

Licence MI2E- 3ème année  
Mention Informatique  
et Mention Mathématiques Mineure Informatique

# **Bases de données relationnelles**

## **Maude Manouvrier**

- Modélisation Entité/Association et UML (Vocabulaire)
- Modèle relationnel et passage au modèle relationnel
- Algèbre relationnelle
- Calcul relationnel
- SQL / *Embedded SQL* / ODBC / JDBC
- Dépendances fonctionnelles
- Décomposition de schéma
- Formes Normales

# BIBLIOGRAPHIE

## Ouvrages de référence utilisés pour le cours :

**T. Connolly, C. Begg et A. Strachan, *Database Systems A Pratical Approach to Desigh, Implementation and Management*, 1998, ISBN: 0-201-34287-1,**

**G. Gardarin, *Bases de Données - objet/relationnel*, Eyrolles, 1999, ISBN: 2-212-09060-9, disponible à la BU 005.74 GAR**

**R. Ramakrishnan et J. Gehrke, *Database Management Systems, Second Edition*; McGraw-Hill, 2000, ISBN: 0-07-232206-3, disponible à la BU 055.7 RAM**

**A. Silberschatz, H.F. Korth et S. Sudarshan, *Database System Concepts*, McGraw-Hill, 1996, ISBN: 0-07-114810-8, disponible à la BU 005.7 DAT**

**J.D. Ullman et J. Widom, *A first Course in Database Systems*, Prentice Hall, 1997, ISBN: 0-13-887647-9, disponible à la BU 005.7 ULL**

# BIBLIOGRAPHIE

**Autres ouvrages de référence, disponibles à la BU :**

**C.J. Date, *An Introduction to Database Systems*, Addison Wesley**

**C.J. Date, *A Guide to SQL Standard*, Addison Wesley**

**R.A. El Masri et S.B. Navathe, *Fundamentals of Database Systems*, Prentice Hall**

**Ouvrages pédagogiques contenant des exercices corrigés :**

**Philip J. Pratt, *Initiation à SQL - Cours et Exercices corrigés*, Eyrolles, 2001 –  
BU : 005.72 SQL**

**Christian Soutou, *De UML à SQL - Conception de bases de données*, Eyrolles,  
2002 – BU : 005.72 SOU**

**F. Brouard, C. Soutou , *SQL (Synthèse de cours et exercices corrigés)*. Pearson  
Education 2005 – BU : 005.72 SQL**

**Christian Soutou, *SQL Pour Oracle (avec exercices corrigés)*, Eyrolles, 2005 –  
BU 005.72 SOU**

**Nicolas Larousse, *Création de bases de données*, Coll. Synthex, Pearson  
Education, 2006**

# Chap. I - Introduction

- **Base de données** :

- collection d'informations ou de données qui existent sur une **longue période de temps** [UW97] et qui décrivent les activités d'une ou plusieurs organisations [RG00]
- ensemble de données **modélisant les objets d'une partie du monde réel** et servant de support à une application informatique [Gar99]

- **SGBD** : Systèmes de Gestion de Bases de Données  
(*DataBase Management Systems - DBMS*)

**ensemble de logiciels systèmes** permettant aux utilisateurs d'insérer, de modifier, et de rechercher efficacement des données spécifiques dans **une grande masse d'informations** (pouvant atteindre plusieurs milliards d'octets) **partagée par de multiples utilisateurs** [Gar99]

# SGBD

Principaux composants :

- **Système de gestion de fichiers**
- **Gestionnaire de requêtes**
- **Gestionnaire de transactions**

Principales fonctionnalités :

- **Contrôle de la redondance d'information**
- **Partage des données**
- **Gestion des autorisations d'accès**
- **Vérifications des contraintes d'intégrité**
- **Sécurité et reprise sur panne**

# Abstraction des données

- **Niveau interne ou physique :**
  - plus bas niveau
  - indique **comment** (avec quelles structures de données) sont stockées physiquement les données
- **Niveau logique ou conceptuel :**
  - décrit par un **schéma conceptuel**
  - indique quelles sont les données stockées et quelles sont leurs relations **indépendamment de l'implantation physique**
- **Niveau externe ou vue :**
  - propre à chaque utilisateur
  - décrit par un ou plusieurs **schémas externes**

# Instances et schéma

- **Instances de base de données :**
  - données de la base à un instant donné
  - manipulées par un **langage de manipulation de données (DML - *Data Manipulation Language*)**
- **Schéma de base de données :**
  - description de la structure des données
  - ensemble de définitions exprimées en **langage de description de données (DDL - *Data Definition Language*)**

# Petit historique

- **1960** : systèmes de gestion de fichiers
- **1970** : début des SGBD réseaux et hiérarchiques proches des systèmes de gestion de fichiers  $\Rightarrow$  pas d'interrogation sans savoir où est l'information recherchée ("navigation") et sans écrire de programmes
- **1970** : papier fondateur de CODD sur la théorie des relations  
**fondement de la théorie des bases de données relationnelles**  
INGRES à Berkeley - langage QUEL  
System R IBM à San Jose - langages SEQUEL et QBE
- **1980** : **Apparition des SGBD relationnels sur le marché** (Oracle, Ingres, Informix, Sybase, DB2 ...)
- **1990** : **début des SBGD orientés objet** (Gemstone, O<sub>2</sub>, Orion, Objectstore, Versant, Matisse...).
- **Aujourd'hui** : **relationnel-objet, semi-structuré, multimédia ...**

# Chap II - Modélisation

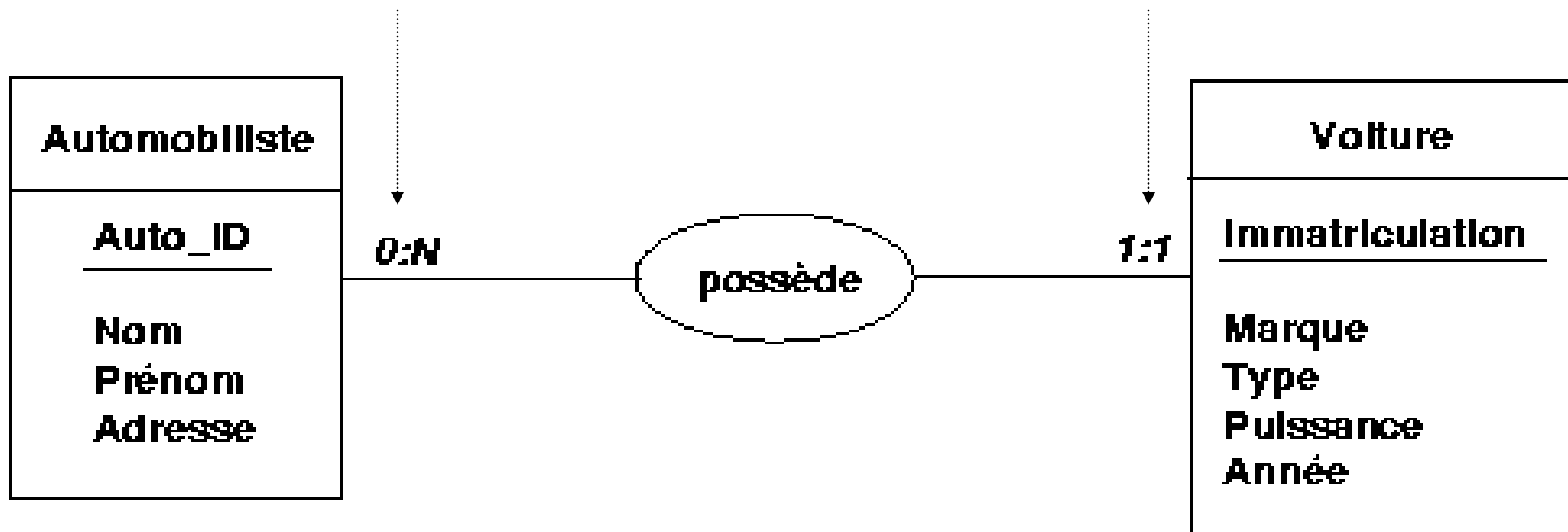
## Méthodologie à suivre pour modéliser un problème

- Déterminer les **entités/classes** et **attributs** :
  - entité/instance de classe = objet décrit par de l'information
  - objet caractérisé uniquement par un identifiant = attribut
  - attribut multi-valué ou avec une association 1:N = entité ou instance
  - attacher les attributs aux ensemble d'entités/classes qu'ils décrivent le plus directement
  - éviter au maximum les identificateurs composites
- Identifier les **généralisations-spécialisations/héritage**
- Définir les **associations**
  - éliminer les associations redondantes
  - éviter les associations n-aires
  - calculer les **cardinalités** de chaque association

# Modélisation Entité/Association (Format Merise)

*Un automobiliste possède  
entre zéro et N voitures*

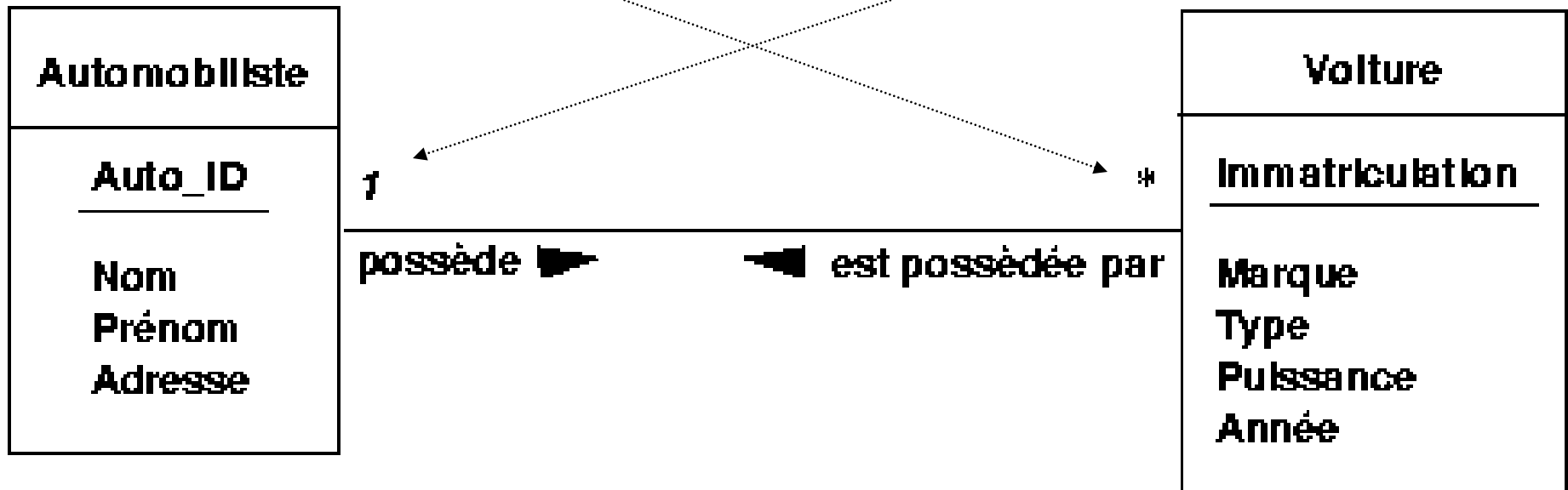
*Une voiture a un et un  
seul propriétaire*



# Modélisation UML

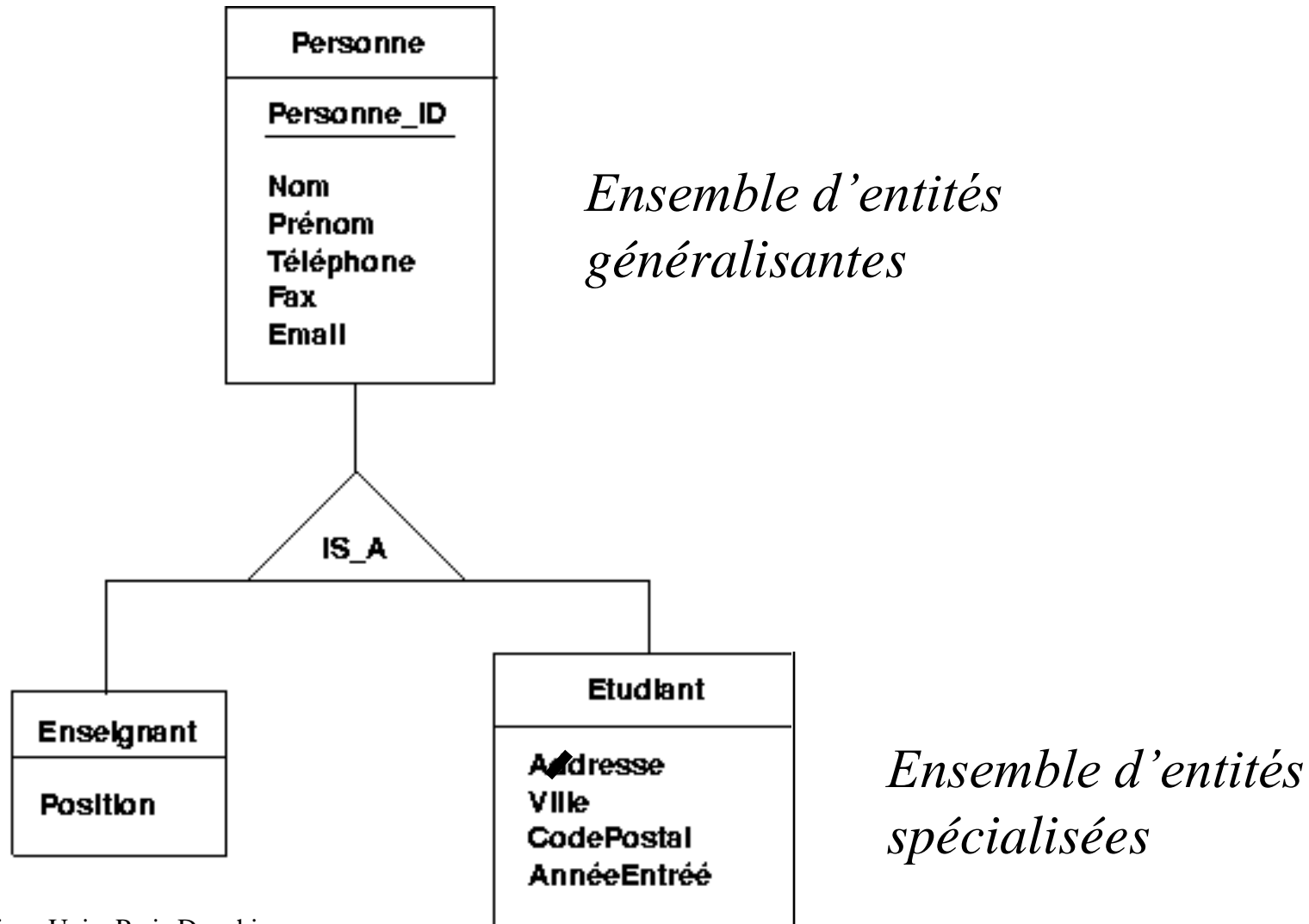
*Un automobiliste possède  
entre zéro et N voitures*

