

# Introduction aux Bases de données

## Cours 1 : Introduction–Conception

UFR 919 – Licence  
informatique

LU2IN009

# Informations pratiques

Responsable de l'UE :

- Camelia Constantin

Chargés de cours :

- Mercredi 8h45 : Mohamed-Amine Baazizi
- Jeudi 14h00 : Camelia Constantin

Informations supplémentaires :

Moodle : Introduction aux bases de données relationnelles - S2

# Bibliographie conseillée

## Notes de cours

- S. Gancarski. Introduction aux bases de données. UPMC, Paris 6, janvier 2003 – lien sur le site de l'UE

## Livres en anglais

- R. Ramakrishnan and J. Gehrke. Database Management Systems 3e édition, McGraw Hill, 2002 -<http://pages.cs.wisc.edu/~dbbook/>  
(Disponible bib. MIR et MIE)
- A. Silberschatz, HF. Korth and S. Sudarshan. Database System concepts 6e édition, McGraw Hill, 2011-<http://db-book.com/>

## Livres en français :

- S. Abiteboul, R. Hull, V. Vianu, Les fondements des bases de données, Vuibert  
(Disponible bib. MIR et MIE)
- G. Gardarin. Bases de données - objet et relationnel. Eyrolles.  
(Disponible bib. L1-L2 scientifique et MIE)

# Aperçu sur les bases de données

Qu'est ce qu'une base de données ?

- Collection de données structurées suivant la réalité modélisée

Où trouve-t-on des bases de données ? Exemple:

- web: sites marchands, réseaux sociaux, ...
- finance: applications financières, gestion de comptes, ...
- économie : e-commerce (amazon), services de ventes/achats, ...
- industrie: gestion de centrales nucléaires, chaînes de production, ...
- transports: réservation de billets, gestion de trains/avions, ...
- science: données d'expérimentation,...
- services publics: impôts, police, open-data, ...

Quels sont les types de Bases de Données ?

- Les BD relationnelles (prédominantes : données de gestion)
- Les BD objet, XML, JSON, RDF (orientées web, données techniques)

# SGBD\* vs système de fichiers

## Accès aux données

- écrire un programme dédié à chaque tâche

## Redondance des données

- la même donnée stockée dans plusieurs fichiers

## Cohérence des données

- difficulté d'exprimer et de garantir des contraintes d'intégrité

## Performance d'accès

- les données souvent volumineuses, plusieurs usagers

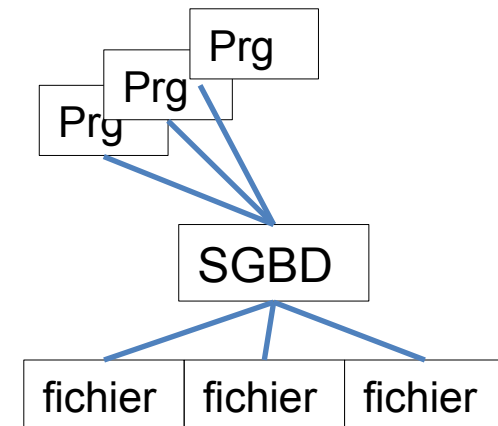
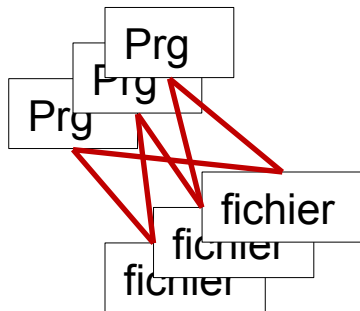
## Concurrence d'accès

- interaction entre plusieurs programmes

## Sécurité et protection des données

- données de sensibilités différentes

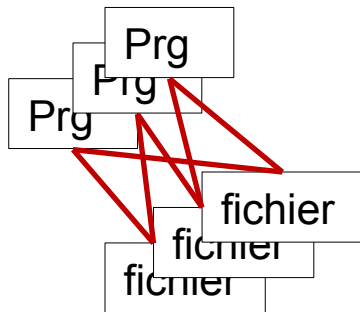
(\*) Système de Gestion de Bases de Données  
*Database Management System (DBMS)*



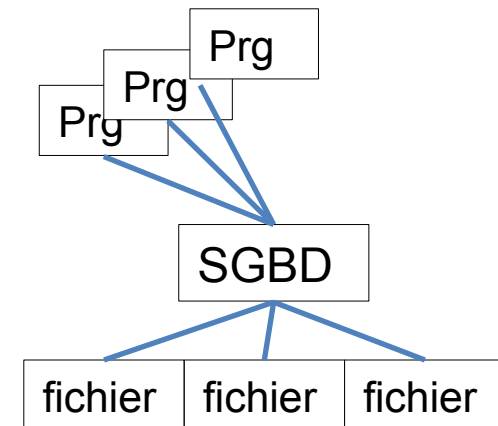
# SGBD\* vs système de fichiers

SGBD :

- Contient les BDs ainsi que leur description et contraintes (méta-données stockes dans le catalogue du SGBD)
- Centralisation de l'information, représentation de relations complexes entre les données, permet de trouver et de mettre à jour efficacement des données reliées
- Contrôle de la redondance des données
- Contrôle des niveaux d'accès aux données (sécurité des données)
- Stockage persistant pour les données des applications, structures de stockage et interrogation efficace des données
- Abstraction des données, isolation entre les données et les traitements
- Plusieurs vues des mêmes données, interfaces utilisateur multiples
- Accès simultané aux données centralisées par plusieurs applications, contrôle de concurrence, partage de données
- Définition de contraintes d'intégrité et automatisation de leur maintenance (triggers)



(\*) Système de Gestion de Bases de Données  
*Database Management System (DBMS)*



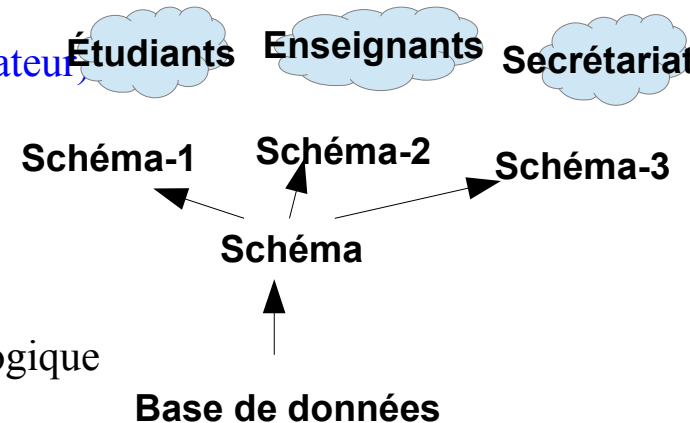
# L'architecture à 3 niveaux

## Niveaux d'abstraction

- Niveau des vues (accessible à l'utilisateur)  
Quelles données peut-on voir → schéma externe
- Niveau logique (accessible au concepteur/programmeur)  
Quelles données sont stockées → schéma logique
- Niveau physique (accessible au concepteur et à l'administrateur)  
Comment les données sont stockées → schéma physique

## Indépendance entre les niveaux

- Indépendance physique des données  
le changement du schéma physique n'affecte pas le schéma logique
- Indépendance logique  
le changement du schéma logique n'affecte pas le schéma externe

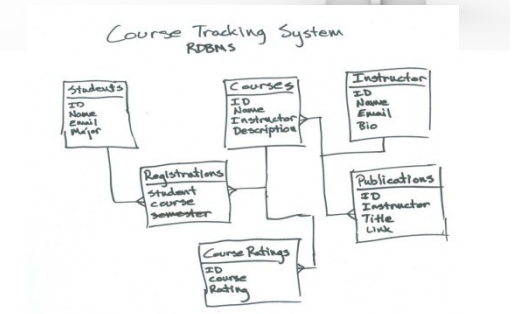


# Comment construire une base de données

1 Analyse des besoins  
*observer le monde réel*  
*identifier les informations pertinentes*



2 Modélisation des données  
*formaliser les besoins*

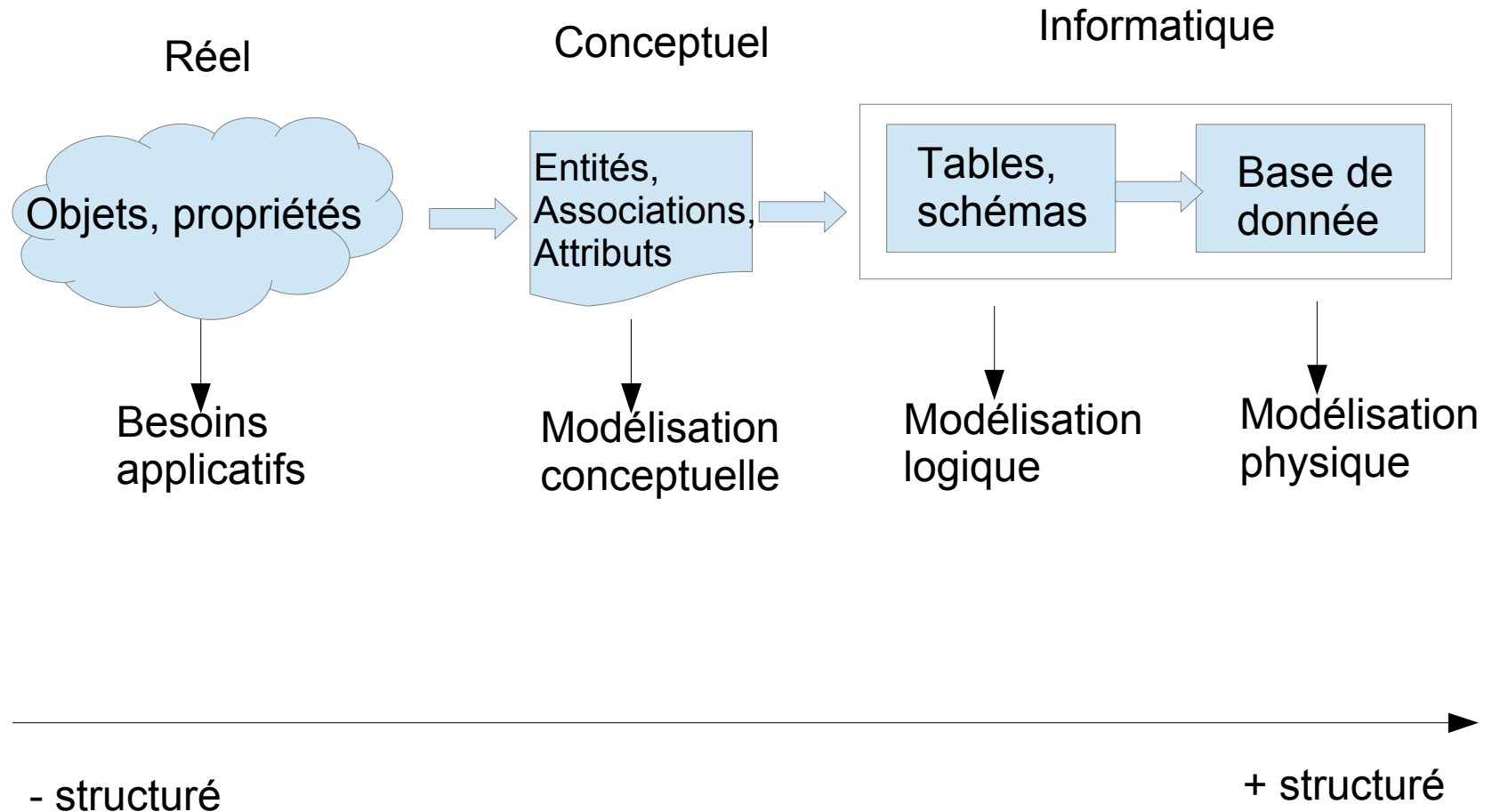


3 Implantation des données et des procédures  
d'interrogation et de mise à jour  
*BD proprement parlé*

```
DBProvider = "Database=MS Access; Username=Admin; Password=; Server=;"
SelectSQL1 = "Select id, name, quantity from all;"
QuerySQL1 = " where id between 1 and 100;"
QuerySQL2 = " group by id, name;"
SelectQuery = SelectSQL1 & QuerySQL1 & QuerySQL2
Execute Query; Commit Transaction; Select new data
Form Navigation
If KeyAscii = 13 Then Execute Query
If KeyAscii < 48 Then Exit
If Not Chr(KeyAscii) Like "#;" And KeyAscii < 48 Then
```



# Construction d'une base de données



# Quels outils ?

## 1 Analyse des besoins :

- discussion ‘informelle’ qui découle sur une documentation technique

## 2 Modélisation des données :

- traduction des besoins en des concepts de base : entités et lien entre elles (modèle Entite-Association)

## 3 Implantation des données :

- langages compréhensibles par la machine
- SQL (*Structured Query Language*) est le standard pour les données relationnelles

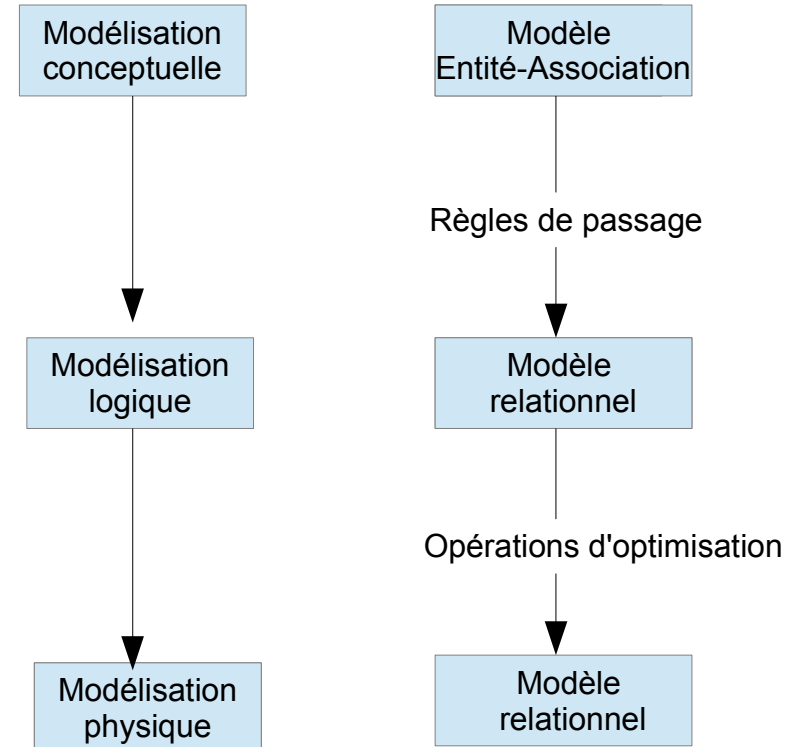
# Modèles de Données

## Modèle entité-association

- Description haut niveau des données
  - entité et les liens entre elles (associations)
- ⇒ concepteur, client

