BD3

Bases de données

Amélie Gheerbrant
IRIF, Université Paris Diderot
amelie@irif.fr

à partir de transparents de Cristina Sirangelo

Introduction à la modélisation

Modéliser plusieurs concepts

Rappel : la table Films vu au dernier cours

Films

titre	*annee*	*realisateur*
Alien	1979	Scott
Vertigo	1958	Hitchcock
Psychose	1960	Hitchcock
Kagemusha	1980	Kurosawa
Volte-face	1997	Woo
Pulp Fiction	1995	Tarantino
Titanic	1997	Cameron
Sacrifice	1986	Tarkovski

Et si on voulait représenter plusieurs informations sur les réalisateurs (nom prénom, date de naissance, …) ?

Modéliser plusieurs concepts

Pourquoi ne pas tout mettre dans la même table?

Films

titre	*annee*	*realisateur*	*prenom*	*naissance*
Alien	1979	Scott	Ridley	1943
Vertigo	1958	Hitchcock	Alfred	1899
Psychose	1960	Hitchcock	Alfred	1899
Kagemusha	1980	Kurosawa	Akira	1910
Volte-face	1997	Woo	John	1946
Pulp Fiction	1995	Tarantino	Quentin	
Titanic	1997	Cameron	James	1954
Sacrifice	1986	Tarkovski	Andrei	1932

Problèmes avec la table simple

Redondance

Les informations sur les réalisateurs sont dupliquées pour chaque film qu'ils ont réalisé.

Anomalies d'insertion

Possibilité d'insérer des données incohérentes

exemple : le même réalisateur avec deux dates de naissance différentes

Anomalies de mise à jour

Si on a besoin de rectifier une erreur sur l'année de naissance, il faut penser à le faire pour tous les films. Sinon, la table contient des informations incohérentes…

Anomalies de suppression

La suppression d'un film de la table, entraîne la suppression des informations associées sur le réalisateur.

Si le réalisateur n'était présent que pour un seul film, la suppression de ce film entraîne la disparition de toutes les informations relatives au réalisateur.

Solutions

- Utiliser plusieurs tables pour représenter les films et les réalisateurs indépendamment les uns des autres :
  - » insertions, mises-à-jour et suppressions indépendantes.

- Identifier les films (et les réalisateurs) pour s'assurer qu'aucun doublon ne figure dans nos tables

- Lier les films et les réalisateurs sans introduire de redondance d'information

Solution : ébauche

Films: 2 films ne peuvent avoir le même titre (supposons-le)

Réalisateurs : 2 réalisateurs peuvent avoir le même nom ; on les distingue grâce à un identificateur (`id`)

Films

titre	*annee*
Alien	1979
Vertigo	1958
Psychose	1960
Kagemusha	1980
Volte-face	1997
Pulp Fiction	1995
Titanic	1997
Sacrifice	1986

Realisateurs

id	*nom*	*prenom*	*naissance*
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910
4	Woo	John	1946
5	Tarantino	Quentin	
6	Cameron	James	1954
7	Tarkovski	Andrei	1932

Solution : ébauche

Il n'y a plus de redondance dans la base de données :

Les informations sur chaque réalisateur n'apparaissent qu'une fois

Plus d'incohérence en cas de mise à jour

Mais l'information :

"Qui est le réalisateur du film ?" a disparu

Solution

Ajout d'un attribut dans la table film : "`realisateur`"

avec une contrainte de clef étrangère Films[realisateur] ⊆ Realisateurs[id]

Toujours pas de redondance

Films

titre	*annee*	*realisateur*
Alien	1979	1
Vertigo	1958	2
Psychose	1960	2
Kagemusha	1980	3
Volte-face	1997	4
Pulp Fiction	1995	5
Titanic	1997	6
Sacrifice	1986	7

Realisateurs

id	*nom*	*prenom*	*naissance*
1	Scott	Ridley	1943
2	Hitchcock	Alfred	1899
3	Kurosawa	Akira	1910
4	Woo	John	1946
5	Tarantino	Quentin	
6	Cameron	James	1954
7	Tarkovski	Andrei	1932

Solution

Insertion

Les informations concernant un même réalisateur sont représentées une seule fois dans la base : pas possible de stocker des information incohérentes (e.g. deux dates de naissance différentes)

ou bien il s'agit d'un réalisateur different !

Mise à jour

Plus de redondance, donc une mise à jour ne risque pas d'introduire d'incohérence

Suppression

La suppression d'un film n'affecte pas le réalisateur

Modélisation

Schéma final avec ses contraintes (une clef primaire choisie pour chaque table, et les clefs étrangères)

Films

titre	annee	realisateur

Realisateurs

id	nom	prenom	naissance

Aussi dénoté :

Films (titre, année, réalisateur)

Realisateurs (id, nom, prénom, naissance) Films[realisateur] ⊆ Realisateurs[id]

Remarque :

la modélisation ne concerne que le schéma de la base de données,

pas les données (tuples)

Un schéma de base de données plus complexe

- On veut représenter:
  - Des films,
  - Leurs réalisateurs et acteurs,
  - Les pays où ces films ont été réalisés,
  - Des utilisateurs du site des films
- On veut aussi permettre aux utilisateurs de noter les films

Solution : ébauche

- Les informations concernant les acteurs et les réalisateurs seront vraisemblablement les mêmes (nom, prénom, année de naissance)
  - on les représente avec une unique table Artistes

Artistes

id	nom	prenom	naissance

- Avec chaque film on représente l'id de l'artiste ayant réalisé le film

Films

titre	annee	id_realisateur

- Comment représenter les acteurs qui jouent dans un film sans redondance?
  - Pourquoi la solution adoptée pour le réalisateur (ajouter son id dans la table Films) n'est pas valable?

Solution : ébauche

- Un film a plusieurs acteurs, un attribut id_acteur ne peut représenter qu'une seule valeur !
- Solution : une nouvelle table qui fait le "lien" entre films et acteurs
  - Seuls les identifiants apparaissent dans cette table, pour éviter la redondance

Casting

titre_film	id_acteur

Films

titre	annee	id_realisateur

Artistes

id	nom	prenom	naissance

- Et si on voulait représenter également les rôles des acteurs dans les film ?

Solution : ébauche

- Le rôle n'est pas associé au film ou à l'acteur, mais à la participation de l'acteur dans le film.
  - attribut de la table Casting

Casting

titre_film	id_acteu	rôle

Films

titre	annee	id_realisateur

Artistes

id	nom	prenom	naissan

Solution : ébauche

- Représentation des pays :

Pays

code	nom	langue

- Et des utilisateurs : chaque utilisateur a un pseudo, nom, prénom, mdp, mais également un pays
  - comment représenter le pays?

Solution : ébauche

- Représentation des pays :

Pays

code	nom	langue

- Et des utilisateurs : chaque utilisateur a un pseudo, nom, prénom, mdp, mais également un pays
  - comment représenter le pays?

Utilisateurs

pseudo	email	nom	prenom	mdp	code_pays

- Comment représenter les notes que les utilisateurs donnent aux films?

Solution : ébauche

- Un utilisateur peut noter plusieurs films et un film peut être noté par plusieurs utilisateurs
  - ⇒ la note ne peut pas être un attribut du film, ni de l'utilisateur
- Solution : une nouvelle table qui fait le "lien" entre films et utilisateurs
  - Seuls les identifiants apparaissent dans cette table, pour éviter la redondance
  - la note est un attribut additionnel de cette table

Notation

titre_film	pseudo	note

Solution : schema complet

Films

titre	annee	id_realisateur

Artistes

id	nom	prenom	naissance

Cast

titre_film	id_acteur

Pays

code	nom	langue

Utilisateurs

pseudo	email	nom	prenom	mdp	code_pays

Notation

titre_film	pseudo	note

Schéma relationnel : spécification des contraintes

- Une clé primaire est choisie pour chaque relation

  Film (titre, année, id_realisateur)

  Artiste (id, nom, prénom, naissance)

  Utilisateur (pseudo, e-mail, nom, prénom, mdp, code_pays)

  Pays (code, nom, langue)

  Casting (titre_film, id_acteur)

  Notation (titre_film, pseudo, note)

- Remarques :
  - clef primaire pour Casting : un film a plusieurs acteurs et un acteur peut jouer dans plusieurs films
  - clef primaire pour Notation : un film peut être noté par plusieurs utilisateurs et un utilisateur peut noter plusieurs films, en revanche un utilisateur donne une seule note à un film

Schéma relationnel : spécification des contraintes

- Les attributs qui font référence à d'autres tables génèrent des contraintes de clé étrangère

  Film (titre, année, id_realisateur)
  Film[id_realisateur] ⊆ Artiste[id]

  Artiste (id, nom, prénom, naissance)

  Utilisateur (pseudo, e-mail, nom, prénom, mdp, code_pays)
  Utilisateur[code_pays] ⊆ Pays[code]

  Pays (code, nom, langue)

  Casting (titre_film, id_acteur)
  Casting[titre_film] ⊆ Film[titre] Casting[id_acteur] ⊆ Artiste[id]

  Notation (titre_film, pseudo, note)
  Notation[titre_film] ⊆ Film[titre] Notation[pseudo] ⊆ Utilisateur[pseudo]

- D'autres contraintes (attributs requis, clef candidates,…) sont exprimées en phase d'implémentation

Implémentation du schéma relationnel

Le schéma relationnel est ensuite implémenté dans le SGBD avec une suite de commandes CREATE TABLE, après avoir choisi le type de chaque attribut

De plus :

- les clefs candidates génèrent des contraintes UNIQUE
- les attributs requis sont identifiés et génèrent des contraintes NOT NULL
- des contraintes plus complexes sont implémentées avec des conditions de CHECK et/ou des assertions

Implémentation du schéma relationnel

```
CREATE TABLE Artiste (
      id INTEGER PRIMARY KEY,
      nom VARCHAR(50) NOT NULL,
      prenom VARCHAR(50) NOT NULL,
      naissance INTEGER,
      UNIQUE (nom, prenom, naissance)
);
```

Implémentation du schéma relationnel

```
CREATE TABLE Pays (
      code CHAR(2) PRIMARY KEY,
      nom VARCHAR(50) NOT NULL,
      langue VARCHAR(30)
 );
```

Implémentation du schéma relationnel

```
CREATE TABLE Utilisateur (
      pseudo VARCHAR(50) PRIMARY KEY,
      email VARCHAR(50) NOT NULL UNIQUE,
      nom VARCHAR(20) NOT NULL,
      prenom VARCHAR(20),
      mdp VARCHAR(60) NOT NULL,
      naissance INTEGER,
      code_pays CHAR(2) NOT NULL,
      FOREIGN KEY (code_pays) REFERENCES Pays(code)
);
```

Implémentation du schéma relationnel

```
CREATE TABLE Films (
   titre VARCHAR(50) PRIMARY KEY,
   annee INTEGER CHECK (annee BETWEEN 1930 AND 2018) NOT NULL,
   genre VARCHAR(30)
            CHECK (genre IN ('Documentaire', 'Action', 'Drame'));
   id_realisateur INTEGER,
   FOREIGN KEY (id_realisateur) REFERENCES Artiste (id)
);
```

