

Part 3

CONCEPTION DE BASES DE DONNÉES RELATIONNELLES

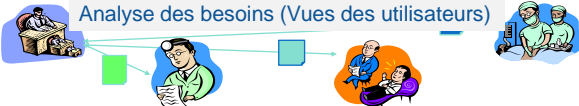

Objectifs de la modélisation / conception

- Permettre une meilleure compréhension
 - Le monde réel est trop complexes
 - Abstraction des aspects cruciaux du problème
 - Omission des détails
- Permettre une conception progressive
 - Abstractions et raffinements successifs
 - Possibilité de prototypage rapide
 - Découpage en modules ou packages
 - Génération des structures de données (et de traitements)

Les niveaux de conception d'une BD

- On a coutume de distinguer trois niveaux de représentation ou d'abstraction pour les bases de données et les systèmes d'information de manière plus générale :
 - le **niveau externe** (utilisateur)
 - le **niveau conceptuel** (concepteur, administrateur)
 - le **niveau interne** (stockage)
- Les trois niveaux et les modèles qui en découlent offrent un cadre rigoureux de description des données pour les concepteurs d'une BD.

Les niveaux de conception d'une BD

Niveau externe		 <p>Analyse des besoins (Vues des utilisateurs)</p>					
Niveau conceptuel		<ul style="list-style-type: none">▪ Indépendant du modèle de données▪ Indépendant du SGBD	<p>Description graphique des données et leurs liens</p> 				
Niveau interne	Modèle logique	<ul style="list-style-type: none">▪ Dépendant du modèle de données▪ Indépendant du SGBD	<p>Description de la structure de la BD</p> <table><tr><td>Relationnel</td><td>Objet</td><td>XML</td></tr></table>		Relationnel	Objet	XML
	Relationnel	Objet	XML				
Modèle physique	<ul style="list-style-type: none">▪ Dépendant du modèle de données▪ Dépendant du SGBD	<ul style="list-style-type: none">▪ Organisation physique des données▪ Structures de stockage des données▪ Structures accélératrices (index)					

Analyse des besoins

Analyse des besoins

- Avant la modélisation conceptuelle des données il faut faire une analyse des besoins.
- Analyse des documents représentatifs des données que l'on souhaite modéliser.
 - Documents papier
 - Fichiers
 - Compte-rendu d'entretien oral
- Liste complète des données à représenter dans la base
- Liste des besoins fonctionnels connus

Exemple

- On souhaite gérer des étudiants qui suivent différents enseignements d'un diplôme.
- On dispose de :
 - La liste des étudiants avec leurs données personnelles
 - Les bulletins de notes des étudiants
 - La liste des enseignants avec pour chacun la matière enseignée
- Règles de gestion :
 - Un étudiant a 1 note par matière
 - Un enseignant enseigne 1 seule matière

Dictionnaire des données

- Extraire les informations élémentaires
 - Attributs ou champs du dictionnaire des données
- Pour chaque attribut on précisera :
 - Nom
 - Descriptif
 - Type de donnée
 - Contraintes d'intégrité
 - Règle de calcul

Modèle conceptuel

La modélisation conceptuelle de la BD

- Isoler les concepts fondamentaux
 - Que vont représenter les données de la BD ?
 - Découvrir les concepts élémentaires du monde réel
 - Décrire les concepts agrégés et les sous-concepts
- Le schéma conceptuel exprime une **vue abstraite de la base de données**.
- Cette vue est représentée de manière graphique – on parle de **diagramme**, de **schéma** et de **modèle**.
 - Compréhension visuelle et non seulement intellectuelle

Avantages de la modélisation conceptuelle

- L'attention est portée sur les applications
- Indépendante des technologies
 - Portabilité
 - Longévité
- Orientée utilisateur
 - Compréhensibilité
 - Support du dialogue concepteurs / utilisateurs
 - Permet la collaboration et la validation par les utilisateurs
- Spécifications formelles, non ambiguës
- Facilite les échanges d'informations entre SGBD différents

Etapes de l'élaboration modèle conceptuel

- Analyse du monde réel
 - Identification des phénomènes à représenter dans la BD
- Représentation à l'aide des concepts du modèle
 - contenu
 - structure
 - règles
 - dynamique
- Représentation en général
 - Partielle
 - Subjective

Méthodes

■ Cartésienne

- ❑ 1ère génération basée sur des arbres fonctionnels
- ❑ décompose et résout de manière linéaire et analytique
- ❑ Descartes, Newton, PERT, ...