*Une voiture a un et un  
seul propriétaire*

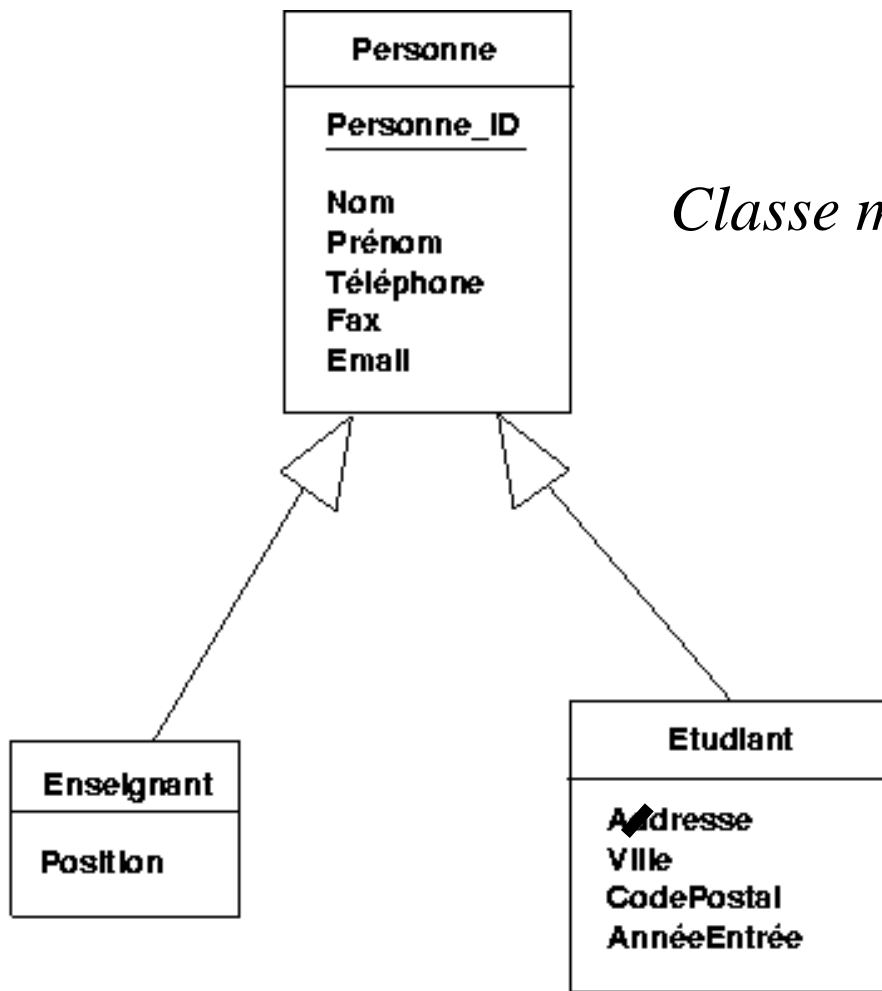


*Attention : petite liberté prise avec UML, les attributs soulignés ici ne correspondent pas à des attributs dérivés mais aux identificateurs (pour ne pas les oublier lors du passage au relationnel!!)*

# Généralisation/Spécialisation (E/A - Merise)



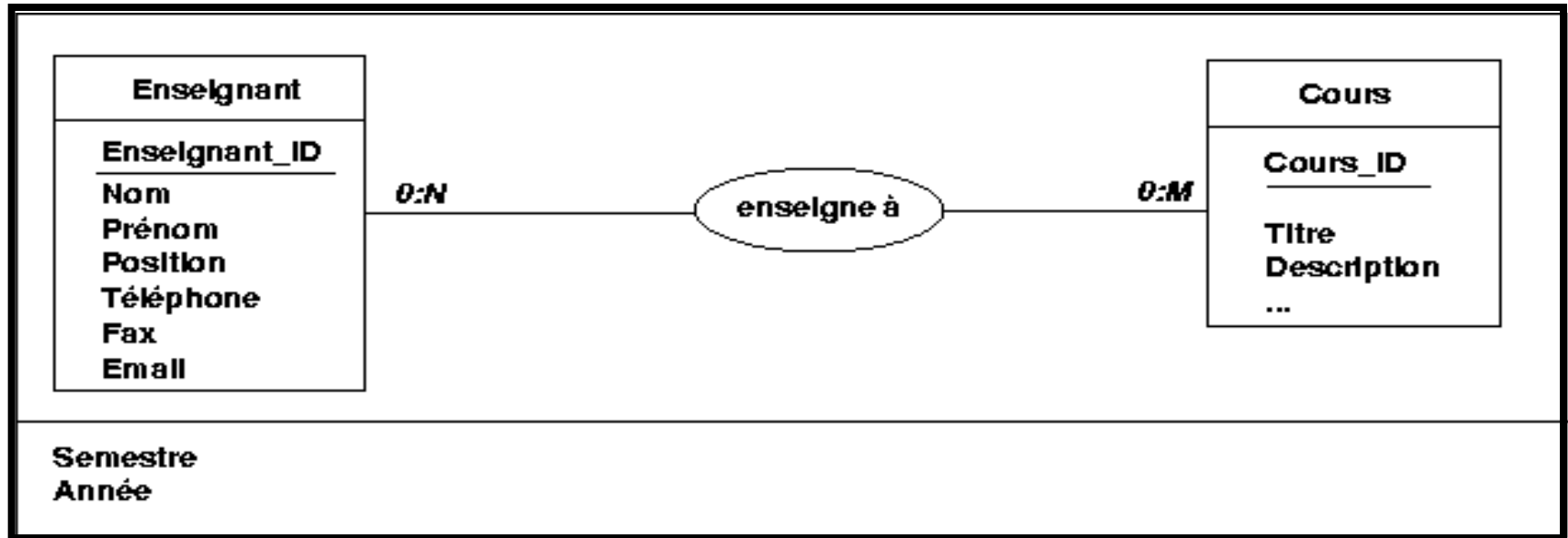
# Héritage (UML)



*Classe mère / Sur-classe*

*Classes dérivées ou filles / sous-classes*

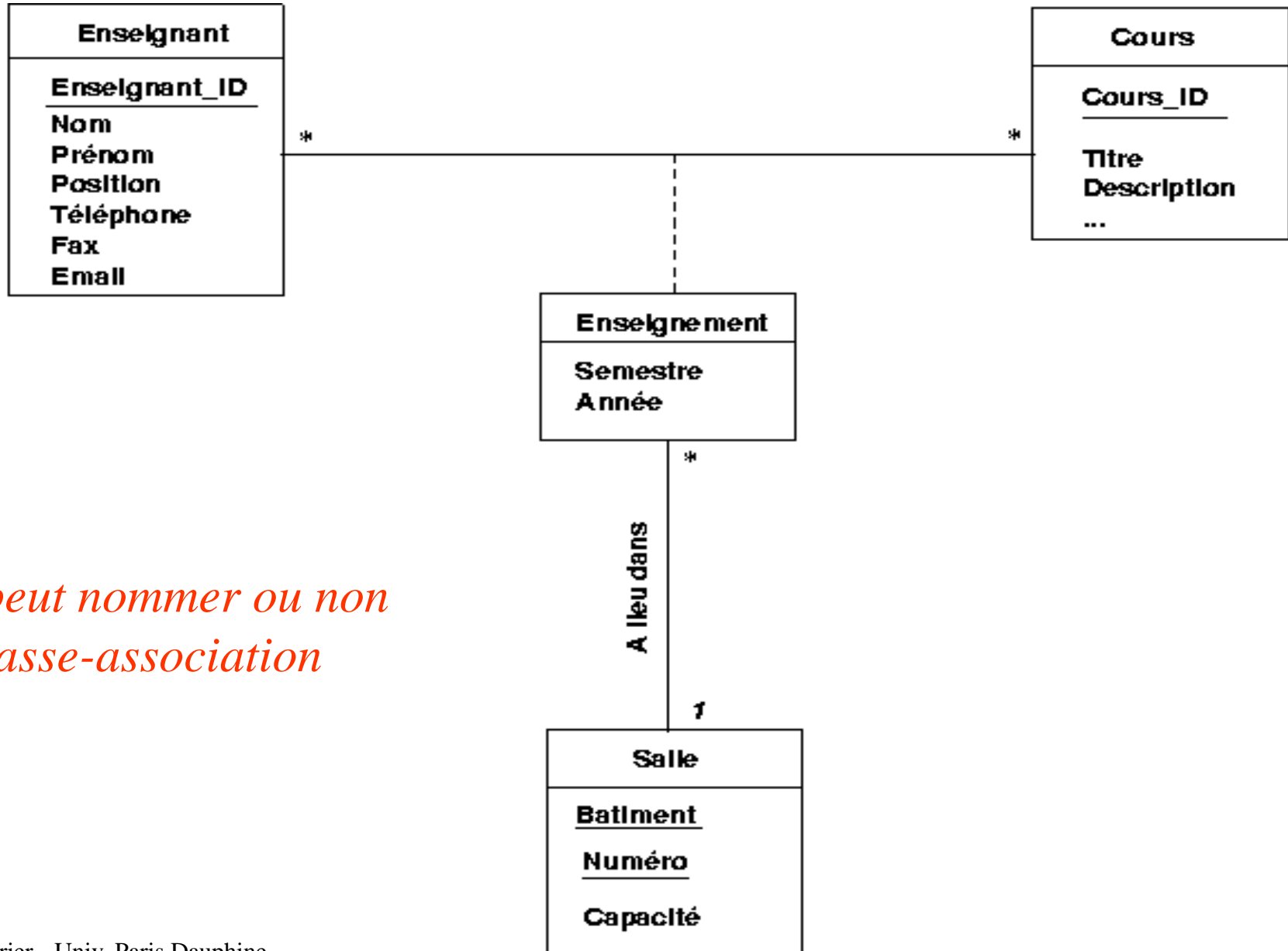
# Agrégat (E/A - Merise)



*On peut nommer ou non l'agrégat*

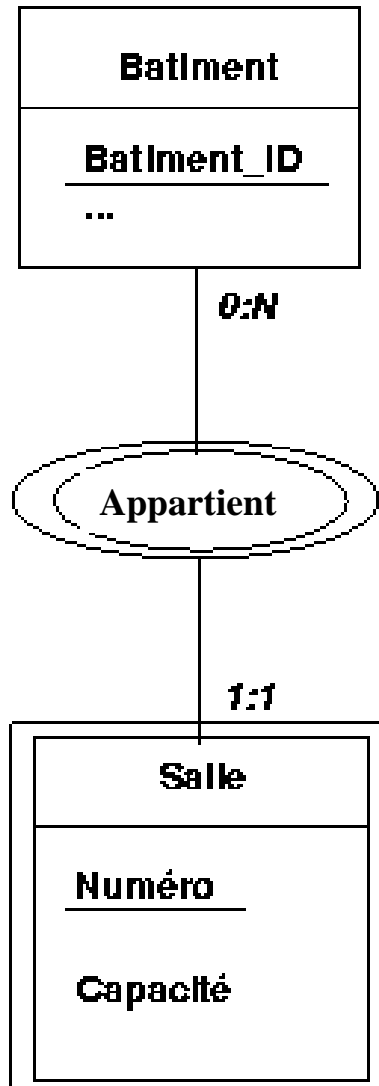


# Classe-Association (UML)



*On peut nommer ou non la classe-association*

# Entité Faible (E/A - Merise)

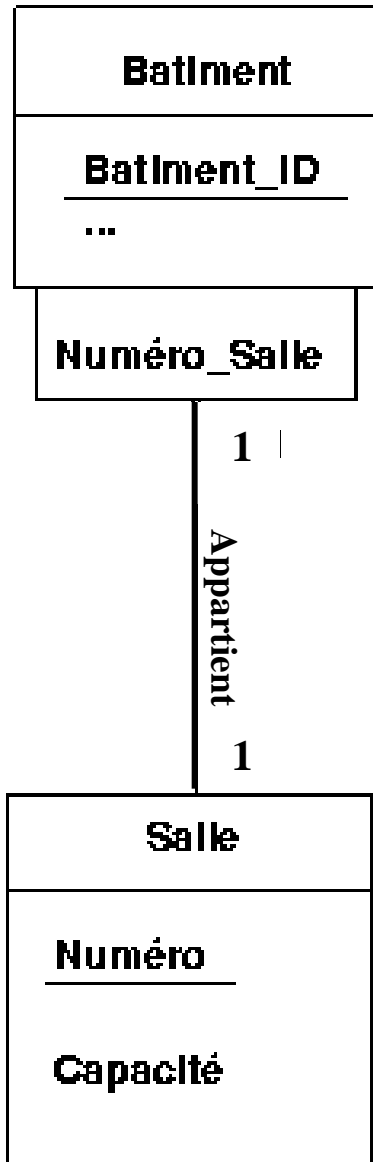


*Chaque salle a un numéro unique dans un bâtiment donné*

*Ex. Salle 1 du bâtiment A et Salle 1 du bâtiment C*

*Pour distinguer une salle d'une autre, il faut connaître le bâtiment auquel elle est rattachée*

# Association qualifiée (UML)

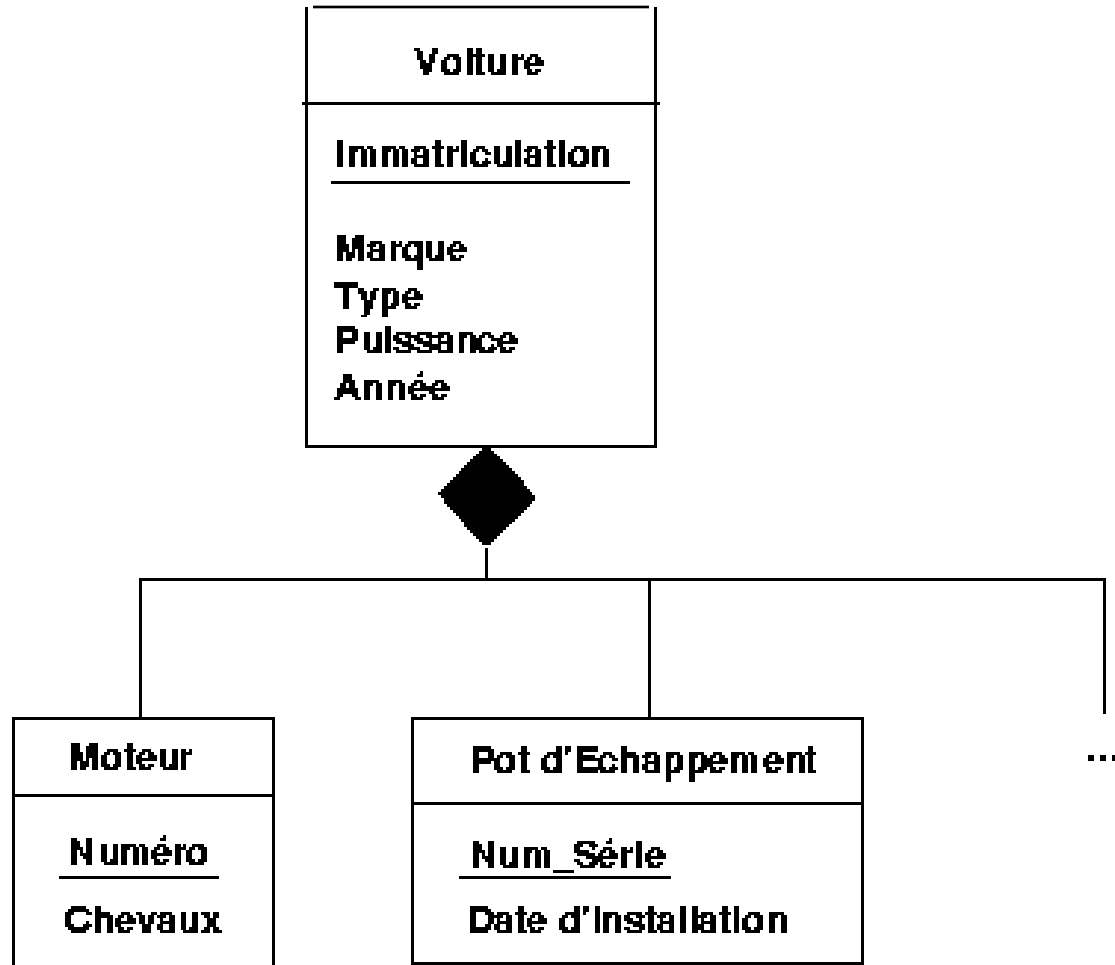


*Chaque salle a un numéro unique dans un bâtiment donné*

*Ex. Salle 1 du bâtiment A et Salle 1 du bâtiment C*

*Pour distinguer une salle d'une autre, il faut connaître le bâtiment auquel elle est rattachée*

# Composition (UML)



# Contraintes

## Contraintes d'intégrité :

toutes règles implicites ou explicites que doivent suivre les données [Gar99]

- **Contraintes d'entité**: toute entité doit posséder un identificateur
- **Contraintes de domaine** : les valeurs de certains attributs doivent être prises dans un ensemble donné
- **Contraintes d'unicité** : une valeur d'attribut ne peut pas être affectée deux fois à deux entités différentes
- **Contraintes générales** : règle permettant de conserver la cohérence de la base de manière générale

# Exemples de contraintes

– **Contraintes de domaine :**

"La fonction d'un enseignant à l'Université prend sa valeur dans l'ensemble {vacataire, moniteur, ATER, MCF, Prof., PRAG, PAST}."

