (composite)
  - Nombre assez limité d'opérations
  - Intégration de SQL à un langage de programmation : ensembliste vs procédural, dysfonctionnement (conversion de types)

Généralités - Historique

- Objectifs des SGBDOO
  - Réduire le dysfonctionnement entre langage de programmation et langage de base de données en offrant un unique langage de programmation de base de données
  - Supporter directement les objets arbitrairement complexes par un modèle objet
  - Partager le code réutilisable au sein du SGBD

Généralités - Historique

- Définition des SGBDOO
  - The Object-Oriented Database System Manifesto (1989)
    - Défini par des chercheurs
    - Ensemble de règles caractérisant un SGBDOO: obligatoires, facultatives, ouvertes

Généralités - Historique

- Définition des SGBDOO
  - ODMG 93
    - Défini par des éditeurs
    - Standard de portabilité des SGBDOO

INTRODUCTION

ORIGINE DE L ’APPROCHE OBJET

- DOMAINES DU GENIE LOGICIEL ET INTELLIGENCE ARTIFICIELLE
  - CONCEPTS D ’OBJET, DE CLASSES, ET D ’HERITAGE SONT APPARUS DANS LES LANGAGES SIMULA, SMALLTALK (67-72)
  - PLUS TARD IL YA EU EIFFEL ET DES EXTENSIONS OBJET DE C ET LISP (C++, OBJECTIVE C, COMMON LISP OBJECT SYSTEM).
  - CONCEPT D ’OBJET A ETE UTILISE COMME MOYEN DE REPRESENTATION DES CONNAISSANCES DANS L ’IA.
  - LE FORMALISME DES « FRAMES » PROPOSES PAR MINSKY en 1975 A ETE PRECURSEUR EN LA MATIERE.

INTRODUCTION

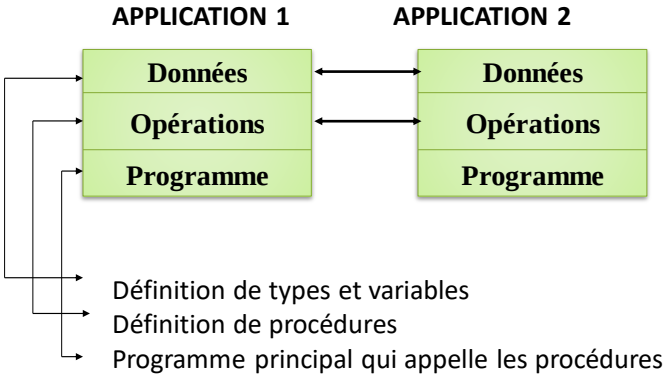
- **Première génération**
  - Fortran I
  - Algol
  - Expressions mathématiques
    - Forte dépendance vis-à-vis de la machine
    - Programme non portable
- **Deuxième génération**
  - Langages procéduraux: Fortran II, Algol 60, Cobol, Lisp
  - Structures de blocs/ sous-programmes (signature + corps)
  - Pas de protection d ’accès aux données car visibilité implicite

INTRODUCTION

- **Troisième génération**
  - **Langages modulaires : Algol 68, Pascal, SIMULA, ADA**
  - Module : signature : nom, liste d'importation, liste d 'exportation : masquage de l 'information mais structure figée à la compilation.
- **Quatrième génération**
  - **Langages orientés objet : Smaltalk, C++, Java ...**
  - Module : collection logique d'objets : les procédures sont rattachées aux objets

INTRODUCTION

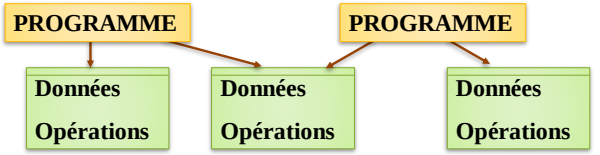
- Les 2ième et 3ième génération de langage obéissent au principe de la programmation structurée



INTRODUCTION

- Limites de la programmation structurée
  - lorsque données et procédures doivent être partagées par plusieurs programmes
  - lorsque les données évoluent

Programmation à objets



Concepts fondamentaux

- **Abstraction**
  - Identifier les aspects essentiels d'une entité et ignorer les propriétés d'importance secondaire ou sans importance
  - Se concentrer sur ce qui est l'objet avant de se concentrer sur la façon dont il sera implémenté.
- **Encapsulation**
  - L'objet contient à la fois la structure de données et les opérations qui servent à la manipuler.
- **Masquage d'information**
  - Séparer les aspects externes des d'un objet de ses détails internes qui sont masqués aux yeux du monde extérieur

Objectifs

- Réutilisation, modularité, couplage, etc.

- CAPACITE DU MODELE A DECRIRE NATURELLEMENT LES OBJETS DU MONDE REEL.
- ENSEMBLE DE CONCEPTS DESTINE A DECRIRE LES OBJETS ET LEURS ASSOCIATIONS.
- LES ENTITES DU MONDE REEL SONT REPRESENTEES A L 'AIDE D 'UN CONCEPT UNIQUE : **L 'OBJET.**
- TOUT OBJET EST DEFINI CONFORMEMENT A UN **TYPE ABSTRAIT.**
- UN TYPE ABSTRAIT EST SPECIFIE PAR
  - UNE STRUCTURE DE DONNEES (ENSEMBLE DES VALEURS POSSIBLES)
  - UN ENSEMBLE D 'OPERATIONS (METHODES) APPLICABLES A CES VALEURS.

- **APPLICATIONS CLASSIQUES**
- DUALITE DONNEES – TRAITEMENTS
- **APPROCHE OBJET**
- APPLICATION = ENSEMBLE D'OBJETS AYANT UN COMPORTEMENT ET ECHANGEANT DES MESSAGES

OBJET

- ABSTRAIT
- CONCRET

- ETAT
- IDENTITE
- COMPORTEMENT

- **ETAT :** ENSEMBLE DE VALEURS D 'ATTRIBUTS = CONTENU INFORMATIONNEL
- **IDENTIFIANT :** IDENTIFIE DE MANIERE UNIQUE UN OBJET ET N 'EST NI MODIFIABLE NI REUTILISABLE (INTERNE AU SYSTEME).
- **COMPORTEMENT:** DIFFERENTS MESSAGES AUXQUELS L 'OBJET PEUT REAGIR.