■ Systémiques

- ❑ 2ème génération basée sur entité-association
- ❑ prend en compte l'ensemble du système et ses interactions.
- ❑ **Merise**, Axial, SSADM, ...

■ Orientées-Objets

- ❑ 3ème génération basée sur les objets
- ❑ modélise des objets avec des données et des comportements, favorisant la réutilisation et l'abstraction.
- ❑ **UML**, OMT, RUP, ...

Méthodes

- Dans le cadre de ce cours nous allons se concentré sur les modèles pour les donnée en utilisant:
 - ❑ Méthode systémique (type **MERISE**)
 - ❑ Méthode orientée objet (type **UML**)
- **Nous réaliserons un face à face entre le modèle conceptuel des données de Merise et le diagramme de classes UML.**

Merise

- La méthode Merise est une évolution ou plus précisément une standardisation du **modèle entité-association**.
- Merise est une méthode d'analyse, de conception et de réalisation de systèmes d'informations.
- Cette méthode préconise plusieurs étapes pour passer d'un système d'informations manuel à un système d'informations automatisé.

Merise

- Dans la méthode **Merise** (**M**éthode d'**E**tude et de **R**éalisation Informatique pour les **S**ystèmes d'**E**ntreprises), le modèle conceptuel correspond au modèle entité-association celui-ci s'appelle le **Modèle Conceptuel de Données** : MCD.
- La méthode Merise propose d'autres modèles tels que :
 - Les **Modèles de Traitement** (MCT, MOT) qui permettent de modéliser les événements, les opérations et les processus.
 - Les **Modèles de Communication** (MCC) qui permettent de modéliser les interactions.
 - Ces aspects de la méthode ne seront pas développés ici car ils sont plus orientés application que base de données.

UML

- **UML** (Unified Modeling Language) est un langage de modélisation graphique et textuel, standardisé par l'OMG (Object Management Group).
- UML permet de construire plusieurs modèles d'un système :
 - certains montrent le système du point de vue des utilisateurs,
 - d'autres montrent sa structure interne,
 - d'autres encore en donnent une vision globale ou détaillée.
- Les modèles se complètent et peuvent être assemblés.
- Ils sont élaborés tout au long du cycle de vie du développement d'un système (depuis le recueil des besoins jusqu'à la phase de conception)

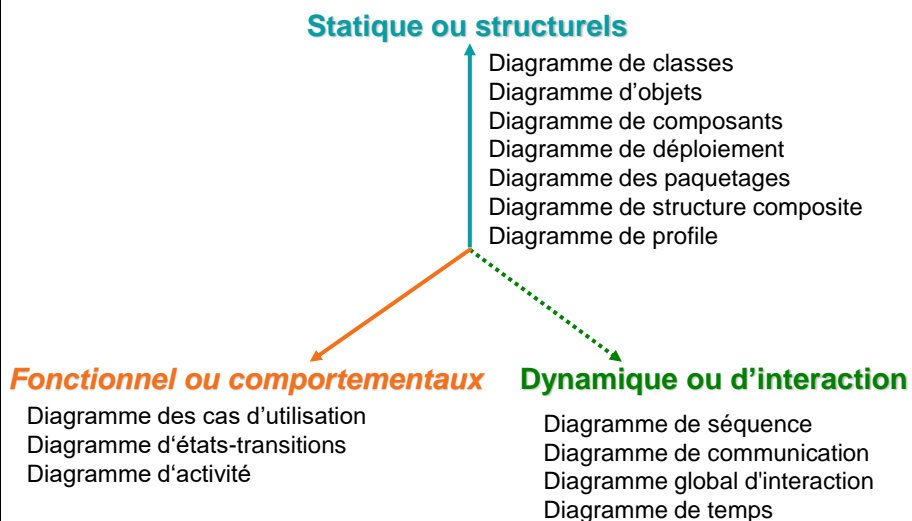
Diagrammes UML

- UML 2.5 propose 14 types de diagrammes.
- UML n'étant pas une méthode, leur utilisation est laissée à l'appréciation de chacun, même si le **diagramme de classes** est généralement considéré comme l'élément central d'UML.
- Les 14 diagrammes UML sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie.