– **Contraintes d'unicité :**

"Un département, identifié par son numéro, a un nom unique (il n'y a pas deux départements de même nom)."

– **Contraintes générales :**

"Un même examen ne peut pas avoir lieu dans deux salles différentes à la même date et à la même heure. "

# Dépendances fonctionnelles

Un attribut (ou un groupe d'attributs)  $Y$  **dépend fonctionnellement** d'un attribut (ou groupe d'attributs)  $X$  si :

étant donné une valeur de  $X$ , il lui correspond une valeur unique de  $Y$  ( $\forall$  l'instant considéré)

$X \rightarrow Y$  :  $Y$  **dépend fonctionnellement de  $X$**   
ou  $X$  **détermine  $Y$**

Déclaration des dépendances **au niveau du schéma conceptuel**

# Exemple de dépendances fonctionnelles

<b>Volture</b>
<u>Immatriculation</u>
Marque Type Puissance Année

*identificateur*

*Tous les autres attributs*

Immatriculation → Marque, Type, Puissance, Année

~~Marque, Type, Puissance, Année → Immatriculation~~

Type → Marque

*Ex. Le type "Twingo" sera toujours associé, dans la base de données, à la marque "Renault".*

<b>Enseignant</b>
<u>Enseignant_ID</u>
Nom Prénom Position Téléphone Fax Email

EnseignantID → Nom, Prénom, Position ...

Nom, Prénom, Position, ... → Enseignant\_ID

*Si un numéro de téléphone est associé à un seul enseignant :*

Telephone → Enseignant\_ID

# Chap III - Modèle relationnel

- **Domaine** : ensemble de valeurs caractérisé par un nom
- **Relation** : sous-ensemble du produit cartésien d'une liste de domaines caractérisé par **un nom unique**
  - représentée sous forme de **table à deux dimensions**
  - colonne = un domaine du produit cartésien
  - **un même domaine peut apparaître plusieurs fois**
  - ensemble de **nuplets sans doublon**
- **Attribut** : une colonne dans une relation
  - caractérisé par un nom et dont **les valeurs appartiennent à un domaine**
  - **les valeurs sont atomiques**
- **Nuplet** : une ligne d'une relation
  - correspondant à un enregistrement, c-à-d **une entité/instance de classe**
  - **les nuplets d'une relation sont tous différents**

# Exemple de relation

Nom d'attribut



NSS	Nom	Prénom	Fonction
273...	<i>Manouvrier</i>	<i>Maude</i>	<i>MCF</i>
...			
...			

*La relation Enseignant*

**Nuplets** ou *tuples*

# Instances et schéma

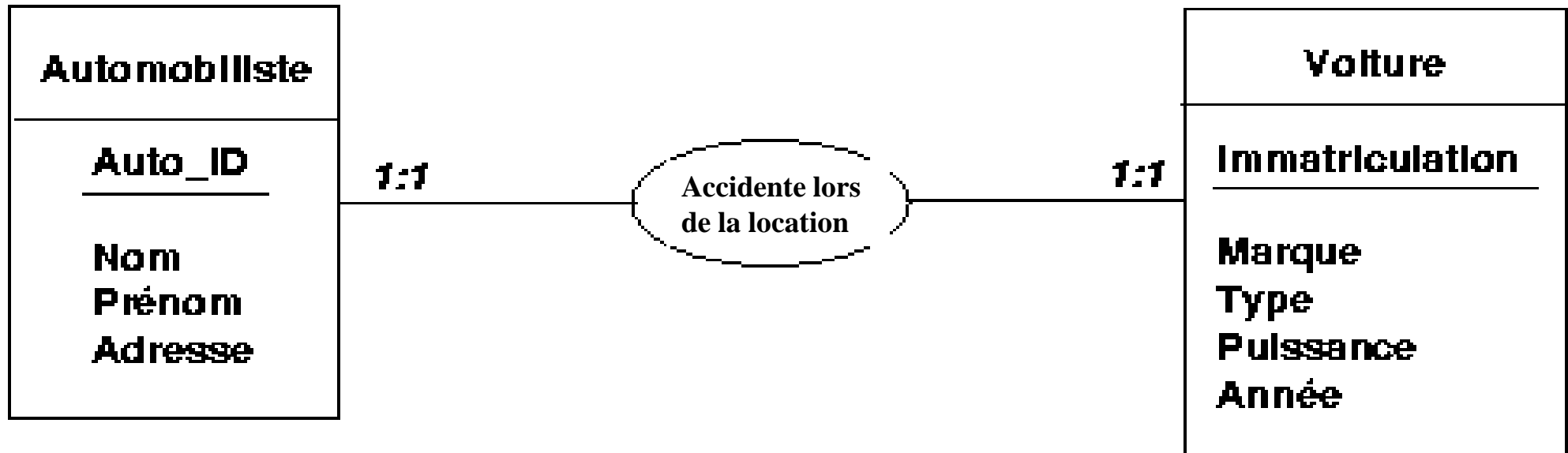
- **Instances de base de données :**
  - les nuplets (les valeurs) contenus dans la base à un instant donné
- **Schéma de base de données :**
  - ensemble de **schémas de relation**
  - modélisation logique de la base de données à l'aide du modèle relationnel
- **Schéma de relation :**
  - liste d'attributs et leurs domaines

# Passage au relationnel

## Transformation des ensembles d'entités :

- **chaque ensemble d'entités/classes  $E \Rightarrow$** 
  - une relation  $R$  dont le schéma est celui de l'ensemble d'entités/classe
  - l'identificateur de  $E$  devient la clé de  $R$
- **chaque ensemble d'entités faibles/association qualifiée  $E \Rightarrow$** 
  - une relation  $R$  qui comprend tous les attributs de  $E$  +  
l'identificateur de l'ensemble d'entités fortes/classe associé(e)
- **généralisation-spécialisation/héritage  $\Rightarrow$** 
  - l'ensemble d'entités généralisante/classe mère  $E \Rightarrow$  une relation  $R$
  - chaque ensemble d'entités  $E_i$  spécialisé/classe fille  
 $\Rightarrow$  une relation  $R_i$  dans laquelle l'identifiant est de même domaine que l'identifiant de  $E$

# Transformation des ensembles d'associations E/A

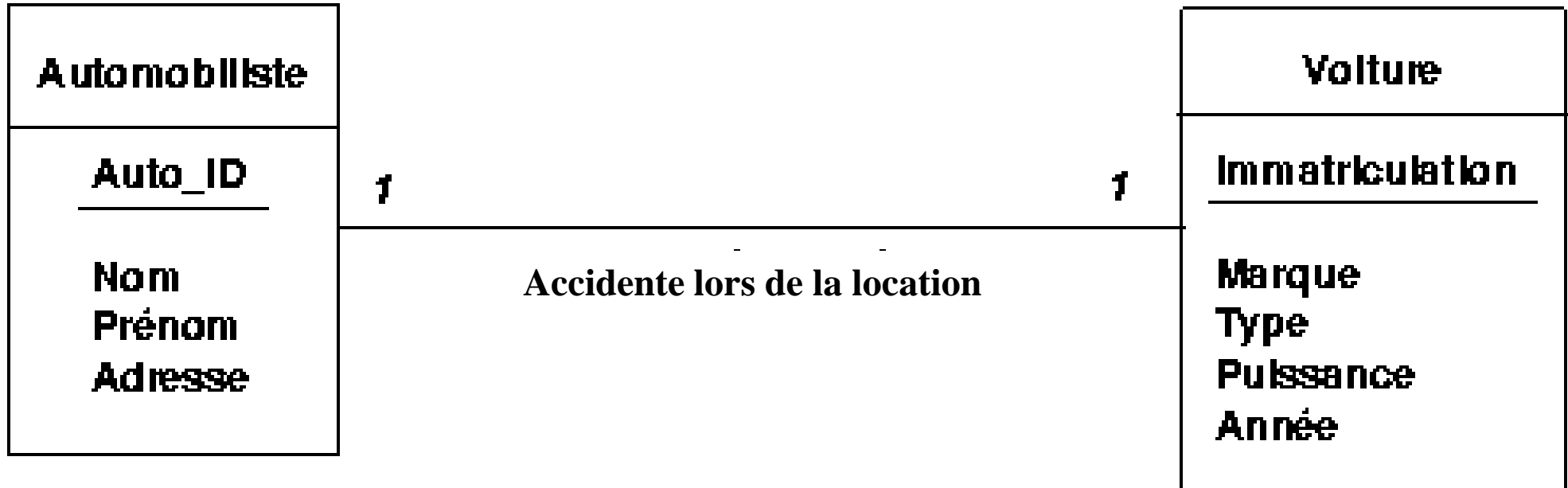


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse)

*Voiture* ( Immatriculation, Marque, Type, Puissance, Année )

Comment faire le lien ?

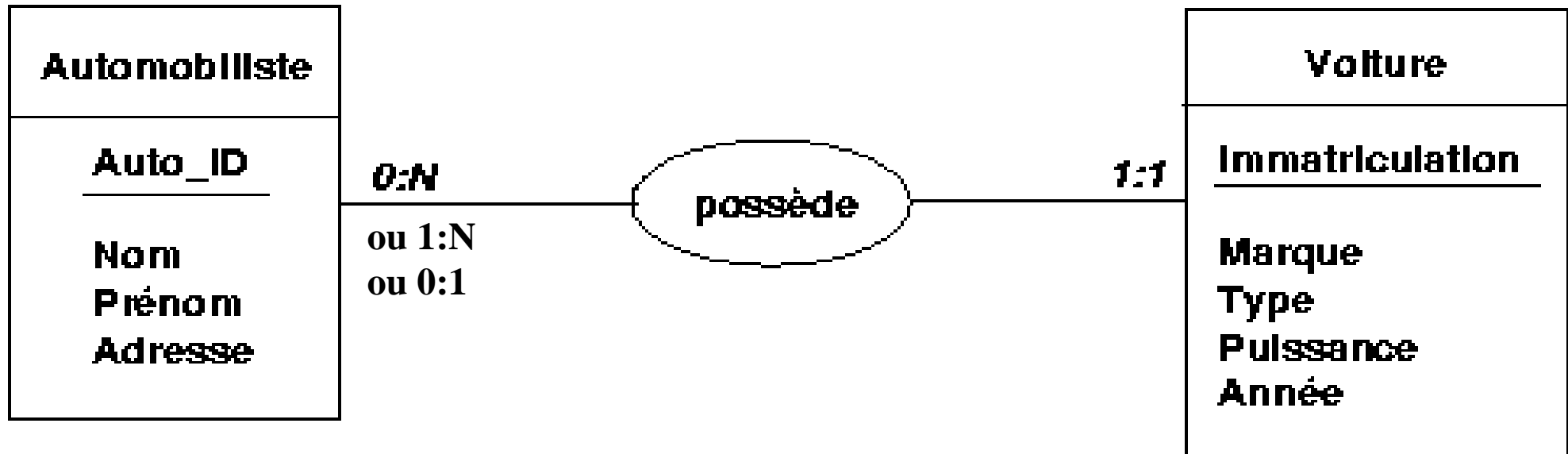
# Transformation des ensembles d'associations UML



*Accidente* ( Auto\_ID, Nom, Prénom, Adresse, Immatriculation, Marque, Type, Puissance, Année )

*On peut choisir l'un ou l'autre comme clé primaire*

# Transformation des ensembles d'associations E/A

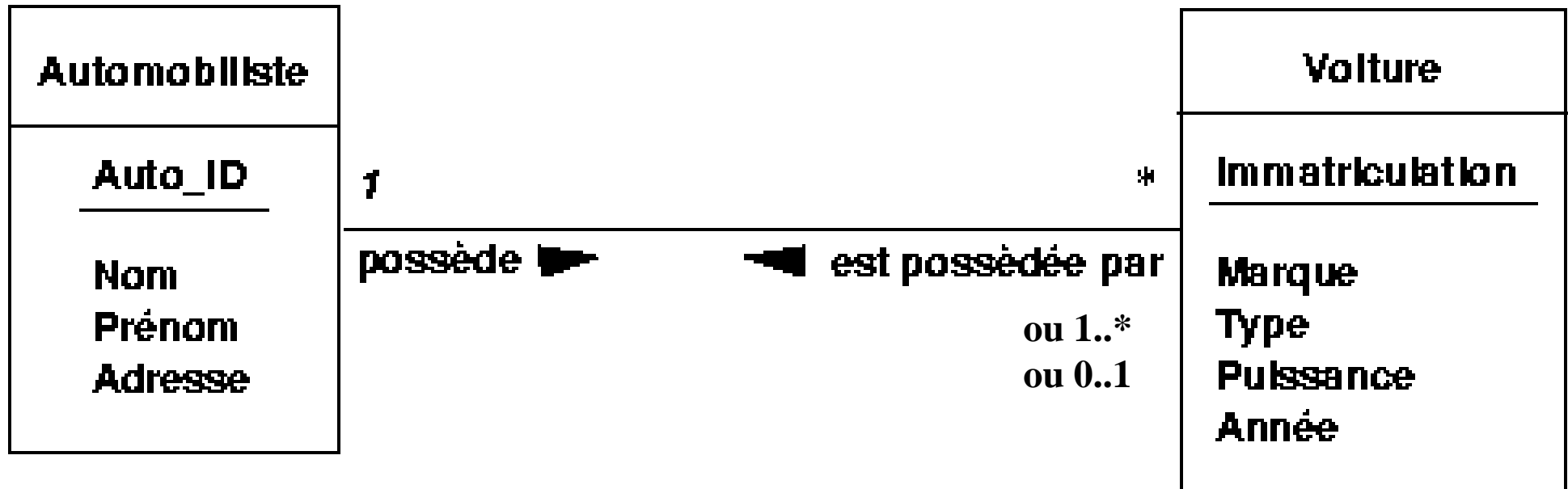


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année, #Auto\_ID )