Implémentation du schéma relationnel

```
CREATE TABLE Notation (
    titre_film VARCHAR(50),
    pseudo VARCHAR(50),
    note INTEGER NOT NULL,
    PRIMARY KEY (titre_film, pseudo),
    FOREIGN KEY (titre_film) REFERENCES  Films(titre),
    FOREIGN KEY (pseudo) REFERENCES  Utilisateurs(pseudo),
);
```

Similaire pour la table `Casting`

Modélisation

- Comment arriver à produire un schéma correct ? (tables, attributs, contraintes)

  Film (titre, année, id_realisateur)
  Film[id_realisateur] ⊆ Artiste[id]

  Artiste (id, nom, prénom, naissance)

  Utilisateur (pseudo, e-mail, nom, prénom, mdp, code_pays)
  Utilisateur[code_pays] ⊆ Pays[code]

  Pays (code, nom, langue)

  Casting (titre_film, id_acteur)
  Casting[titre_film] ⊆ Film[titre] Casting[id_acteur] ⊆ Artiste[id]

  Notation (titre_film, pseudo, note)
  Notation[titre_film] ⊆ Film[titre] Notation[pseudo] ⊆ Utilisateur[pseudo]

- Modélisation : Passage d'une spécification informelle des données à une schéma de BD qui les représente correctement
  - une procédure qui donne plus de garantie que le "bon sens"

Modélisation conceptuelle :
le modèle Entité/Association (E/R)

Sources (quelques slides empruntés et réadaptés) :

- *DB Systems concepts* , A. Silberschatz, Yale U. & H. Korth, Lehigh U. & S.Sudarshan, IIT Bombay

Modele E/R (*Entity/Relationship* en anglais)

- Un formalisme pour la modélisation conceptuelle des données:
  - quelles données
  - comment elles sont reliées
  - quelle contraintes elle satisfont
- Introduit par Chen :
  - *Peter P.-S. Chen. The Entity-Relationship Model, Toward a Unified View of Data, ACM Transactions on Database Systems (TODS) 1:(1), 1976*

- La plupart des modèles conceptuels utilisés encore aujourd'hui dans l'industrie s'inspirent du modele E/R (e.g., diagrammes de classe UML, méthodes Merise…)
- Plusieurs variantes dans la notation : on fixera une notation, beaucoup de notations alternatives existent !

Modélisation des données avec E/R

- Les données peuvent être modélisées comme :
  - un ensemble d'*entités*,
  - des *associations* entre ces entités

Entité

- **Entité :** une classe d'objets avec des propriétés communes
  - Exemples : les étudiants de Paris 7, les Employés d'une entreprise, les spectacles dans Paris,…

- Les entités ont des **attributs** (i.e. leurs propriétés)
  - Exemple: les étudiants de Paris 7 ont un *nom*, un *numéro étudiant*, une *adresse*
  - Les spectacles ont une *description*, un *genre*, un *lieu*, des *horaires*

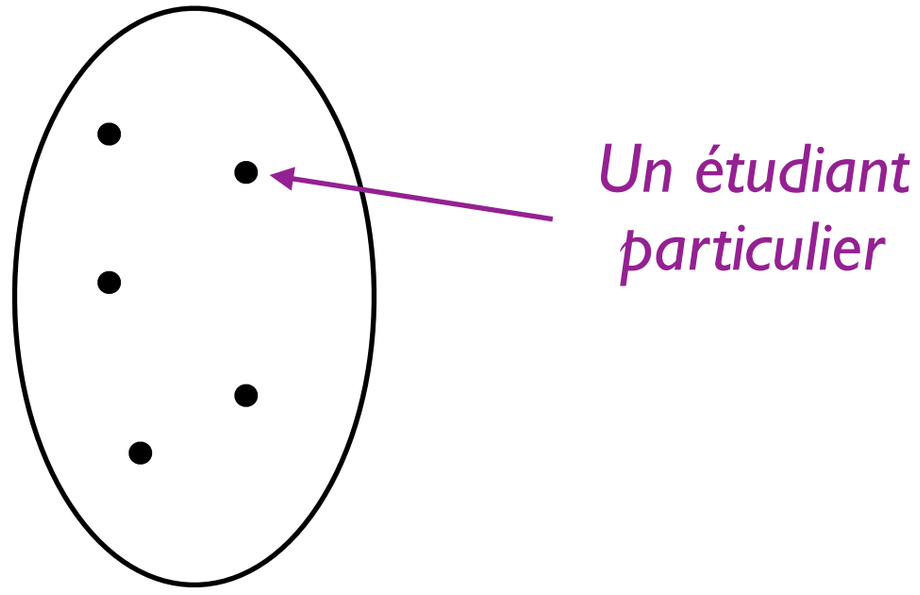
Entités et attributs

Syntaxe :

Étudiant
numero étudiant nom adresse

Sémantique :

- Une entité dénote un ensemble d'objets, appelés *instances* de l'entité



instances(**Étudiant**)

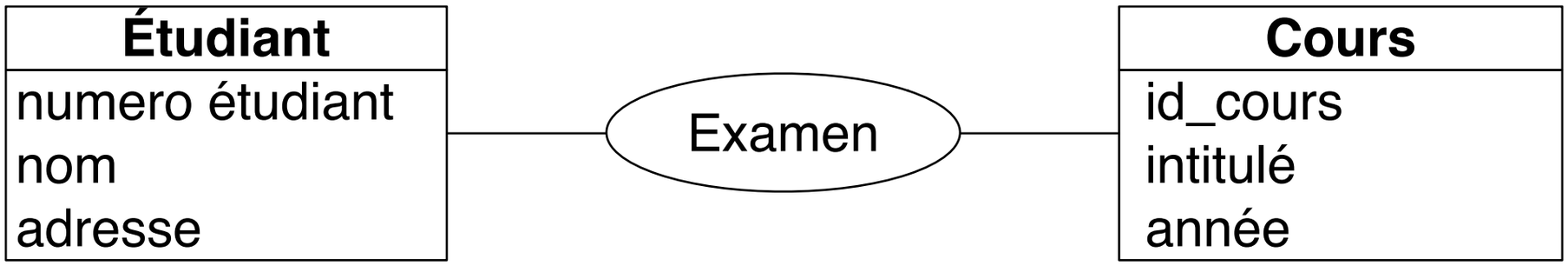
- Un attribut dénote une fonction qui associe à chaque instance de l'entité une valeur dans un domaine
  - Ex. nom : Instances (Etudiant) → String

Associations

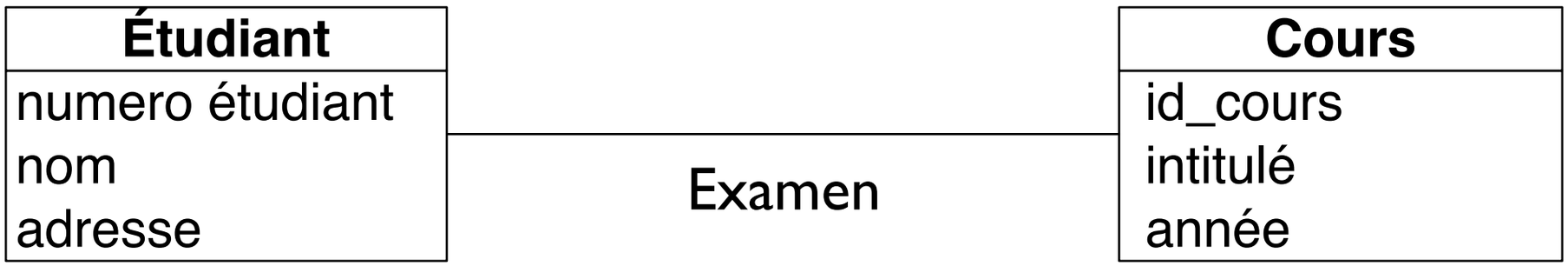
- **Association :** représente un lien entre deux ou plusieurs entités
- Exemples d'associations :
  - l'association **Examen** relie l'entité **Etudiant** à l'entité **Cours**
  - l'association **Affectation** relie l'entité **Employé** à l'entité **Département**

Associations

Syntaxe :



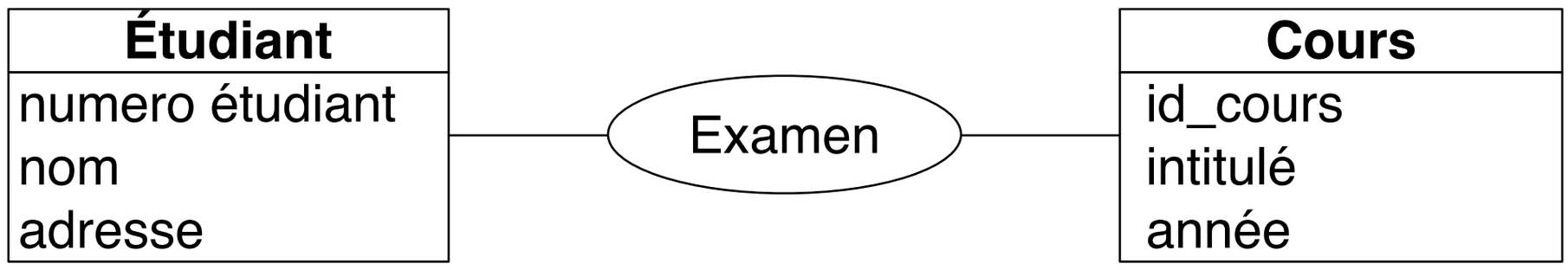
alternative :