Axes de modélisation des diagrammes

- UML modélise le système selon trois modes de représentations:
 - **Fonctionnel**: décrit les services fonctionnels rendus par le système;
 - **Statique**: décrit la structure statique du système;
 - **Dynamique**: concerne le dynamique fonctionnel du système.
- Les trois représentations sont nécessaires et complémentaires pour schématiser la façon dont le système est composé et le fonctionnement de ses composants.

Axes de modélisation des diagrammes



UML pour les BDs

- Parmi tous les diagrammes proposés par la méthode UML, seul le **diagramme de classes** permet de modéliser les bases de données.
- Les autres diagrammes ne seront donc pas mentionnés dans cette partie du cours.

Terminologie

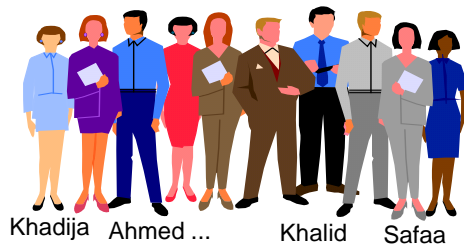
Entité-association	UML
Entité	Classe
Association (Relation)	Association (Relation)
Occurrence	Objet
Cardinalité	Multiplicité
Modèle conceptuel de données (Merise)	Diagramme de classes

Entité / Classe

- Un objet du monde réel qui peut être identifié et que l'on souhaite représenter
 - La **classe d'entité** correspond à une collection d'entités décrites par leur type commun (le format)
 - L'**instance d'entité** correspond à un élément particulier de la classe d'entité (un objet)
 - Attention: souvent on dit entité pour les deux ! Comprendre selon le contexte.
- Il existe généralement plusieurs *instances* d'entités dans une *classe* d'entité.

Entité / Classe

- **Instance d'entité :**
représentation d'un objet du monde réel ayant une existence propre.



- **Classe d'entités :**
Représentation d'un ensemble d'entités perçues comme similaires et ayant les mêmes caractéristiques.



Personne

Entité / Classe

Merise

Entité
Identifiant
Attribut 2
....
Attribut n

UML

Classe
Attribut 1
Attribut 2
....
Attribut n

- ❑ **Attributs:** Description des propriétés des entités
- ❑ Exemple: No=2, Nom=«Ahmed Chaouki», DateNaiss=16/03/1987, Sexe=«M»

Identifiant ou Clé

- **Un identifiant aussi appelé clé est un attribut qui permet de retrouver une instance d'entité unique à tout instant parmi celles de la classe.**
 - ❑ Exemple: **Matricule** dans Voitures, **CIN** dans Personnes
- **Un identifiant peut être constitué de plusieurs attributs (clé composée)**
 - ❑ Exemple:
 - [No , Rue, Ville] pour Maisons
 - [Nom, Prénom] pour Personnes
- **Clés candidates et clés primaires**
 - ❑ Une clé candidate est un ensemble d'attributs *permettant* d'identifier de manière unique une instance d'une entité
 - ❑ Parmi les clés candidates, on en choisit une, qu'on nomme clé primaire qui va identifier l'entité

Association

- Les entités sont reliées ensemble par des associations
 - Entre instances: par exemple 1 véhicule est associé à 1 personne
 - Entre classes: abstraction des associations entre instances
- Une association peut avoir des attributs (propriétés)
- Elle peut relier plusieurs entités ensemble
- Il est possible de distinguer le rôle d'une entité (elle peut en avoir plusieurs)

Cardinalité	nom	Cardinalité
rôle		rôle

Association: quelques définitions

- Association (Association)
 - Une relation entre des instances de deux (ou plus) classes
- Lien (Link)
 - Une instance d'association
- Rôle (Role)
 - Une extrémité d'une association
- Attribut de lien (Link attribute)
 - Un attribut de l'association instancié pour chaque lien
- Cardinalité (Multiplicity)
 - Le nombre d'instance d'une entité pour chaque instance de l'autre