**NB** : #Auto\_ID fait référence à Auto\_ID de Automobiliste

# Transformation des ensembles d'associations UML

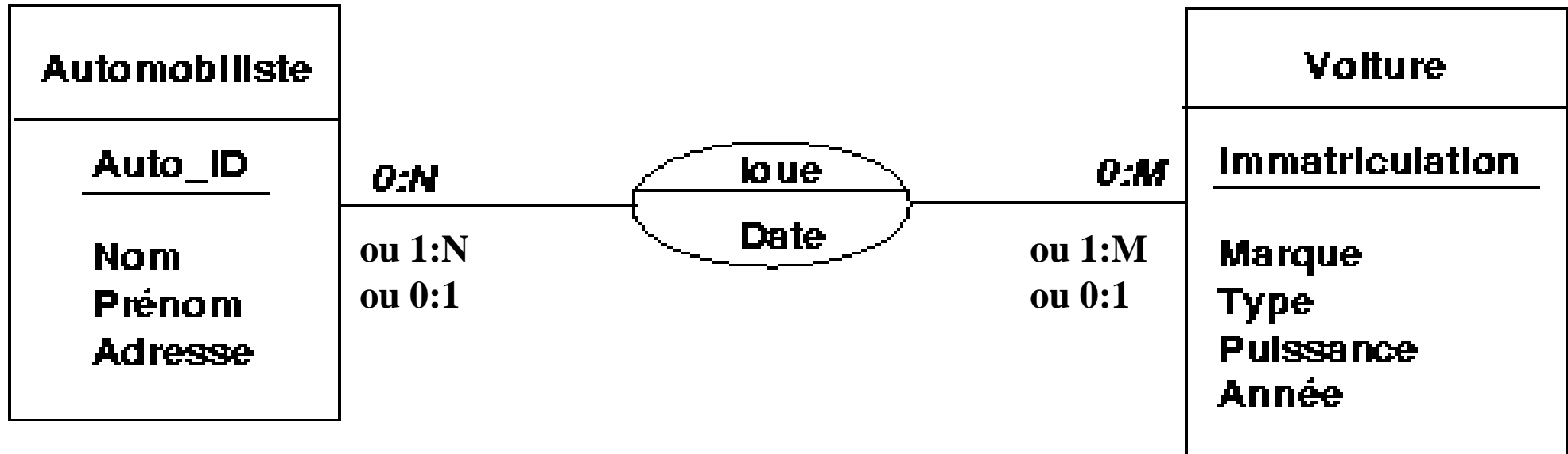


*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année, **#Auto\_ID** )

**NB** : #Auto\_ID fait référence à Auto\_ID de Automobiliste

# Transformation des ensembles d'associations E/A



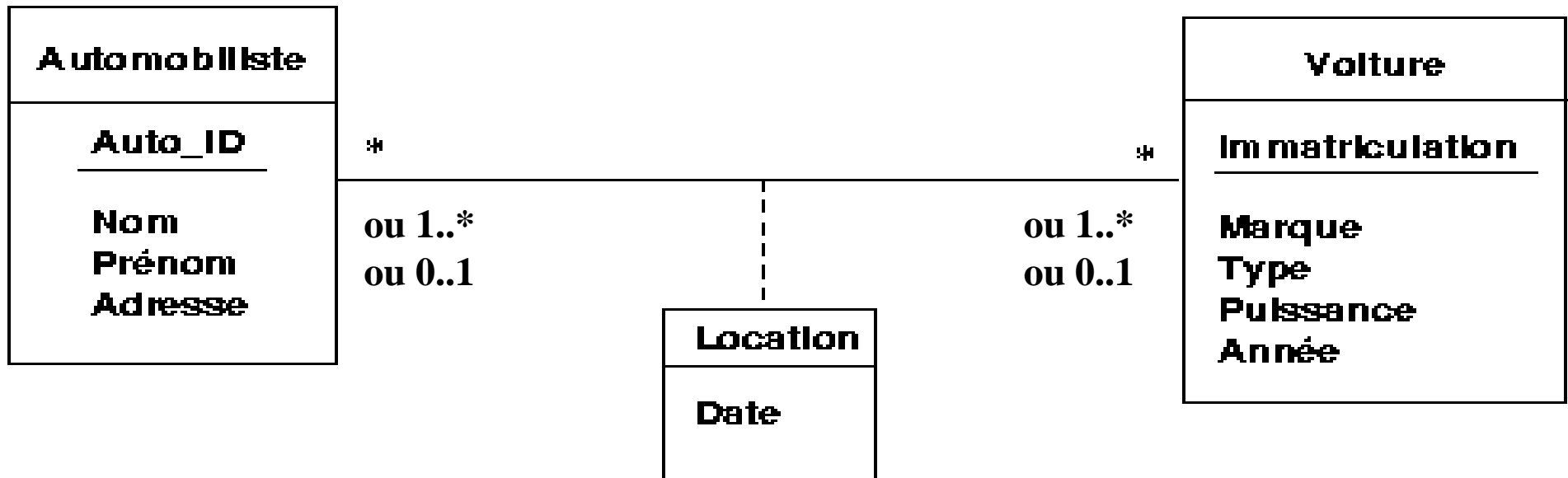
*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )

*Location* ( #Auto\_ID, #Immatriculation, Date ) ou

*Location* ( Loc\_ID, #Auto\_ID, #Immatriculation, Date )

# Transformation des ensembles d'associations UML



*Automobiliste* ( Auto\_ID, Nom, Prénom, Adresse )

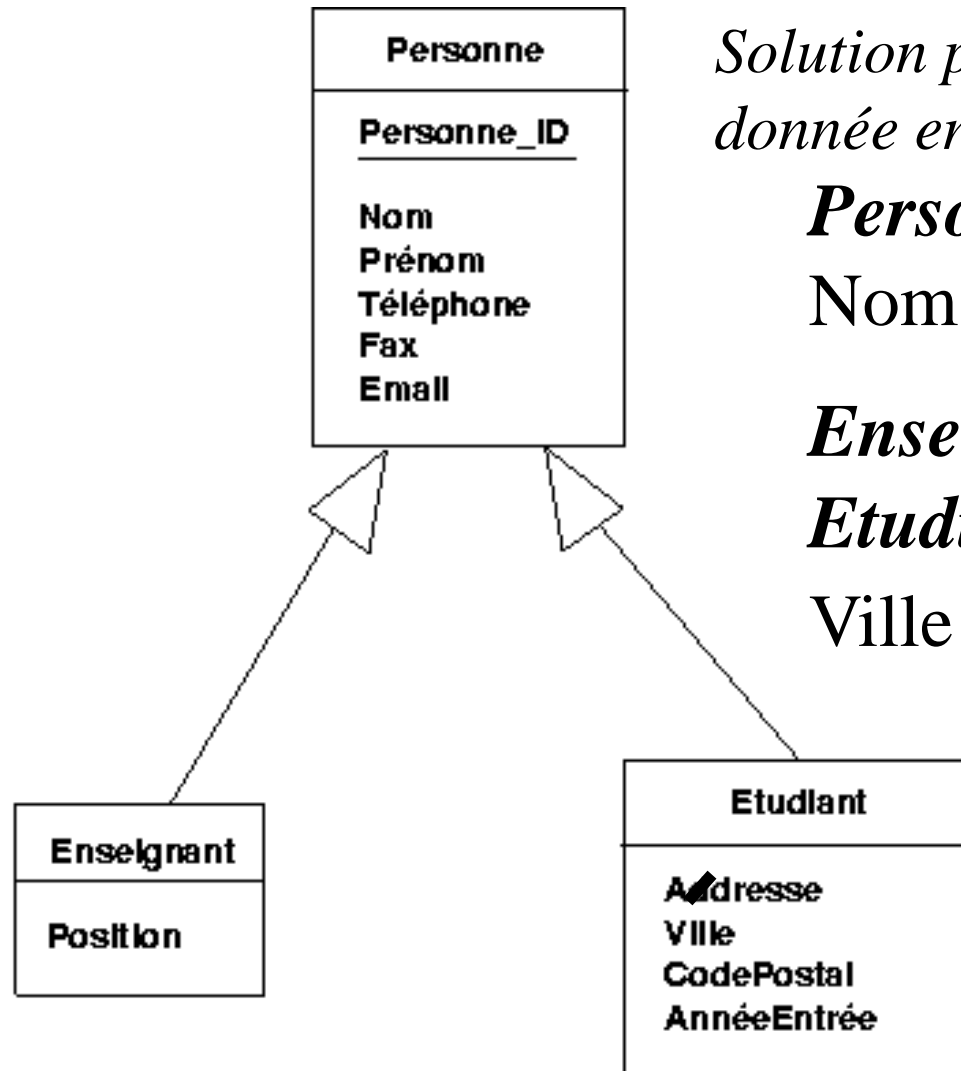
*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )

*Location* ( #Auto\_ID, #Immatriculation, Date ) ou

*Location* ( Loc\_ID, #Auto\_ID, #Immatriculation, Date )

# Transformation des concepts

## Généralisation-Spécialisation / Héritage



*Solution possible (une autre sera donnée en cours) :*

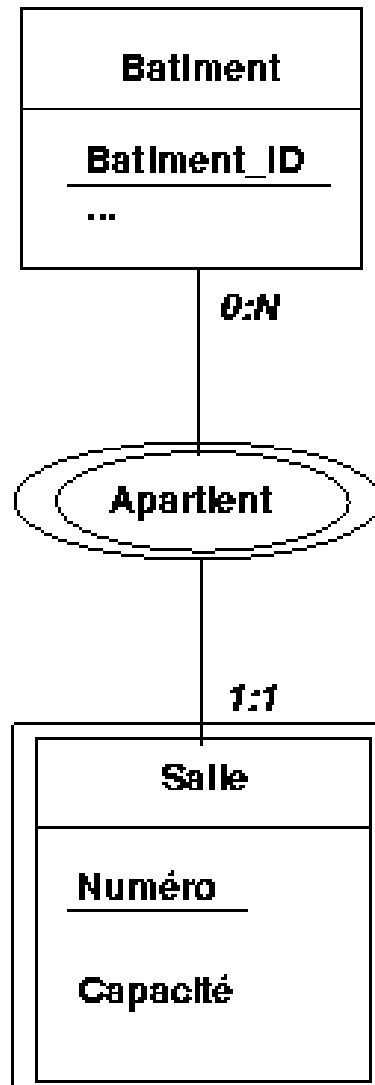
***Personne** ( Personne\_ID,  
Nom, Prénom, Téléphone ... )*

***Enseignant** ( #Personne\_ID, Position )*

***Etudiant** ( #Personne\_ID, Adresse,  
Ville ... )*

**NB** : #*Personne\_ID* dans  
*Enseignant* et *Etudiant* font  
référence à *Personne\_ID* dans  
*Personne*

# Transformation des entités faibles E/A



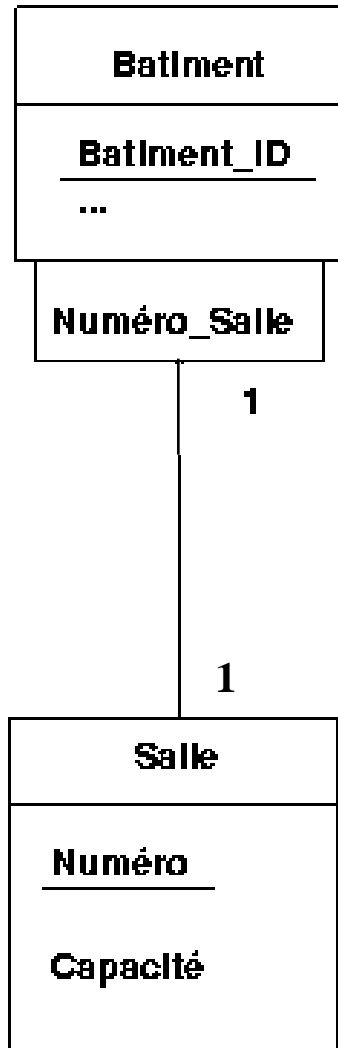
*Bâtiment* ( Bâtiment\_ID, ... )

*Salle* ( Numéro, #Bâtiment\_ID, Capacité)

**NB :** *Une salle est identifiée par le couple (Numéro,#Bâtiment\_ID)*

*#Bâtiment\_ID fait référence à Bâtiment\_ID de Bâtiment*

# Transformation des associations qualifiées UML



*Bâtiment* ( Bâtiment\_ID, ... )

*Salle* ( Numéro, #Bâtiment\_ID, Capacité)

**NB** : *Une salle est identifiée par le couple (Numéro, #Bâtiment\_ID) ;*

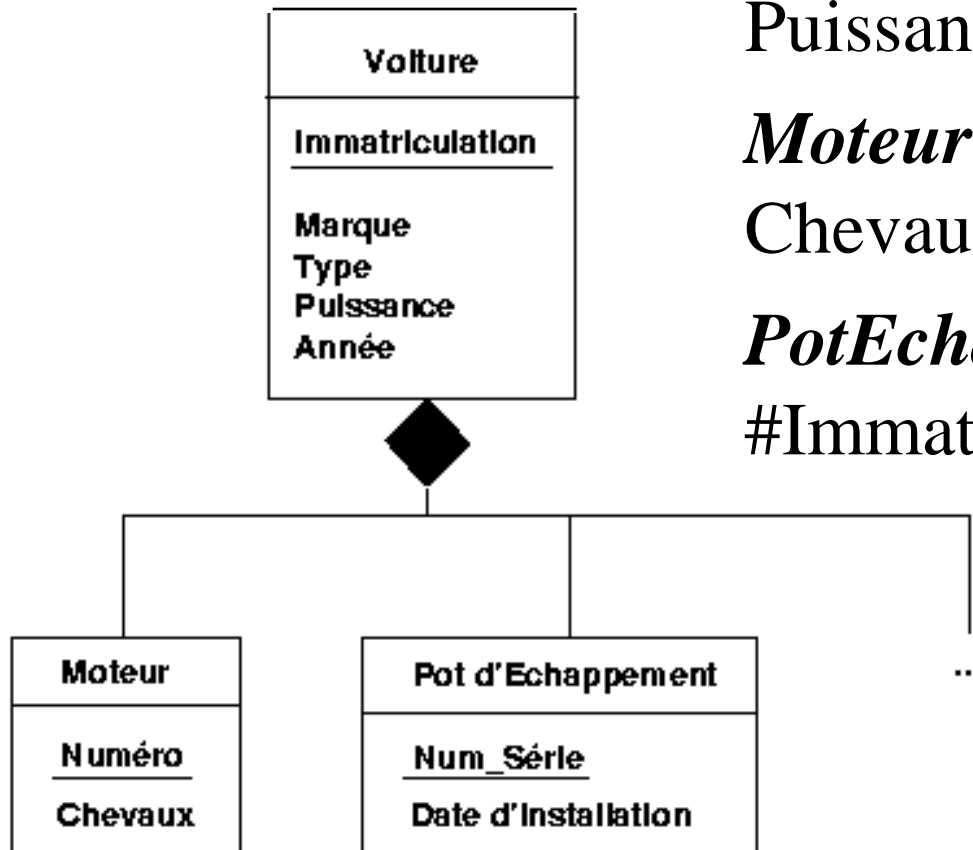
*#Bâtiment\_ID fait référence à Bâtiment\_ID de Bâtiment*

# Transformation de la composition UML

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année )

*Moteur* ( Numéro, #Immatriculation, Chevaux )

*PotEchappement* ( Num\_Série, #Immatriculation, DateInstallation )



# Dépendances fonctionnelles

**Ne pas oublier de définir les DF :**

*Accidente* ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

Auto\_ID → Nom, Prénom, Adresse

Immatriculation → Marque, Type, Puissance, Année

Type → Marque

Auto\_ID → Immatriculation et Immatriculation → Auto\_ID

*Voiture* ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

Immatriculation → Auto\_ID    ~~Auto\_ID → Immatriculation~~

+ les Dépendances fonctionnelles de Voiture

*Location* ( Auto\_ID, Immatriculation, Date )

Pas de dépendance non triviale

# Intégrité structurelle

- **Unicité des clés**

- **ensemble minimal d'attributs** dont la connaissance des valeurs permet d'identifier un nuplet unique de la relation considérée

- $R$  a pour clé  $K$  si :  $\forall t_1, t_2$  nuplets d'une instance de  $R$   
 $t_1.K \neq t_2.K$

- **Contraintes de référence**

- **contrainte référentielle** : contrainte d'intégrité portant sur une relation  $R$  qui consiste à imposer que la valeur d'un groupe d'attributs apparaissent comme valeur de clé dans une autre relation

