I.U.T. LUMIÈRE

UNIVERSITÉ LUMIÈRE LYON 2

Bases de Données Avancées – Module B

IUT Lumière, License CE-STAT
2006-2007
Pierre Parrend

# Conception de Bases de Données Avec UML

Plan du Cours

## Table of Contents

UML et la conception de Bases de Données....2
Les Modèles de conception de Base de Données....2
UML et Modèle Entité-Relation....2
Classes et Schémas relationnels....2
Les Associations....3
Modèles UML Spécifiques....3
Les Design Patterns....3
Associations....3
Clés secondaires....4
Répétition d'attributs....4
Attributs multivalués....5
Domaines....5
L'héritage....5
Aggrégation....6
Associations récursives....6

# I. UML et la conception de Bases de Données

## A. Les Modèles de conception de Base de Données

Les principaux modèles de représentation de Base de Données sont les suivants:

- ***Modèles UML*** (Unified Modeling Language, Langage de Modèles Unifié)
Utilisé pour la modèlisation des grand systèmes d'information basés sur la Programmation orientée Objects. Les Objects dont les attributs doivent être gardés en mémoire sont stockés dans une Base de Données. C'est ce qu'on appelle la persistence.

- ***Modèles Entité relation***
Modèle le plus couramment utilisé. Il est souvent représenté graphique par un MCD (Modèle Conceptuel de Données)

- ***Modèle relationnel***
Modèle formel datant du début des années 197O, basé sur la théorie des ensembles. Il contient des informations qui ne sont pas exprimées en UML (charactérisation des clés, des contraintes sur les attributs, etc.)

- ***Algèbre relationnelle***
Langage formel de manipulation d'objets du modèle relationnel

- ***Modèles par tables*** (informel)

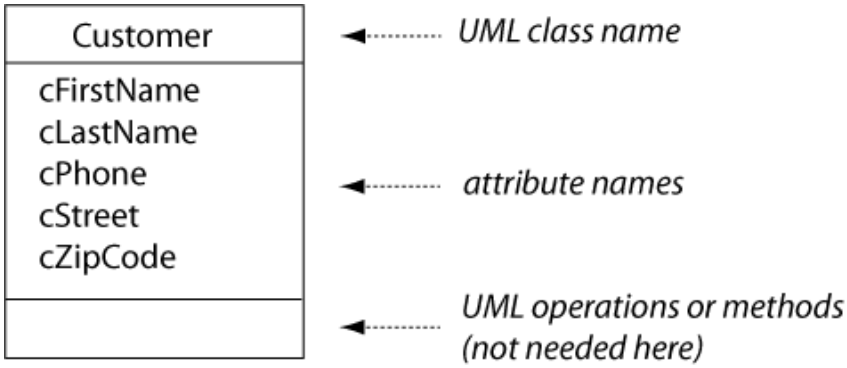
- ***Modèles SQL*** (programmatique)
Supporte la définition d'ensemble de données et leur manipulation.

## B. UML et Modèle Entité-Relation

| UML | Modèle ER |
|---|---|
| Classe | Entité |
| Association | Relations |
| Type et dénomination des champs | Type et dénomination des champs |
| Attributs naturels seulement | Attributs naturels + attributs de conception (Identifiants, etc.) |
| Clés non représentées | Charactérisation des clés |
| Permet de générer des Objects Java; Persistence transparente possible | Gestion de la l'accès aux données par le concepteur |

### C. Classes et Schémas relationnels

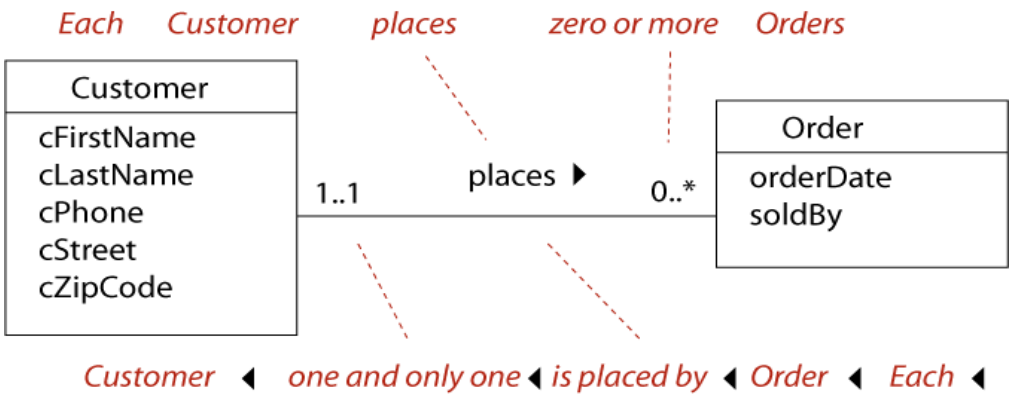
Un exemple de classe:




Le schéma relationnel correspondant:


### D. Les Associations

Exemple d'association en UML:



## II. Modèles UML Spécifiques

### A. Les Design Patterns

Un Design Pattern ou `Patron de Conception', est un modèle exprimant la relation existant entre plusieurs classes pour résoudre un problème particulier.