Association

Merise

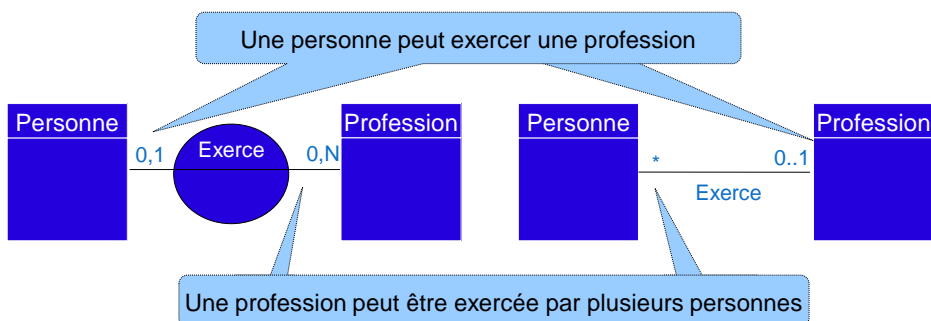
UML



Exemple d'association

Merise

UML



- Les cardinalités d'une association binaire dans le modèle de **Merise** sont **inversées** par rapport au modèle **UML** au niveau de l'axe de représentation de l'association.

Cardinalités, Multiplicités

- Elle apparaît à **chaque extrémité d'une association** et indique Le nombre d'instance d'une entité pour chaque instance de l'autre.

Merise

Cardinalités	Lecture
0 , 1	Lien vers 0 ou 1
1 , 1	Lien vers 1
0 , n	Lien vers 0 ou N
1 , n	Lien vers 1 ou N

UML

Multiplicités	Lecture
0..1	Lien vers 0 ou 1
1	Lien vers 1
* ou 0..*	Lien vers 0 ou plusieurs
1..*	Lien vers 1 ou plusieurs

Associations un à un

Merise

Cardinalités	
0 , 1	0 , 1
0 , 1	1 , 1
1 , 1	1 , 1

UML

Multiplicités	
0 .. 1	0 .. 1
0 .. 1	1
1	1

Associations un à plusieurs

Merise

UML

Cardinalités		Multiplicités	
0 , 1	0 , N	0 .. 1	*
0 , 1	1 , N	0 .. 1	1 .. *
1 , 1	0 , N	1	*
1 , 1	1 , N	1	1 .. *

Associations plusieurs à plusieurs

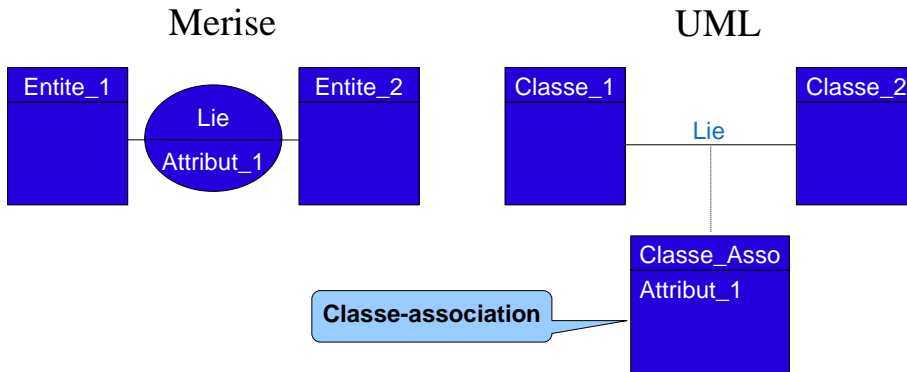
Merise

UML

Cardinalités		Multiplicités	
0 , N	0 , N	*	*
0 , N	1 , N	*	1 .. *
1 , N	0 , N	1 .. *	*
1 , N	1 , N	1 .. *	1 .. *

Association avec attributs

- Une association peut être raffinée et avoir ses propres attributs, qui ne sont disponibles dans aucune des entités/classes qu'elle lie.