- **clé étrangère** : un groupe d'attributs qui doit apparaître comme clé dans une autre relation

# Clé /Clé minimale /Surclé

***Accident*** ( Auto\_ID, Nom, Prénom, Adresse,  
Immatriculation, Marque, Type, Puissance, Année )

*Clés primaires possibles* : Auto\_ID ou Immatriculation

*Surclé* : (Auto\_ID, Immatriculation) + d'autres attributs

***Voiture*** ( Immatriculation, Marque, Puissance, Type, Année,  
Auto\_ID )

*Clé primaire* : Immatriculation

*Surclé* : (Immatriculation, Marque, Puissance, Type, Année, Auto\_ID)

***Location*** ( Auto\_ID, Immatriculation, Date )

*Clé primaire* : (Auto\_ID, Immatriculation, Date)

# Intégrité structurelle

- **Valeur nulle**

- valeur conventionnelle introduite dans une relation pour représenter une information inconnue ou inapplicable
- tout attribut peut prendre une valeur nulle **excepté les attributs de la clé primaire** (contrainte d'entité)

- **Contraintes de domaine**

contrainte d'intégrité qui impose qu'une colonne d'une relation doit comporter des valeurs vérifiant une assertion logique

# Langages d'interrogation

- **Algèbre relationnelle**

Pour comprendre comment le SGBD exécute les requêtes

- **Calcul relationnel à variable nuplet**

La base logique du langage SQL

- **Calcul relationnel à variable domaine**

La base logique pour les langages de requêtes graphiques

- **SQL (*Structured Query Language*)**

Ces langages sont équivalents : ils permettent de désigner les mêmes ensembles de données

# Chap IV - Algèbre relationnelle

## Opérations unaires :

- **sélection** des nuplets satisfaisant un certain prédicat

**Etudiant**(Etudiant\_ID, Nom, Prénom, Rue, Ville, Code-Postal, Téléphone, Fax, Email, NumAnnées)

$\sigma_{(Ville='Paris')}(\text{Etudiant})$

$\sigma_{(Ville='Paris') \wedge (NumAnnées \geq 2)}(\text{Etudiant})$

- **projection** : élimination de certains attributs d'une relation

$\Pi_{Nom, Prénom}(\text{Etudiant})$

$\Pi_{Nom, Prénom}(\sigma_{(Ville='Paris')}(\text{Etudiant}))$

# Exemples de résultats d'opérations unaires

## Relation *Enseignant*

L..	enseignant_id (...)	departement_id ...	nom (varchar)	prenom ...	grade (...)	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	2	1	NAIJA	Yosr	Moniteur			naija@lmasade.dauphine.fr
3	3	1	BAHRI	Afef	Moniteur			bahri@lmasade.dauphine.fr
4	4	1	LIMAM	Medhi	ATER			
5	5	5	MyTaylor	IsRich	Vacataire			
6	6	1	RIGAUX	Philippe	PROF			
7	7	1	CHAKHAR	Salem	ATER			chakhar@lamsade.dauphine.fr
8	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr

## Résultat de la sélection $\sigma_{(grade='MCF')}$ (*Enseignant*) :

L..	enseignant_id...	departement_id...	nom (varchar)	prenom...	grade ...	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr

## Résultat de la projection

### $\Pi_{Nom,Prénom}(Enseignant)$ :

L..	nom (varchar)	prenom...
1	MANOUVRIER	Maude
2	NAIJA	Yosr
3	BAHRI	Afef
4	LIMAM	Medhi
5	MyTaylor	IsRich
6	RIGAUX	Philippe
7	CHAKHAR	Salem
8	MURAT	Cécile

## Résultat de la requête

### $\Pi_{Nom,Prénom}(\sigma_{(grade='MCF')}(Enseignant))$ :

Ligne	nom (varchar)	prenom...
1	MANOUVRIER	Maude
2	MURAT	Cécile

# Opérations binaires

- **Union** : rassemblement des nuplets de 2 relations compatibles

*Enseignant*( Enseignant\_ID, Département\_ID, Nom, Prénom, Grade, Téléphone, Fax, Email )

$$\Pi_{\text{Nom,Prénom}}(\text{Etudiant}) \cup \Pi_{\text{Nom,Prénom}}(\text{Enseignant})$$

- **Différence** : des nuplets de 2 relations compatibles

$$\Pi_{\text{Nom,Prénom}}(\text{Enseignant}) - \Pi_{\text{Nom,Prénom}}(\text{Etudiant})$$

- **Produit cartésien** : combinaison des nuplets de 2 relations 

*Département*(Département\_ID, Nom\_Département)

Produit cartésien de *Enseignant* × *Departement* a pour schéma :

(Enseignant\_ID, Enseignant.Département\_ID, Nom, Prénom, Grade, Téléphone, Fax, Email, Département.Département\_ID, Nom\_Département)

# Exemple d'union et de différence

$\Pi_{\text{Nom,Prénom}}(\text{Etudiant}) \cup \Pi_{\text{Nom,Prénom}}(\text{Enseignant}) :$

Ligne	nom (varc...	prenom...
1	BAHRI	Afef
2	CHAKHAR	Salem
3	Debécé	Aude
4	GAMOTTE	Albert
5	HIBULAIRE	Pat
6	LIMAM	Medhi
7	MANOUVRIER	Maude
8	MURAT	Cécile
9	MyTaylor	IsRich
10	NAIJA	Yosr
11	ODENT	Jamal
12	RASLATABLE	Deborah
13	RIGAUX	Philippe

$\Pi_{\text{Nom,Prénom}}(\text{Enseignant}) - \Pi_{\text{Nom,Prénom}}(\text{Etudiant}) :$

Ligne	nom (varchar)	prenom ...
1	MANOUVRIER	Maude
2	MURAT	Cécile
3	MyTaylor	IsRich
4	RIGAUX	Philippe

# Produit cartésien

NSS	Nom	Prénom	Grade	Dept
12345	Manouvrier	Maude	MCF	1
45678	Toto	Titi	Prof	2

*La relation Enseignant*

Dept_ID	Nom_Dept_
1	Info
2	Math

*La relation Département*

NSS	Nom	Prénom	Grade	Dept	Dept_ID	Nom_Dept
12345	Manouvrier	Maude	MCF	1	1	Info
45678	Toto	Titi	Prof	2	1	Info
12345	Manouvrier	Maude	MCF	1	2	Math
45678	Toto	Titi	Prof	2	2	Math

*La relation Enseignant  $\times$  Département*

# Autres opérations

- **Renommage :**

$$\Pi_{A',B', \dots} (\mathbf{r}_{A \rightarrow A', B \rightarrow B', \dots})$$

- **Intersection :**

$$\mathbf{r} \cap \mathbf{s} = \mathbf{r} - (\mathbf{r} - \mathbf{s})$$

- **Théta-jointure :**

$$\mathbf{r} \infty_{\Theta} \mathbf{s} = \sigma_{\Theta} (\mathbf{r} \times \mathbf{s})$$

- **Jointure naturelle :**  $\mathbf{r}(\mathbf{R})$  et  $\mathbf{s}(\mathbf{S})$  avec  $\mathbf{R} \cap \mathbf{S} = \{A_1, A_2, \dots, A_n\}$

$$\mathbf{r} \infty \mathbf{s} = \Pi_{\mathbf{R} \cup \mathbf{S}} (\sigma_{(\mathbf{r}.A_1=\mathbf{s}.A_1) \wedge (\mathbf{r}.A_2=\mathbf{s}.A_2) \wedge \dots \wedge (\mathbf{r}.A_n=\mathbf{s}.A_n)} (\mathbf{r} \times \mathbf{s}))$$

# Exemple de renommage et d'intersection

$\Pi_{\text{Last\_Name, First\_Name}}(\text{Enseignant } \text{Nom} \rightarrow \text{Last\_Name}, \text{Prénom} \rightarrow \text{First\_Name}) :$

Ligne	last_name ...	first_name ...
1	MANOUVRIER	Maude
2	LIMAM	Medhi
3	MyTaylor	IsRich
4	RIGAUD	Philippe
5	MURAT	Cécile
6	CHAKHAR	Salem
7	NAIJA	Yosr
8	BAHRI	Afef

$\Pi_{\text{Nom, Prénom}}(\text{Enseignant}) \cap \Pi_{\text{Nom, Prénom}}(\text{Etudiant}) :$

Ligne	nom (varchar)	prenom...
1	BAHRI	Afef
2	CHAKHAR	Salem
3	LIMAM	Medhi
4	NAIJA	Yosr

# Exemple de produit cartésien

La relation *Enseignant* :

Ligne	enseignant_id ...	departement_id ...	nom (varchar)	prenom ...	grade ...	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	4	1	LIMAM	Medhi	ATER			
3	5	5	MyTaylor	IsRich	Vacataire			
4	6	1	RIGAUX	Philippe	PROF			
5	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr
6	7	1	CHAKHAR	Salem	ATER			chakhar@lamsade.dauphine.fr
7	2	1	NAIJA	Yosr	Moniteur			naija@lmasade.dauphine.fr
8	3	1	BAHRI	Afef	Moniteur			bahri@lmasade.dauphine.fr

La relation *Departement* :

Ligne	departement_id ...	nom_departement...
1	1	INFO
2	2	MATHS
3	3	GESTION
4	4	ECO
5	5	LANGUES
6	7	COMM
7	8	OPTION

*Enseignement* × *Departement* :

L..	enseignant_id...	departement_id (...	nom (varchar)	prenom...	grade...	telephone ...	fax ...	email (va...	departement_id...	nom_departement ...
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	1	INFO
2	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	2	MATHS
3	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	3	GESTION
4	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	4	ECO
5	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	5	LANGUES
6	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	7	COMM
7	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	8	OPTION
8	4	1	LIMAM	Medhi	ATER				1	INFO
9	4	1	LIMAM	Medhi	ATER				2	MATHS

# Exemple de jointure

La relation *Enseignant* :

Ligne	enseignant_id ...	departement_id ...	nom (varchar)	prenom ...	grade ...	telephone ...	fax ...	email (varchar)
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvrier@lmasade.dauphine.fr
2	4	1	LIMAM	Medhi	ATER			
3	5	5	MyTaylor	IsRich	Vacataire			
4	6	1	RIGAUX	Philippe	PROF			
5	8	1	MURAT	Cécile	MCF			murat@lamsade.dauphine.fr
6	7	1	CHAKHAR	Salem	ATER			chakhar@lamsade.dauphine.fr
7	2	1	NAIJA	Yosr	Moniteur			naija@lmasade.dauphine.fr
8	3	1	BAHRI	Afef	Moniteur			bahri@lmasade.dauphine.fr

La relation *Departement* :

Ligne	departement_id ...	nom_departement...
1	1	INFO
2	2	MATHS
3	3	GESTION
4	4	ECO
5	5	LANGUES
6	7	COMM
7	8	OPTION

*Enseignement*  $\infty_{\text{Departement\_ID}}$  *Departement* :

L...	enseignant_id...	departement_id ...	nom (varchar)	prenom...	grade ...	telephone ...	fax ...	email (va...	departement_id...	nom_departement ...
1	1	1	MANOUVRIER	Maude	MCF	4185	4091	manouvri...	1	INFO
2	4	1	LIMAM	Medhi	ATER				1	INFO
3	5	5	MyTaylor	IsRich	Vacata...				5	LANGUES
4	6	1	RIGAUX	Philippe	PROF				1	INFO
5	8	1	MURAT	Cécile	MCF			murat@la...	1	INFO
6	7	1	CHAKHAR	Salem	ATER			chakhar...	1	INFO
7	2	1	NAIJA	Yosr	Moniteur			naija@lm...	1	INFO
8	3	1	BAHRI	Afef	Moniteur			bahri@lm...	1	INFO

# Division

**Requête qui contient le terme « pour tous »**

Soient  $r(R)$  et  $s(S)$  avec  $S \subseteq R$

**la relation  $r \div s$  a pour schéma  $R - S$**

un nuplet  $t$  appartient à  $r \div s$  si :

①  $t \in \Pi_{R-S}(r)$

②  $\forall t_s$  nuplet de  $s$ ,  $\exists t_r$  dans  $r$  qui satisfait :

◆  $t_r(S) = t_s(S)$

◆  $t_r(R-S) = t$

$$r \div s = \Pi_{R-S}(r) - \Pi_{R-S} [ (\Pi_{R-S}(r) \times s) - \Pi_{R-S,S}(r) ]$$

# Division

La relation *Enseignement* :

L..	enseignement_id ...	departement_id ...	intitule (varchar)	description (varchar)
1	①	①	Bases de Données	Niveau Licence : Modélisation E/A et UML, Modèle relationnel, Algèbre Relation...
2	②	①	Mise à Niveau Informatique	Pour les étudiants de GMI entrant directement en IUP2: Architecture, Algorithm...
3	③	①	Mise à Niveau Bases de Données	Pour les étudiants de DESS ID ou DEA127 - Programme Licence et Maîtrise en ...
4	④	⑤	Anglais	

La relation  
*Inscription* :

Ligne	etudiant_id (int4)	enseignement_id (int4)	departement_id (int4)	date_inscription (date)
1	①	①	①	2004-02-25
2	①	②	①	2004-07-22
3	3	2	1	2004-07-22
4	5	2	1	2004-07-22
5	4	2	1	2004-07-22
6	①	③	①	2004-07-22
7	①	④	⑤	2004-07-22
8	2	4	5	2004-07-22

$\Pi_{Etudiant\_ID, Enseignement\_ID, Departement\_ID} (Inscription) \div \Pi_{Enseignement\_ID, Departement\_ID} (Enseignement) :$

Ligne	etudiant_id (int4)
1	1

# Contraintes et DF

- **Expressions des contraintes d'intégrité référentielle :**

$$\Pi_{\text{Département\_ID}}(\text{Enseignant}) \subseteq \Pi_{\text{Département\_ID}}(\text{Département})$$

$$\Pi_{\text{Département\_ID}}(\text{Département}) - \Pi_{\text{Département\_ID}}(\text{Enseignant}) = \emptyset$$

- **Expressions des dépendances fonctionnelles :**

$$X \rightarrow Y \Leftrightarrow \forall r \text{ et } \forall t_1, t_2 \in r \text{ on a :}$$

$$\Pi_X(t_1) = \Pi_X(t_2) \Rightarrow \Pi_Y(t_1) = \Pi_Y(t_2)$$

# Chap V - Calcul relationnel

- Langage déclaratif (ou non-procédural): le quoi
- Algèbre relationnelle (procédural) : le comment
- Calcul à variable nuplet :

$\{ t \mid P(t) \}$  avec  $P(t)$  tel que :

$r(t)$  :  $t$  est un nuplet de  $r$

$t.att_1 = valeur_1$

$t.att_1 > s.att_2 \dots$

$\{t \mid Etudiant(t) \wedge (t.Ville='Paris') \wedge (t.NumAnnées \geq 2)\}$

$\{t.Nom, t.Prénom \mid Etudiant(t) \wedge (t.Ville='Paris')\}$

# Quantificateurs

- $\exists t (Q(t))$  : il existe un nuplet dans la base qui vérifie  $Q(t)$
- $\forall t (Q(t))$  : pour tous les nuplets,  $Q(t)$  est vrai
- Variable liée
- Variable libre
- Expression saine
- Expression non saine

# Expression des opérateurs algébriques

- $\sigma_{\Theta}(\mathbf{r}) :$
- $\prod_{A_1, A_2, \dots, A_n}(\mathbf{r}) :$
- $\mathbf{r} \cup \mathbf{s} :$
- $\mathbf{r} - \mathbf{s} :$
- $\mathbf{r} \times \mathbf{s} :$
- $\mathbf{r} \infty \mathbf{s} :$
- $\mathbf{r} \div \mathbf{s} :$