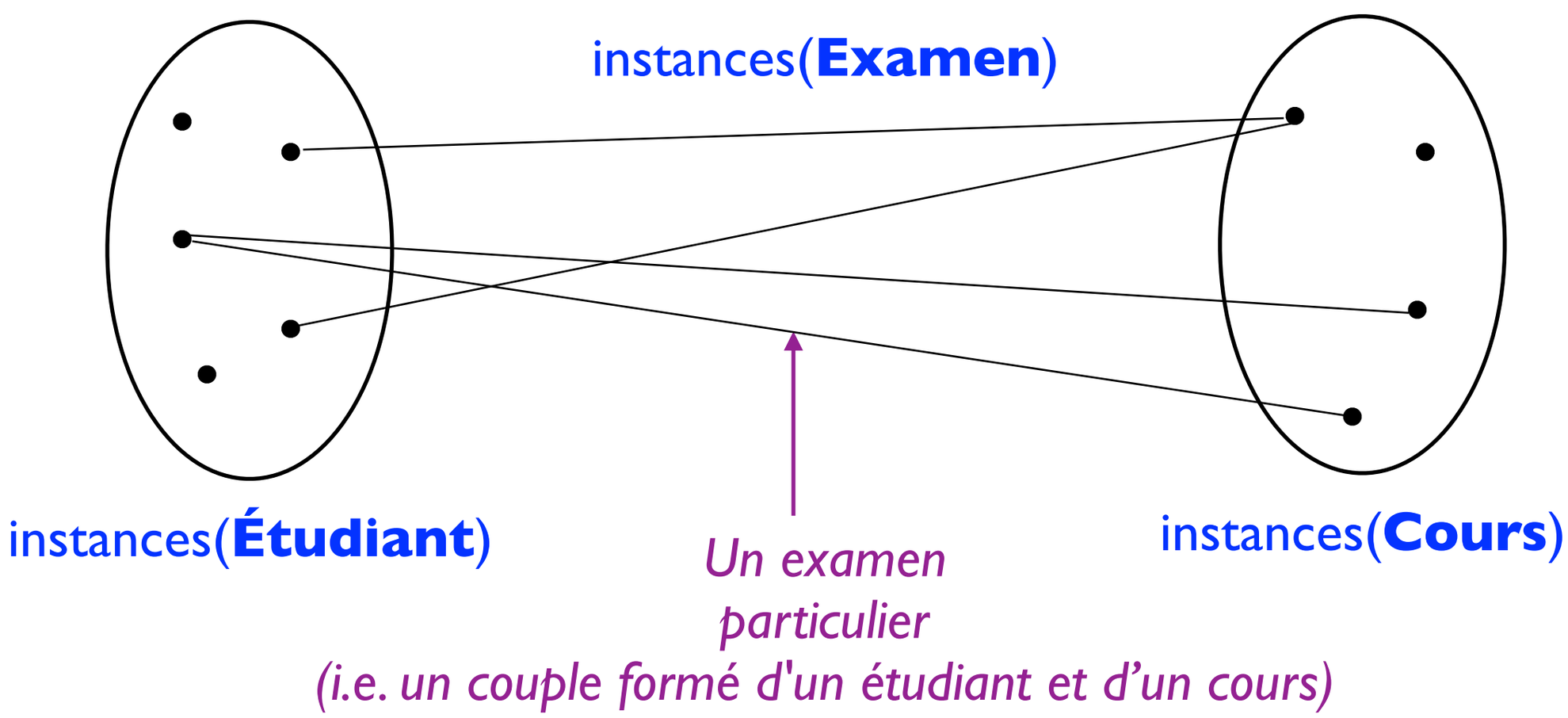
(Le nom de l'association peut être omis)

Associations

Syntaxe :

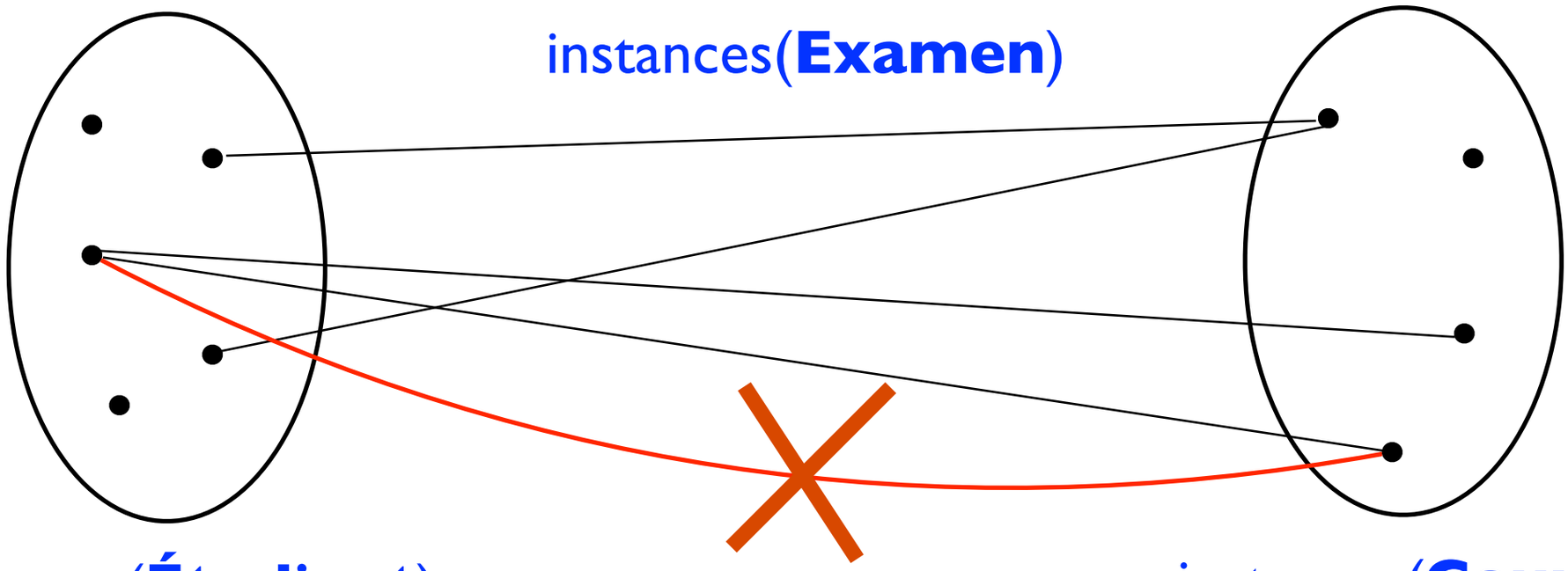


Sémantique : instances(Examen) ⊆ instances(Etudiant) × instances(Cours)



Associations

- **Remarques sur la sémantique**



instances(**Examen**) est un **ensemble** de couples : pas de doublons

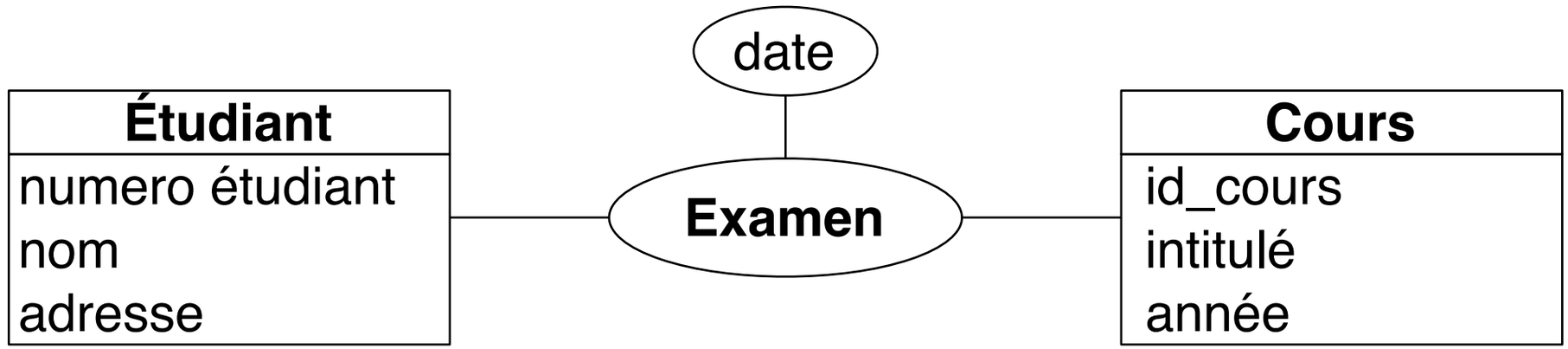
⇒ modéliser Examen comme association introduit automatiquement la contrainte suivante : un étudiant ne peut pas passer l'examen d'un même cours plusieurs fois

Si ce n'est pas le cas dans la réalité que nous modélisons
(e.g. examens non réussis, à répéter…) il faut changer la modélisation

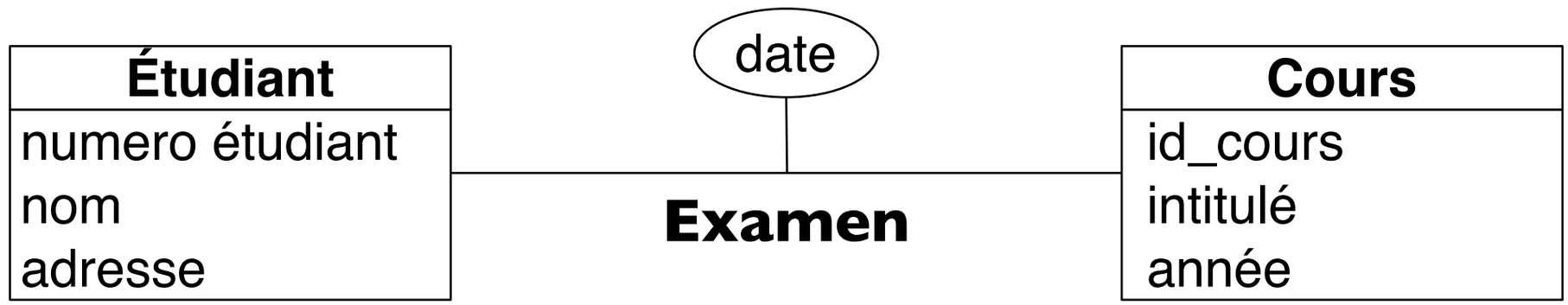
Attributs d'associations

- **Attribut d'une association** : décrit une propriété des instances de l'association
  - Exemple : chaque examen a une date

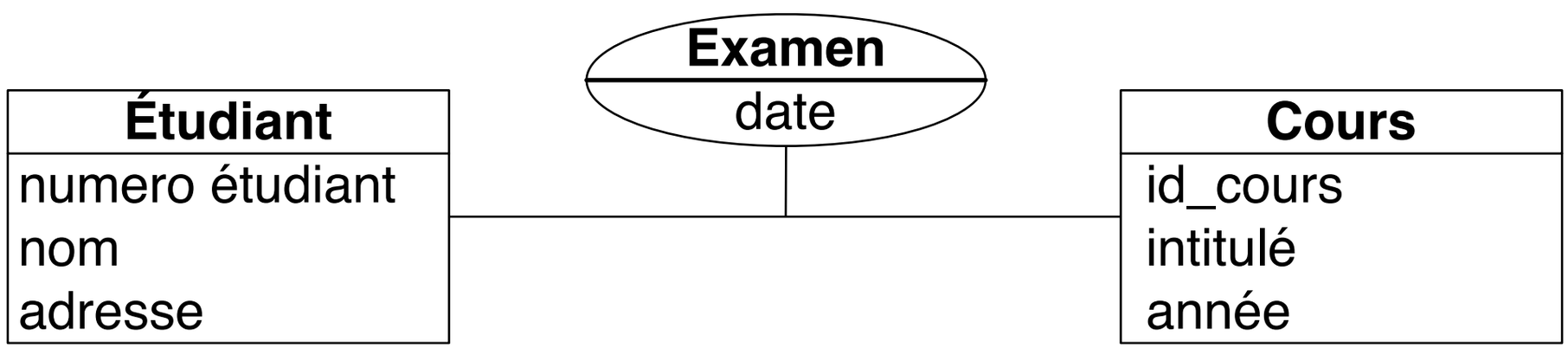
Syntaxe :



alternative :