## Modèles relationnel

- Représentation logique des données
  - Concept de tables
- ⇒ concepteur, développeur



# Langages pour les BD

Langage **déclaratif** (se concentrer sur la logique de l'application, pas sur l'accès aux données)

- Quelles données retourner, supprimer, modifier?
- Le système génère un programme « optimisé » pour réaliser la tâche demandée
- Langage de définition des données (*Data Definition Language*)
  - Création des tables
  - Définition des contraintes
- Langage de manipulation des données (*Data Manipulation Language*)
  - Interrogation des données
  - Insertions, modification et suppression données

# But du cours

Introduction aux Bases de données relationnelles sous un angle applicatif

- Modélisation et représentation des données
- Implantation et manipulation des données
- Programmation en lien avec les données

Aperçu des cours liés

- LUIN3009 : mise en œuvre et optimisation dans les systèmes BD
- MLBDA (M1) : modèles et langages pour les BD
- BDR (M1) : BD réparties
- BLDE (M2) : Big Data

# Plan du cours

- Conception d'un schéma de Bases de Données

Modèle Entité-Association

- Interrogation des données

Langage de requêtes : Calcul relationnel et SQL

- Saisie, modification et cohérence des données

Langage de Définition de Données (DDL)

Langage de Manipulation de Données (DML)

- Manipulation complexe des données

Programmation en PL/SQL, Déclencheurs (triggers)

# Conception d'un schéma Entité-Association

# Démarche

## Analyse des besoins

- Discussion **informelle** avec les futurs utilisateurs
  - Identifier les objets du monde réel et des liens entre eux
  - Identifier les opérations sur ces objets et les éventuelles évolutions
- Document technique décrivant les données de l'application

## Etablissement du schéma conceptuel

- Langage de **haut niveau** (ex. le modèle entité-association)
  - Décider des données devant être stockées, de leur propriétés et des relations entre elles
  - Définir les contraintes à respecter
- Schéma dans un langage de haut niveau (Entité-Association)



# Etude de cas : BD d'une université

Les besoins :

- Gérer les inscriptions des étudiants à des modules
- Gérer l'affectation des tuteurs à des étudiants
- Gérer le planning des salles

Les objets à modéliser :

- Les étudiants Les modules
- Les tuteurs
- Les salles

Les liens entre les objets (scénario) :

- Les étudiants s'inscrivent à un ou plusieurs modules pour une année universitaire
- Le cours d'un module a lieu dans une salle donnée ; il débute à une heure connue et se déroule pendant une durée connue.

# Le modèle Entité-Association

- Proposé par Peter Chen en 1976
- **Principe** : Transcrire les besoins en terme de *classes* d'*entités* et de *classes* d'*associations*
- Les *entités* = les objets du monde réel
- Une classe d'entités = ensemble d'entités possédant les mêmes propriétés
- Les *associations* = les relations liant les entités
- Une classe associations = ensemble d'associations reliant des entités de la même classe
- Les *attributs* = les propriétés qui renseignent certaines informations sur une entité ou une associatio

# Entité

- Tout objet du monde réel pertinent pour l'application, peut être concret ou abstrait.
- Exemple d'entités concrètes :
  - Le médecin "Anne DUPONT" est une entité
  - La salle 24-34/208 est une entité
  - L'enseignement "2I009" est une entité
- Exemple d'entités abstraites (qui ne correspondent pas à des objets physiques) :
  - Le virement n° XXX ayant eu lieu le 20/01/19 est une entité
  - Le compte en banque d'un client est une entité
  - Un contrat d'assurance est une entité

# Classe d'entités

- Permet de décrire un ensemble d'entités de même type (ayant les mêmes caractéristiques)
- Exemple de classes d'entités :
  - La classe *Étudiant* décrit l'ensemble des étudiants de l'université (tout étudiant a un nom, prénom, adresse, matricule)
  - La classe *Module* décrit toutes les modules de l'université (tout module a un code, un intitulé et un niveau)
  - La classe *Médecin* décrit tous les médecins de l'application (tout médecin a un matricule et un nom)
  - etc

# Attribut

- Modélise une propriété/caractéristique d'une classe d'entités
- Possède un nom et un domaine de valeurs (=ensemble de valeurs permises)
- Est *atomique* (ne peut pas être multivalué)
- Un attribut prend une seule valeur à la fois pour chaque entité
- **Exemple d'attributs** :
  - Le nom, le prénom, l'adresse et le matricule sont des attributs de la classe *Étudiant*
  - Le nom est une chaîne de 1 à 20 caractères et l'attribut correspondant est Nom.
  - Le matricule est un nombre entier avec 4 chiffres
  - L'attribut prénom *ne peut pas contenir plusieurs chaînes de caractères* (e.g "Jean", "Louis") mais une seule chaîne de caractères (e.g "Jean, Louis")
  - Le nom de l'étudiant de matricule 1234 (=une entité) ne peut pas être à la fois "Dupont" et "Martin"

# Identifiants

- Un sous ensemble d'attributs d'une classe d'entités
- Permet de distinguer les entités de la même classe (deux entités de la même classe ne peuvent pas avoir le même identifiant)
- Toute classe d'entités doit avoir au moins un identifiant (si plusieurs identifiants potentiels → en choisir un seul )
- Un identifiant peut être :
  - **Naturel** (construit à partir des attributs de la classe)
  - **Artificiel** (rajouté aux attributs de la classe, lorsque les attributs de la classe ne permettent pas de définir un identifiant)

# Exemples d'identifiants

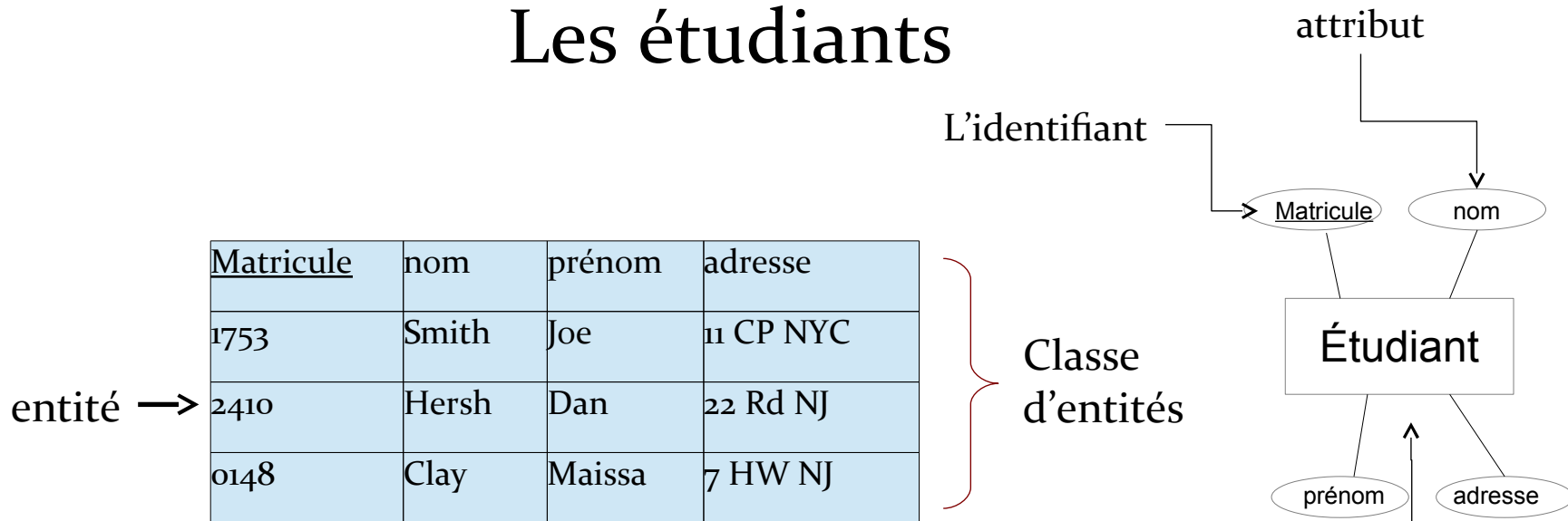
- Le matricule de l'étudiant permet de distinguer l'ensemble d'étudiants de l'université → peut être identifiant de la classe *Étudiant*
- Pour une salle d'enseignement, son numéro (e.g 207) ne permet pas de la distinguer des autres salles de l'université (il peut y avoir une autre salle 207), on a besoin aussi de connaître son emplacement (e.g 24-34), donc le couple (numéro, emplacement) (e.g (207, 24-34)) sera l'identifiant de la classe *Salle*
- Si chaque étudiant a un nom différent des autres étudiants ainsi qu'un matricule différent des autres étudiants, choisir un des deux (e.g matricule) comme identifiant de la classe *Étudiant*
- Si plusieurs enseignants peuvent avoir le même nom et le même prénom (le couple nom, prénom ne peut pas jouer le rôle d'identifiant), alors ajouter un attribut artificiel id à la classe *Enseignant* qui sera désigné comme identifiant.

# Etude de cas : BD d'une université

## Classe d'entités

ensemble d'entités possédant les mêmes propriétés

### Les étudiants

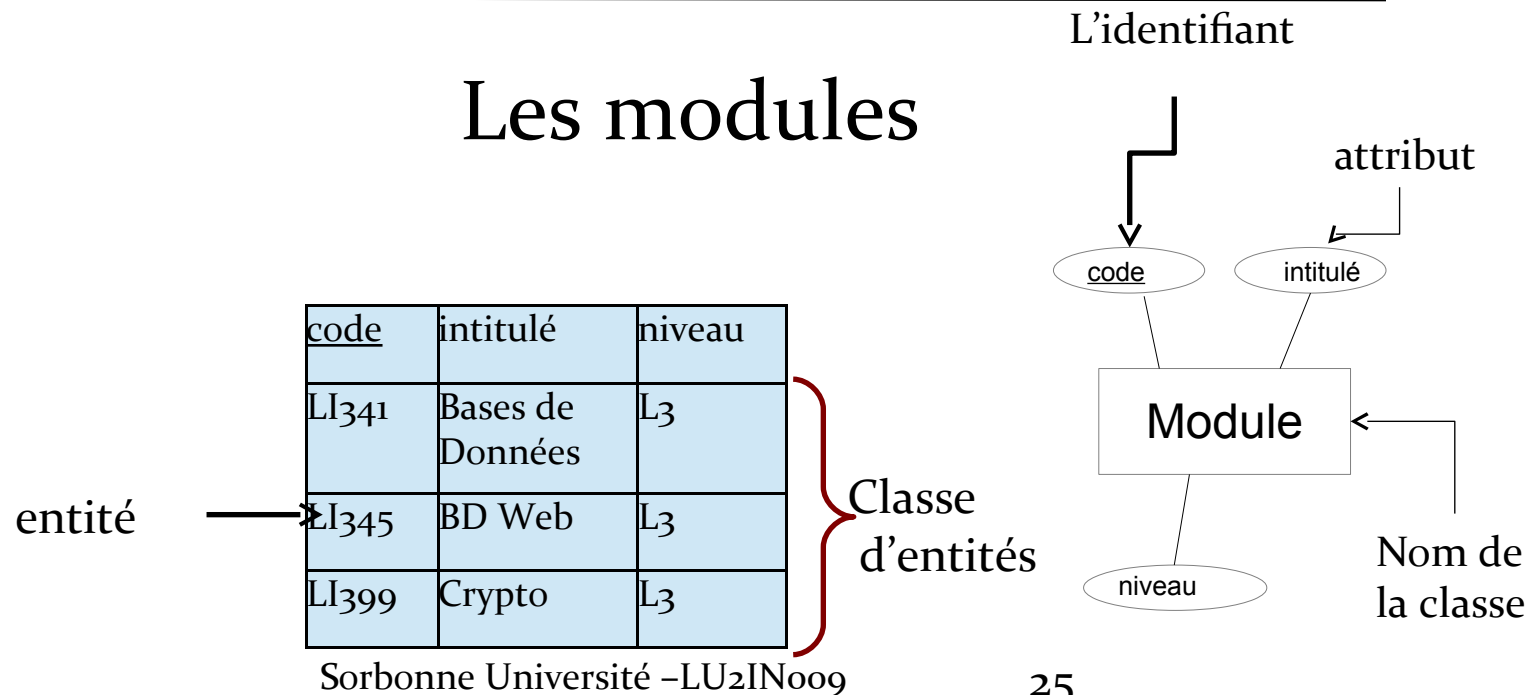




# Etude de cas : BD d'une université

## Classe d'entités

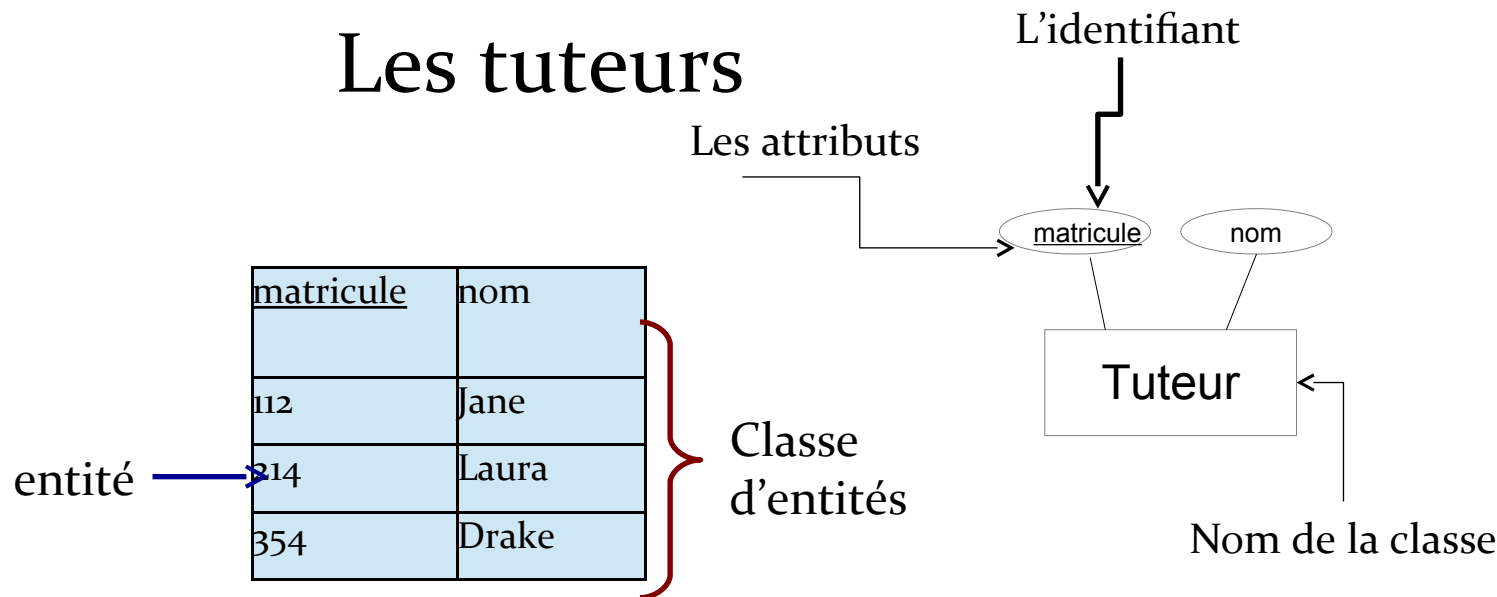
ensemble d'entités possédant les mêmes propriétés