IDENTIFIANT D'UN OBJET (OID)

- PROPRIETE D 'UNICITE
- NON GERE PAR LE PROGRAMMEUR
- INITIALEMENT, L 'OID (OBJECT IDENTIFIER) ETAIT UN MECANISME DE DESIGNATION ET DE RESTITUTION : **OID PHYSIQUE.**
- LES MECANISMES D 'ACCES AGISSENT SUR UN RESEAU D 'IDENTIFIANTS : **OID LOGIQUE.**

- **IDENTIFIANT D 'UN OBJET**

  PROPRIETES D'IDENTITE ET D'EGALITE
- **IDENTITE DE DEUX OBJETS :** DEUX OBJETS SONT IDENTIQUES S 'ILS ONT MEME IDENTIFIANT.
- **EGALITE DE DEUX OBJETS :** DEUX OBJETS SONT EGAUX SI LEURS ETATS SONT EGAUX.

INTERET

- EXISTENCE DE L 'OBJET INDEPENDEMMENT DE SON ETAT.
- ON PEUT COMPRENDRE LA SEMANTIQUE DE MODIFICATION DE L 'OBJET COMME DES CHANGEMENTS D 'ETATS.
- DEUX OBJETS DIFFERENTS PEUVENT AVOIR LE MEME ETAT.

- **LE COMPORTEMENT**
  - IL REPRESENTE LA FAÇON DONT CELUI-CI REAGIT AUX MESSAGES QUI LUI SONT ENVOYES
  - COMPORTEMENT = CONTRAINTES + OPERATIONS
    - **CONTRAINTES :** ETATS POSSIBLES ET CHANGEMENTS D 'ETATS POSSIBLES DE L 'OBJET
    - **OPERATIONS :** MANIPULATION DES ETATS DE L 'OBJET = INTERFACE DE L 'OBJET, c-à-d L 'OBJET N 'EST ACCESSIBLE QU 'A TRAVERS SON INTERFACE.

- **TYPE D 'UN OBJET**
  - UN OBJET EST ATOMIQUE (OU PRIMITIF) SI SA VALEUR EST DEFINIE PAR UN TYPE ATOMIQUE (ENTIER, CHAINE DE CARACTERES, TEXTE NON STRUCTURE).
  - UN OBJET EST COMPLEXE S'IL FAIT REFERENCE A D'AUTRES OBJETS
- **Définition plus formelle:**
  - Un objet O est un triplet O = (Id, V, M) id: identifiant, V: valeur, M: comportement
  - O est atomique si V a un type atomique O= (Id, A1, A2, …,An)
  - O est complexe si V contient des identificateurs d'autres objets
- **EXEMPLES D 'OBJETS**

- OBJET ATOMIQUE

O1= (id1, 123), O2 = (id2, 234), O3 =(id3 , ' AHMED ') , O4 = (id4, 7)

- OBJET COMPLEXE

O5 = (id5, {MATRICULE : id1, NOM : id3}) ; O6 = (id6, {NUMERO: id2, PUISSANCE : id4})

O8 = (id8, set(id5, id6))

- **TYPE ABSTRAIT DE DONNEES (TAD)**
- **UN TYPE DE DONNEES ABSTRAIT PERMET DE DEFINIR DES OBJETS EN TERMES DE STRUCTURES DE DONNEES (ATTRIBUTS) ET DE COMPORTEMENT (METHODES).**

UN TAD EST ENTIEREMENT DEFINI PAR :

INTERFACE OU PARTIE PUBLIQUE

- SPECIFICATION QUI DECRIT LES OPERATIONS VISIBLES PERMETTANT LA MANIPULATION DES OBJETS APPARTENANT AU TYPE

PARTIE PRIVEE OU CACHEE

- REPRESENTATION DE LA STRUCTURE DE DONNEES

- IMPLEMENTATION QUI DECRIT LA REALISATION DES OPERATIONS


```
Package  TDA-PILE IS
Type ELEMENT IS NATURAL RANGE 0..100
Type PILE IS (Taille : Positive : 20) IS Private
Procedure DEPILER (P: IN PILE; ELEM : OUT ELEMENT)
Procedure EMPILER (P : IN PILE; ELEM : IN ELEMENT)
Function  PILE-VIDE (P : PILE) return Boolean ;
Function PILE-PLEINE (P: PILE) return Boolean ;
VIDE, PLEINE : exception ;
PRIVATE
Définition de la structure interne du type PILE (tableau, liste,..)
END TDA-PILE
Package Body TDA-PILE IS
….
END
```

Partie spécification publique

Partie spécification privée

Partie implémentation privée

- **CLASSES**

  UNE CLASSE REGROUPE UN ENSEMBLE D 'OBJETS AYANT MEME STRUCTURE ET MEME COMPORTEMENT (C-A-D DE MEME TYPE).

  UNE CLASSE EST DECRITE A LA FOIS AU NIVEAU CONCEPTUEL (TYPE) ET AU NIVEAU INTERNE (CORPS DES METHODES).

- **CLASSIFICATION D 'OBJETS :** *PROCESSUS DE CREATION DES CLASSES PAR REGROUPEMENT D 'OBJETS DE MEME NATURE.*

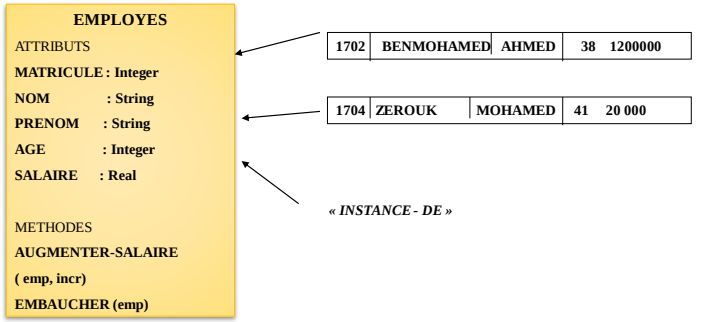
- **INSTANCIATION D 'UNE CLASSE :** *PROCESSUS DE CREATION D 'UN OU PLUSIEURS OBJETS A PARTIR D 'UNE DESCRIPTION DE CLASSE.*

UN OBJET EST INSTANCE D 'UNE OU PLUSIEURS CLASSES

Remarques

- Un objet ne peut exister individuellement dans le système, il est nécessairement instance d'au moins une classe
- Une classe n'est qu'une abstraction alors qu'un objet est une entité concrète du système
- Une classe possède nécessairement une structure ; elle est décrite par au moins un attribut autre que son nom.
- Une classe possède au moins deux méthodes : création et suppression
- Une classe peut être elle-même un objet: notion de méta-classe
- La différence entre classe et type : un type correspond à une définition conceptuelle (spécification de la structure et des opérations applicables) alors qu'une classe englobe les aspects spécification et réalisation (corps des méthodes) : définition du type en intention alors que la définition de la classe est en extension.