# Division

## Division :

Livre(ISBN, Titre, Editeur)

Emprunt(EmpruntID, ISBN, DateEmprunt, EtudiantID)

Etudiant(EtudiantID, Nom, Prenom)

« Quels sont les noms et prénoms des étudiants ayant emprunté tous les livres ? »

*La requête retourne les valeurs des att. Nom et Prenom des nuplets t de la relation Etudiant tq :*

$$\{ \overbrace{t.Nom, t.Prenom} / Etudiant(t) \wedge [ \forall u \neg ( Livre(u) ) \vee \underbrace{ ( \exists v Emprunt(v) \wedge (v.Etudiant\_ID=t.Etudiant\_ID) \wedge (v.ISBN=u.ISBN) ) }_{\text{Il existe un nuplet v dans Emprunt associant le livre u à l'étudiant t}} ] \}$$

*Quel que soit u, u n'est pas dans Livre ou (u est dans Livre et) il existe v ...*

# Calcul à variable domaine

$$\{ \langle x_1, x_2, \dots, x_n \rangle \mid P(x_1, x_2, \dots, x_n) \}$$

avec  $x_1, x_2, \dots, x_n$  variable domaine et  $P$  formule

**Exemples :**

**Marin**(Mid, Nom, Grade, Date-Nais)

**Bateau**(Bid, Bnom, Couleur)

**Reservation**(Mid, Bid, Date)

Nom et grade des marins :  $\{ \langle n, g \rangle \mid \exists id, dn \text{ Marin}(id, n, g, dn) \}$

Marins ayant réservé tous les bateaux :

$$\{ \langle i, n, g, dn \rangle \mid \text{Marin}(i, n, g, dn) \wedge [ \forall b, bn, c \neg \text{Bateau}(b, bn, c) \vee (\exists m, br, d \text{ Reservation}(m, br, d) \wedge (m = i) \wedge (br = b) ) ] \}$$

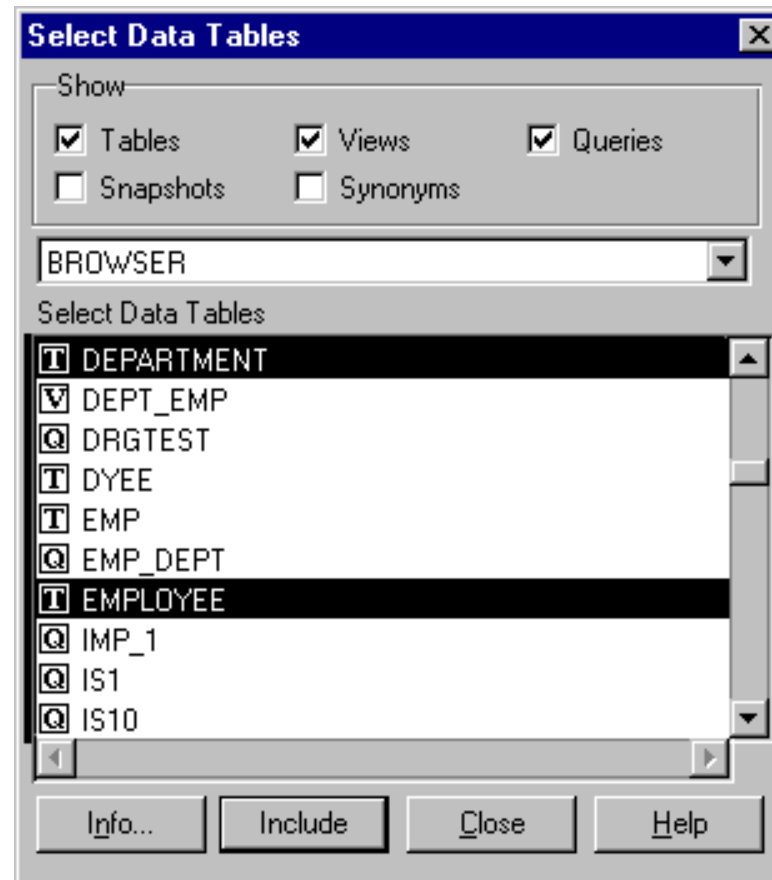
# QBE

## *Query By Exemple :*

- langage de requête graphique
- mise en œuvre du calcul à variable domaine

## *Oracle Query Builder*

*Etape 1 : sélection des relations de la requête*



# QBE

*Etape 2 :  
sélection des  
attributs de  
la requête*

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	DEPARTMENT	<input type="checkbox"/>	T
<input type="checkbox"/>		DEPARTMENT ID	789	
<input checked="" type="checkbox"/>		NAME	A	
<input type="checkbox"/>		LOCATION ID	789	

SAL > 500
↵

<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	EMPLOYEE	<input type="checkbox"/>	T
<input type="checkbox"/>		EMPLOYEE ID	789	
<input checked="" type="checkbox"/>		LAST NAME	A	
<input checked="" type="checkbox"/>		FIRST NAME	A	
<input type="checkbox"/>		MIDDLE INITIAL	A	
<input type="checkbox"/>		JOB ID	789	
<input type="checkbox"/>		MANAGER ID	789	<input type="checkbox"/>
<input type="checkbox"/>		HIRE DATE		<input type="checkbox"/>
<input checked="" type="checkbox"/>		SALARY	789	
<input type="checkbox"/>		COMMISSION	789	
<input type="checkbox"/>		DEPARTMENT ID	789	

# QBE

*Etape 3 :*  
*possibilité*  
*d'appliquer*  
*des fonction*  
*d'agrégation*  
*sur les nuplets*  
*résultats*

	NAME	LAST_NAME	FIRST_NAME	SALARY
1	ACCOUNTING	KING	FRANCIS	\$5,000
2		CLARK	CAROL	\$2,450
3		MILLER	BARBARA	\$1,300
4	Total			\$8,750
5	OPERATIONS	SOMMERS	DENISE	\$1,850
6		LEWIS	RICHARD	\$1,800
7	Total			\$3,650
8	RESEARCH	ALBERTS	CHRIS	\$3,000
9		FORD	JENNIFER	\$3,000
10		SCOTT	DONALD	\$3,000
11		FISHER	MATTHEW	\$3,000
12		JONES	TERRY	\$2,975
13		ROBERTS	GRADE	\$2,875
14		ADAMS	DIANE	\$1,100

# Expression des contraintes en logique des prédicats

**Agence**(nom\_banque,ville ...)

**Emprunt**(num\_emprunt,nom\_banque,num\_client, montant ...)

**Compte**(nom\_banque,num\_client,num\_compte,solde ...)

*« Chaque emprunteur possède un compte en banque dans l'agence dont le solde est au minimum égal à la moitié de son emprunt »*

$$\{ \neg ( \exists e \text{ Emprunt}(e) \wedge \neg ( \exists c \text{ Compte}(c) \\ \wedge (c.\text{num\_client}=e.\text{num\_client}) \\ \wedge (c.\text{nom\_banque} = e.\text{nom\_banque}) \\ \wedge (c.\text{solde} \geq (e.\text{montant} / 2) \\ ) \\ ) \\ ) \\ \}$$

# Chap VI - Algèbre relationnelle étendue

- **Projection généralisée :**

ajout d'expressions arithmétiques dans une projection

$$\Pi_{\text{Nom\_Client}, (\text{Crédit} - \text{Débit})} (\text{Compte\_en\_Banque})$$

- **Jointure externe (*outer-join*) :**

- jointure externe à gauche :  $]^\infty$

- jointure externe à droite :  $^\infty[$

- jointure externe :  $]^\infty[$

$R ]^\infty S \Rightarrow R \infty S$  et conservation des attributs des nuplets de R qui ne joignent avec aucun nuplet de S (les valeurs des attributs de S sont mises à NULL)

*Personnel*

Nom_Employé	Ville
Tom	Marseille
Jerry	Paris
Alex	Limoges
Marthe	Perpignan

*Employé*

Nom_Employé	Filiale	Salaire
Tom	SUD_EST	10000
Jerry	IDF	25000
Sophie	IDF	15000
Marthe	SUD_OUEST	12000

*Personnel*

]∞

*Employé*

Nom_Employé	Ville	Filiale	Salaire
Tom	Marseille	SUD_EST	10000
Jerry	Paris	IDF	25000
<b>Alex</b>	<b>Limoges</b>	<b>NULL</b>	<b>NULL</b>
Marthe	Perpignan	SUD_OUEST	12000

*Personnel*

∞[

*Employé*

Nom_Employé	Ville	Filiale	Salaire
Tom	Marseille	SUD_EST	10000
Jerry	Paris	IDF	25000
<b>Sophie</b>	<b>NULL</b>	<b>IDF</b>	<b>15000</b>
Marthe	Perpignan	SUD_OUEST	12000

# Fonction d'agrégation

- Somme des places disponibles dans l'Université

**Sum**<sub>Capacité</sub>(Salle)

- Nombre moyen de places disponibles dans les salles de l'Université

**Avg**<sub>Capacité</sub>(Salle)

- Nombre d'étudiants à l'Université

**Count**<sub>Etudiant\_ID</sub>(Etudiant)

- Capacité de la plus petite salle

**Min**<sub>Capacité</sub>(Salle)

- Nombre d'enseignants par départements :

**Count**<sub>Enseignant\_ID</sub>(Enseignant  $\bowtie$  Département)

Nom\_Département

# Mise à jour de la base

- **Insertion**

$$\text{Salle} \leftarrow \text{Salle} \cup \{(\ll \mathbf{B} \gg, \ll \mathbf{038} \gg, \mathbf{15})\}$$

- **Suppression**

$$\text{Salle} \leftarrow \text{Salle} - \sigma_{\text{Salle} \leq 10} (\text{Salle})$$

- **Mise à jour** : utilisation de la projection généralisée

$$r \leftarrow \Pi_{\text{Etudiant\_ID}} [\sigma_{(\text{Nom}=\text{' Dupont '}) \wedge (\text{Prénom}=\text{' Jacques '})} (\text{Etudiant})]$$

$$\text{Etudiant} \leftarrow \sigma_{(\text{Etudiant.Etudiant\_ID} \neq r.\text{Etudiant\_ID})} (\text{Etudiant})$$

∪

*Mise à jour du  
téléphone*

$$\Pi_{\text{Etudiant\_ID}, \text{Nom}, \text{Prénom}, \text{Rue}, \text{Ville}, \text{Code-Postal},$$

$$\text{Téléphone} \leftarrow \ll \mathbf{45\ 12\ 45\ 86} \gg, \text{Fax}, \text{Email}, \text{NumAnnées}$$

$$[\sigma_{(\text{Etudiant.Etudiant\_ID} = r.\text{Etudiant\_ID})} (\text{Etudiant}) ]$$

# Vue

Table virtuelle dont le schéma et les instances sont dérivés de la base réelle par une requête et qui est utilisée pour :

- Cacher certaines informations à un groupe d'utilisateurs
- Faciliter l'accès à certaines données

**create view** nom\_vue **as** < requête >

Exemple :

**create view** Info\_Non\_Confidentielle\_Etudiant

**as**  $\Pi$  Etudiant\_ID, Nom, Prénom, Email (Etudiant)

# Chap VII - SQL

## *Structured Query Language*

- **SQL2** : standard adopté en 1992
- **SQL3** : extension de SQL2 avec "gestion" d'objets

## **SQL :**

- **Langage de Manipulation de Données (DML)** : interroger et modifier les données de la base
- **Langage de Définition de Données (DDL)** : définir le schéma de la base de données
- **Langage de contrôle d'accès aux données**

# Bibliographie

- *SQL2 - Application à Oracle, Access et RDB*  
**Pierre DELMAL, 2ème Edition, De Boeck Université, 1998**  
**BU: 005.74 SQL**
- *SQL Pour Oracle (avec exercices corrigés)*  
**Christian Soutou, Eyrolles, 2005 – BU: 005.72 SOU**
- *Initiation à SQL (cours et exercices corrigés)*  
**Philip J. Pratt, Eyrolles, 2001 – BU : 005.72 SQL**
- *SQL (cours et exercices corrigés)*  
**Frédéric Brouad et Christian Soutou, Coll. Synthex, Pearson Education, 2005 – BU : 005.72 SQL**
- *Oracle PL/SQL - Précis & concis*  
**Steven Feuerstein, Bill Pribyl et Chip Dawes, O 'Reilly, 2000**

# DML

```
SELECT [DISTINCT] *  
FROM table_1 [synonyme_1], table_2 [synonyme_1], ...  
[WHERE prédicat_1  
AND [ou OR] prédicat_2 ...]
```

```
SELECT [DISTINCT] exp_1 [AS nom_1], exp_2 ...  
FROM table_1 [synonyme_1], table_2 [synonyme_1], ...  
[WHERE prédicat_1  
AND [ou OR] prédicat_2 ...]
```

# DML

```
SELECT Nom, Prénom  
FROM Etudiant  
WHERE Ville = ' Paris ' ;
```

```
SELECT Nom, Prénom  
FROM Etudiant  
WHERE Ville = ' Paris '  
AND Nom  
LIKE ' _AR% ' ;
```

```
SELECT Nom, Prénom  
FROM Etudiant  
WHERE Fax IS NULL;
```

```
SELECT Intitulé,  
(NbSeances*3) AS NbHeures  
FROM Cours  
WHERE (NbSeances*3)  
BETWEEN 24 AND 27 ;
```

```
SELECT Nom, Prénom  
FROM Enseignant  
WHERE Département_ID IN  
( ' INFO ', ' MATH ', ' ECO ' )
```

# DML

## Prédicats du WHERE de la forme :

exp1 = exp2

exp1 != exp2

exp1 > exp2

exp1 < exp2

exp1 <= exp2

exp1 >= exp2

exp1 BETWEEN exp2 AND exp3

exp1 LIKE exp2

exp1 IN (exp2, exp3, ...)

exp1 NOT IN (exp2, exp3, ...)

exp1 IS NULL

exp1 IS NOT NULL

exp *op* ANY (SELECT ...)

exp *op* ALL (SELECT ...)

avec *op* tel que =, !=, <, > ...

exp IN (SELECT ...)

exp NOT IN (SELECT ...)

```
SELECT Intitulé,  
FROM Cours  
WHERE NbSeances <=  
( SELECT AVG(NbSeances)  
FROM Cours);
```

# DML

## Clause EXISTS :

- Retourne VRAI si au moins un nuplet est renvoyé par la requête
- FAUX si aucun nuplet n'est retourné.
- La valeur NULL n'a aucun effet sur le booléen résultat

```
SELECT Nom, Prénom  
FROM Enseignant E  
WHERE NOT EXISTS  
( SELECT *  
FROM Reservation_Salle S  
WHERE S.Enseignant_ID = E.Enseignant_ID  
);
```

# DML

## Fonctions de groupe :

COUNT, MIN, MAX, AVG, SUM, ORDER BY, GROUP BY

```
SELECT COUNT(*)
```

```
FROM Etudiant ;
```

```
SELECT AVG(Capacité), SUM(Capacité)
```

```
FROM Salle ;
```

```
SELECT Département_ID, Nom, Prénom
```

```
FROM Enseignant
```

```
ORDER BY Département_ID DESC, Nom, Prénom ;
```

```
SELECT Département_ID, COUNT(*)
```

```
FROM Réservation_Salle
```

```
GROUP BY Département_ID HAVING COUNT(*) >=4 ;
```

# DML

## Jointure :

```
SELECT Nom, Prénom, Nom_Département  
FROM Enseignant E, Département D  
WHERE E.Département_ID = D.Département_ID ;
```

## Jointure externe : sous Oracle

```
SELECT Nom, Prénom, Nom_Département  
FROM Enseignant E, Département D  
WHERE E.Département_ID = D.Département_ID (+);
```