# Etude de cas : BD d'une université

## Classe d'entités

ensemble d'entités possédant les mêmes propriétés

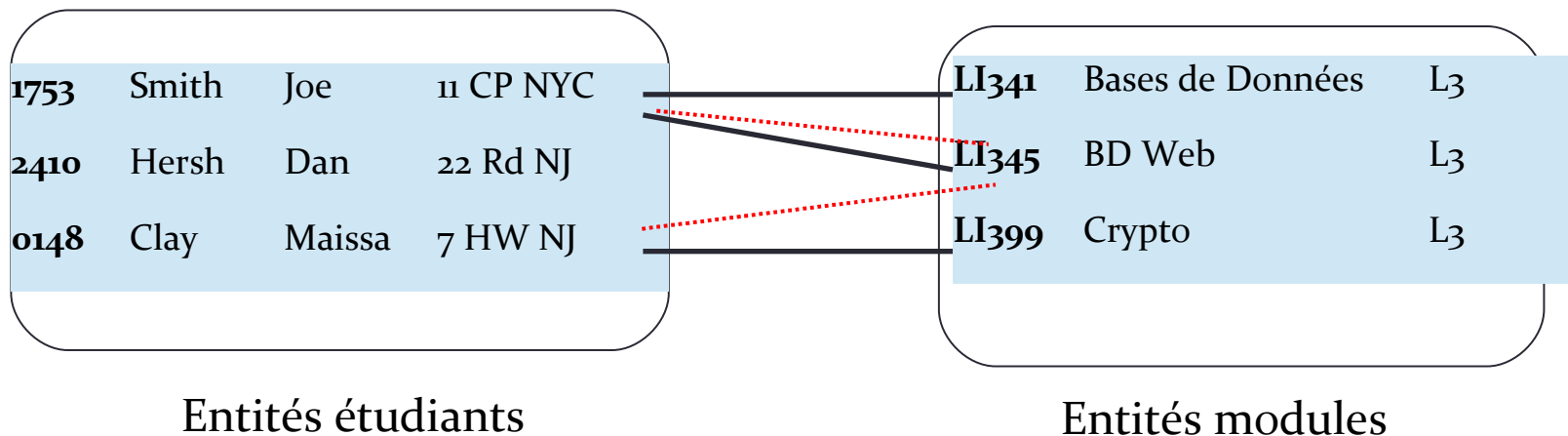


# Etude de cas : BD d'une université

## Association

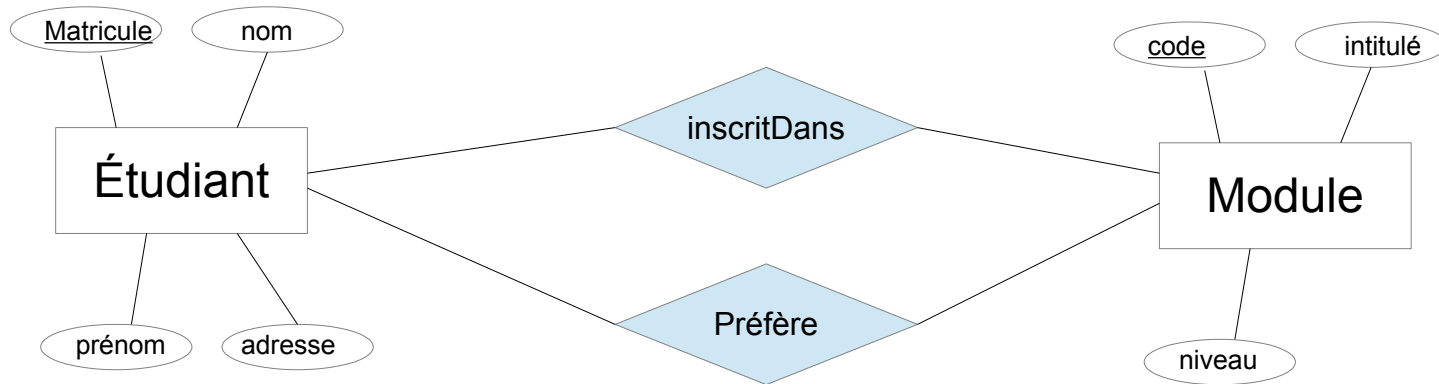
- Lien entre entités (relation entre deux ensembles)
  - Il est possible d'avoir plusieurs associations entre les mêmes entités avec des sémantiques différentes
- 

« Etudiant *inscritDans* Module » et « Etudiant *prefère* Module »



# Classe d'associations

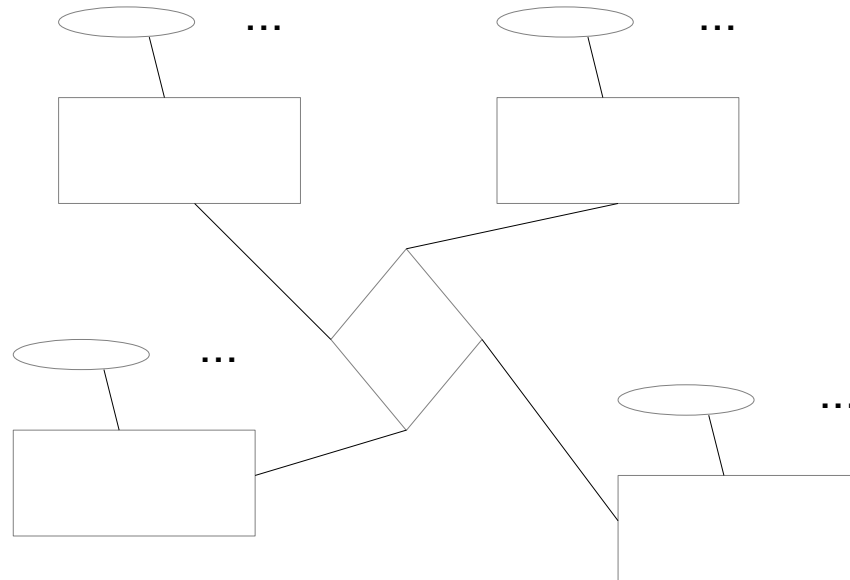
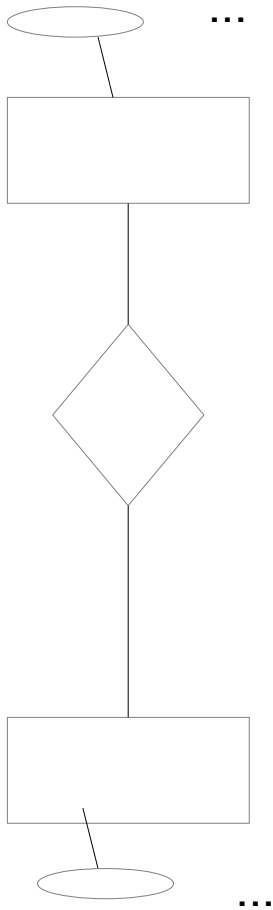
- L'ensemble d'associations de même type (qui relient les mêmes ensembles d'entités et qui ont la même sémantique)
- Désignée généralement par un verbe
- Exemple (représentation graphique) :



# Types de classes d'association

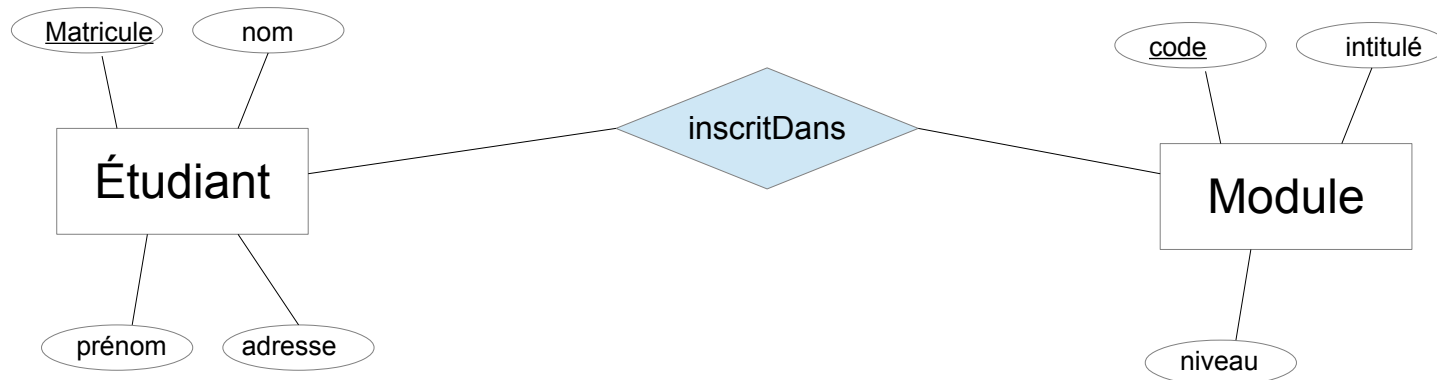
Associations binaires (relient deux entités)

Associations N-aires (relient plus de deux entités)



Association quaternaire

# Exemple d'associations appartenant à une classe



|      |       |        |           |
|------|-------|--------|-----------|
| 1753 | Smith | Joe    | 11 CP NYC |
| 2410 | Hersh | Dan    | 22 Rd NJ  |
| 0148 | Clay  | Maissa | 7 HW NJ   |

Entités étudiants

1753, LI341

1753, LI345

0148, LI399

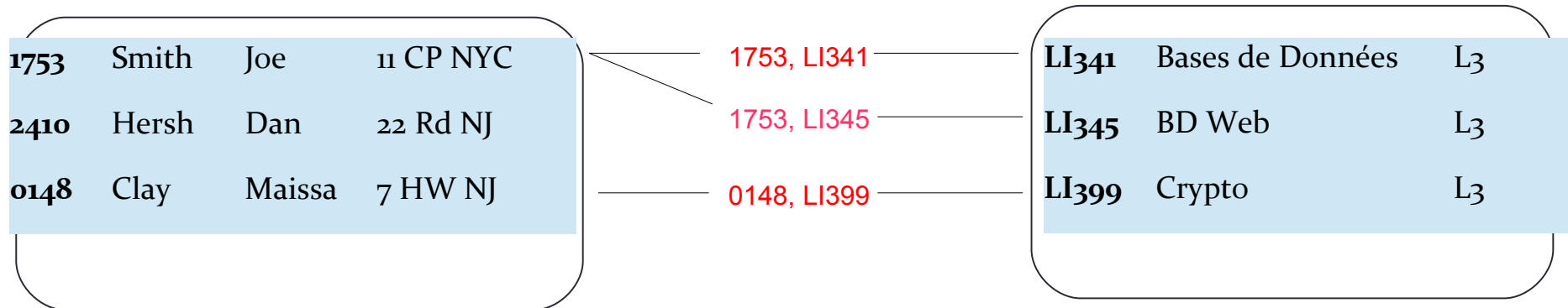
|       |                  |    |
|-------|------------------|----|
| LI341 | Bases de Données | L3 |
| LI345 | BD Web           | L3 |
| LI399 | Crypto           | L3 |

Entités modules

# L'identifiant d'une association

Une association est identifiée au moyen des identifiants des entités qu'elle met en relation <sup>7</sup> on ne peut pas associer les mêmes entités plusieurs fois avec des associations de même sémantique

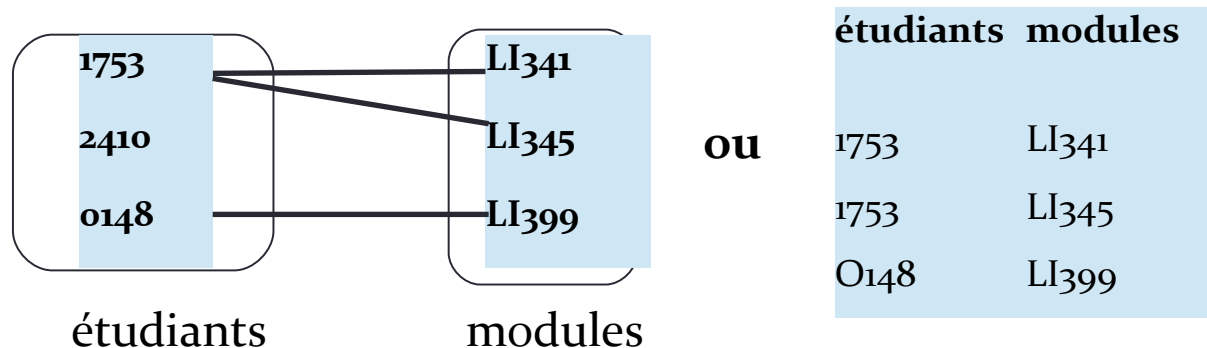
**Exemple:** l'étudiant 1753 ne peut pas s'inscrire plusieurs fois au module LI341