EXEMPLE DE DESCRIPTION DE CLASSE



- **ASSOCIATION :**

  RELATION ENTRE PLUSIEURS CLASSES, CARACTERISES PAR UN VERBE DECRIVANT CONCEPTUELLEMENT LES LIENS ENTRE LES OBJETS DE CES CLASSES.

- **CARACTERISTIQUES :**

1-1 : CHAQUE OBJET DE C1 EST CONNECTE AU PLUS A UN OBJET DE C2 ET RECIPROQUEMENT.

1-N : CHAQUE OBJET DE C1 EST CONNECTE A ZERO, UN OU PLUSIEURS OBJETS DE C2. CHAQUE OBJET DE C2 EST CONNECTE AU PLUS A UN OBJET DE C1.

M-N : CHAQUE OBJET DE C1 EST CONNECTE A PLUSIEURS OBJETS DE C2 ET CHAQUE OBJET DE C2 EST CONNECTE A PLUSIEURS OBJETS DE C1.

- **LE POLYMORPHISME ET LA LIAISON DYNAMIQUE**

Le polymorphisme est la faculté d 'une opération à pouvoir s 'appliquer à des objets de classes différentes.

- **SURCHARGE** : redéfinition d 'une opération avec un code différent

-**LIAISON DYNAMIQUE :** mécanisme de sélection de code d 'une opération à l 'exécution, en fonction de la classe d 'appartenance de l 'objet receveur

- **LA GENERICITE**

Le moyen de paramétrer une classe .Une classe générique est un modèle de classes.

Une classe générique a pour instances des classes

- **LA MODULARITE**

Le module est une construction séparée.

L 'utilisation de modules permet de contrôler la complexité de grosses applications.

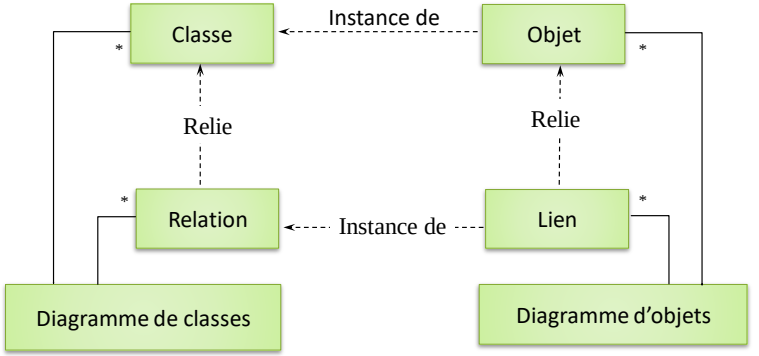
La modularité est la propriété d 'un système qui a été décomposé en ensemble de modules regroupant des abstractions logiquement reliées et faiblement couplés

DIAGRAMMES DE CLASSE ET D'ETATS-TRANSITIONS

Z. Alimazighi, Latifa Mahdaoui & K. Boukhalfa

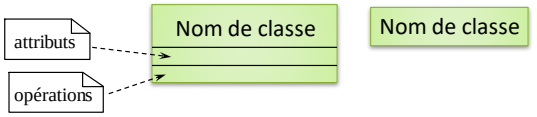
Diagrammes de classes et d'objets

Un diagramme de classes représente la structure du système en termes de classes et de relations entre ces classes; Un diagramme d'objets illustre les objets et leurs relations



Classe

Une classe est une description abstraite d'un ensemble d'objets ayant des propriétés similaires, un comportement commun, des relations communes avec d'autres objets et des sémantiques communes

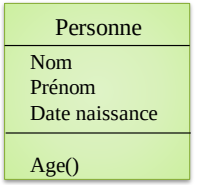


Exemples:

Personne

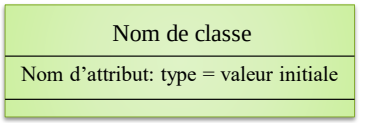
Poste de travail

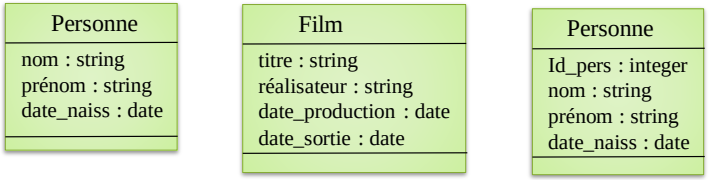
Département



Attributs de classe

Un attribut définit une propriété commune aux objets d'une classe

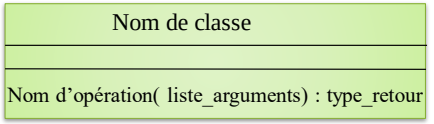


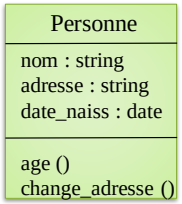


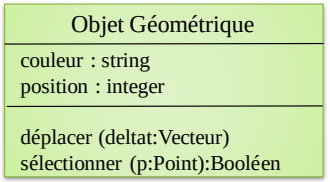
- Les noms d'attributs d' une classe sont uniques
- Chaque objet, instance d'une classe a sa propre identité indépendante des valeurs de ses attributs. L'identification d'un objet est donc facultative

Opérations de classe

Une opération définit une fonction appliquée à des objets d'une classe





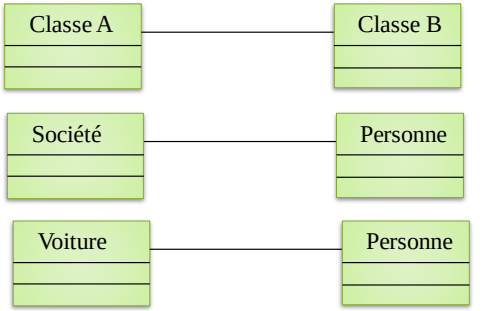


La syntaxe de description d'une opération est la suivante:

```
Nom_opération (Nom_Argument : Type_Argument =
Valeur_Par_Défaut, …): Type_Retourné
```