**S'il existe des enseignants attaché à aucun département, la valeur de Département\_ID sera NULL.**

**En SQL2 : [RIGHT | LEFT | FULL] OUTER JOIN**

# DML

## Opérateurs ensemblistes :

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' INFO '  
INTERSECT
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' MATH '
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' INFO '  
UNION
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' MATH '  
ORDER BY Nom,Prénom
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' INFO '  
MINUS
```

```
SELECT Nom,Prénom FROM Enseignant WHERE Département_ID = ' MATH '
```

**MINUS = EXCEPT en standard SQL2**

# DML

## Division :

Livre(ISBN, Titre, Editeur)

Emprunt(EmpruntID, ISBN, DateEmprunt, EtudiantID)

Etudiant(EtudiantID, Nom, Prenom)

« Quels livres ont été empruntés par tous les étudiants? »

$$\{ \text{t.Titre} / \text{Livre}(t) \wedge \neg [ \exists u \text{ Etudiant}(u) \wedge \neg (\exists v \text{ Emprunt}(v) \wedge (v.\text{Etudiant\_ID}=u.\text{Etudiant\_ID}) \wedge (v.\text{ISBN}=t.\text{ISBN})) ] \}$$

*Il n'y a pas de mot-clé "quel que soit" en SQL2*

```
SELECT t.Titre FROM Livre t WHERE NOT EXISTS
  ( SELECT * FROM Etudiant u WHERE NOT EXISTS
    ( SELECT * FROM Emprunt v
      WHERE u.EtudiantID=v.EtudiantID AND
        v.ISBN=t.ISBN
    )
  );
```

# DML

- **Insertion**

**INSERT INTO table(col1, col2, ... coln)**

**VALUES (val1, val2, ... valn)**

**INSERT INTO table(col1, col2, ... coln)**      *Sous Oracle*

**SELECT**

- **Suppression**

**DELETE FROM table**

**WHERE prédicat**

- **Mise à jour**

**UPDATE table**

**SET col1 = exp1, col2 = exp2 WHERE prédicat**

- **Transactions : COMMIT, ROLLBACK [TO], SAVE POINT**

# DDL

```
CREATE TABLE table (col1 type 1 [NOT NULL] ,  
                    col2 type2 [NOT NULL] ...  
                    )
```

**Contraintes :**

**CONSTRAINT** *nom\_contrainte*

**PRIMARY KEY** (liste attributs clé primaire)

| **NOT NULL** *immédiatement après la déclaration de l'attribut*

| **CHECK** (condition) *après la déclaration de l'attribut*

/ **UNIQUE** *après la déclaration de l'attribut*

| **FOREIGN KEY** (clé étrangère)

**REFERENCES** nom\_table (liste-colonne)

```
CREATE TABLE table  
AS SELECT ...
```

# DDL

**CREATE TABLE Enseignant**

```

(
  Enseignant_ID          integer,
  Departement_ID        integer NOT NULL,
  Nom                   varchar(25) NOT NULL,
  Prenom                varchar(25) NOT NULL,
  Grade                 varchar(25)
  CONSTRAINT CK_Enseignant_Grade
  CHECK (Grade IN ('Vacataire', 'Moniteur', 'ATER', 'MCF', 'PROF')),
  Telephone              varchar(10) DEFAULT NULL,
  Fax                   varchar(10) DEFAULT NULL,
  Email                  varchar(100) DEFAULT NULL,
  CONSTRAINT PK_Enseignant PRIMARY KEY (Enseignant_ID),
  CONSTRAINT "FK_Enseignant_Departement_ID"
  FOREIGN KEY (Departement_ID)
  REFERENCES Departement (Departement_ID)
  ON UPDATE RESTRICT ON DELETE RESTRICT
);

```

*Contrainte  
de domaine*

*Définition de la  
clé primaire*

*Définition d'une clé  
étrangère*

## **CREATE TABLE Reservation**

```
(  
Reservation_ID integer,  
Batiment        varchar(1) NOT NULL,  
Numero_Salle    varchar(10) NOT NULL,  
Enseignement_ID integer NOT NULL,  
Departement_ID  integer NOT NULL,  
Enseignant_ID   integer NOT NULL,  
Date_Resa       date NOT NULL DEFAULT CURRENT_DATE,  
Heure_Debut     time NOT NULL DEFAULT CURRENT_TIME,  
Heure_Fin       time NOT NULL DEFAULT '23:00:00',  
Nombre_Heures   integer NOT NULL,  
CONSTRAINT PK_Reservation PRIMARY KEY (Reservation_ID),  
CONSTRAINT "FK_Reservation_Salle" FOREIGN KEY (Batiment,Numero_Salle) REFERENCES  
Salle (Batiment,Numero_Salle) ON UPDATE RESTRICT ON DELETE RESTRICT,  
CONSTRAINT "FK_Reservation_Enseignement" FOREIGN KEY (Enseignement_ID,Departement_ID)  
REFERENCES Enseignement (Enseignement_ID,Departement_ID) ON UPDATE RESTRICT ON  
DELETE RESTRICT,  
CONSTRAINT "FK_Reservation_Enseignant" FOREIGN KEY (Enseignant_ID) REFERENCES  
Enseignant (Enseignant_ID) ON UPDATE RESTRICT ON DELETE RESTRICT,  
CONSTRAINT CK_Reservation_Nombre_Heures CHECK (Nombre_Heures >=1),  
CONSTRAINT CK_Reservation_HeureDebFin  
CHECK (Heure_Debut < Heure_Fin)  
);
```

# DDL

```
CREATE ASSERTION <nom contrainte>  
[ {BEFORE COMMIT |  
  AFTER {INSERT | DELETE | UPDATE[OF (Attributs)]} ON  
  <Relation>} ...]  
CHECK <Condition>  
[FOR [EACH ROW OF] <Relation> ]
```

```
CREATE ASSERTION CA_Place_Université  
BEFORE COMMIT  
CHECK( (SELECT SUM(Capacité) FROM Salle)  
  >= (SELECT COUNT(*) FROM Etudiant)  
  )
```

# DDL

**CREATE [OR REPLACE] TRIGGER nom {BEFORE | AFTER}  
événement\_déclencheur ON nom\_table  
[FOR EACH ROW]  
[WHEN (condition) ]  
bloc PL/SQL *sous Oracle*  
| inst\_de\_suppr | inst\_de\_modif | instr\_d\_ajout | ERROR *en SQL2***

*événement\_déclencheur* = INSERT, UPDATE, DELETE

- Déclencheur de *niveau instruction* : pas de clause FOR EACH ROW
- Déclencheur de *niveau ligne* : variables liens *:new* et *:old*
  - INSERT : valeurs à insérer dans *:new.nom\_colonne*
  - UPDATE : valeur originale dans *:old.nom\_colonne*, nouvelle valeur dans *:new.nom\_colonne*
  - DELETE : valeur en cours de suppression *:old.nom\_colonne*

# DDL

```
CREATE OR REPLACE TRIGGER Enseignant_Actif
BEFORE DELETE ON Enseignant
FOR EACH ROW
declare
    counter number;
begin
    SELECT count(*) INTO counter
    FROM Enseignements
    WHERE Enseignant_ID = :old.Enseignant_ID;
    if counter > 0 then
        raise_application_error (-20800, 'Enseignant actif ne
        pouvant être supprimé');
    end if;
end;
```

# DDL

```
CREATE OR REPLACE TRIGGER UPD_salaire_personnel
BEFORE UPDATE salaire ON Personnel
FOR EACH ROW
WHEN (:old.salaire > :new.salaire)
declare
    salaire_diminution EXCEPTION;
begin
    raise salaire_diminution ;
    when salaire_diminution then
        raise_application_error(-20001, 'Le salaire ne peut pas
diminuer ')
end;
```

# DDL

## Sous PostgreSQL :

```
CREATE OR REPLACE FUNCTION GetSalleCapaciteSuperieurA(int)  
RETURNS SETOF Salle  
AS '  
    SELECT * FROM Salle WHERE Capacite > $1;  
    '  
LANGUAGE SQL;  
  
SELECT * FROM GetSalleCapaciteSuperieurA(300) ;
```

# DDL

```
CREATE OR REPLACE FUNCTION FunctionTriggerReservation()  
RETURNS trigger AS  
' DECLARE  
resa Reservation.Reservation_ID%TYPE;  
BEGIN  
SELECT INTO resa Reservation_ID  
FROM Reservation  
WHERE ...  
IF FOUND THEN RAISE EXCEPTION "Réservation impossible, salle  
occupée à la date et aux horaires demandés";  
ELSE RETURN NEW;  
END IF;  
END;'  
LANGUAGE 'plpgsql';
```

# DDL

Sous PostgreSQL :

```
CREATE TRIGGER InsertionReservation  
BEFORE INSERT ON Reservation  
FOR EACH ROW  
EXECUTE PROCEDURE  
FunctionTriggerReservation();
```

# DDL

**ALTER TABLE table**

**ADD (col1 type1, col2 type2 ...)**

**| MODIFY (col1 type1, col2 type2 ...)**

**| DROP PRIMARY KEY**

**| DROP CONSTRAINT nom\_contrainte**

**DROP TABLE table**

**CREATE VIEW vue (col1, col2)**

**AS SELECT ...**

**DROP VIEW vue**

**CREATE [UNIQUE] INDEX nom\_index ON table (col1,col ...)**

# Embedded SQL

Utilisation de commandes SQL à l'intérieur d'un langage hôte :

- Commandes SQL remplacée par des appel de fonctions du langage hôte par le précompilateur.
- Commandes SQL reconnues par **EXEC SQL**

```
/* déclaration de variables hôtes */  
EXEC SQL BEGIN DECLARE SECTION  
char d_name[20];  
char d_id;  
EXEC SQL END DECLARE SECTION  
...  
EXEC SQL INSERT INTO Department  
VALUES (:d_id, :d_name);
```

# Embedded SQL

Gestion des erreurs :

```
EXEC INCLUDE SQLCA;
```

```
...
```

```
EXEC SQL WHENEVER SQLERROR GOTO erreur
```

```
...
```

```
erreur :
```

```
    printf(`erreur : les transactions en  
cours vont être annulées'\n');
```

```
    EXEC SQL ROLLBACK WORK RELEASE;
```

```
    exit(1);
```

# Embedded SQL

Gestion de curseur :

```
/* Déclaration d'un curseur pour manipuler la
   table Department */
EXEC SQL DECLARE c1 CURSOR FOR
   SELECT * FROM Department ;

/* Ouverture du curseur */
EXEC SQL OPEN c1;

/* Lecture de la première ligne de la table */
EXEC SQL FETCH c1 INTO :d_id, :d_name ;
printf(` `Nom du département %s, identifiant :
   %s\n' ', d_name, d_id);

/* fermeture du curseur */
EXEC SQL CLOSE c1;
```

# Middleware d'accès aux bases de données

- Architecture logicielle définissant une interface standard d'accès aux SGBD
  - A chaque SGBD correspond un pilote (*driver*)
    - ⇒ Indépendance vis à vis du SGBD par simple configuration du pilote
    - ⇒ Possibilité pour un même programme d'interroger différentes bases de données dans différents SGBD
1. *Java DataBase Connectivity (JDBC)*
  2. *Open DataBase Connectivity (ODBC) - Middleware propriétaire (Windows)*

# JDBC/ODBC

- Appel des fonctions de l'API :  
⇒ lien entre l'application et le **gestionnaire de pilotes**
- **Pilote** = librairie qui contient les appels JDBC/ODBC et traduit les requêtes en requêtes propres au SGBD
- **Gestionnaire de pilotes (ODBC)** = librairie qui charge les pilotes associés à chaque **source de données** (BD + SGBD)

# JDBC

## Exemple de programme Java sur la base exemple :

```
import java.sql.*; // Tout ce qui est nécessaire pour JDBC
import java.text.*; // Pour formater les dates
import java.util.*;

public class TestJDBCPostgresql
{
    Connection db=null; // Variable de connexion
    Statement sql=null; // Variable pour un ordre SQL
    DatabaseMetaData dbmd; // Méta-données nécessaires
                          // au pilote.
```

# JDBC

```
public TestJDBCPostgresql(String argv[])
    throws ClassNotFoundException, SQLException,
           java.io.IOException
{ String database = argv[0];
  String username = argv[1];
  String password = argv[2];

  // Récupération du pilote du SGBD (ici PostgreSQL)
  Class.forName("org.postgresql.Driver");

  // Connexion à la base de données
  db = DriverManager.getConnection("jdbc:postgresql:"
                                   +database, username,
                                   password);

  // Si AUTOCOMMIT=true => le SGBD exécute COMMIT
  // après chaque requête
  // Si AUTOCOMMIT= false => le SGBD attend un COMMIT
  // explicite
  db.setAutoCommit(false);
```

# JDBC

```
dbmd = db.getMetaData(); // Récupération des méta-données
// Pour tester que la connexion est OK
System.out.println("Connection to SGBD «
    + dbmd.getDatabaseProductName()+ " version «
    + dbmd.getDatabaseProductVersion()+ " database "
    + dbmd.getURL()+ " \nusing " + dbmd.getDriverName()
    + " version "+ dbmd.getDriverVersion()+ " " +
    "successful.\n");

// Création d'un ordre SQL pour créer une relation
sql = db.createStatement();
String sqlText = "CREATE TABLE ... ";
System.out.println("Executing this command: "+sqlText+"\n");
sql.executeUpdate(sqlText);
// Ne pas oublier le COMMIT si AUTOCOMMIT = false
db.commit();

// Code équivalent pour un INSERT, UPDATE, DELETE
```

# JDBC

```
// Exemple de requête de sélection
ResultSet rset = sql.executeQuery("SELECT Capacité FROM
    Reservation " + "WHERE Batiment='B' " + "AND
    Numero_Salle='022'");

// Parcours des nuplets résultat
while (rset.next()) {
    System.out.println("Nom de la table : (généralement vide
    bug du driver) "
    + rset.getMetaData().getTableName(1));
    System.out.println("Type de la colonne : "
    + rset.getMetaData().getColumnTypeName(1));
    System.out.println("Nom de la colonne : "
    + rset.getMetaData().洗getColumnName(1) + "\n");