Entités étudiants

sera notée

Entités modules



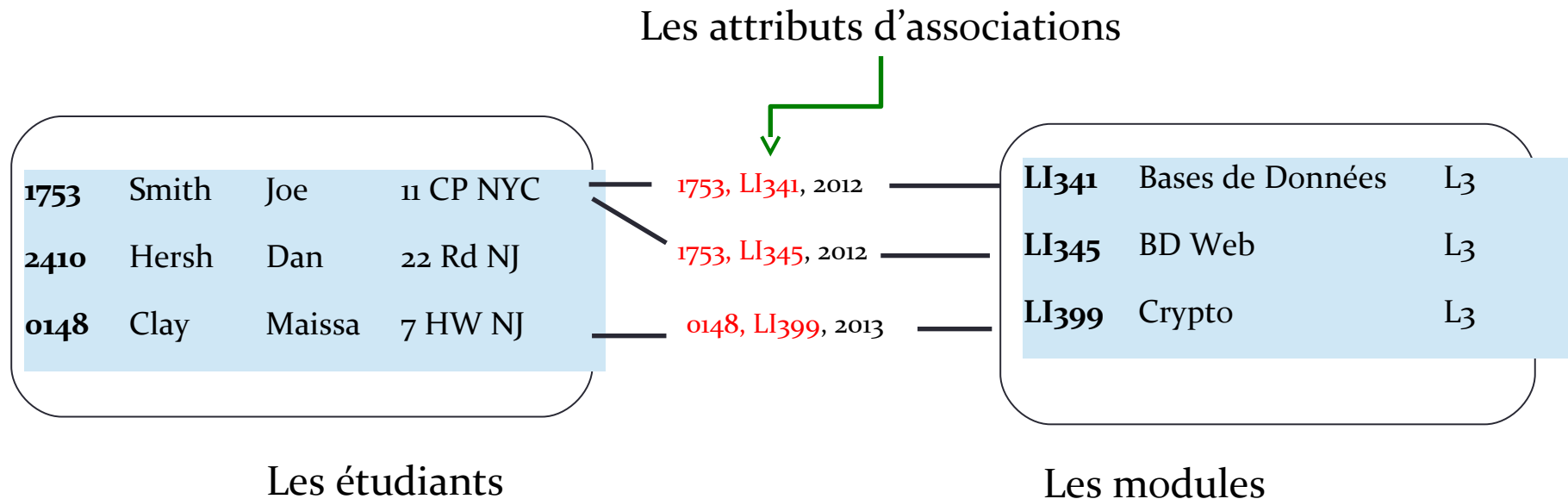
# Etude de cas : BD d'une université

## Attributs d'une association

- Les identifiants des entités qu'elle relie, plus éventuellement d'autres attributs

**Exemple:** Etudiant *inscritDans* Module pour une année

---





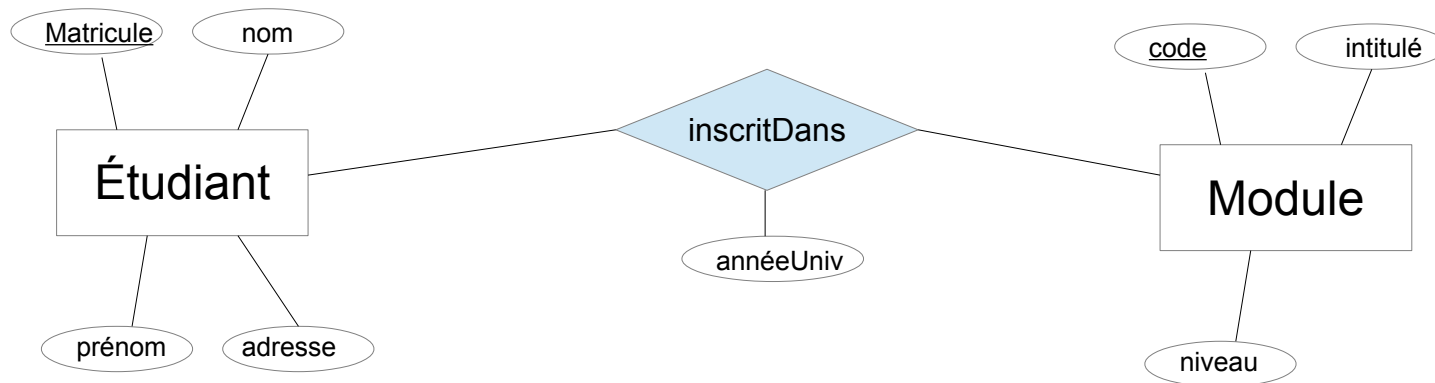
# Attributs d'une classe d'associations

Identifiant: L'ensemble des identifiants des classes d'entités qu'elle relie

Autres attributs: l'ensemble des valeurs des attributs des associations de cette classe

---

**Exemple:** Etudiant *inscritDans* Module pour une année



L'identifiant de la classe *inscritDans* est {**Matricule**, **code**}

# Identifiant d'une classe d'associations

On souhaite qu'un étudiant puisse s'inscrire à un module plusieurs fois,  
à des années différentes

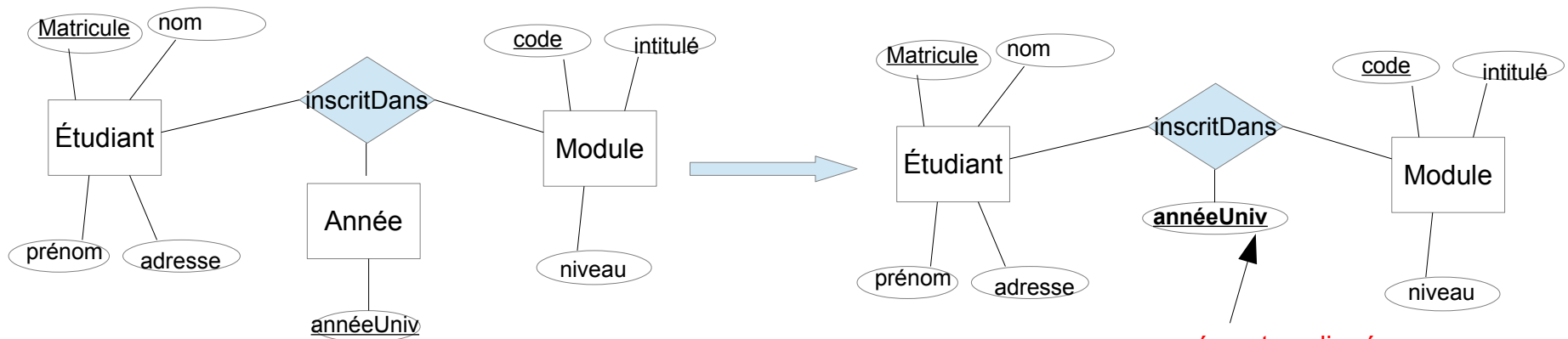
→ **Problème** : l'identifiant d'une association est composé des identifiants des entités reliées

→ **Solution** : intégrer l'année dans l'identifiant de l'association

|      |       |        |           |                   |       |                  |    |
|------|-------|--------|-----------|-------------------|-------|------------------|----|
| 1753 | Smith | Joe    | 11 CP NYC | 1753, LI341, 2014 | LI341 | Bases de Données | L3 |
| 2410 | Hersh | Dan    | 22 Rd NJ  | 1753, LI341, 2012 | LI345 | BD Web           | L3 |
| 0148 | Clay  | Maissa | 7 HW NJ   | 1753, LI345, 2012 | LI399 | Crypto           | L3 |
|      |       |        |           | 0148, LI399, 2013 |       |                  |    |

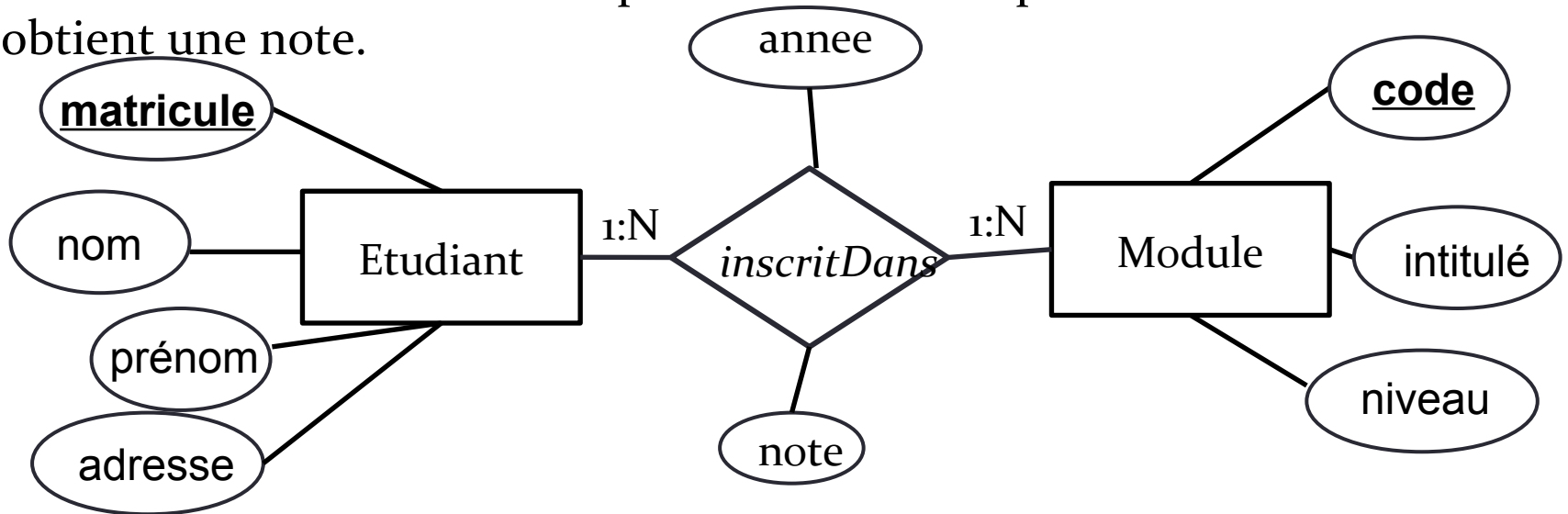
Les étudiants

Les modules



# Attribut d'une classe d'association

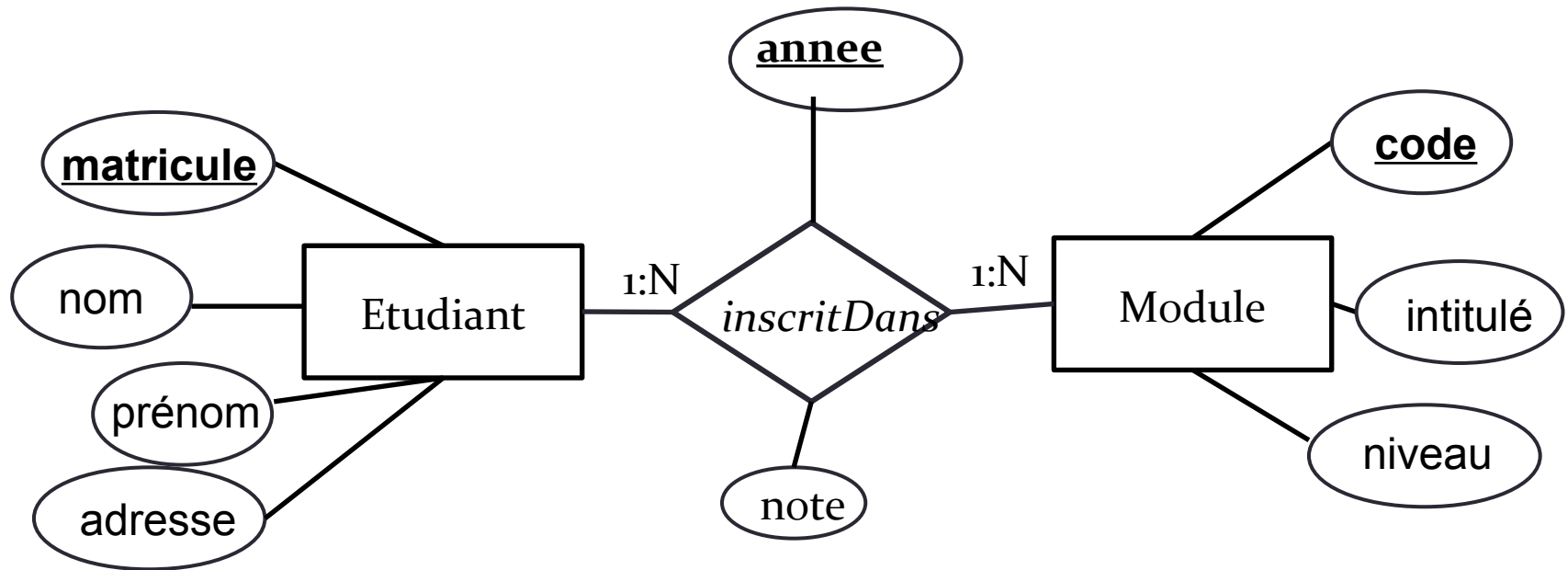
Un étudiant s'inscrit à un ou plusieurs modules pendant une année et obtient une note.