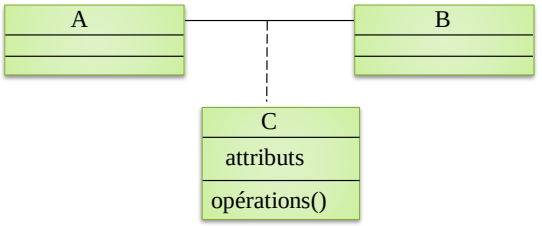

Association

Une association représente une classe de relations structurelles entre classes d'objets

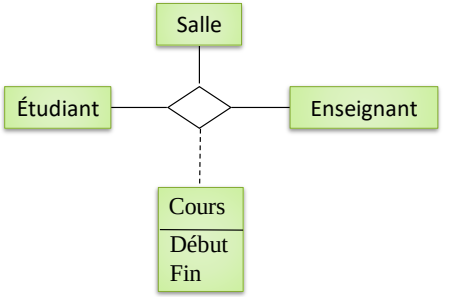


Classe association

- Une association peut être représentée (réifiée) par une classe appelée ***classe associative*** ou ***classe-association***. Utile par exemple, lorsuqe l'association a des attributs ou bien qu'on souhaite lui attacher des opérations. La notation utilise une ligne pointillée pour attacher une classe à une association



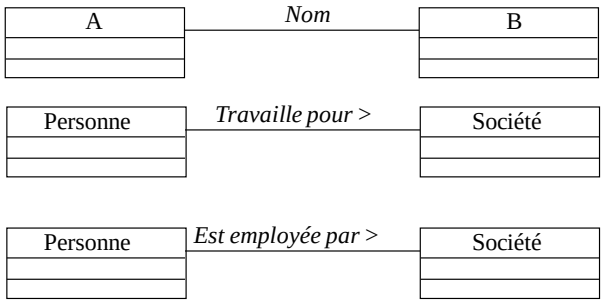
Association n-aire



L' association ternaire entre salle, étudiant et enseignant est réifiée comme une classe cours ayant deux attributs, Début et Fin

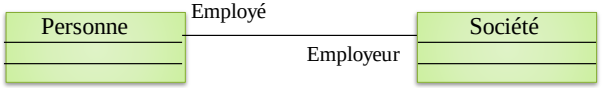
Nommage des associations

- Le nom d'une association apparaît en *Italique* au milieu de la ligne qui symbolise l'association
- L'usage recommande de nommer les associations par une forme verbale, soit active, soit passive



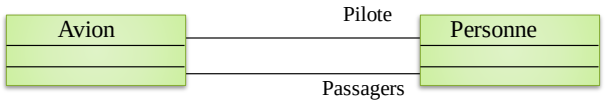
Nommage des rôles

- Chaque association binaire possède 2 rôles
- Le rôle décrit comment une classe intervient dans une association
- Le nommage des associations et le nommage des rôles ne sont pas exclusifs l'un de l'autre



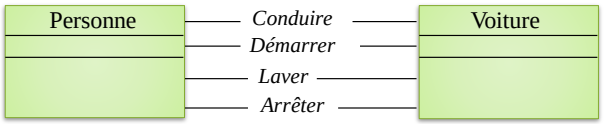
Nommage des rôles

- Le nommage des rôles prend tout son intérêt lorsque il y a plusieurs associations entre les deux mêmes classes.



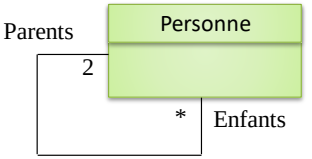
- La présence d'un grand nombre d'associations entre 2 classes est suspecte. Est souvent le signe d'une mauvaise modélisation

Exemple :



*

Association réflexive



Le nommage des rôles est indispensable à la clarté du diagramme

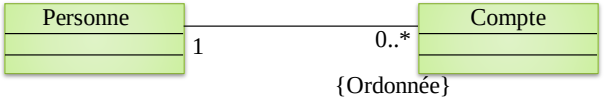
Multiplicité des associations

- La multiplicité est une information portée par le rôle, qui quantifie le nombre de fois où un objet participe à une instance de relation

1	Un et un seul
0 .. 1	Zéro ou un
M .. N	De M à N (entiers naturels)
*	De zéro à plusieurs
0 .. *	De zéro à plusieurs
1 .. *	De un à plusieurs

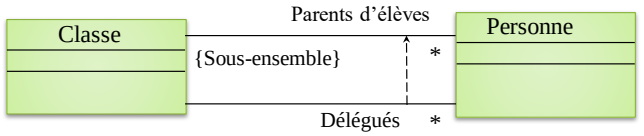
Contraintes sur les associations

- Les contraintes sont représentées sur les diagrammes par des expressions placées entre accolades
- La contrainte **{ordonnée}** placée sur le rôle définit une relation d'ordre sur les objets de la collection

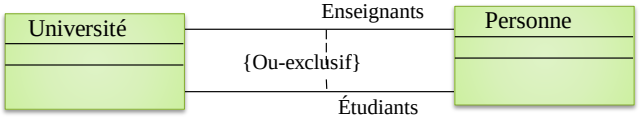


Contraintes sur les associations

- La contrainte **{Sous-ensemble}** indique qu'une collection est incluse dans une autre collection



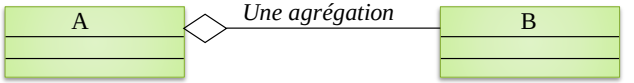
- La contrainte **{Ou-exclusif}** précise que, pour un objet donné, une seule association parmi un groupe d'associations est valide



Agrégation

Une agrégation représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre

- Les propriétés suivantes suggèrent une agrégation:
  - une classe B 'fait partie' d'une classe A
  - les valeurs d'attributs de la classe B se propagent dans les valeurs d'attributs de la classe A
  - une action sur la classe A implique une action sur une la classe B;
  - les objets de la classe B sont subordonnés aux objets de la classe A



Agrégation

- L'agrégation peut être multiple, comme l'association

En tant que 'propriétaire' une personne est un agrégat d'immeubles! Les immeubles dont elle est propriétaire font 'partie de' la description d'une personne