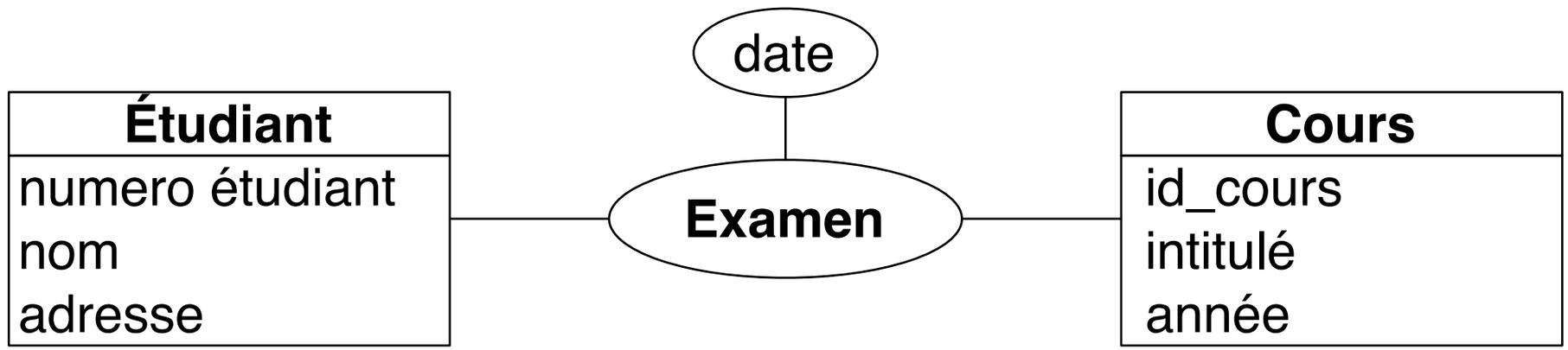
alternative :



Attributs d'associations

- **Attribut d'une association** : décrit une propriété des instances de l'association
  - Exemple : chaque examen a une date

Syntaxe :



Sémantique :

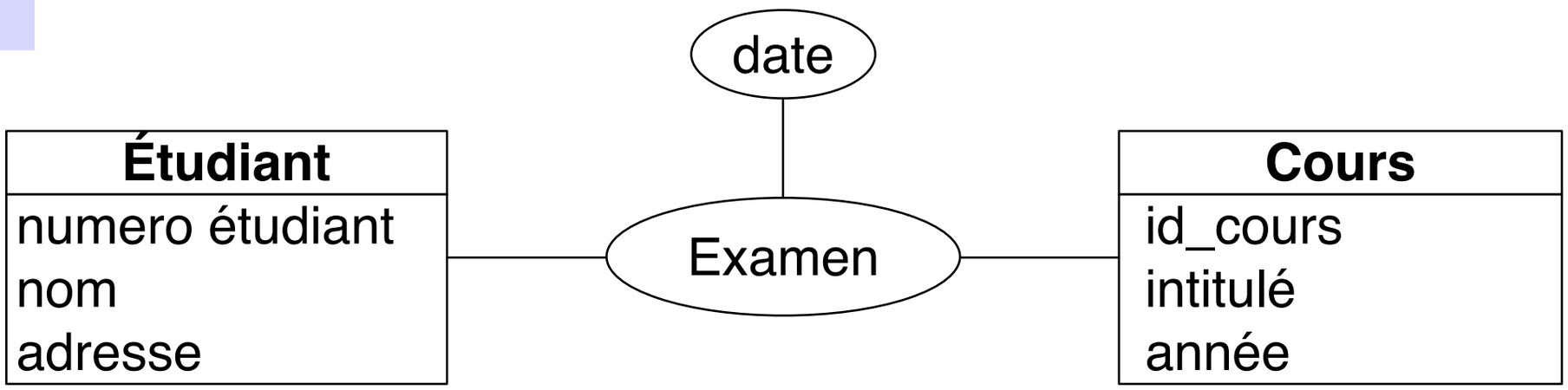
Un attribut d'une association dénote une fonction qui associe à chaque instance de l'association une valeur dans un domaine

- date : Instances (Examen) → Dates
  (i.e date est une propriété des couples <étudiant, cours> qui représentent un examen)

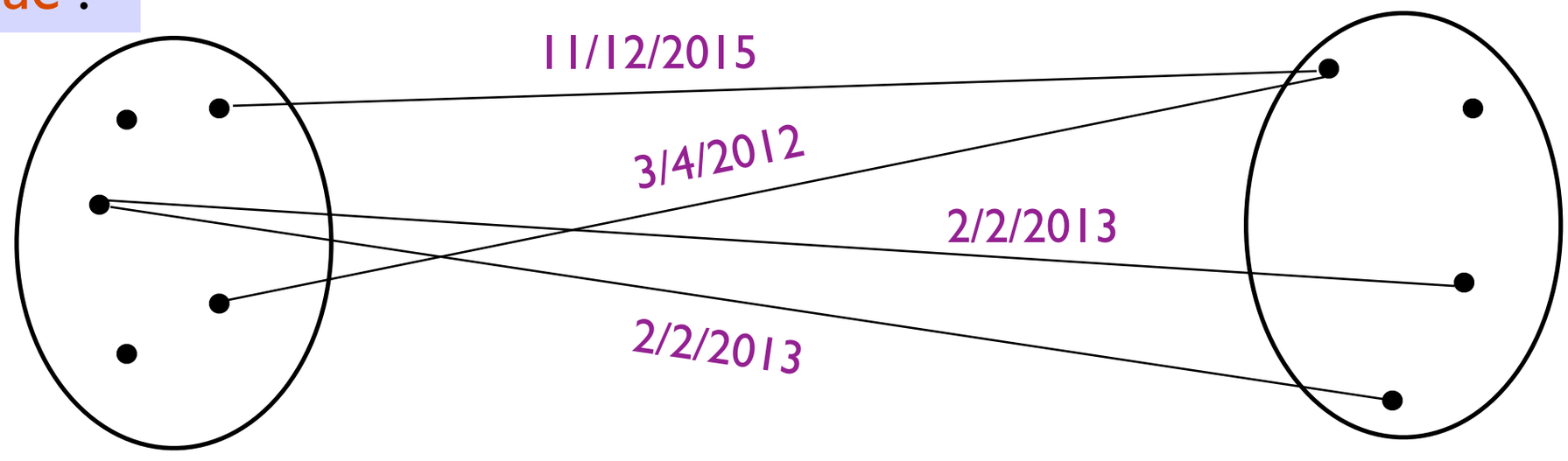
Attributs d'associations

- **Attribut d'une association** : décrit une propriété des instances de l'association
  - Exemple : chaque examen a une date

Syntaxe :



Sémantique :

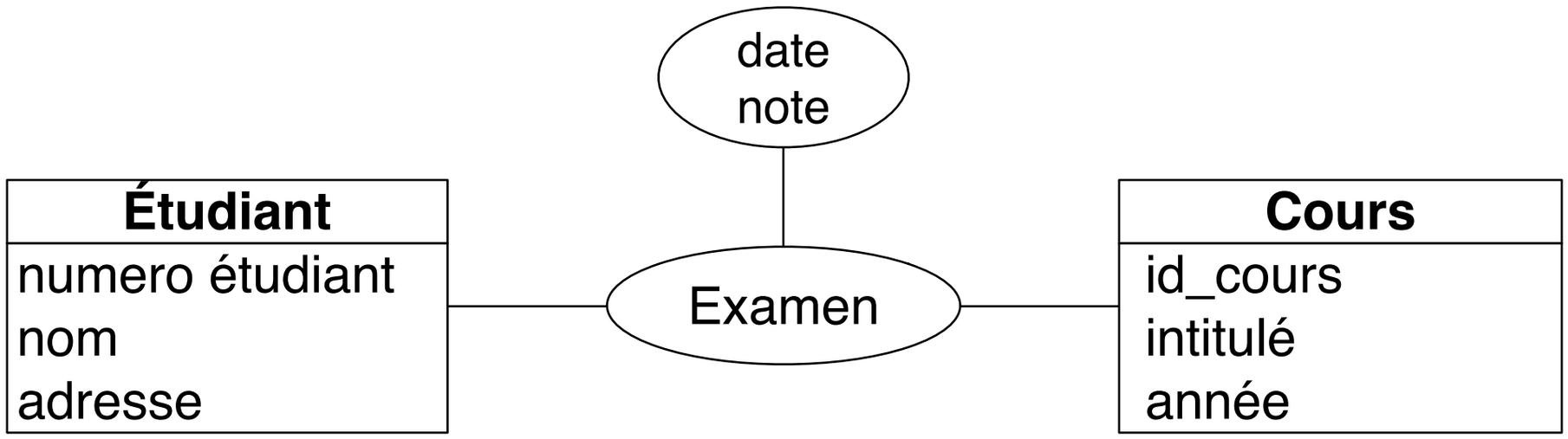


instances(**Étudiant**)

instances(**Cours**)

Attributs d'associations

- Plusieurs attributs sont possibles :