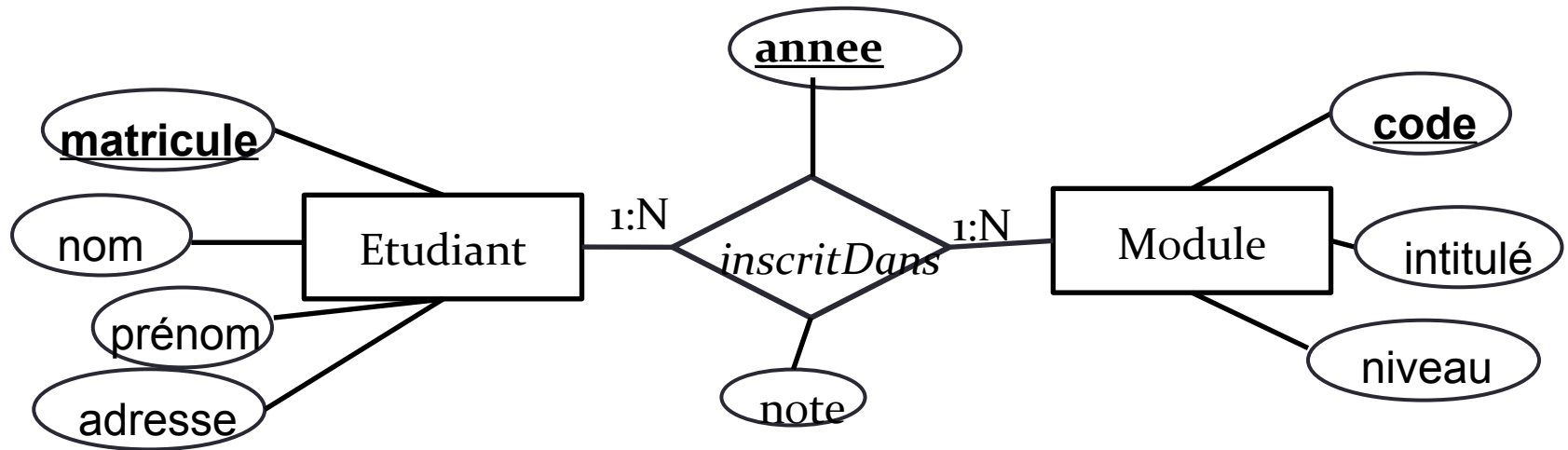
| etudiants | modules | Annee | note |
|-----------|---------|-------|------|
| 1753      | LI341   | 2012  | 9    |
| 0148      | LI345   | 2012  | 13   |

# Identifiant d'une classe d'associations

Un étudiant s'inscrit à un ou plusieurs modules pendant une année ou plusieurs années et obtient à chaque inscription une note.



# Illustration sur une instance



| etudiants | modules | Annee | note |
|-----------|---------|-------|------|
| 1753      | LI341   | 2012  | 9    |
| 1753      | LI341   | 2013  | 10.5 |
| 0148      | LI345   | 2012  | 13   |

# Simplification des appellations

## Omission du terme « classe »

- Entité sous-entend classe d'entités
- Association sous entend classe d'associations
- Au niveau E/A, on se préoccupe seulement des classes, pas des instances particulières

# La cardinalité d'une association

Nombre d'entités liées par les associations

Intervalle de valeurs [m:n]

| Description   | Entités            | cardinalité |
|---|--------------------|-------------|
| Un module n'ouvre que s'il y a au moins un étudiant d'inscrit | Module<br>Etudiant | 1:N         |
| Un module a lieu dans la même salle                           | Module<br>Salle    | 1:1         |

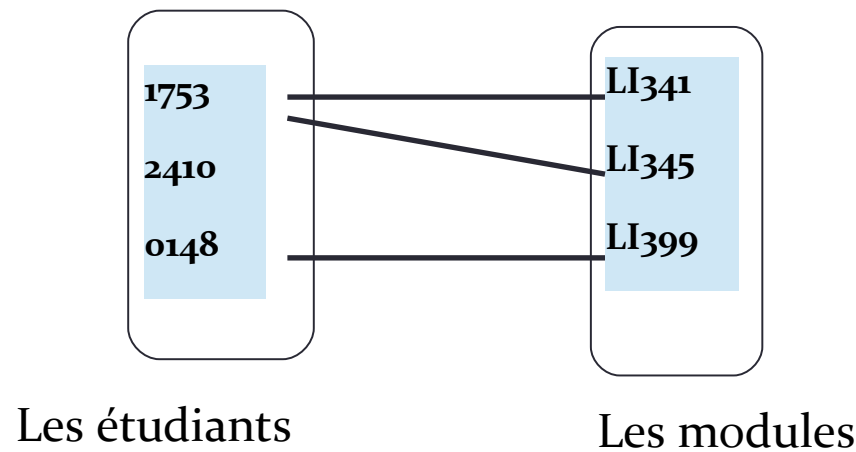
# Etude de cas : BD d'une université

## Les cardinalités d'associations

- Etudiant *inscritDans* Module
  - Un étudiant DOIT s'inscrire dans au moins un module
  - Un module n'ouvre que s'il y a au moins un étudiant

---

## L'association « Etudiant *inscritDans* Module »



| etudiants | modules |
|-----------|---------|
| 1753      | LI341   |
| 1753      | LI345   |
| 2401      |         |
| 0148      | LI399   |

*Quel est le problème?*



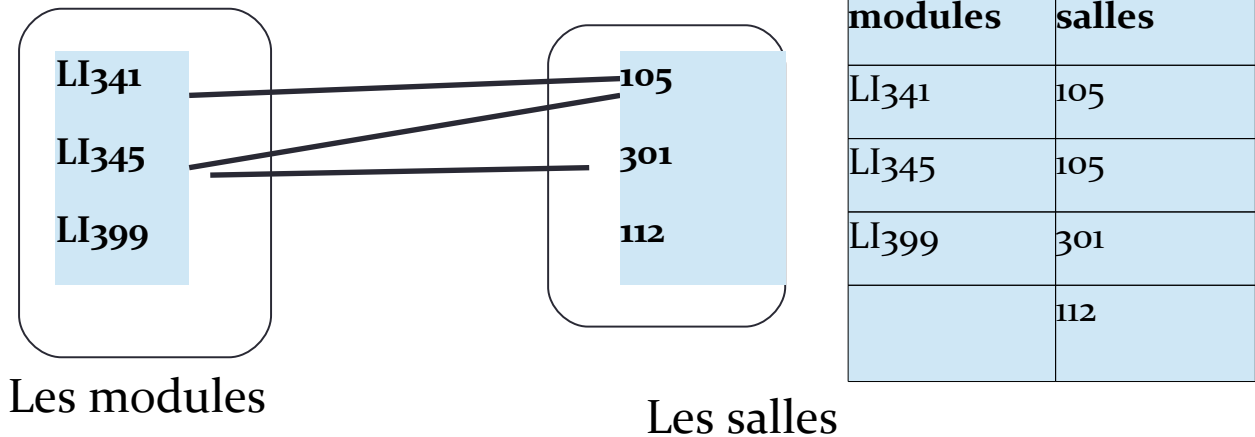
# Etude de cas : BD d'une université

## Les cardinalités d'associations

- Module *aLieuDans* Salle
  - Un module a lieu dans une et une seule salle
  - Une salle peut être utilisée pour plusieurs modules ou rester inoccupée