### B. Associations

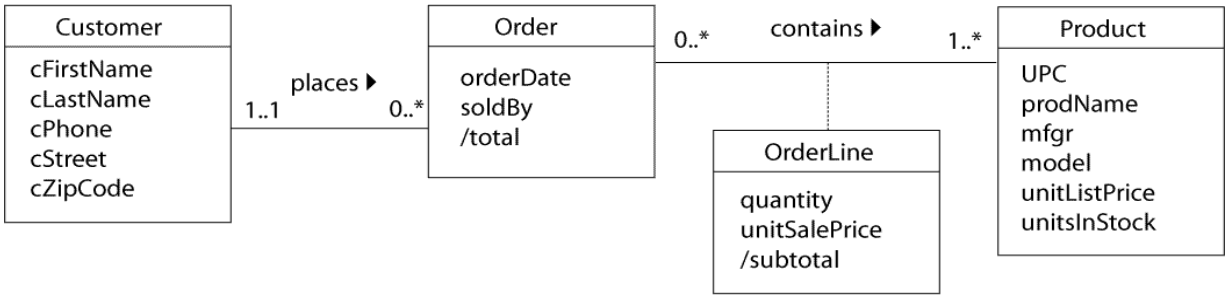
On identifie 3 grands types d'associations:
- 1-to-1

un object A est associé à un objet B
- 1-to-many

un object A est associé à plusieurs objets B, et un objet B donné n'est associé qu'à un seul A
B contient un attribut 'valeur de A associée à B'
- many-to-many

Chaque objet A peut être associé à plusieurs objets B, et chaque objet B peut être associé à plusieurs objets A.
Un objet spécifique doit être réaliser pour représenter l'association. Il contient pour chaque A les valeurs de B associées.

Example d'association `many-to-many':




### C. Clés secondaires

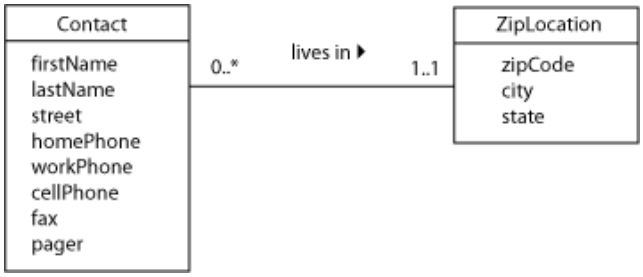
Une clé secondaire est un attribut C qui peut être associé de manière unique à un autre attribut D dans une classe donnée, quand aucun des attributs C et D ne font partie de la clé primaire.

Une classe qui contient une clé secondaire est mal conçue. Il faut créer une autre classe spécifique pour stocker les attributs C et D (voir le cours sur la normalisation).

### D. Répétition d'attributs

Il peut être nécessaire de préciser les propriétés de certains attributs.
Par exemple, un numéro de téléphone peut être celui d'un portable, d'un téléphone personnel, professionnel, etc.

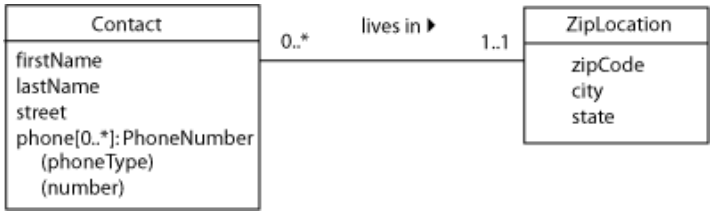
Exemple:




Problèmes avec ce modèle:
- grand nombre de champs non renseignés
- nécessité de modification s'il l'on ajoute un nouveau type de numéro (ex.: numéro Skype)

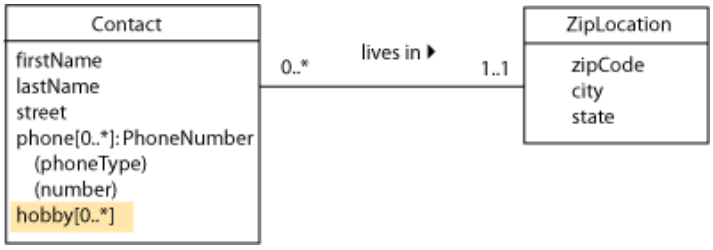
La solution consiste à créer un Classe spécifique, qui contient le nom de la personne, ainsi que le numéro de téléphone et le type de celui-ci.
De tels attributs sont appelés 'attributs répétés', ou 'entités faibles'.
Ils n'ont pas d'intérêt sans les informations principales (ici: les coordonnées d'une personne).

Exemple de Modèle UML avec des entités faibles:




E. Attributs multivalués

Un même attribut peut prendre plusieurs valeurs différentes simultanément pour une même occurrence d'une Table de Données. C'est cec qu'on appelle un '**Attribut Multivalué'.**
Exemple: Hobbies d'une personne.