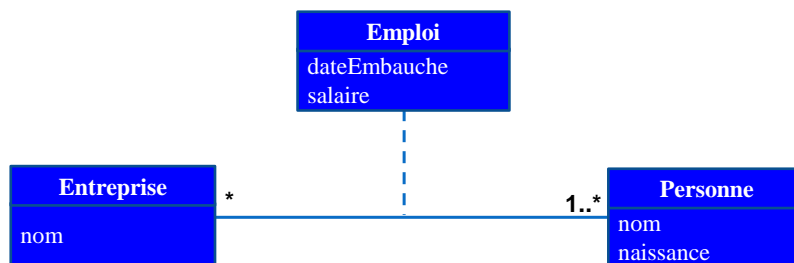
Prof. Asmaa El Hannani

2TTE-S1

244

Association avec attributs (Exemple UML)

- L'association *Emploi* possède comme propriétés le *salaire* et la *date d'embauche*.
- Ces deux propriétés n'appartiennent ni à l'entreprise, qui peut employer plusieurs personnes, ni aux personnes, qui peuvent avoir plusieurs emplois.



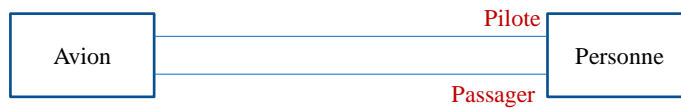
Prof. Asmaa El Hannani

2TTE-S1

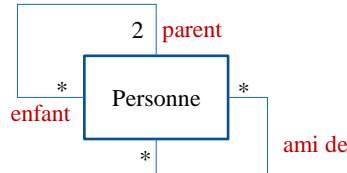
245

Rôle d'une association

- Décrit le rôle d'une classe dans une association
- Utile surtout dans deux cas :
 - Lorsqu'on a plusieurs associations entre deux classes avec des rôles différents

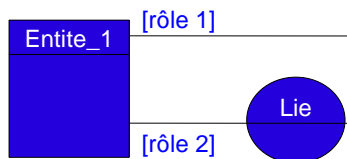


- Une relation réflexive : relation entre deux instances d'une même classe

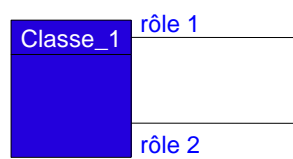


L'association réflexive

Merise

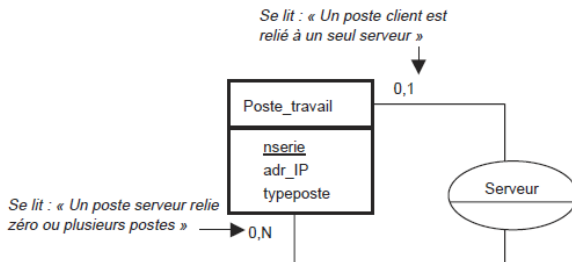


UML

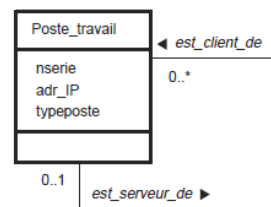


L'association réflexive

Merise



UML



Exercice 2

■ Inscription des étudiants

Lors de son inscription en début d'année scolaire, chaque étudiant remplit une fiche sur laquelle il indique certains renseignements comme son code d'identification nationale (cin), ses nom et prénom (nom, prenom), son adresse (adresse) et la liste des modules (MD) qu'il s'engage à suivre (8 au plus sur les 15 possibles). Un code lui est automatiquement attribué (codetu).

Un MD est caractérisé par un code (codemd) et un intitulé (intmd). Par exemple le code CIP41 identifie Math V. Chaque MD est placée sous la responsabilité d'un enseignant identifié par ses initiales (initens) et caractérisé par un nom (nomens), un numéro de bureau (bureauens) et un numéro de téléphone (telens). Cet enseignant se rend disponible un jour de la semaine (jsem) et durant une plage horaire précise (hrens) afin de donner tout renseignement concernant les MD qu'il dirige.

Exercice 2 - suite

1. Composer les schémas conceptuels de type Merise et UML afin de modéliser les faits ci-dessus.
2. Que faut-il modifier pour qu'un enseignant se rende disponible à différents moments (un seul créneau par MD qu'il dirige) ?
3. Que faut-il modifier pour que différents créneaux soient disponibles par MD qu'il dirige ?