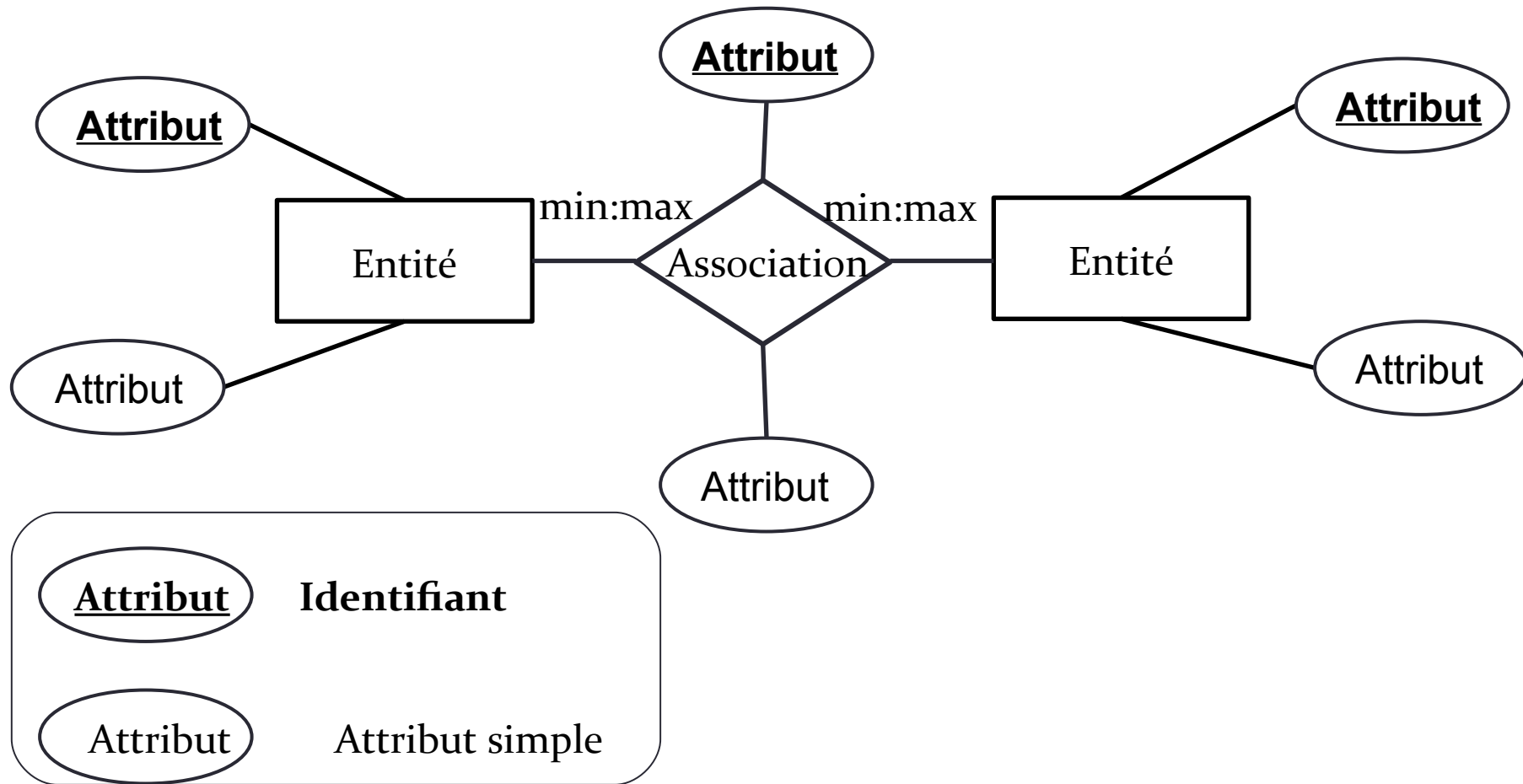
---

## L'association « Module *aLieuDans* Salle »



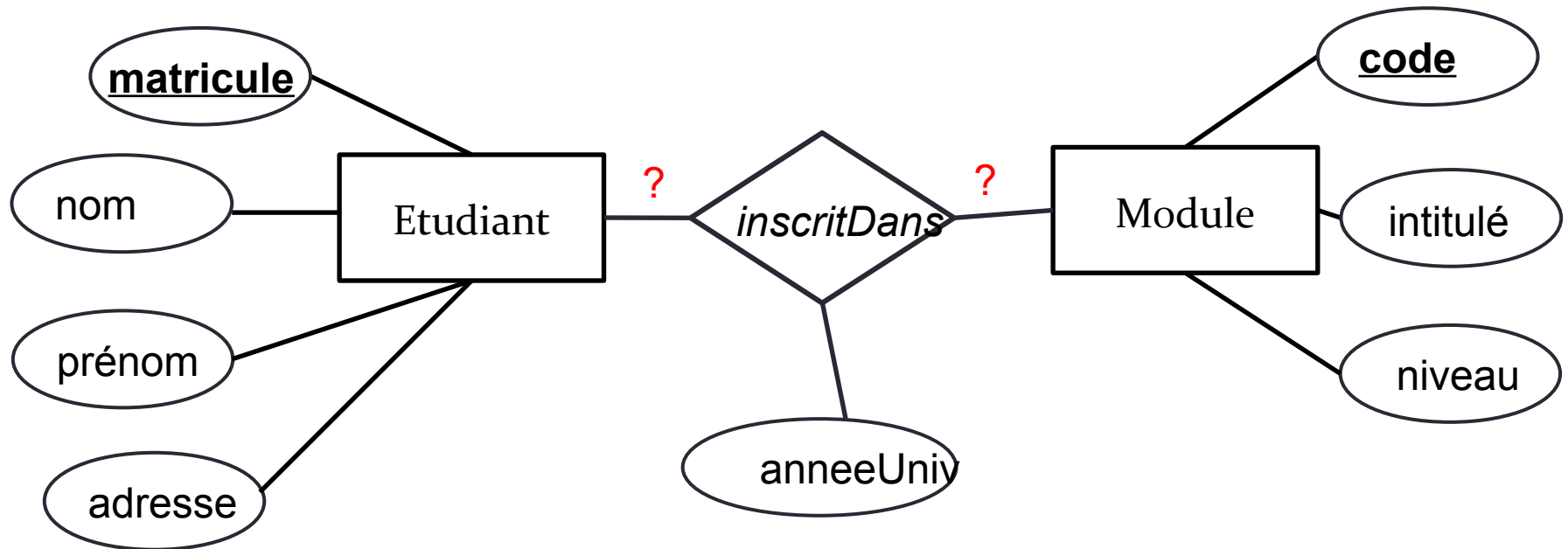
*Quel est le problème?*

# Modèle E-A : Notation graphique

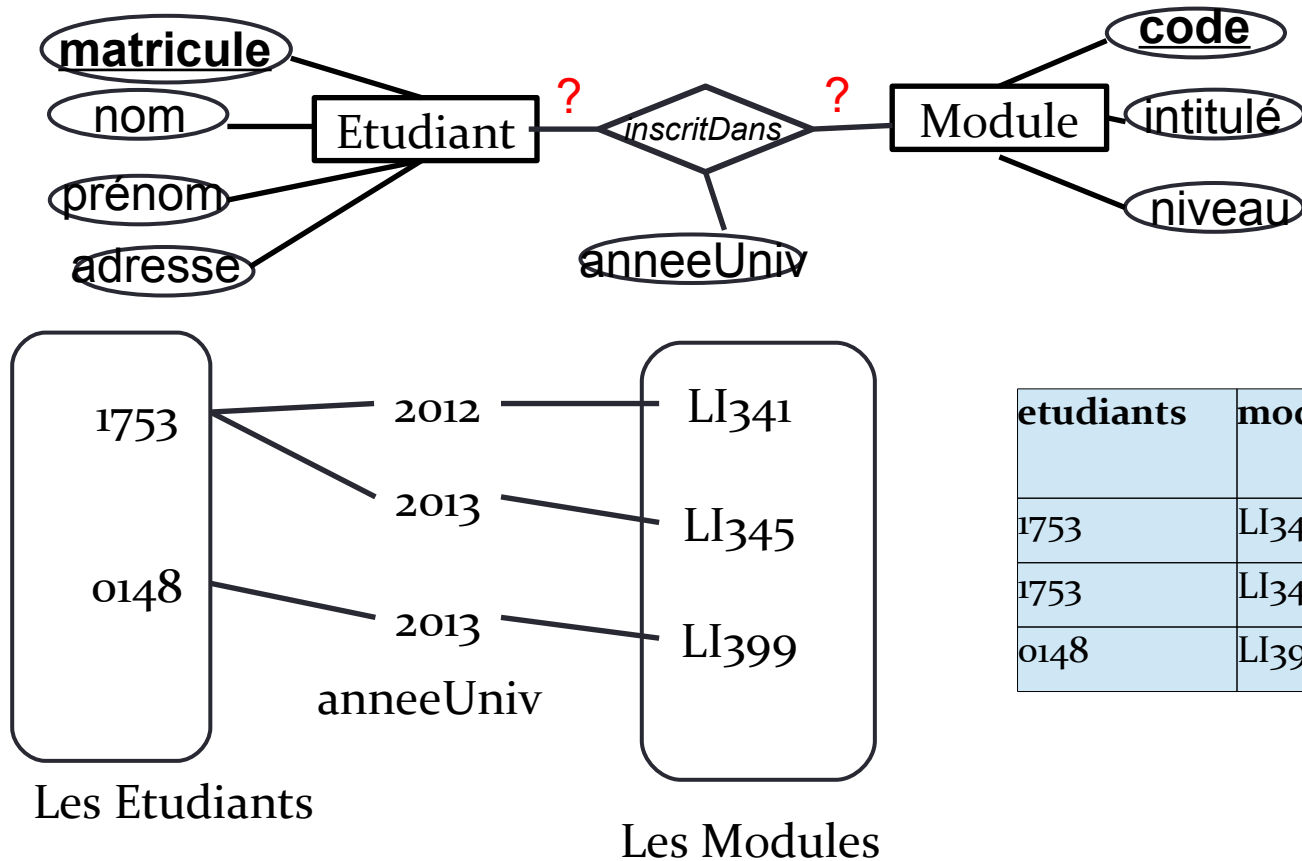


# Etude de cas : BD d'une université

|   |   |
|---|---|
| Un étudiant DOIT s'inscrire à au moins un module              | ? |
| Un module n'ouvre que s'il y a au moins un étudiant d'inscrit | ? |

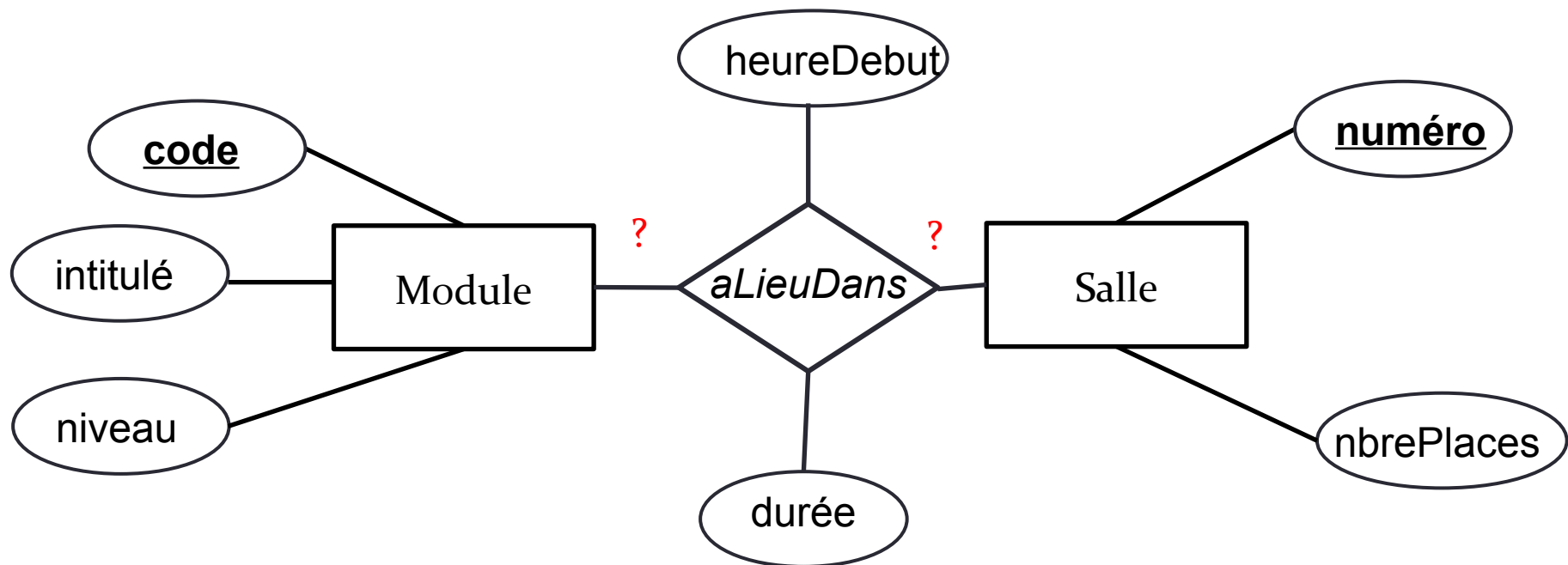


# Illustration sur une instance

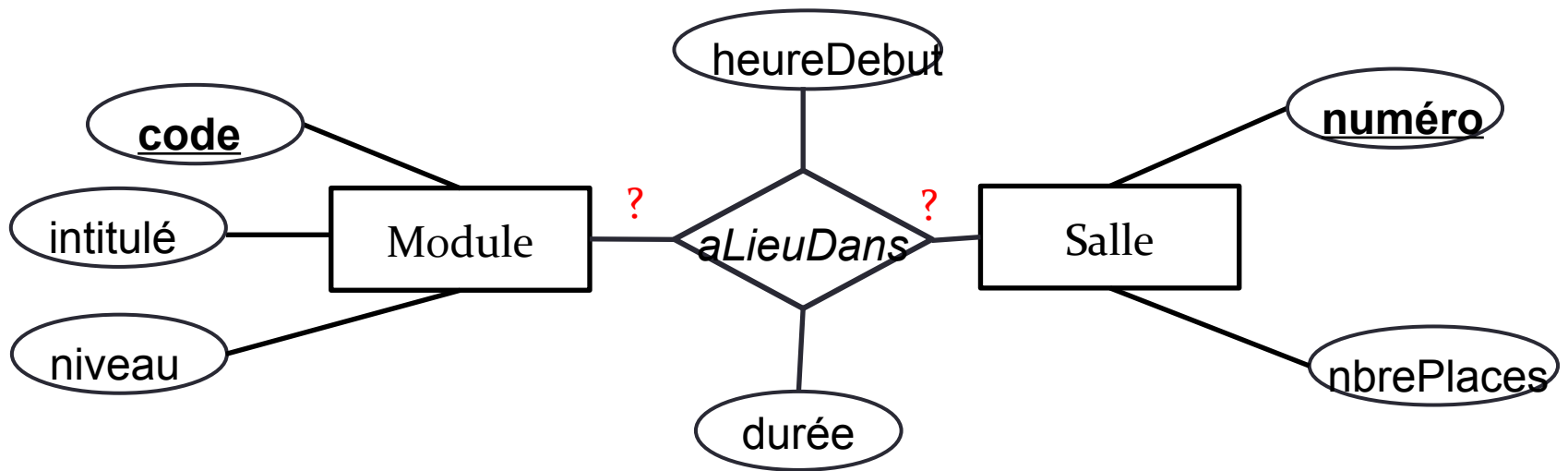


# Etude de cas : BD d'une faculté

|  |   |
|--|---|
| Un module a lieu dans une salle et une seule                           | ? |
| Un salle peut être utilisée pour plusieurs modules ou rester inoccupée | ? |



# Illustration sur une instance

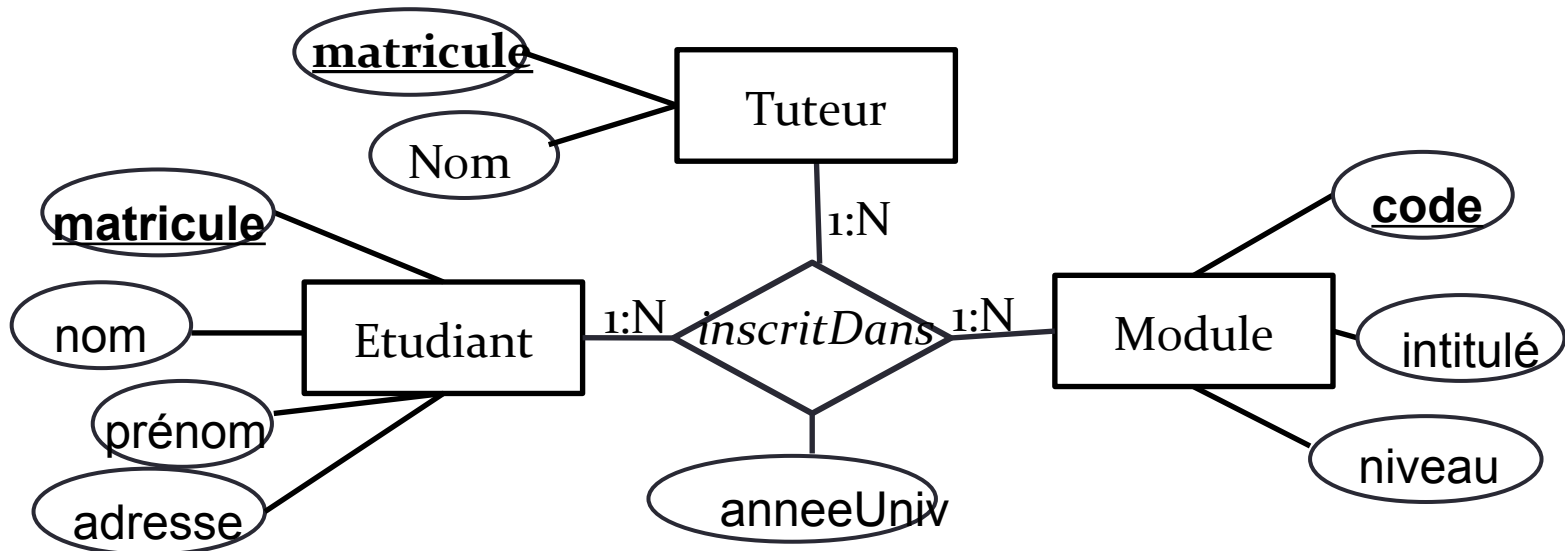


| modules | salles | heureDebut | duree |
|---------|--------|------------|-------|
| Ll341   | 105    | 830        | 120   |
| Ll345   | 105    | 1030       | 90    |
|         | 214    |            |       |

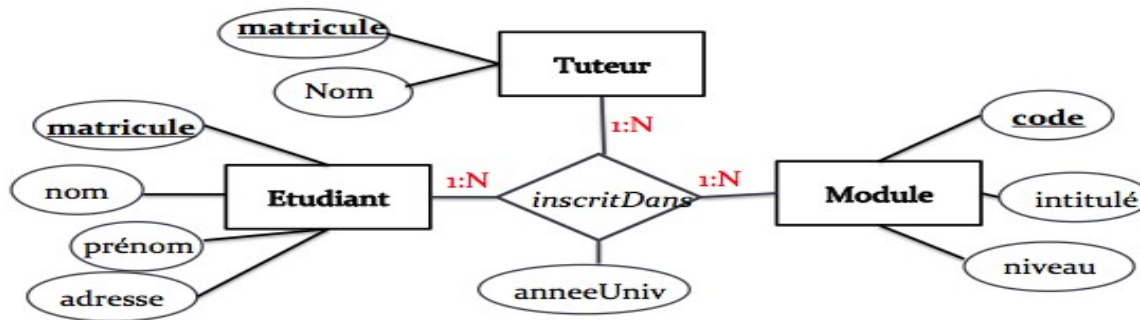
# Association ternaire

- Un étudiant doit s'inscrire dans au moins un module à une année donnée.
- Il se voit affecté un tuteur, un tuteur doit être affecté à au moins un étudiant.
- Un module doit avoir au moins un étudiant.