Composition

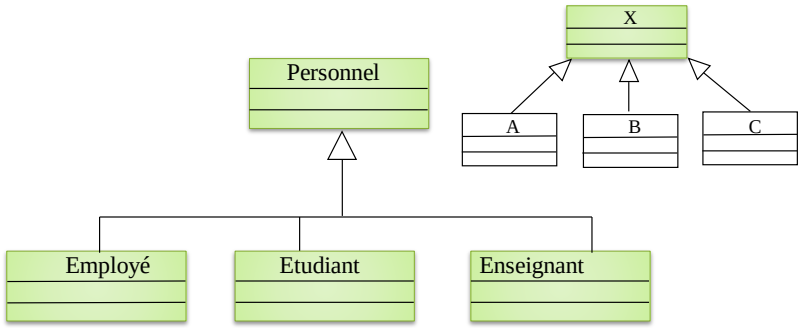
La composition est une forme particulière d'agrégation

Le composant est physiquement contenu dans l'agrégat

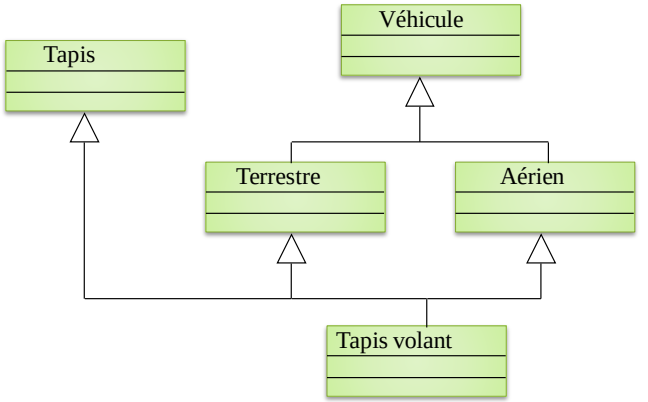
- La composition implique une contrainte sur la valeur de la multiplicité du côté de l'agrégat: elle ne peut prendre que les valeurs 0 ou 1

Généralisation

- Dans le cas des classes, la relation de généralisation signifie ***est un*** ou ***est une sorte de***

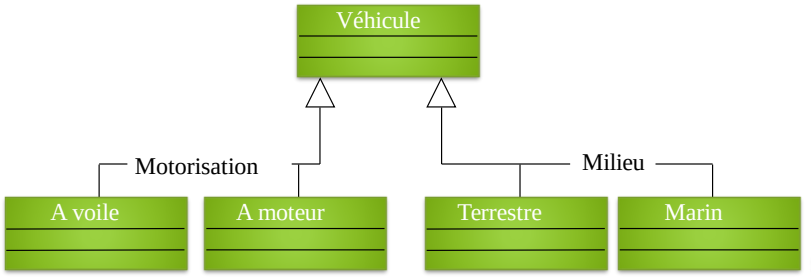


Généralisation multiple



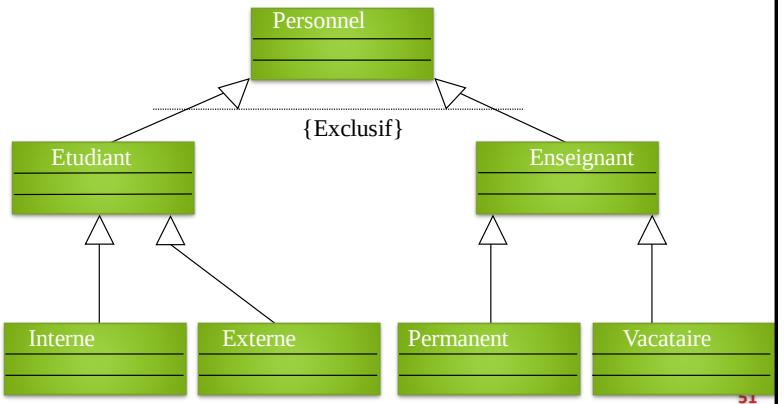
Généralisation

- Une classe peut être spécialisée selon plusieurs critères
- Différentes contraintes peuvent être appliquées aux relations de généralisation
- Par défaut, la généralisation symbolise une décomposition exclusive



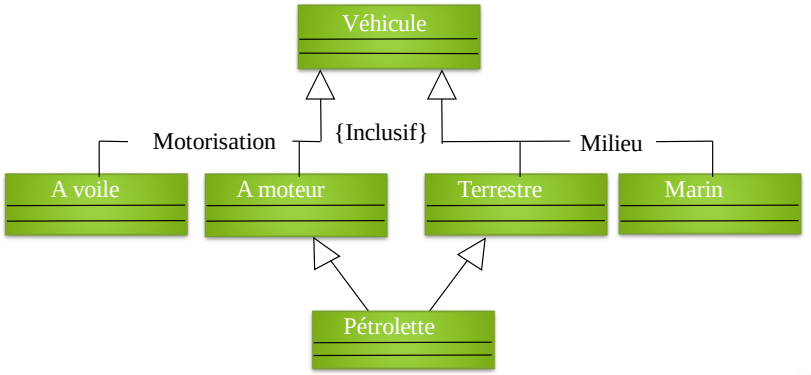
Contraintes de généralisation

- La contrainte **{Disjoint}** ou **{Exclusif}** indique la participation exclusive d'un objet à l'une des collections spécialisées



Contraintes de généralisation

- La contrainte **{Chevauchement}** ou **{Inclusif}** indique qu'un objet peut appartenir à plusieurs collections spécialisées .



Contraintes de généralisation

- La contrainte **{Complète}** indique que la généralisation est terminée et qu'il n'est pas possible de rajouter des sous-classes. Inversement, la contrainte **{Incomplète}** désigne une généralisation extensible

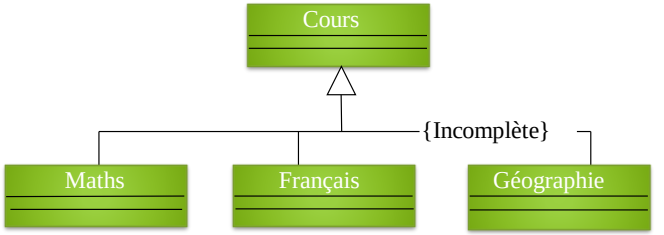


Diagramme d'objets

- Permet de visualiser la structure du système au niveau des instances
- Facilite la compréhension des structures de données complexes
- Trois possibilités de représentation :