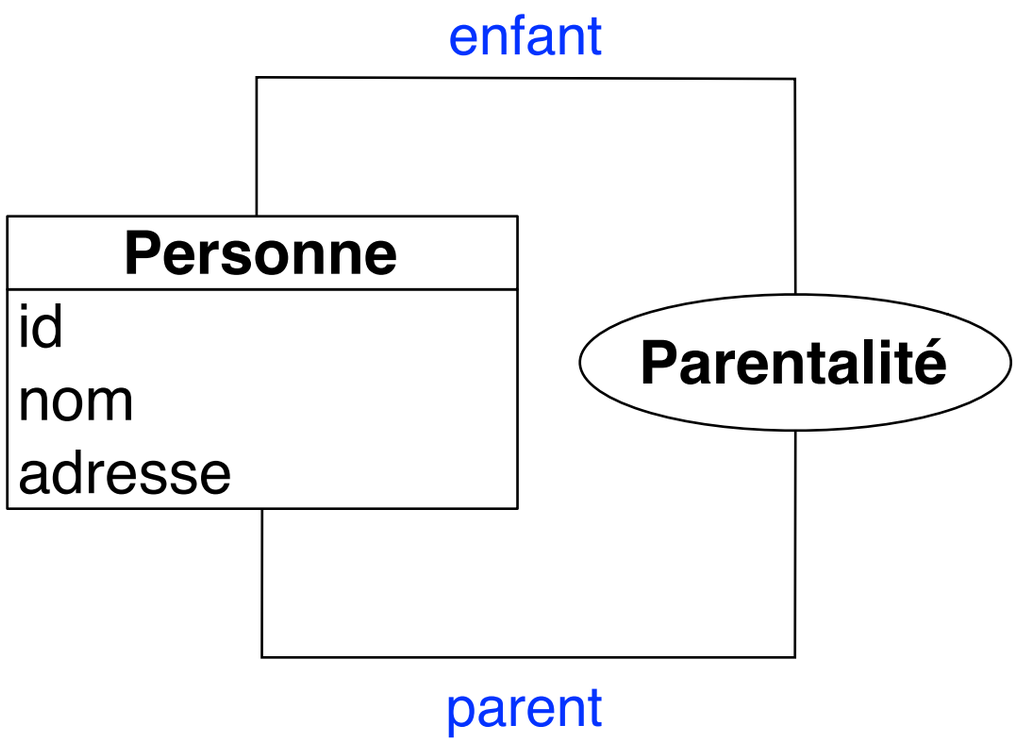


chacun représente une fonction différente

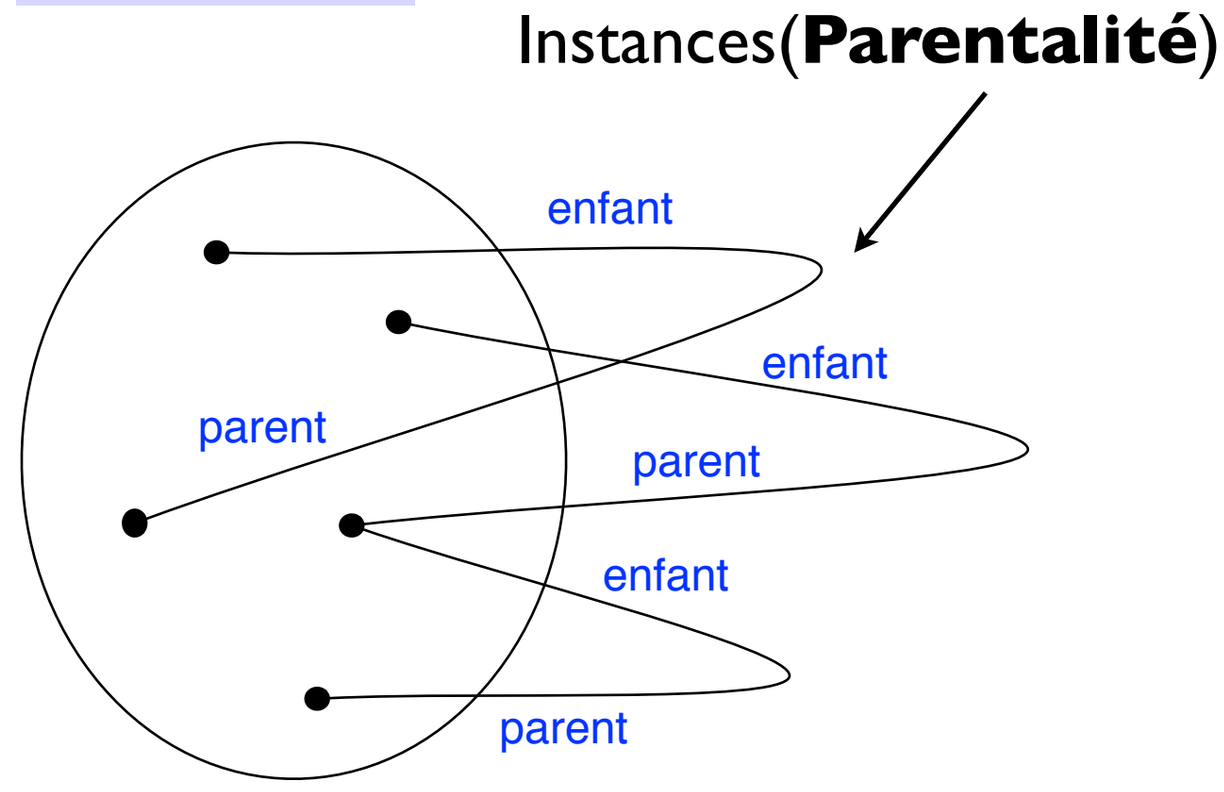
Associations réflexives

- Une association peut relier une entité à elle même, dans ce cas une étiquette (appelé rôle) doit distinguer les deux cotés de l'association :

Syntaxe :



Sémantique :



Instances(**Personne**)

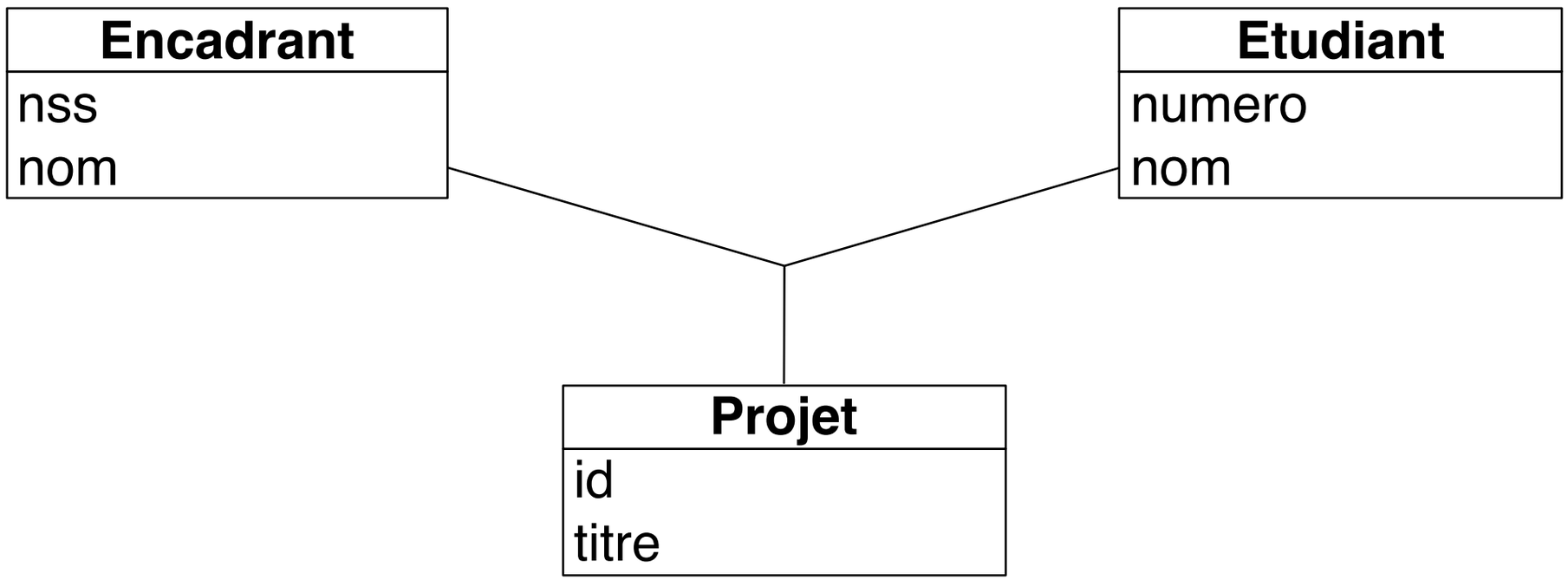
- Sémantique : Instances(Parentalité) est un ensemble de couples de personnes étiquetées par les rôles. Dans chaque couple : une personne a le rôle "parent" et l'autre le rôle "enfant"

Associations n-aires

- Une association peut relier plus que deux entités. Ex. : l'association Encadrement

  (Les étudiants sont encadrés par des encadrants sur des projets)

Syntaxe :

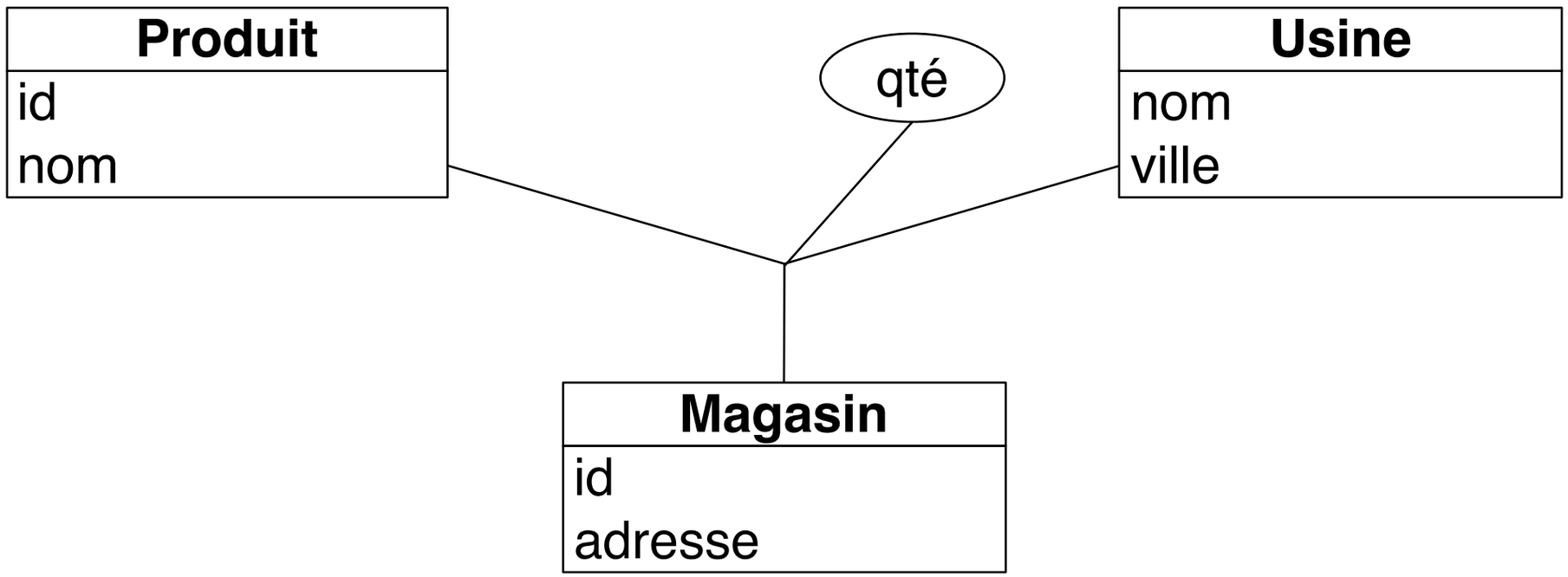


Associations n-aires

- Une association peut relier plus de deux entités. Ex. : l'association Disponibilité

  (Dans chaque magasin il y a une certaine quantité disponible de produits fabriqués dans des usines)

Syntaxe :