Les contraintes

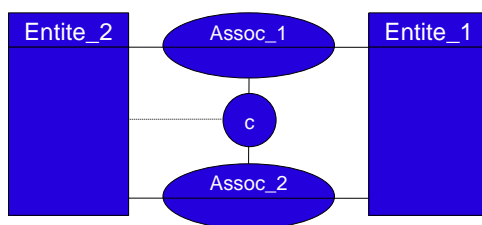
- Les contraintes sont des expressions qui précisent le rôle ou la portée d'un élément de l'association.
- Les contraintes les plus courantes sont :
 - ❑ L'**exclusivité** : peut être l'un ou l'autre
 - ❑ La **totalité** : doit être l'un, l'autre ou les deux
 - ❑ La **partition** : doit être l'un ou l'autre
 - ❑ L'**inclusion** : ce qui est dans l'un est dans l'autre
 - ❑ La **simultanéité** : doit être dans l'un et dans l'autre
 - ❑ L'**unicité** : ce qui est dans l'un ne peut être qu'une fois dans l'autre

Les contraintes

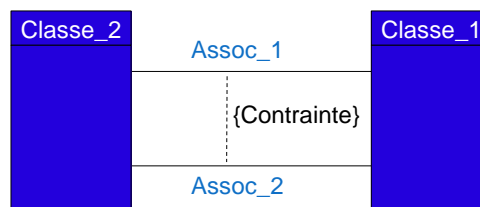
- Dans Merise les contraintes sont modélisés par **des codes** :
 - ❑ L'exclusivité : **X**
 - ❑ La totalité : **T**
 - ❑ La partition : **XT (parfois P)**
 - ❑ L'inclusion : **I**
 - ❑ Simultanéité : **S**
 - ❑ L'unicité : **U**
- En contrepartie, UML permet de définir tout type de contrainte en **langage naturel**. Graphiquement, il s'agit **d'un texte encadré d'accolades**.
 - ❑ {ordonnée} : une relation d'ordre décrit les objets
 - ❑ {sous-ensemble} : une collection est incluse dans une autre collection
 - ❑ {ou-exclusif} : pour un objet donné, une seule association est valide

Les contraintes

Merise

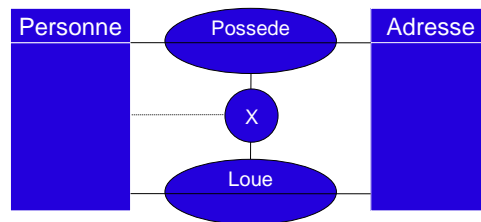


UML



Les contraintes

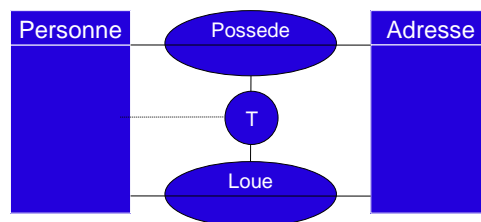
Exemple de contrainte d'exclusivité :



- Ici, on modélise le fait qu'une personne ne peut pas posséder et louer une même adresse.

Les contraintes

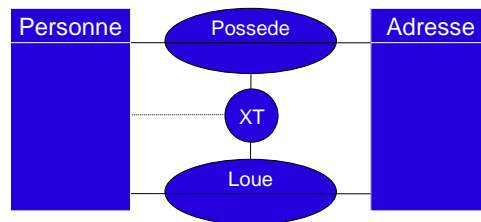
Exemple de contrainte de totalité :



- Ici, on modélise le fait qu'une personne est forcément propriétaire, locataire ou les deux.

Les contraintes

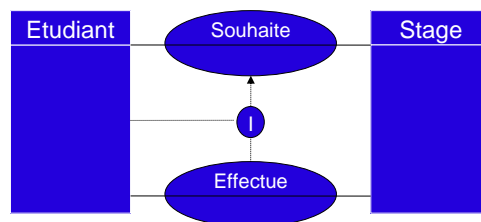
Exemple de contrainte de partition :



- Ici, on modélise le fait qu'une personne est forcément propriétaire ou locataire mais pas les deux.

Les contraintes

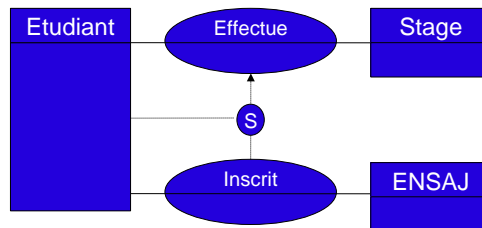
Exemple de contrainte d'inclusion :



- Ici, on modélise le fait que le stage qu'un étudiant effectuera est un des stage qu'il a souhaité.