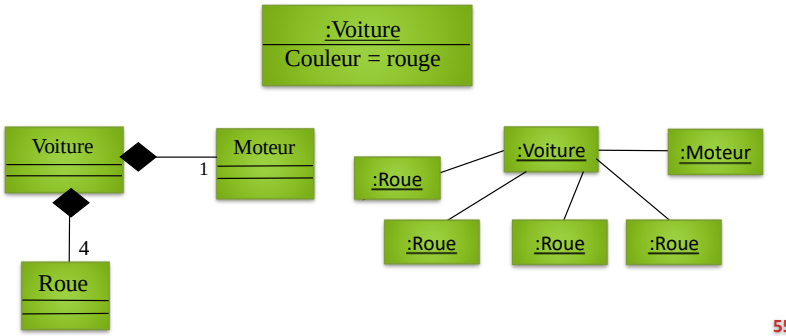
Nom de l'objet

Nom de l'objet : Classe

: Classe

Diagramme d'objets

- On peut faire apparaître des valeurs d'attributs dans un objet
- ainsi que les liens entre objets

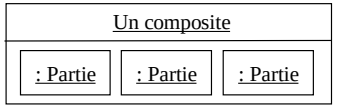


Liens entre les objets

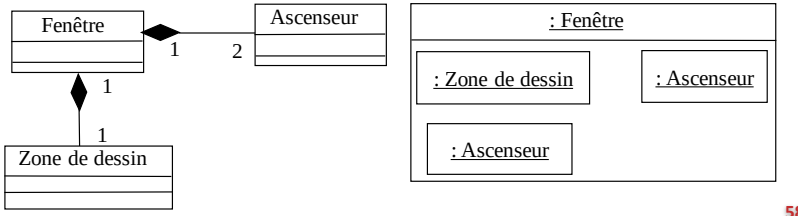
❒ Les liens instances des associations réflexives peuvent relier un objet à lui-même

Objet composite

❒ Les objets composés de sous-objets peuvent être visualisés



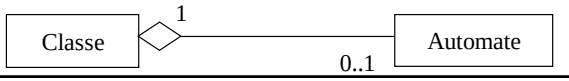
❒ Les objets composites sont instances de classes composites:



DIAGRAMMES D'ÉTATS-TRANSITIONS

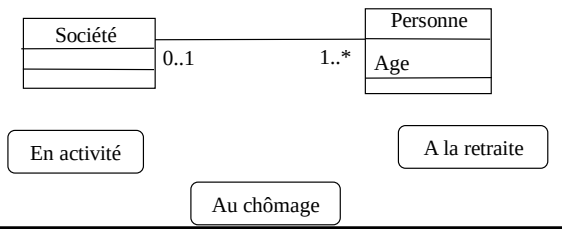
Diagramme d'états-transitions

- Décrit le comportement des objets d'une classe au moyen d'un automate d'états associé à la classe
- Le comportement est modélisé dans un graphe dont les nœuds sont les états possibles des objets de la classe et les arcs sont les transitions d'état à état. Chaque transition résulte de l'exécution d'une action et représente la réaction des objets aux événements qui surviennent
- Les objets qui n'ont pas un comportement réactif très important peuvent être considérés comme des objets qui restent toujours dans le même état: leurs classes ne possèdent alors pas d'automate



La notion d'état

- Un état est une étape dans le cycle de vie d'un objet durant lequel l'objet satisfait certaines conditions, réalise certaines actions ou attend certains évènements
- Chaque objet est à un moment donné, dans un état particulier
- Chaque état possède un nom qui l'identifie
- Un état est stable et durable

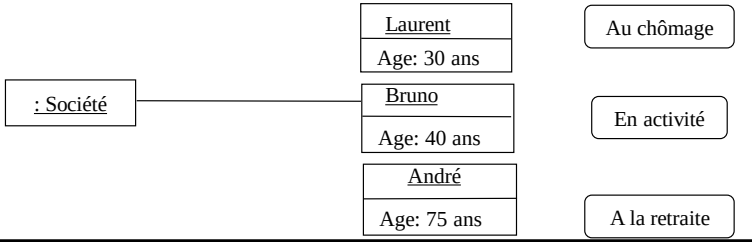


Notion d'état

- Un état est l'image de la conjonction instantanée des valeurs des attributs de l'objet, et de la présence ou non de ses liens à d'autres objets

Exemple: L'état d'une personne résulte de la conjonction suivante :

- âge de la personne,
- présence d'un lien vers une société

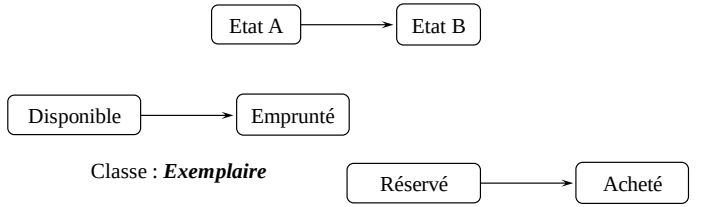


Notion d'état

- Chaque diagramme d'états-transition contient un ***état initial***
- Pour un niveau hiérarchique donné, il y a ***un et un seul état initial*** mais il est possible qu'il y ait ***plusieurs états finaux*** qui correspondent chacun à une fin de vie d'objet différente
- Il est également possible de n'avoir ***aucun état final*** (par exemple, un système qui ne s'arrête jamais)

Notion de transition

- Lorsque des événements se produisent, les objets changent d'état en respectant les règles décrites dans l'automate associé à leur classe
- Les diagrammes d'états-transitions sont des graphes dirigés
- Les états sont reliés par des connexions unidirectionnelles, appelées ***transitions***



Notion d'événement

- Un événement correspond à l'occurrence d'une situation donnée dans le domaine du problème
- Un événement est par nature une information instantanée qui doit être traitée sans plus attendre.
- L'événement est le déclencheur de la transition d' état à état
- Un objet, placé dans un état donné, attend l'occurrence d'un événement pour passer dans un autre état

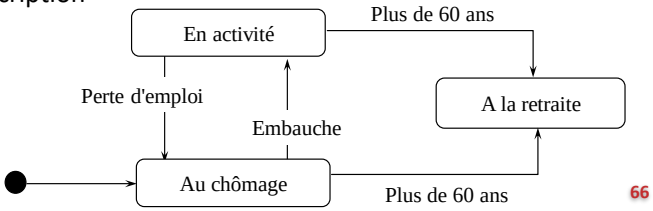
Classe : ***Exemplaire***

Notion d'événement

- La syntaxe générale d'un événement est la suivante:

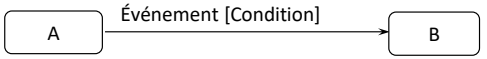
```
Nom de l'événement (Nom de paramètre : Type, …)
```

- La spécification complète d'un événement comprend:
  - le nom de l'événement
  - la liste des paramètres
  - l'objet expéditeur
  - l'objet destinataire
  - sa description



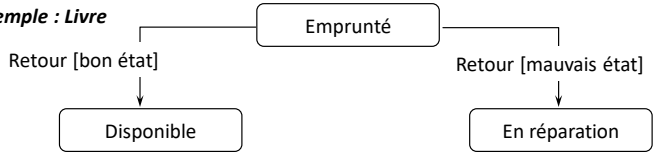
Notion de garde

- Une garde est une condition booléenne qui permet ou non le déclenchement d'une transition lors de l'occurrence d'un événement



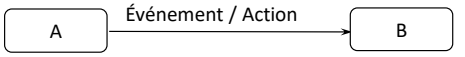
- Les gardes permettent de maintenir l'aspect déterministe d'un automate d'états finis. Lorsque l'événement a lieu, les gardes - qui doivent être mutuellement exclusives - sont évaluées et une transition est validée puis déclenchée

Exemple : Livre



Notions d'opération et d'action

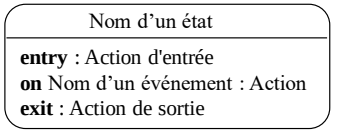
- Le lien entre les opérations définies dans la spécification d'une classe et les événements apparaissant dans le diagramme d'états-transitions de cette classe. Cela est assuré par le biais des *actions* et des *activités*
- Chaque transition peut avoir une action à exécuter lorsque la transition est déclenchée par un événement
- L'action est considérée comme instantanée et atomique



- L'action correspond à une des *opérations* déclarées dans la classe de l'objet destinataire de l'événement. L'action a accès aux paramètres de l'événement, ainsi qu'aux attributs de l'objet

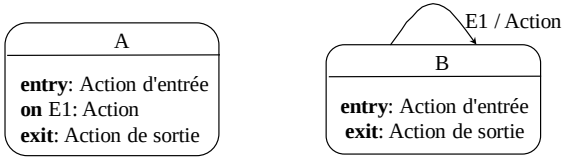
Actions dans un état

- Les états peuvent également contenir des actions; elles sont exécutées à l'entrée ou à la sortie de l'état ou lors de l'occurrence d'un événement pendant que l'objet est dans l'état :
  - L'action d'entrée (***entry:***) est exécutée de manière instantanée et atomique dès l'entrée dans l'état
  - L'action de sortie (***exit:***) est exécutée à la sortie de l'état
  - L'action sur événement interne (***on:***) est exécutée lors de l'occurrence d'un événement qui ne conduit pas un à un autre état



Opération, action, activité

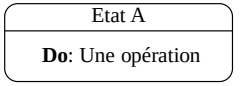
- Un événement interne n'entraîne pas l'exécution des actions de sortie et d'entrée, contrairement au déclenchement d'une transition réflexive



- Les actions correspondent à des opérations dont le temps d'exécution est négligeable. Une opération qui prend du temps correspond à un état plutôt qu'à une action

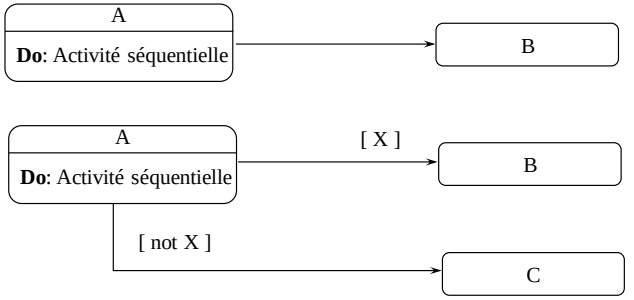
Opération et activité

- Une opération qui prend un temps non négligeable et qui est exécutée pendant que l'objet est dans un état donné est appelée une *activité*
- Le mot-clé ***do:*** indique une *activité*
- Contrairement aux actions, les activités peuvent être interrompues à tout moment, dès qu'une transition de sortie de l'état est déclenchée
  - *Activité cyclique:* elle ne s'arrête que lorsqu'une transition de sortie est déclenchée
  - *Activité séquentielle:* elle démarre à l'entrée dans l'état. Lorsqu'elle parvient à son terme, l'état peut être quitté si une des transitions est franchissable. C'est une transition automatique



Notion d'activité

- Lorsque l'activité se termine, les transitions automatiques, sans événements, mais éventuellement protégées par des gardes, sont déclenchées



Points d'exécution des opérations

6 façons d'associer une opération à une transition:

- l'action associée à la transition d'entrée (**Op1**)
- l'action d'entrée de l'état (**Op2**)
- l'activité dans l'état (**Op3**)
- l'action de sortie de l'état (**Op4**)
- l'action associée aux événements internes (**Op5**)
- l'action associée à la transition de sortie de l'état (**Op6**)

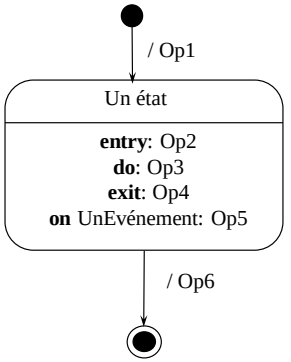
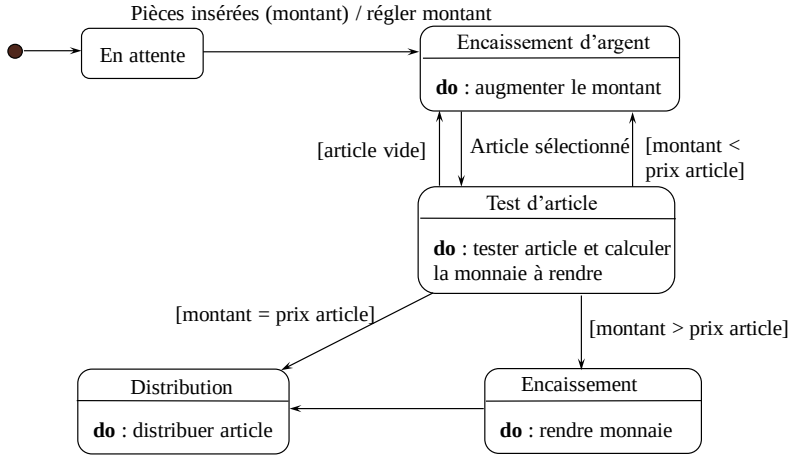
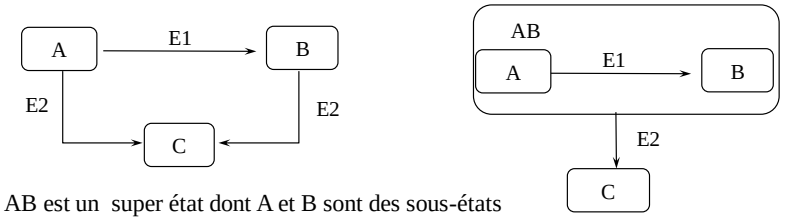