*Remarque:* la cardinalité de l'association d'un étudiant avec un tuteur doit être la même que la cardinalité de l'association d'un étudiant avec un module



# Illustration sur une instance



| etudiants | modules | tuteurs | anneeUniv |
|-----------|---------|---------|-----------|
| 1753      | Ll341   | 112     | 2012      |
| 1753      | Ll345   | 354     | 2102      |
| 0148      | Ll399   | 214     | 2013      |

- <sup>3</sup> *Est-il toujours possible d'exprimer une association n-aires avec des associations binaires?*



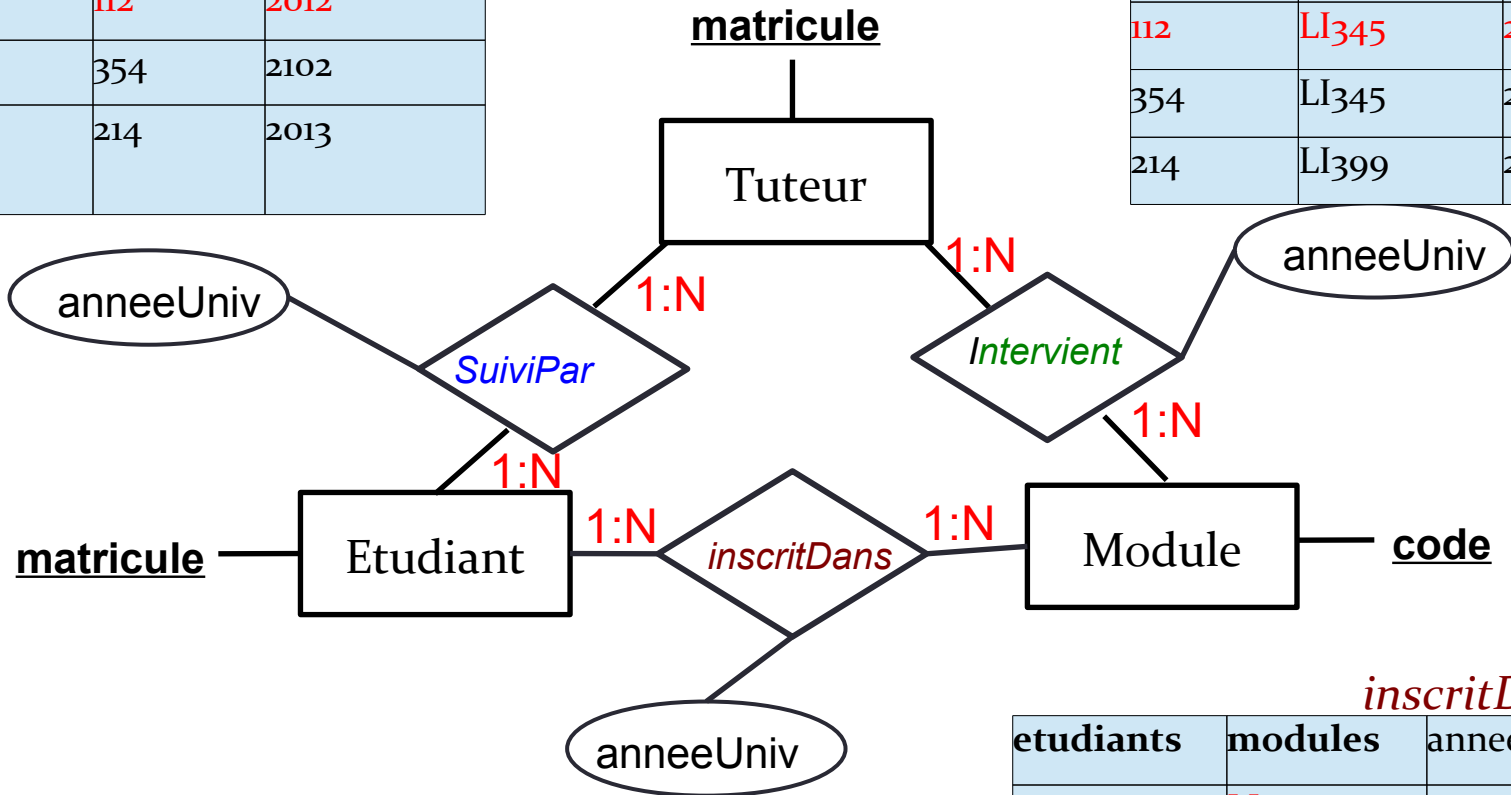
# Trois associations binaires

*SuiviPar*

| etudiants | tuteurs | anneeUniv |
|-----------|---------|-----------|
| 1753      | 112     | 2012      |
| 1753      | 354     | 2102      |
| 0148      | 214     | 2013      |

*Intervient*

| tuteurs | modules | anneeUniv |
|---------|---------|-----------|
| 112     | LI345   | 2012      |
| 354     | LI345   | 2102      |
| 214     | LI399   | 2013      |



- Un étudiant peut être inscrit dans un module et être suivi par un tuteur qui n'intervient pas dans le module !
- On peut aussi enlever l'inscription d'un étudiant à un module alors que le tuteur existe toujours !

*inscritDans*

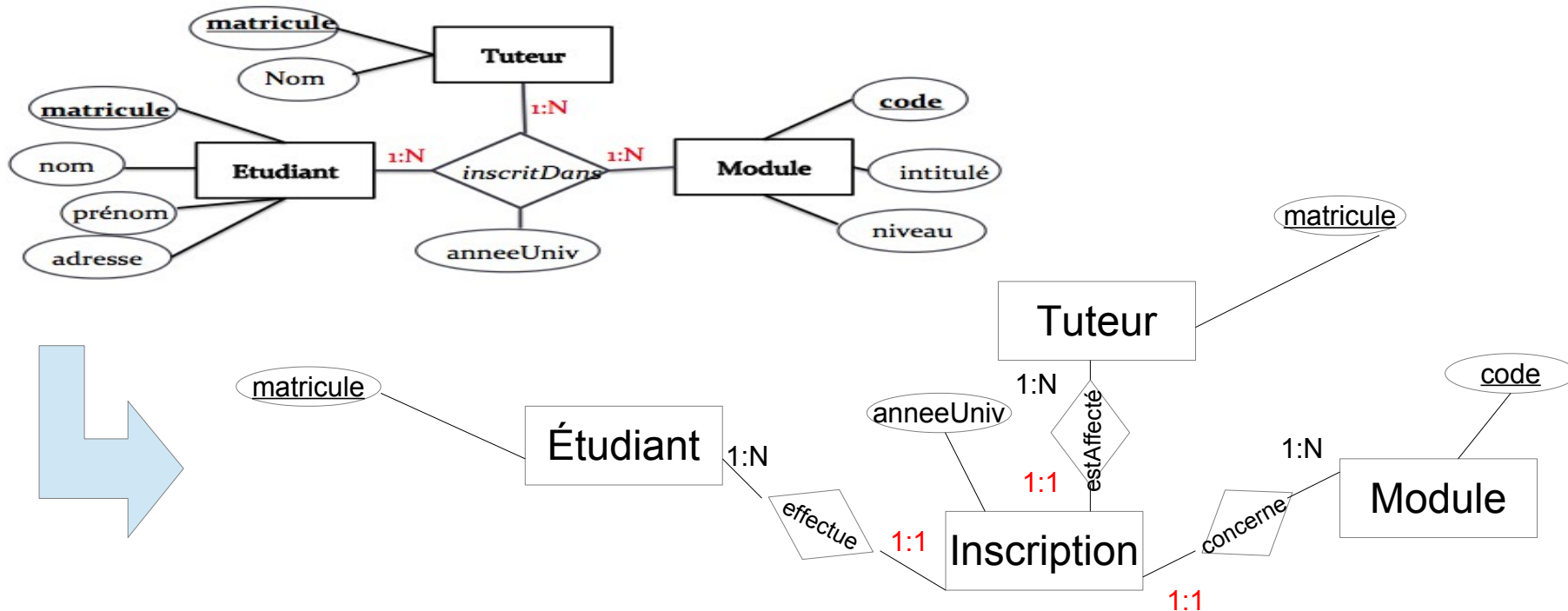
| etudiants | modules | anneeUniv |
|-----------|---------|-----------|
| 1753      | LI341   | 2012      |
| 1753      | LI345   | 2102      |
| 0148      | LI399   | 2013      |

# Règle de transformation d'une association n-aire en entité

Pour une association A entre les entités E1, ..., En

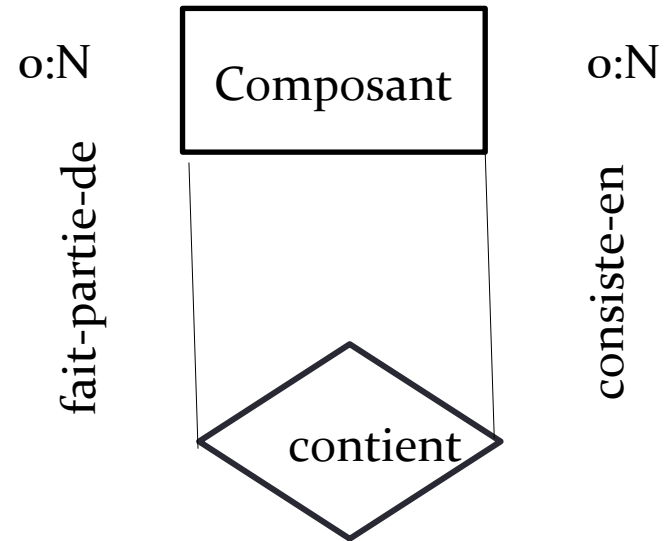
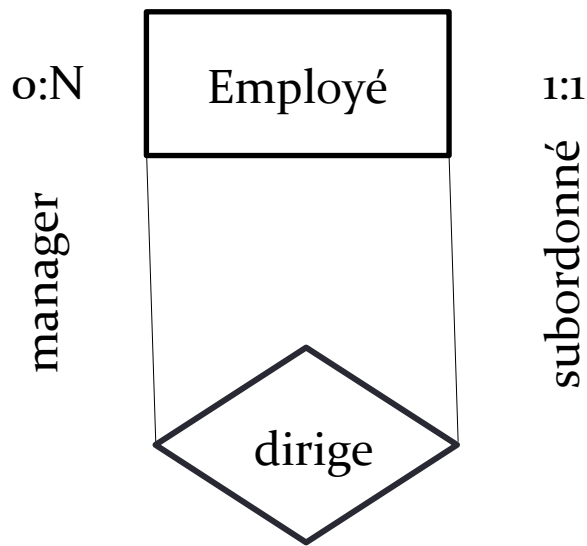
- Construire une entité E à partir des attributs de A hormis son identifiant
- Attribuer un identifiant à E (artificiel si aucun sous-ensemble de E ne peut être choisi comme identifiant)
- Créer entre chaque entité Ei et E une association Ai de cardinalité 1:1

Exemple :



# Association réflexive

- Une entité est associée à elle même
- On distingue deux *rôles*
- Les cardinalités peuvent être distinctes



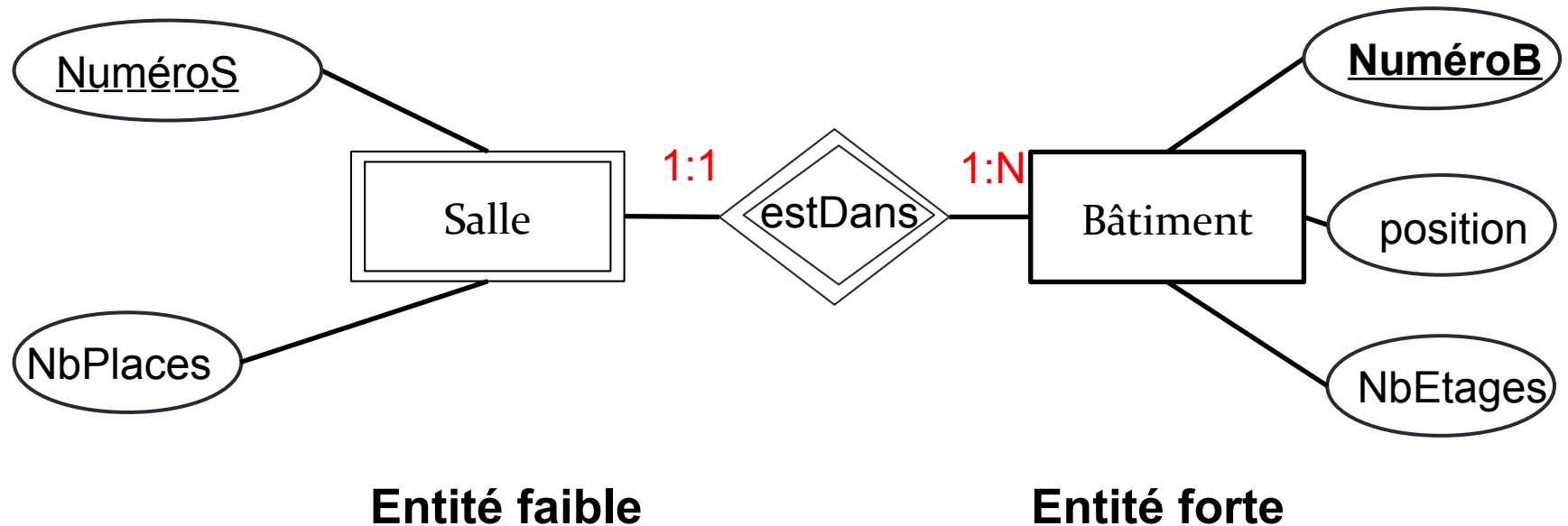
# Entités faibles

- Entités ne possédant pas assez d'attributs leur permettant d'être identifiées
- Elles sont identifiées relativement à une autre entité appelée forte

## Exemples :

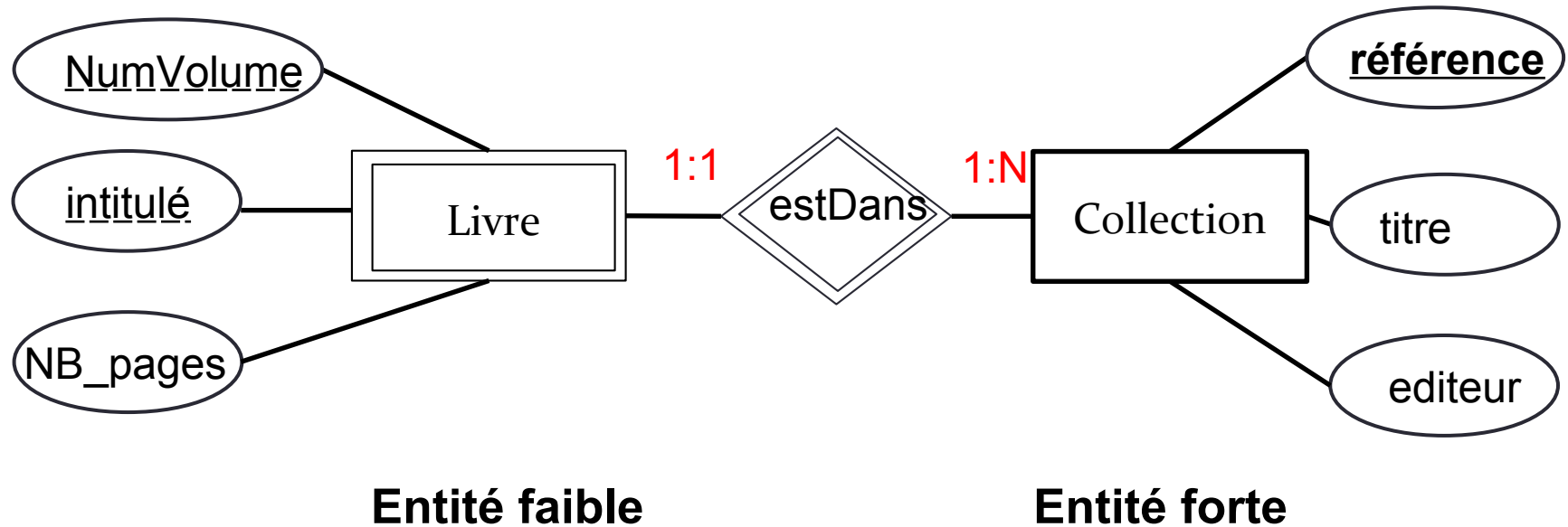
- ❏ Entité Section définie par rapport au Livre qui la contient
- ❏ Entité Livre définie par rapport à une entité Collection
- ❏ Entité Salle définie par rapport à une entité Bâtiment
- ❏ Entité Bâtiment définie par rapport à une entité Campus

# Entités faibles (exemple)



- Les entités faible possèdent des attributs discriminants dont les valeurs sont uniques (e.g *NuméroS*) dans le contexte de l'entité forte
- L'attribut discriminant est toujours souligné en pointillé
- Cardinalité **1:1** *implicite*
- Pas d'attribut pour l'association

# Entités faibles (autre exemple)



# Choix de conception

Analyse des besoins produit une spécification peu précise

→ plusieurs choix de conception possibles

Questions fréquentes :

1. Un objet du monde réel peut-il être modélisé par une entité ou par un attribut ?
2. Un objet du monde réel peut-il être modélisé par une entité ou par une association ?
3. Un attribut décrit-il une association ou une entité ?