Les contraintes

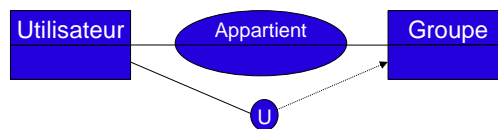
Exemple de contrainte de simultanéité :



- Ici, on modélise le fait que le stage est obligatoire pour tous les étudiants puisque toute occurrence dans 'Inscrit' doit exister dans 'Effectue'.

Les contraintes

Exemple de contrainte d'unicité :



- Ici, on modélise le fait qu'un utilisateur ne peut pas appartenir plusieurs fois au même groupe.
- Cette contrainte n'a d'intérêt que dans une relation plusieurs à plusieurs.

Exercice 3

Modéliser sous forme de diagrammes de classes ce qui suit :

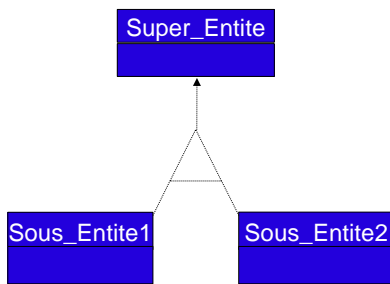
1. Le président d'un comité doit être membre du comité
2. Une personne qui soumet un article à un journal ne peut pas évaluer son propre article.

L'héritage

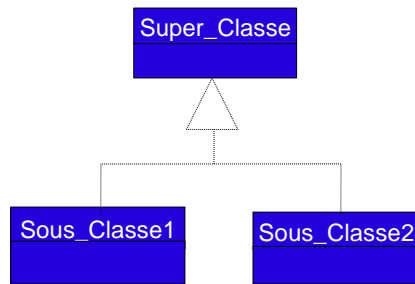
- C'est un mécanisme de **dérivation** entre classes, symbolisant la relation « **est un** ».
- L'héritage permet de reprendre les caractéristiques d'une classe A existante pour les étendre et définir ainsi une nouvelle classe B qui hérite de A.
- Les objets de B possèdent toutes les caractéristiques de A avec en plus celles définies dans B
- L'héritage existe en Merise comme en UML.
 - On modélise l'héritage en Merise par **un triangle contenant les contraintes**, en UML par une flèche de l'héritier vers le parent.

L'héritage

Merise

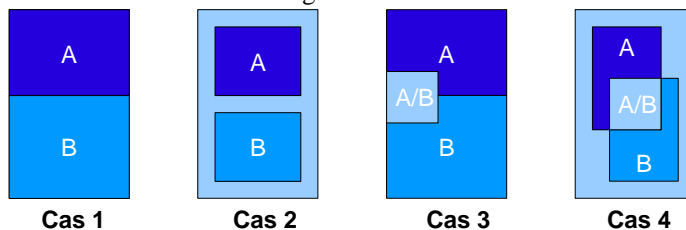


UML



L'héritage

Il existe différents cas d'héritage en fonction des instances des classes :

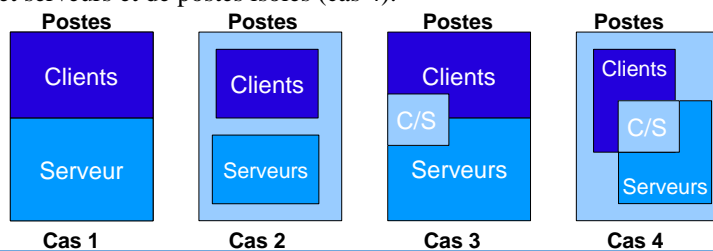


Chacun de ces cas d'héritage peut être modélisé à l'aide de contraintes :

	Couverture	Non-couverture
Disjonction	Cas 1 Merise : Partition (XT) UML : {complete, disjoint}	Cas 2 Merise : Exclusivité (X) UML : {incomplete, disjoint} (par défaut)
Non-disjonction	Cas 3 Merise : Totalité (T) UML : {complete, overlapping}	Cas 4 Merise : Pas de contrainte UML : {incomplete, overlapping}

L'héritage : exemple

- Une population de postes de travail peut être composée :
 - de postes clients ou serveurs exclusivement (cas 1) ;
 - de postes clients, de postes serveurs et de postes isolés qui ne sont ni clients ni serveurs (cas 2) ;
 - de postes clients, de postes serveurs et de postes qui sont à la fois clients et serveurs, mais pas de postes isolés (cas 3) ;
 - de postes clients, de postes serveurs, de postes qui sont à la fois clients et serveurs et de postes isolés (cas 4).