Diagramme E/T du distributeur automatique de boissons



Généralisation d'états

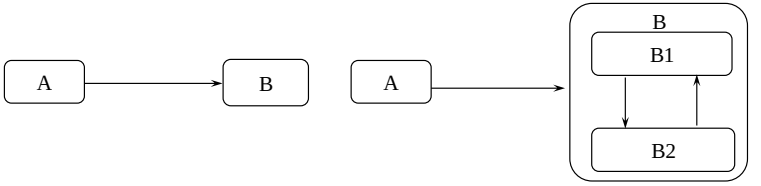
- Un état peut être décomposé en plusieurs sous-états disjoints; les sous-états héritent des caractéristiques de leur super-état
- *Décomposition disjonctive:* l'objet doit être dans un seul sous-état à la fois



AB est un super état dont A et B sont des sous-états
Cette généralisation permet de factoriser la transition E2

Généralisation d'états

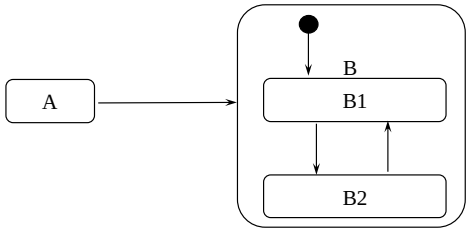
- Les transitions d'entrée ne sont pas héritées par tous les états, seul un état peut être cible de la transition



L'état B est divisé en deux sous-états B1 et B2. La transition d'entrée dans l'état B doit être reportée directement sur un des sous états.

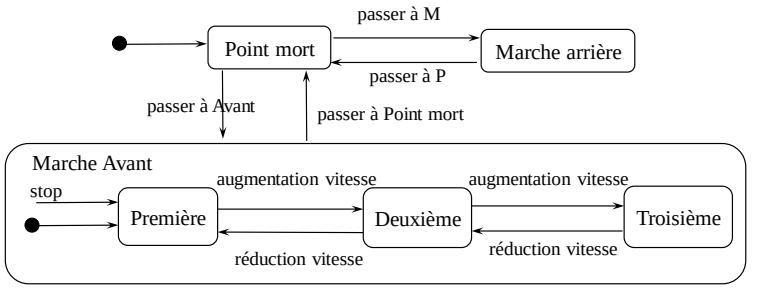
Généralisation d'états

- Il est préférable de limiter les liens entre niveaux hiérarchiques d'un automate en définissant systématiquement en état initial pour chaque niveau



Généralisation d'états

Exemple: Boîte automatique de transmission



La généralisation d'état permet de mettre en facteur les deux transitions 'passer à Avant' et 'passer à Point mort'

Généralités - Aspects Base de Données

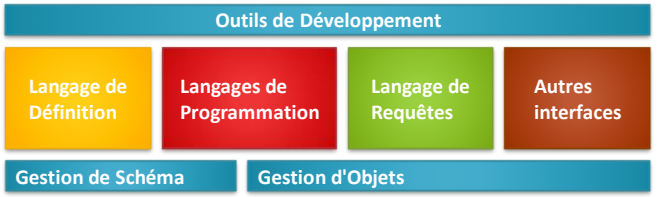
- un SGBDOO est un SGBD
  - Persistance
  - Gestion du disque
  - Partage des données (multi-utilisateurs)
  - Fiabilité des données
  - Sécurité des données
  - Interrogation ad hoc

Généralités - Aspects Orientés Objet

- Un SGBDOO est Orienté Objet
  - Objets complexes
  - Identité d'objet
  - Encapsulation, Classes
  - Héritage
  - Surcharge
  - Valeurs, Types
  - Polymorphisme

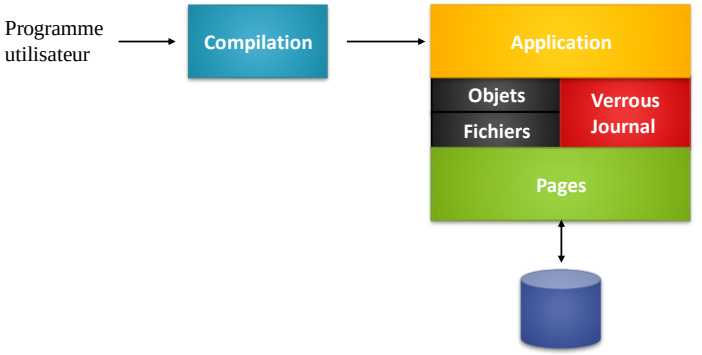
Généralités - Architecture

- Architecture Fonctionnelle



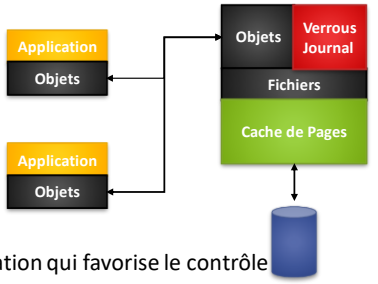
Généralités - Architecture

- Architecture Opérationnelle



Généralités - Architecture

- **Architecture Client/Serveur**
  - Serveur d'objets
  - Unité de transfert
    - Un objet ou groupe d'objets
  - Client léger
  - Fonctions BD sur le serveur Centralisation qui favorise le contrôle
    - d'intégrité
    - Sécurité
  - Interface Client / Serveur
    - Créer ou détruire un objet
    - Lire ou écrire un objet
    - Envoyer un message à un objet



Généralités - Architecture

- Architecture Client/Serveur
  - Multi-serveurs