# Choix de conception:

## Entité ou attribut ?

*Question.* Pour renseigner l'adresse d'un étudiant

- 1) rajouter un attribut adresse à l'entité étudiant ? Ou
- 2) introduire une nouvelle entité, adresse, ayant comme attributs numéro, voie, code postal ?

*Réponse.* Décision relativement facile à prendre si l'on connaît l'application et son évolution.

## Entité ou association ?

*Règle générale :* toute action impliquant deux entités donne lieu à une association.

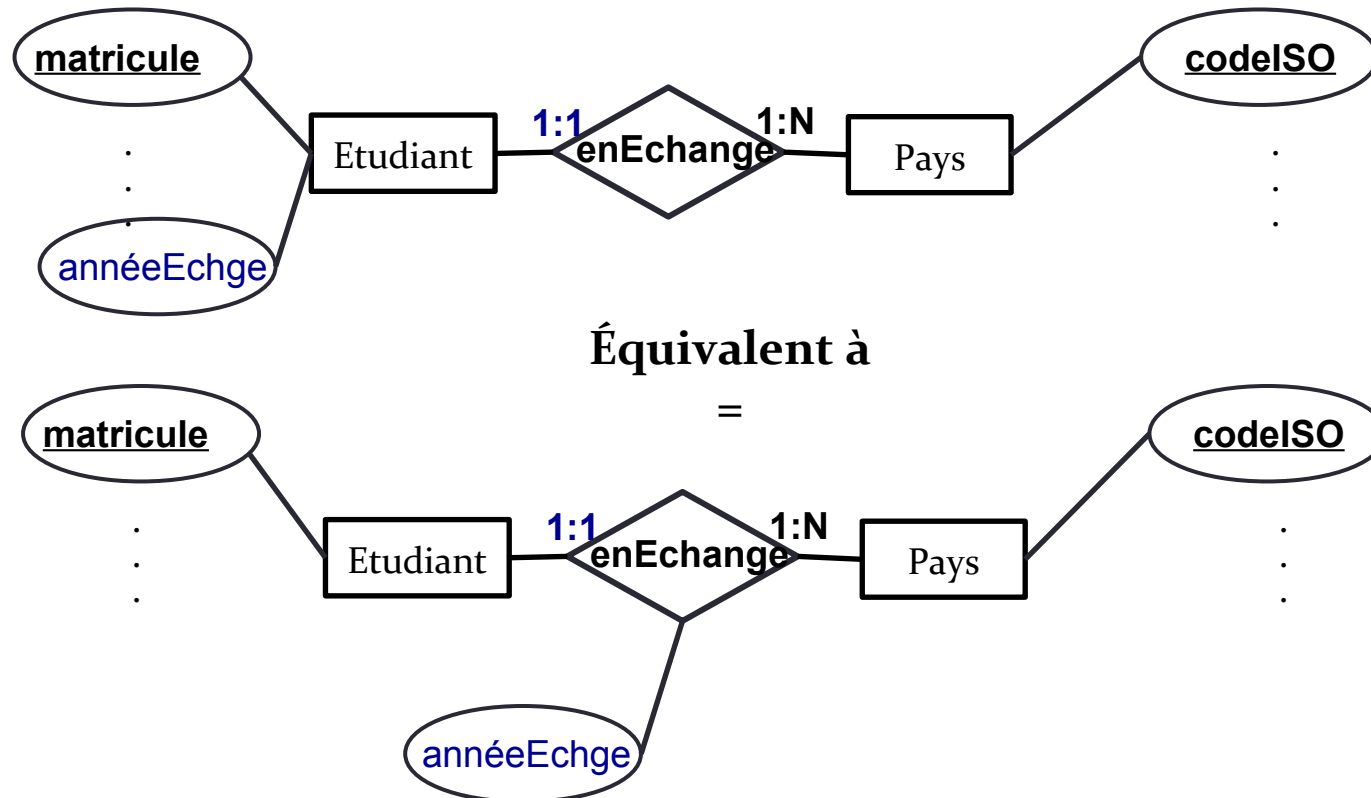
*Exemple:* (le cours d'un) module a lieu dans une salle → association  
*ALieuDans*



# Choix de conception

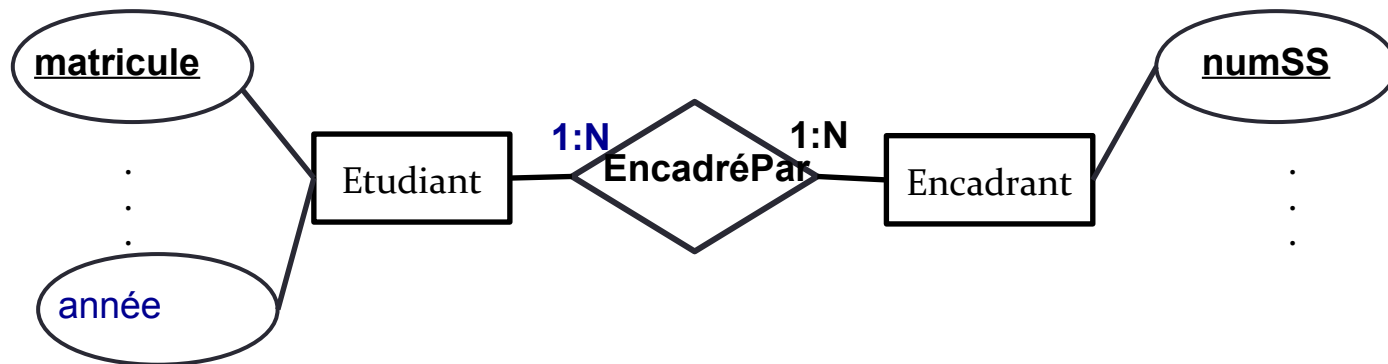
**Attribut d'une association ou d'une entité?** Le choix dépend des cardinalités

- Cardinalité **1 à plusieurs** : les deux alternatives sont équivalentes puisque l'une des deux entités participe une seule fois dans l'association
- Cardinalité **plusieurs-à-plusieurs** : la sémantique diffère selon le cas où l'attribut est au niveau de l'entité ou de l'association



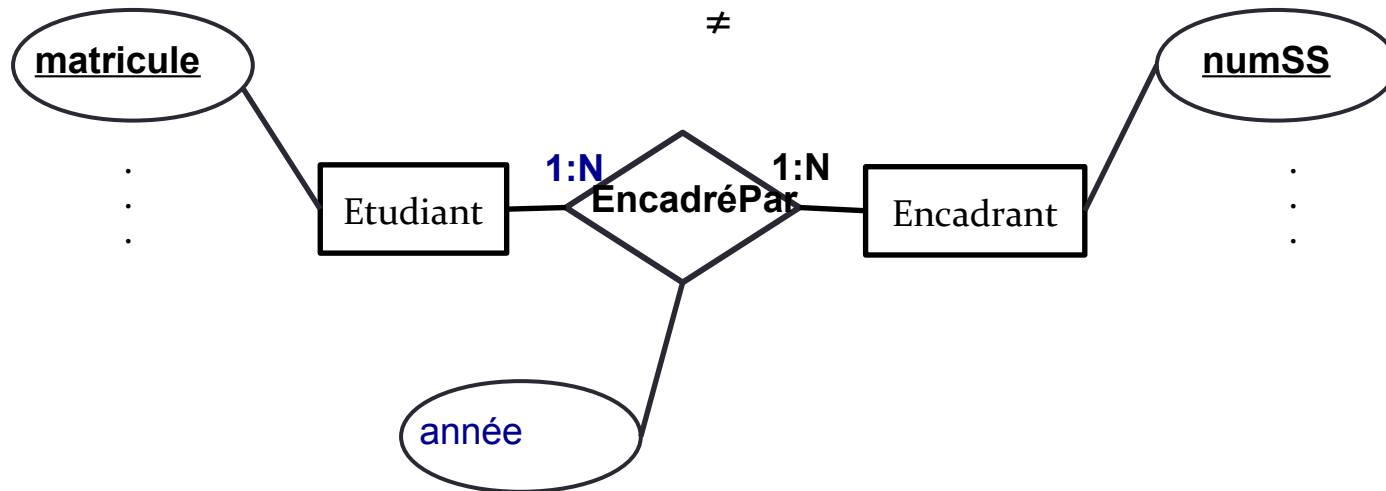
# Attribut d'association ou d'entité?

Cardinalité plusieurs-à-plusieurs



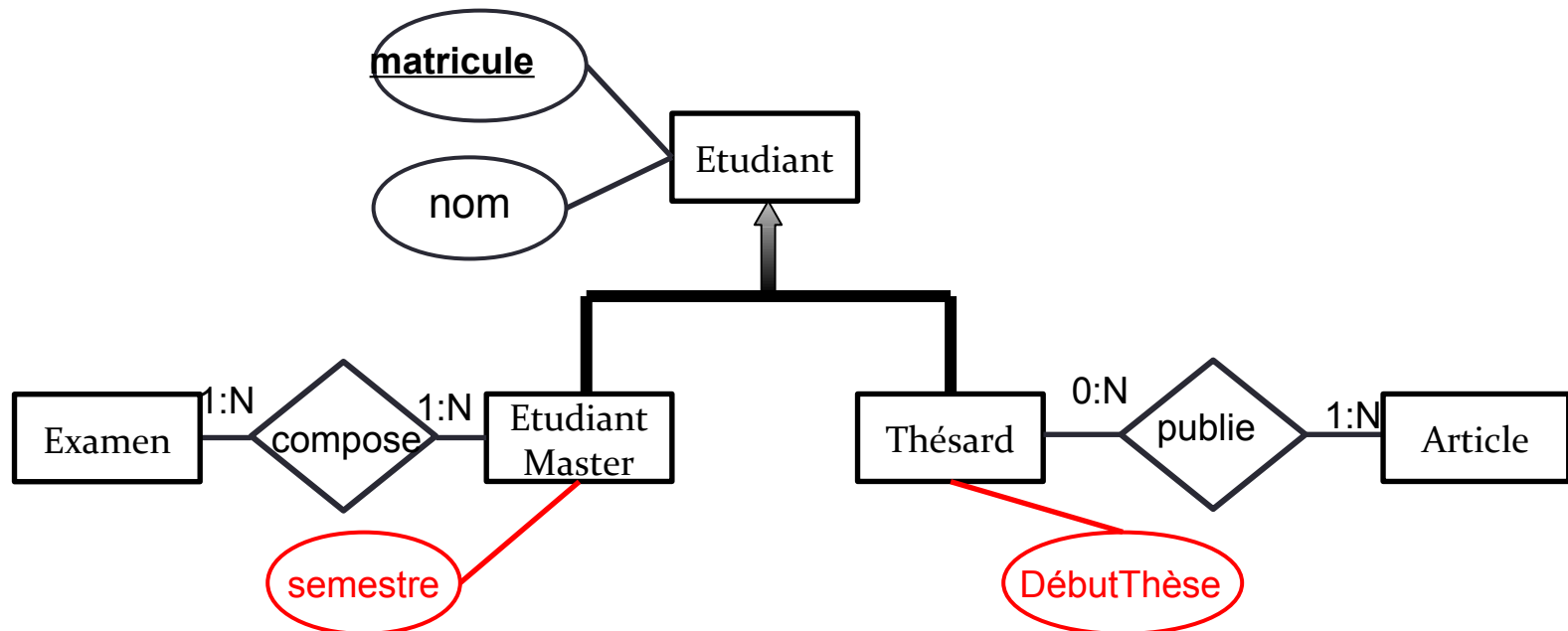
Différent de

≠



# Spécialisation

- Utile lorsque les objets à modéliser partagent certaines propriétés et possèdent d'autres propriétés propre à eux
- **Principe** : créer une entité avec les propriétés en commun dont vont hériter des propriétés plus spécifiques
- **Exemple** : il peut y avoir deux types d'étudiants
  - ❏ *Etudiants en master* passent des examens
  - ❏ *Etudiants en thèse* publient des articles scientifiques



# Conclusion

- L'intérêt des bases de données
  - Méthodologie pour la conception et la structuration de données
  - Différents niveaux d'abstraction qui permettent l'interopérabilité entre les systèmes
- Plusieurs étapes pour créer une base de données
  - Analyse de besoins
  - Modélisation des données
  - Création des données
- Modélisation des données
  - Transcription de la réalité vers le modèle Entité-Association
  - Plusieurs alternatives
- Suite: passage du modèle Entité-Association vers le modèle relationnel