    //Numérotation des colonnes à partir de 1 (et non de 0!!)
    System.out.println("Capacité de la salle B020 : "
    + rset.getInt(1) + "\n");
}
```

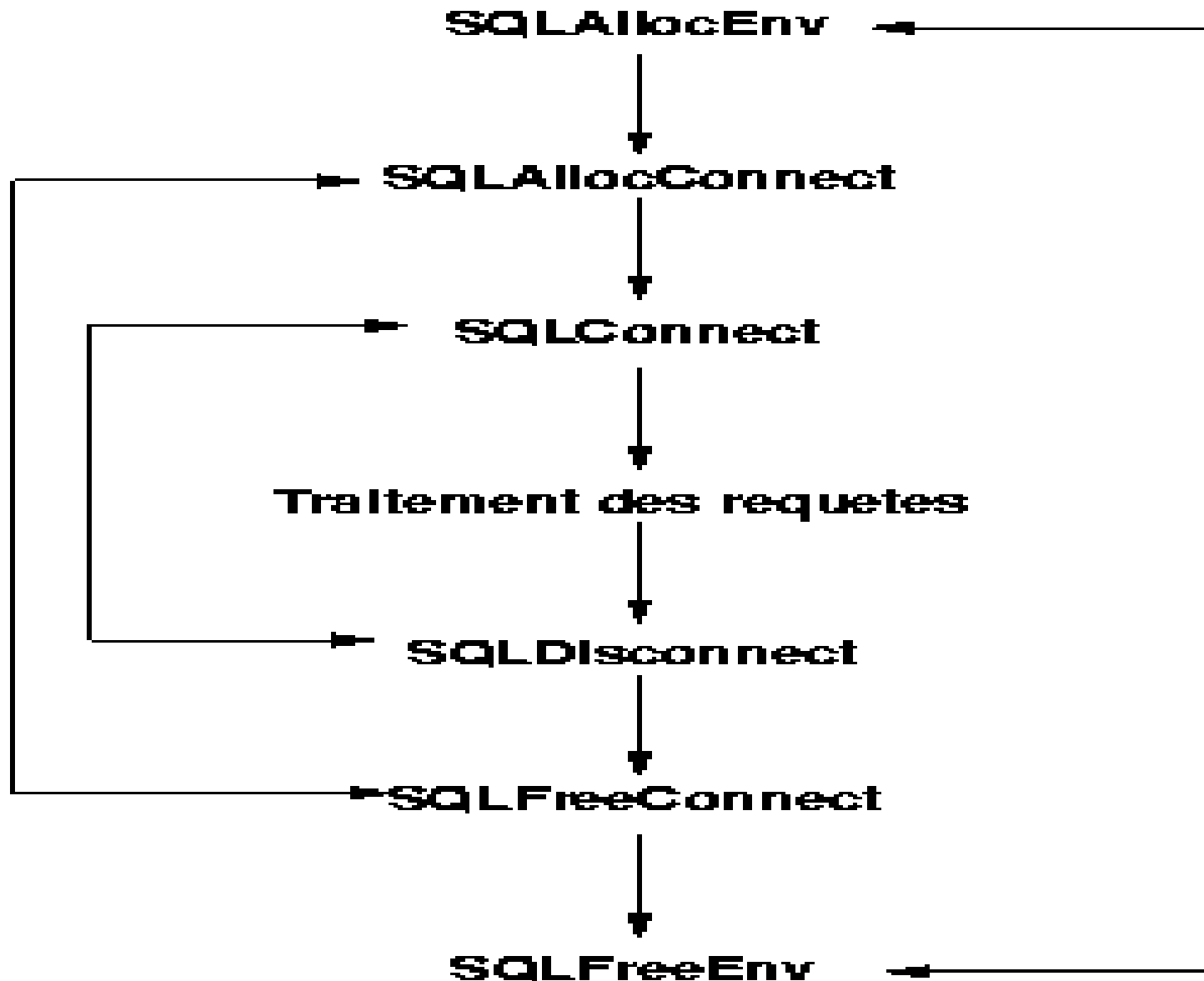
# JDBC

```
// Création d'un ordre SQL paramétré
// Code SQL de la requête
sqlText = "INSERT INTO Salle VALUES (?, ?, ?)";
// Préparation de l'ordre SQL - non encore exécuté
//car il manque des paramètres
PreparedStatement ps = db.prepareStatement(sqlText);
String [] NumBatiment = {"A", "B", "C", "P", "D"};
String [] NumSalle = {"208", "026", "405", "340", "120"};
int lenNB = NumBatiment.length;
for (int i=0, c=30 ; (i<lenNB) && (c<35) ;c++,i++)
    {ps.setString(1,NumBatiment[i]); // Paramètre 1
     ps.setString(2,NumSalle[i]); // Paramètre 2
     ps.setInt(3,c); // Paramètre 3
     ps.executeUpdate(); // Exécution de l'ordre SQL
    }
// Ne pas oublier le COMMIT!!
db.commit();
ps.close(); // Fermeture de la préparation de l'ordre
```

# ODBC

- Appel des fonctions de l'API ODBC :  
⇒ lien entre l'application et le **gestionnaire de pilotes**
- **Gestionnaire de pilotes** = DLL qui charge les pilotes associés à chaque **source de données** (BD + SGBD)
- **Pilote** = DLL qui contient les appels ODBC et traduit les requêtes en requêtes propres au SGBD

# ODBC



# ODBC

## Exemple de programme en C sous Visual C++ :

```
#include <stdio.h>
#include <conio.h>
#include <afxdb.h> // MFC ODBC database classes

char *cBASE ; /* Nom de la source de données */
char *cLOGIN ; /* Login utilisateur */
char *cPASSWD ; /* Mot de passe utilisateur */

void main()
{
    HENV d_env; /* Descripteur d'environnement */
    HDBC d_connex; /* Descripteur de connexion */
    HSTMT curseur; /* Curseur */
    RETCODE retcode; /* Code de retour de fonction */
```

# ODBC

```
UCHAR ucLastName[20],ucCity[20];
SDWORD ceLastName,ceCity;
char *cREQUETESQL; /* Variable recevant une requête SQL */

/* Saisie du nom de la source de données */
cBASE=(char*)malloc(20);
printf("Nom de la base de donnees :"); scanf("%s",cBASE);
/* Saisie du login */
cLOGIN=(char*)malloc(20); printf("Login :");
scanf("%s",cLOGIN);
/* Saisie du password */
cPASSWORD=(char*)malloc(20); printf("Mot de passe : ");

/* Pour ne pas afficher le mot de passe à l'écran */
int iPosCaractere=0; fflush(stdin);
do { if((cPASSWORD[iPosCaractere]=_getch())!='\r')
    printf("*");
    } while(cPASSWORD[iPosCaractere++]!='\r' && iPosCaractere <20);
cPASSWORD[--iPosCaractere]='\0';
```

# ODBC

```

/* Création d'un environnement ODBC */
retcode = SQLAllocEnv(&d_env);
/* Si la création d'un environnement ODBC est correcte */
if (retcode == SQL_SUCCESS)
{
    /* Création d'une connexion ODBC */
    retcode = SQLAllocConnect(d_env, &d_connex);

    /* Si la connexion ODBC s'est bien passée */
    if (retcode == SQL_SUCCESS)
    {
        /* Initialisation du temps de connexion à 5 secondes. */
        SQLSetConnectOption(d_connex, SQL_LOGIN_TIMEOUT, 5);

        /* Connexion à une source de données */
        retcode = SQLConnect(d_connex,(unsigned char*)cBASE,SQL_NTS,(unsigned
char*)cLOGIN,SQL_NTS,(unsigned char*)cPASSWORD,SQL_NTS);

```

↑  
*Longueur de la chaîne ou on indique que la chaîne se termine par le code NULL*

# ODBC

```
/* Si la connexion à la source de données s'est bien passée */
if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
{
    printf("Connection a la base (source de données).\n");

    /* Pause dans l'affichage */
    printf("Taper une touche pour continuer \n"); getchar();
    /* Allocation mémoire du curseur et association du curseur à la
       source de données identifiée par d_connex. */
    retcode = SQLAllocStmt(d_connex, &curseur);

    /* Si l'allocation mémoire du curseur est correcte */
    if (retcode == SQL_SUCCESS)
{ /* Création de la requête SQL */
    CREQUETESQL = "SELECT Nom, Ville FROM Etudiant";

    /* Execution directe de la requête sur la base */
    retcode = SQLExecDirect(curseur, (unsigned char*)CREQUETESQL,
SQL_NTS);
    printf("EXECUTION DE LA REQUETE, CODE ERREUR %d, CODE DE SUCCES
%d \n",retcode,SQL_SUCCESS);
}
```

# ODBC

```
/* Tant le parcours du curseur est valide */
while (retcode == SQL_SUCCESS)
{
    /* Parcours de l'enregistrement résultat de la requête */
    retcode = SQLFetch(curseur);

    /* Si le parcours est incorrect */
    if (retcode == SQL_ERROR || retcode == SQL_SUCCESS_WITH_INFO)
    {
        printf("Erreur %d\n",SQL_ERROR);
    }

    /* Si le parcours des enregistrements est correct */
    if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
    {
        /* récupération des données des colonnes 1 et 2 de la table
        résultat */
        SQLGetData(curseur, 1, SQL_C_CHAR, ucLastName, 30, &ceLastName);
        SQLGetData(curseur, 2, SQL_C_CHAR, ucCity, 30, &ceCity);
        /* Affichage du résultat */
        printf("Etudiant : %s %s\n",ucLastName,ucCity);
    }
}
```

# Chap VIII - Dépendances fonctionnelles

- **Dépendance fonctionnelle sur R**

$$A_1, A_2, \dots, A_n \rightarrow B$$

*"Si deux nuplets de R ont mêmes valeurs pour les attributs de  $A_1, A_2, \dots, A_n$  alors ils ont même valeur pour les attributs de B. »*

- Une dépendance  $A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m$  est :
  - **triviale** : si l'ensemble  $B_1, B_2, \dots, B_m$  est sous-ensemble de  $A_1, A_2, \dots, A_n$
  - **non triviale** : si au moins un  $B_i$  n'appartient pas à l'ensemble de  $A_1, A_2, \dots, A_n$
  - **complètement non triviale** : si aucun des  $B_i$  n'appartient à l'ensemble de  $A_1, A_2, \dots, A_n$

# Règles

## Axiomes de Armstrong :

- **Réflexivité** : si  $Y \subseteq X$  alors  $X \rightarrow Y$
- **Augmentation** : Si  $X \rightarrow Y$  alors  $\forall Z \ XZ \rightarrow YZ$
- **Transitivité** :  
Si  $A \rightarrow B$  et  $B \rightarrow C$  alors  $A \rightarrow C$

## On déduit :

- **Union** :  $\{X \rightarrow Y, X \rightarrow Z\} \models \{X \rightarrow YZ\}$
- **Pseudo-transitivité** :  
 $\{X \rightarrow Y, WY \rightarrow Z\} \models \{XW \rightarrow Z\}$
- **Décomposition** :  
Si  $X \rightarrow Y$  et  $Z \subseteq Y$  alors  $X \rightarrow Z$

# Fermeture

- **Fermeture d'une famille de dépendances fonctionnelles**

$$\mathbf{F}^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$$

- **Fermeture d'un ensemble d'attributs  $X$  par rapport à une famille de dépendances fonctionnelles  $F$**

$$[\mathbf{X}]^+ = \{A \mid F \models X \rightarrow A\}$$

- **Lemme**

La dépendance fonctionnelle  $X \rightarrow Y$  peut être déduite des axiomes d'Amstrong si  $Y \subseteq [X]^+$

# Equivalence et Couverture

- Deux familles de dépendances fonctionnelles  $F$  et  $G$  sont **équivalentes** si  $F^+ = G^+$
- Si  $F^+ \subset G^+$  alors  $G$  est **une couverture** de  $F$
- **Une famille de dépendances fonctionnelles  $F$  est minimale si :**
  1. En partie droite de toute dépendance de  $F$ , il n'y a qu'un seul attribut
  2. Il n'y a pas de dépendance fonctionnelle  $X \rightarrow A$  dans  $F$  telle que  $(F \setminus \{X \rightarrow A\})$  soit équivalente à  $F$
  3. Il n'y a pas de dépendance fonctionnelle  $X \rightarrow A$  et  $Z \subset X$  tels que  $(F \setminus \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$  soit équivalente à  $F$

**Attention: pas d'unicité des couvertures minimales**

# Clé

Soit  $R(A_1, A_2, \dots, A_n)$  et  $F$  une famille de DF sur  $R$

- Un sous-ensemble  $X$  de  $\{A_1, A_2, \dots, A_n\}$  est une **clé minimale** de  $R$  si
  - ① La dépendance fonctionnelle  $X \rightarrow A_1, A_2, \dots, A_n \in F^+$
  - ②  $\forall Y \subset X$ , on a pas  $Y \rightarrow A_1, A_2, \dots, A_n$
- Si  $X$  n'est pas un ensemble minimal alors  $X$  est une **surclé**
- Les dépendances fonctionnelles permettent de déduire les clés des relations

# Chap IX - Décomposition de schéma

- **Décomposition sans perte d'information (*Lossless jointure*)**

La décomposition de  $R$  en  $R_1, R_2, \dots, R_n$  est sans perte d'information si et seulement si  $\forall r$ , instance de  $R$  :

$$r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \bowtie \dots \bowtie \Pi_{R_n}(r)$$

- **Décomposition sans perte de dépendances**

La décomposition de  $R$ , munie d'une famille de dépendances  $F$ , en  $R_1, R_2, \dots, R_n$  est sans perte de dépendance si et seulement si :

$$F^+ = F^+_{R_1} \cup F^+_{R_2} \cup \dots \cup F^+_{R_n} \text{ avec } F^+_{R_i} = \{X \rightarrow Y, XY \subseteq R_i\}$$

# Décomposition SPI

$$\left. \begin{array}{l} \bullet r \subseteq \prod_{i=1}^n R_i(r) ? \\ \bullet r \supseteq \prod_{i=1}^n R_i(r) ? \end{array} \right\} \forall \mathbf{r}$$

Soit  $t(a_1, a_2, \dots, a_n)$ , un nuplet générique issu de  $\prod_{i=1}^n R_i(r)$

$t$  est construit à partir de  $t_1 \otimes t_2 \otimes \dots \otimes t_n$

$$\text{avec } t_1 \in R_1(r)$$

$$t_2 \in R_2(r)$$

...

$$t_n \in R_n(r)$$

Est - ce - que  $t \in r$  ?

# Décomposition SPD

- Association d'une famille  $F_i$  à chaque sous-relation  $R_i$

- Calcul des  $F_i$  à partir de  $F^+$  :

$$\forall X \rightarrow Y \in F^+ \text{ et } XY \subseteq R_i \implies X \rightarrow Y \in F_i$$

- Perte de dépendances  $\implies$

$$\exists X \rightarrow Y \in F \text{ tq } X \rightarrow Y \notin [X]_{\cup_{F_i}}^+$$

$$[X]_{\cup_{F_i}}^+ \text{ calculé itérativement par } X \cup [(X \cap R_i)^+ \cap R_j]$$

# Chap X - Formes normales

## Forme normale de Boyce-Codd (BCNF)

- Un schéma de relation est **BCNF** si et seulement si :

DF élémentaires telles que une clé détermine un attribut

- Un schéma de relation est BCNF si :

$\forall X \rightarrow A$  DF non triviale,  $A \notin X$  :

$X$  est une (sur)clé et  $A$  n'est pas un attribut de clé

- La forme normale BCNF évite la redondance d'information
- La décomposition de schéma BCNF sans perte d'information ne préserve pas les dépendances fonctionnelles

# 3ème forme normale

- Un schéma de relation est **3NF** si :
  - $\forall A$ , attribut  $\notin$  une clé,  
A ne dépend pas d'un ensemble d'attributs qui n'est pas une clé  
(i.e. pas de dépendance transitive ni de dépendance partielle)
- $X \rightarrow A$  est **partielle** si X est une partie de clé et A ne l'est pas
- $X \rightarrow A$  est **transitive** si ni X ni A ne sont des parties de clé
- Si K est clé alors  $K \rightarrow X \rightarrow A$  est une **chaîne non triviale**
- Un schéma R, munie d'une famille F de DF, est 3NF si :

$\forall X \rightarrow A$ , DF non triviale de F :

X est une surclé **ou** A appartient à une clé

# 3ème forme normale

- La 3NF n'élimine pas le redondance d'information
  - Objectif principaux de la décomposition de schéma :
    - BCNF
    - Non perte d'information
    - Non perte de dépendance
- ⇒ On préférera sacrifier la forme normale BCNF  
et avoir une 3NF SPI-SPD

# 1ère, 2ème et 3ème formes normales

- Un schéma de relation R est **1NF** si :
  - $\forall$  attribut de R, il contient une **valeur atomique**
- Un schéma de relation R est **2NF** si et seulement si :
  - le schéma est en 1NF
  - $\forall A$ , attribut  $\notin$  une clé, A ne dépend pas d'une partie de clé  
c-à-d  $\neg( \exists \text{ une dépendance fonctionnelle partielle } )$
- Un schéma de relation R est **3NF** si :
  - le schéma est 2NF
  - $\neg( \exists \text{ une dépendance fonctionnelle transitive } )$