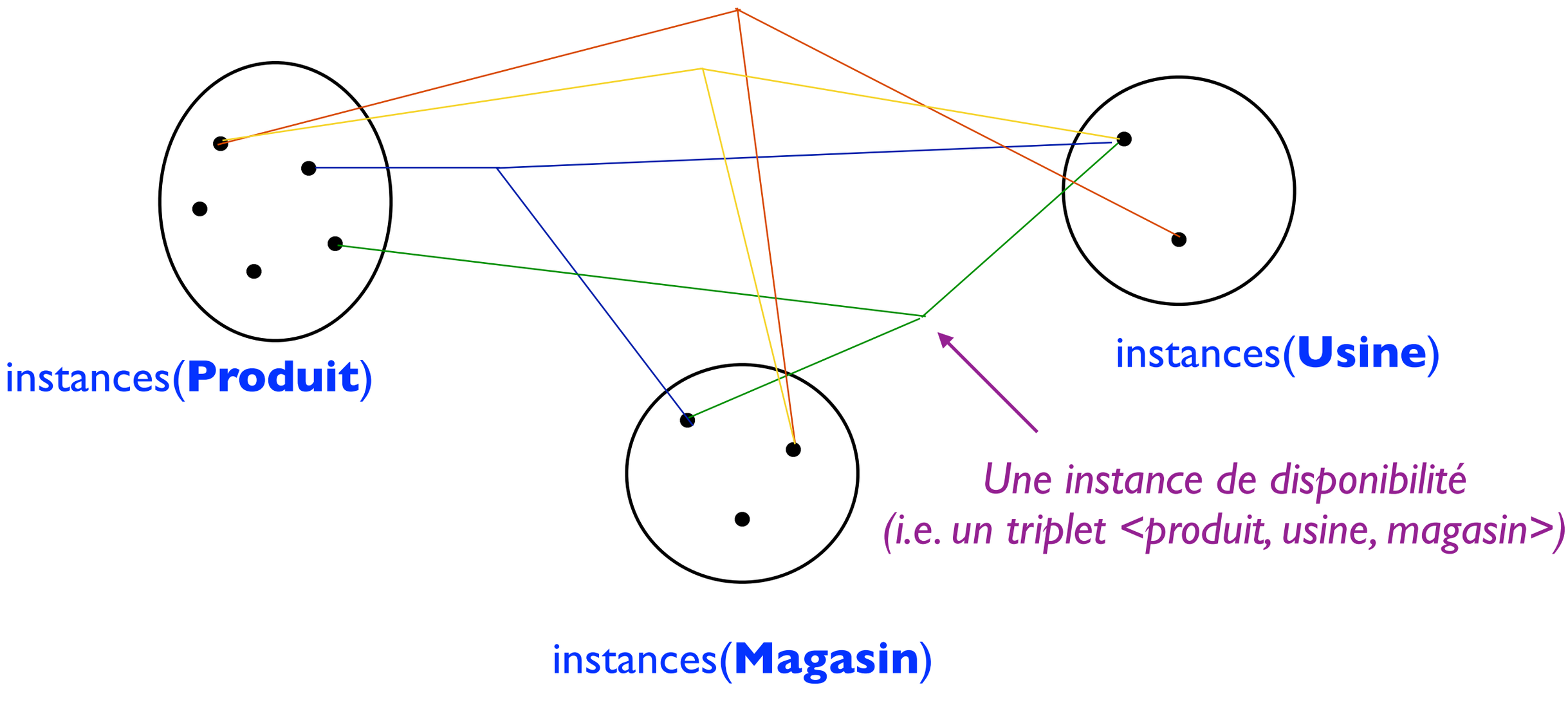


Associations n-aires

- Une association peut relier plus de deux entités. Ex. : l'association Disponibilité

Sémantique :

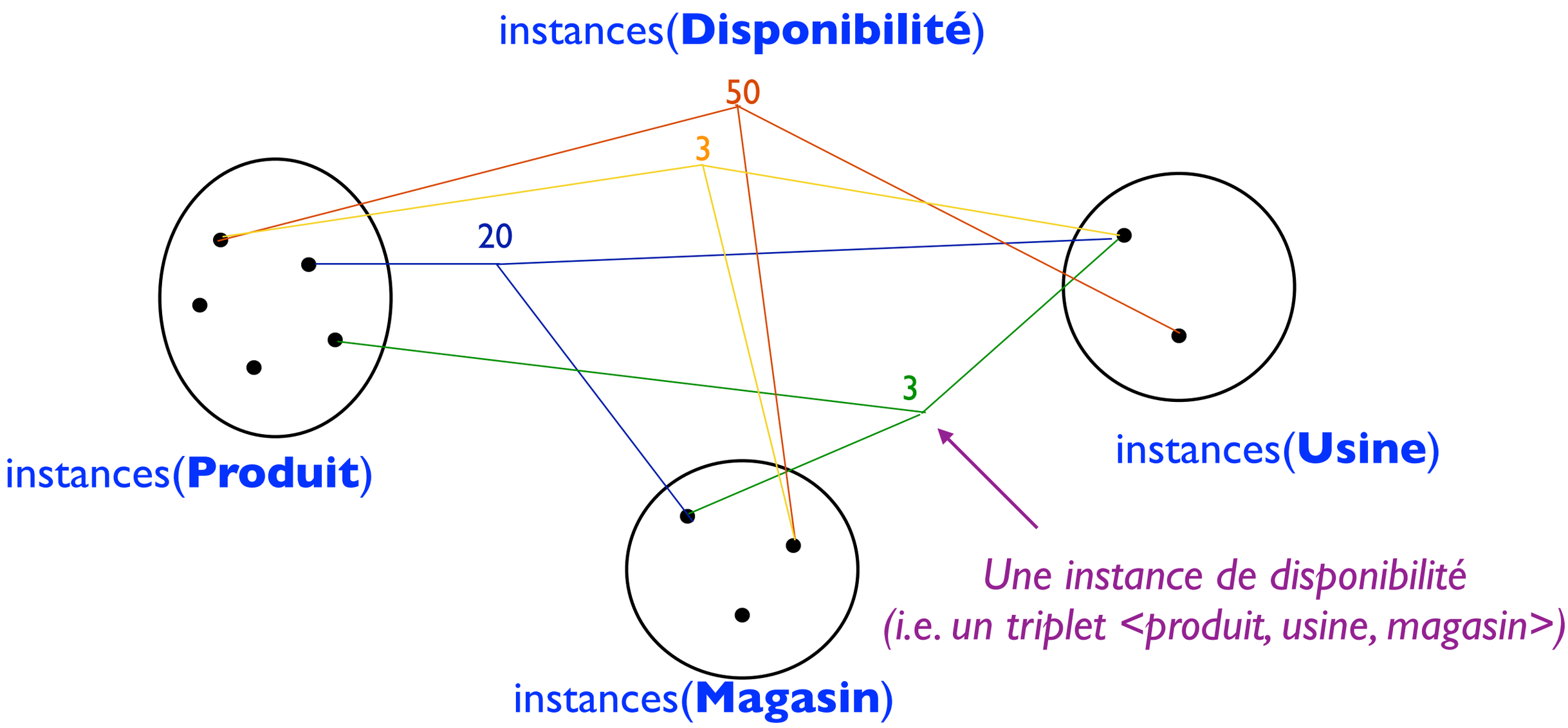
instances(Disponibilité) ⊆
instances(Produit) × instances(Usine) × instances(Magasin)



Associations n-aires

- Une association peut relier plus que deux entités. Ex. : l'association Disponibilité

Sémantique : instances(Disponibilité) ⊆
instances(Produit) × instances(Usine) × instances(Magasin)



qté : instances(**Disponibilité**) → Entiers

Contraintes dans le modele E/R

- Le modele E/R peut exprimer certaines contraintes sur les données :
  - Contraintes : conditions additionnelles pour que les instances d'un diagramme E/R soient valides

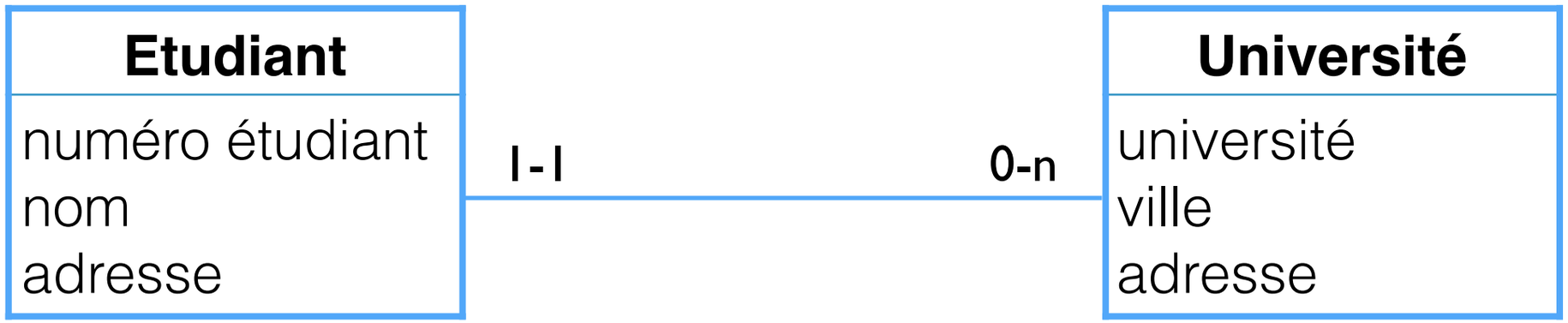
- Deux types de contraintes peuvent être exprimées par le modèle E/R:
  - cardinalité
  - identification

- D'autres contraintes plus complexes doivent être ajoutées au diagramme E/R en tant que contraintes externes

Contraintes de cardinalité

- Expriment le nombre d'instances des entités qui peuvent être associées via une association

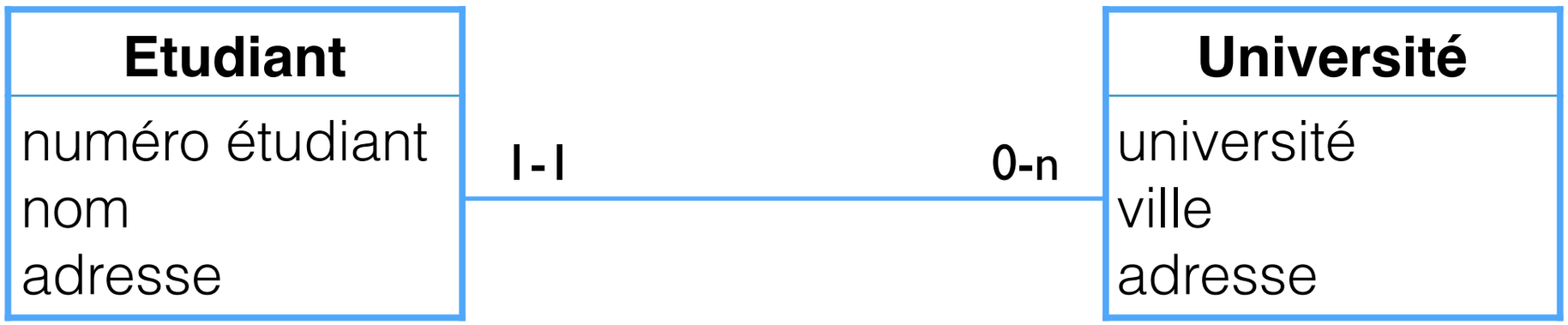
Syntaxe (exemple): Contraintes de cardinalité pour l'association Inscription



Sémantique :