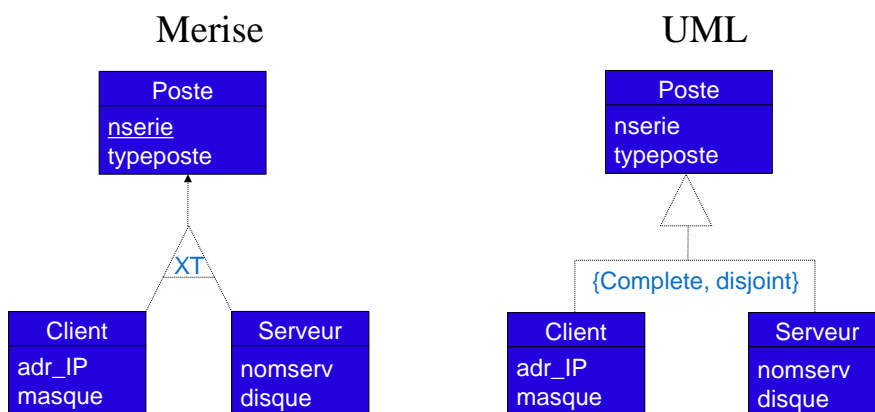


Prof. Asmaa El Hannani

2TTE-S1

268

L'héritage : exemple « cas 1 »



Prof. Asmaa El Hannani

2TTE-S1

269

L'héritage

- L'absence de contraintes sur un graphe d'héritage n'a pas la même signification en Merise et UML :
 - Pour Merise c'est **le cas 4** : non-couverture, non-disjonction
 - Pour UML c'est **le cas 2** : non-couverture, disjonction

UML ou Merise ?

- La notation UML est bien adaptée à la modélisation d'une base de données pour les raisons suivantes :
 - Les bases de données sont majoritairement relationnelles, mais certaines migreront **vers l'objet** ;
 - Les concepts fondamentaux de Merise peuvent être représentés dans le diagramme de classes d'UML qui offre en plus la possibilité de définir **des stéréotypes et des contraintes personnalisées** ;
 - la majorité des contraintes sur les associations n-aires sont implicites si on utilise les classes-associations;
 - les concepteurs travaillent dans un même environnement (comme Eclipse), avec la possibilité **d'interfacer plus facilement les langages évolués C++ ou Java.**

Conception du modèle pour une BD

- Il n'existe pas de modèle de données idéal.
- Le modèle doit correspondre à un besoin précis.
- Il est indispensable que chaque décision, chaque façon de faire soit réfléchie.
- En cas de multiples possibilités, il faut s'assurer que le fonctionnement mis en place répondra au besoin.
- Il peut être tentant de créer un modèle le plus généraliste possible. Mais attention :
 - Ce type de modèles mèneront à des bases de données trop complexes.
 - Les performances seront alors moindre et les fonctionnalités mise en place pas forcément utilisées.

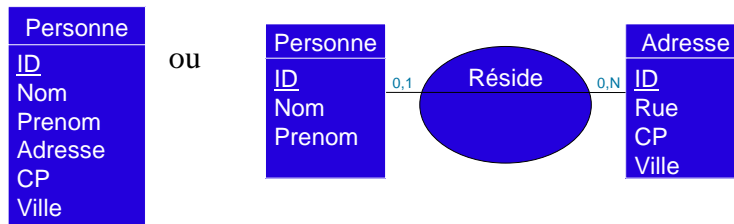
Interprétation (1/3)

- Une base données doit permettre de stocker toutes les informations nécessaires à son utilisation.
- Toute la complexité réside dans l'organisation de ces attributs.
- Toute redondance est interdite.
- Il faut essayer de créer un modèle à la fois évolutif mais aussi suffisant pour le besoin.

Interprétation (2/3)

- Un choix difficile est notamment le fait d'utiliser un attribut dans l'entité ou de créer une association.

Exemple :



Interprétation (3/3)

- Le choix des cardinalités est également primordial.

Exemple :