Solution: créer une Table de Données secondaire qui contient une référence sur la personne concernée, et un hobby associé. Chaque personne peut apparaître plusieurs fois dans la table.

F. Domaines

Un **domaine** est l'ensemble des valeurs que peut prendre un attribut donné.
Il est défini par des `**règle de validation**', dont l'objectif est de renforcer le contrôle de l'**intégrité des données**.

Exemple: le poids d'une personne ne peut pas être négatif.

**Les domaines énumérés**: Un attribut peut prendre des valeurs données. Par exemple, le nom d'un enseignant en CE-Stat est limité à l'ensemble des enseignants du l'IUT Lumière.
Solution: Une Table de Données associée contient les valeurs possibles pour un attribut, en l'occurence les noms des enseignants.

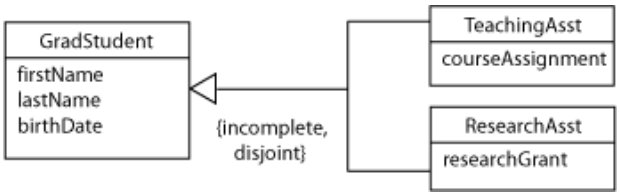
### G. L'héritage

Comme en Programmation Objet, il peut être nécessaire dans le cadre de la conception de Bases de Données de spécialiser des classes (ou des Tables de Données) existantes, ou de les généraliser.

**La spécialisation**

La spécialisation en UML se fait par **héritage**. Il s'agit de créer des classes filles, qui contiennent l'ensemble des attributs de la classe mère, plus des attributs spécifiques.

Exemple:




*Les contraintes de spécialisation*
Une spécialisation peut avoir les propriétés suivantes:
- être **incomplète**, ou **partielle**,

une partie seulement des individus parents sont spécialisés (ex.: certains étudiants peuvent être assistant d'enseignement, mais tous ne le sont pas forcément)
- être **complète**

tous les individus parents sont spécialisés (ex.: tous les animaux appartiennent à une espèce donnée)
- être **disjointe**, ou **exclusive**

un individu de la classe parent ne peut être membre que d'une seulement classe d'enfants (ex.: un animal est SOIT un chien, SOIT un chat, pas les deux)
- être **non exclusive**

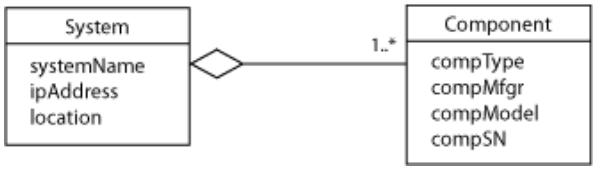
un individu de la classe parent peut être membre de plusieurs classes d'enfant (une personne peut à la fois être un étudiant de CE-Stat et un salarié)

**La généralisation**

La généralisation est l'opération inverse de l'héritage. Il s'agit s'abstraire des informations pertinentes dans une **super-classe** moins détaillée que la classe d'origine.

### H. Aggrégation

Une classe peut contenir des élements définis par une autre classe. Cette relation s'appelle l'**aggrégation**.
Une classe peut contenir des aggrégats. Si la classe principale disparait, les aggrégats ne sont pas modifiés.



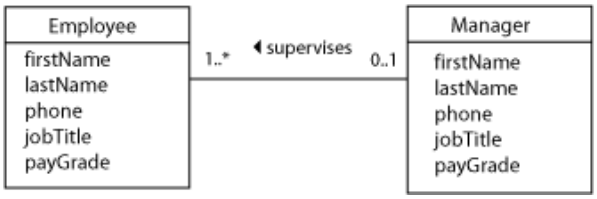

Un forme plus forte de relation entre deux classes est la **composition**: la destruction de la classe principale (ex.: System) a pour conséquencec la destruction de ses **composants**.

La présence d'un aggrégation ou d'une comparaison n'as pas de conséquence directe sur le format d'une Base de Données (pas plus que sur celui d'un programme Java). Par contre, elle indique que la gestion des occurrences doit être gérée de manière cohérente.

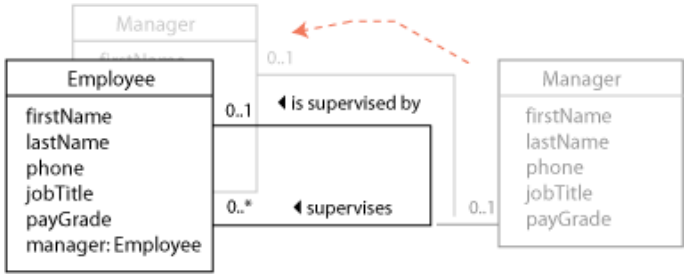
## I. Associations récursives

Une association récursive connecte une Classe (correspondant à un rôle particulier) à elle-même (correspondant à un autre rôle).

Modèle Incorrect:




Solution:




Exemple: relation entre un manager et ses employés. Le manager est lui-même un employé de la même entreprise.