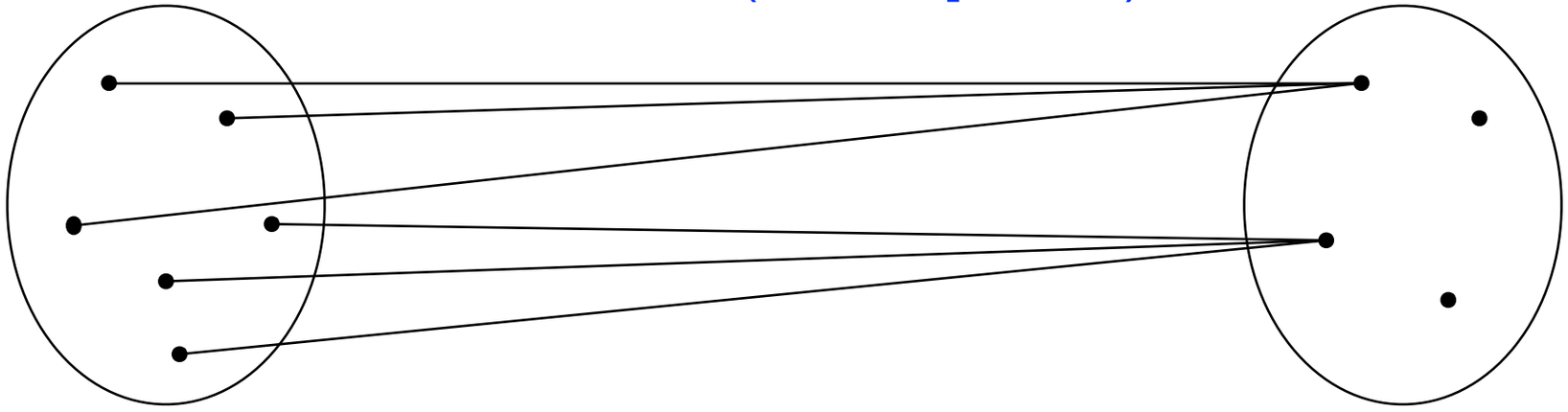
- un même étudiant peut être inscrit à une et une seule université (min : 1, max : 1)
- une université peut avoir entre 0 et plusieurs inscrits (min : 0, n = pas de max)

Contraintes de cardinalité



Sémantique (exemple):

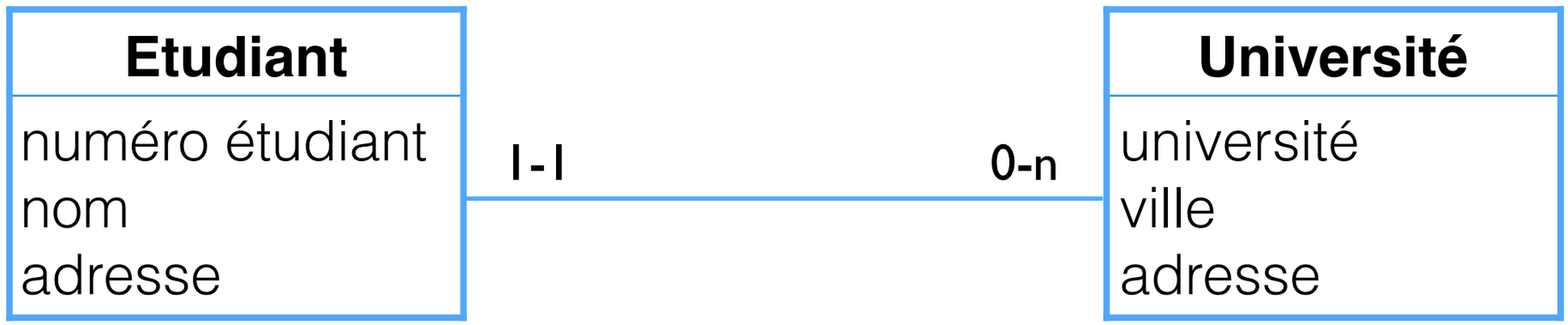
instances(**Inscription**)



instances(**Étudiant**)

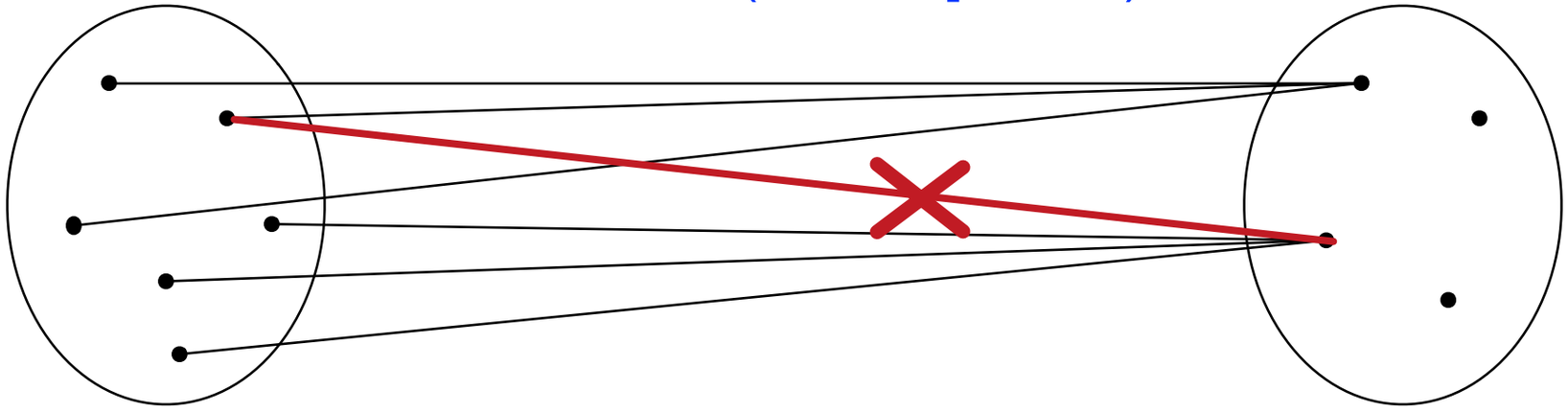
instances(**Université**)

Contraintes de cardinalité



Sémantique (exemple):

instances(**Inscription**)



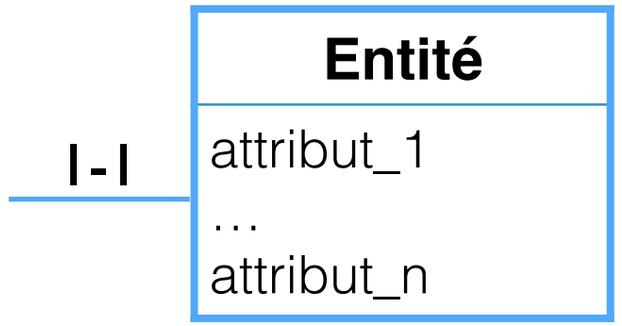
instances(**Étudiant**)

instances(**Université**)

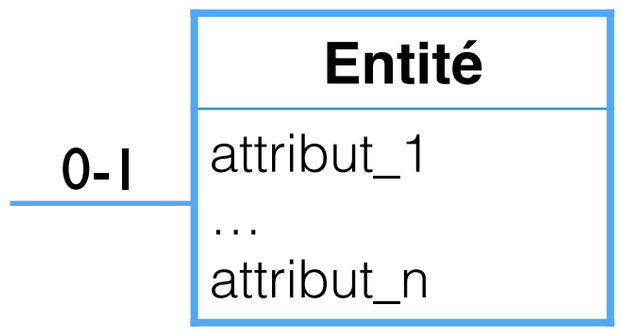
Contraintes de cardinalité

- Cardinalités possibles :

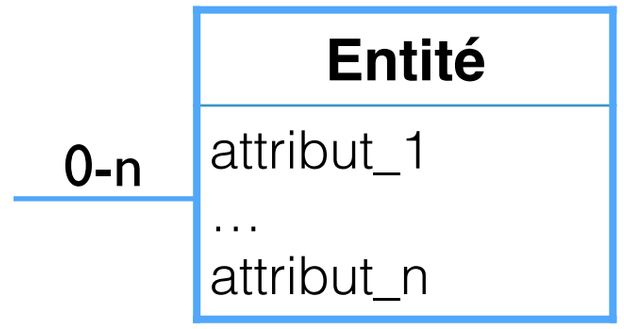
min: 1, max : 1



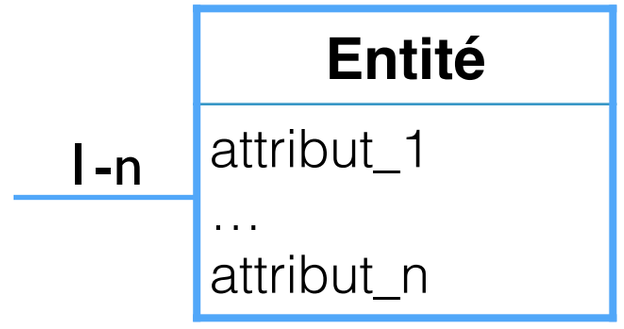
min: 0, max : 1



min: 0, max n

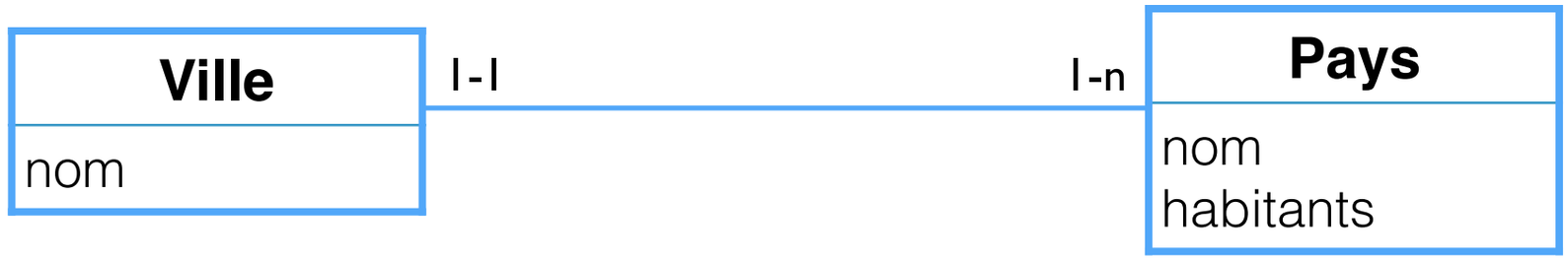


min: 1, max n



Contraintes de cardinalité : exemples

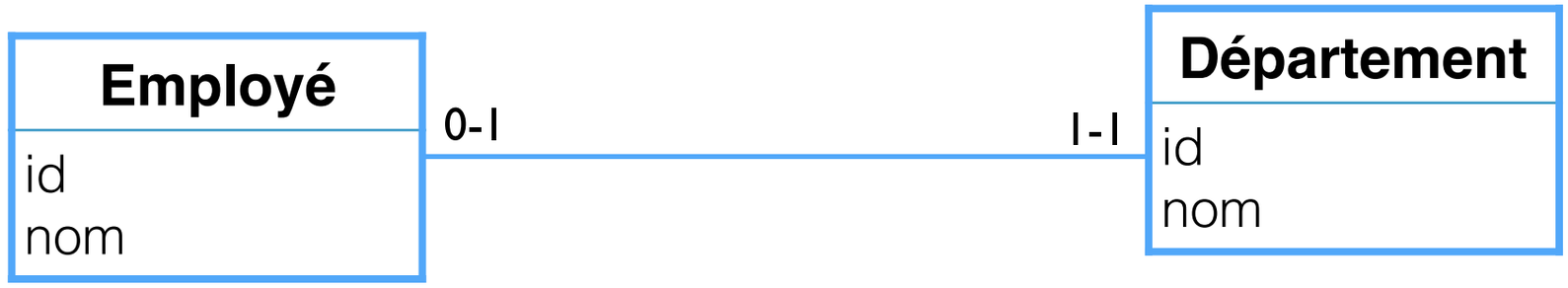
- Localisation :



- Emploi :

- Inscription :

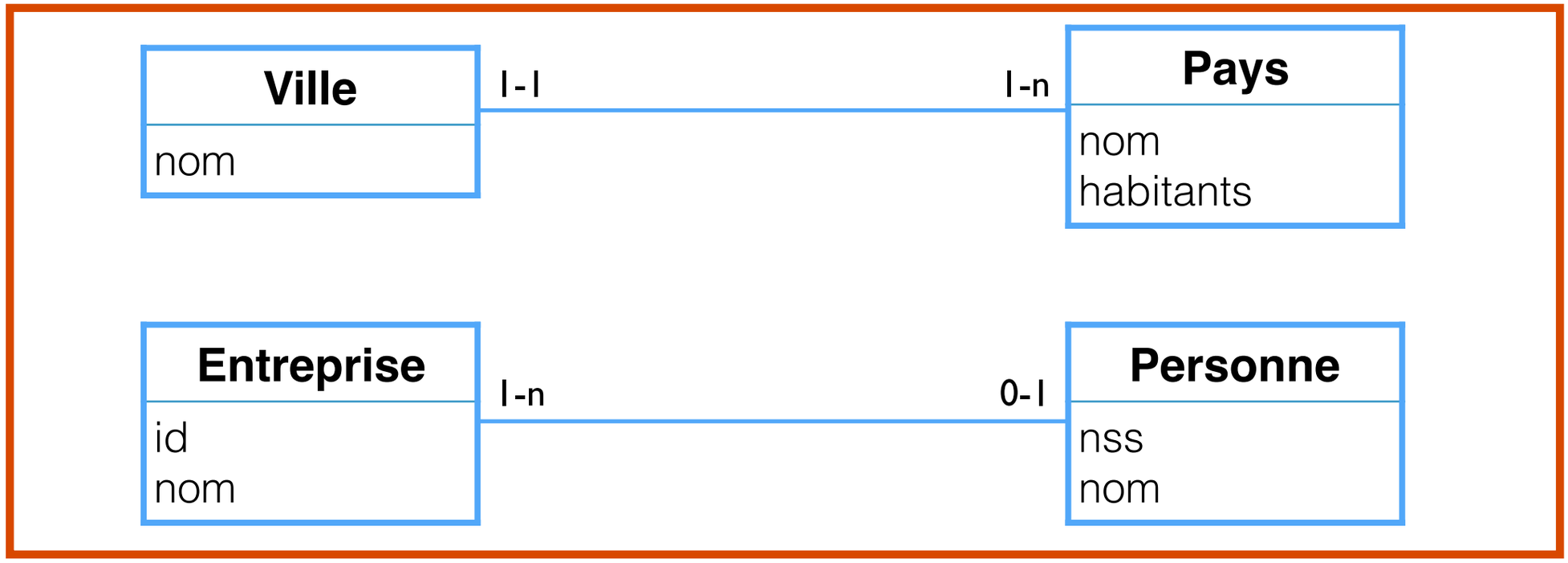
- Directeur :



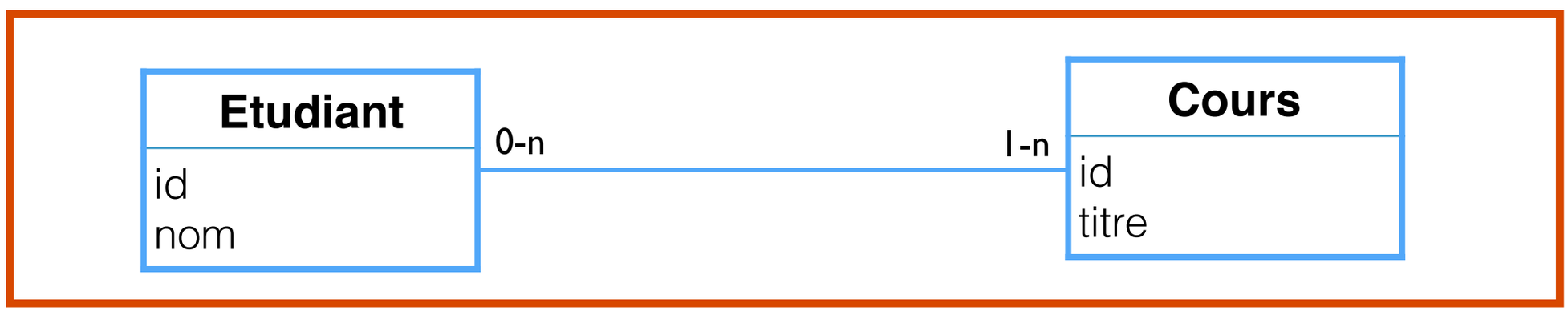
Multiplicité des associations

- En faisant référence uniquement au **cardinalités maximales** de chaque coté, on distingue entre associations :

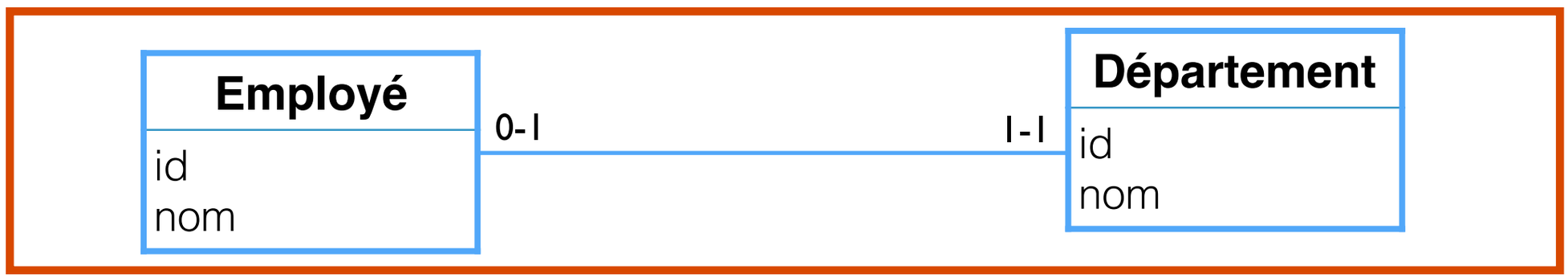
un à plusieurs (1-n)



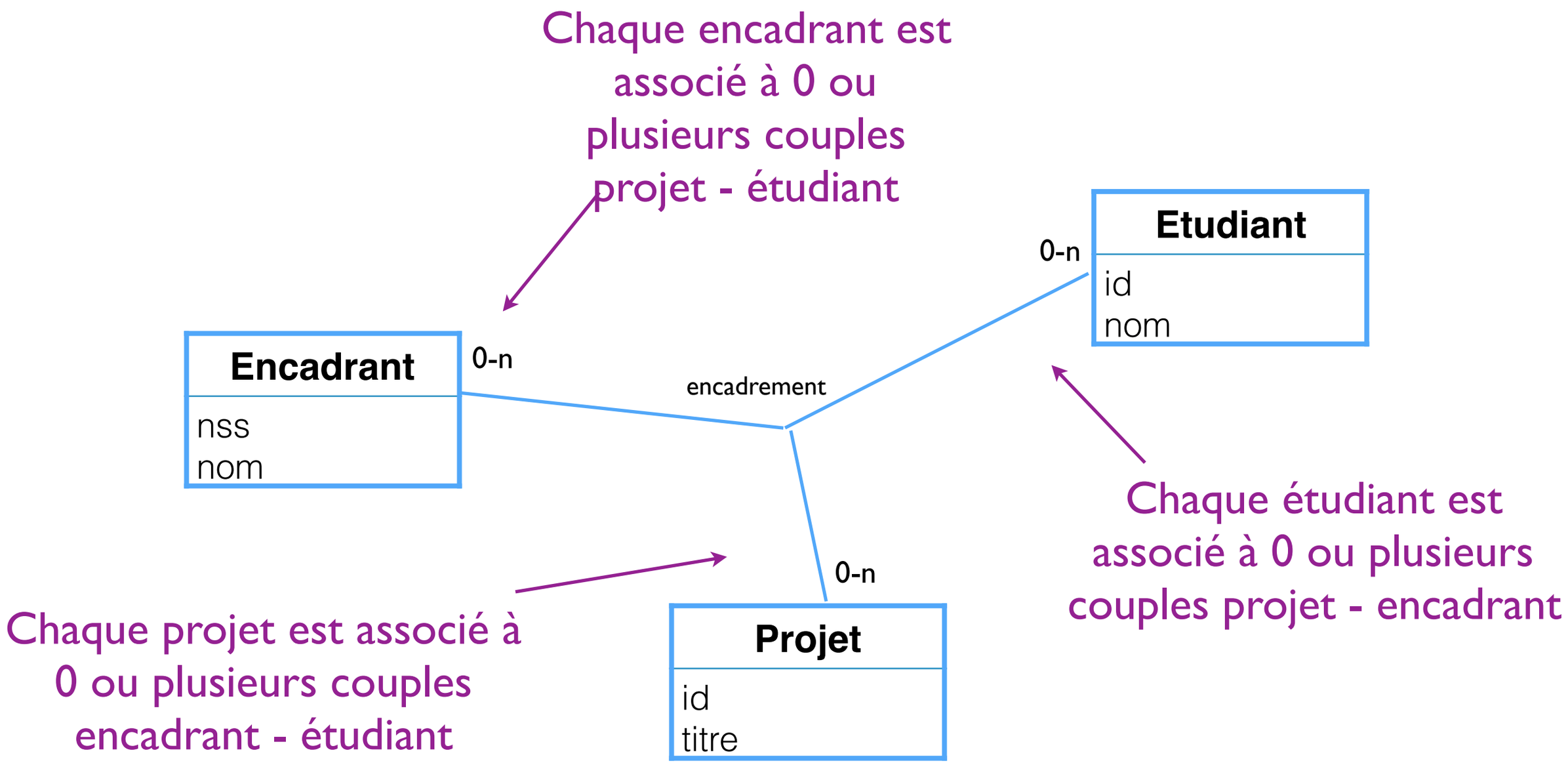
plusieurs à plusieurs (n-n)



un à un (1-1)

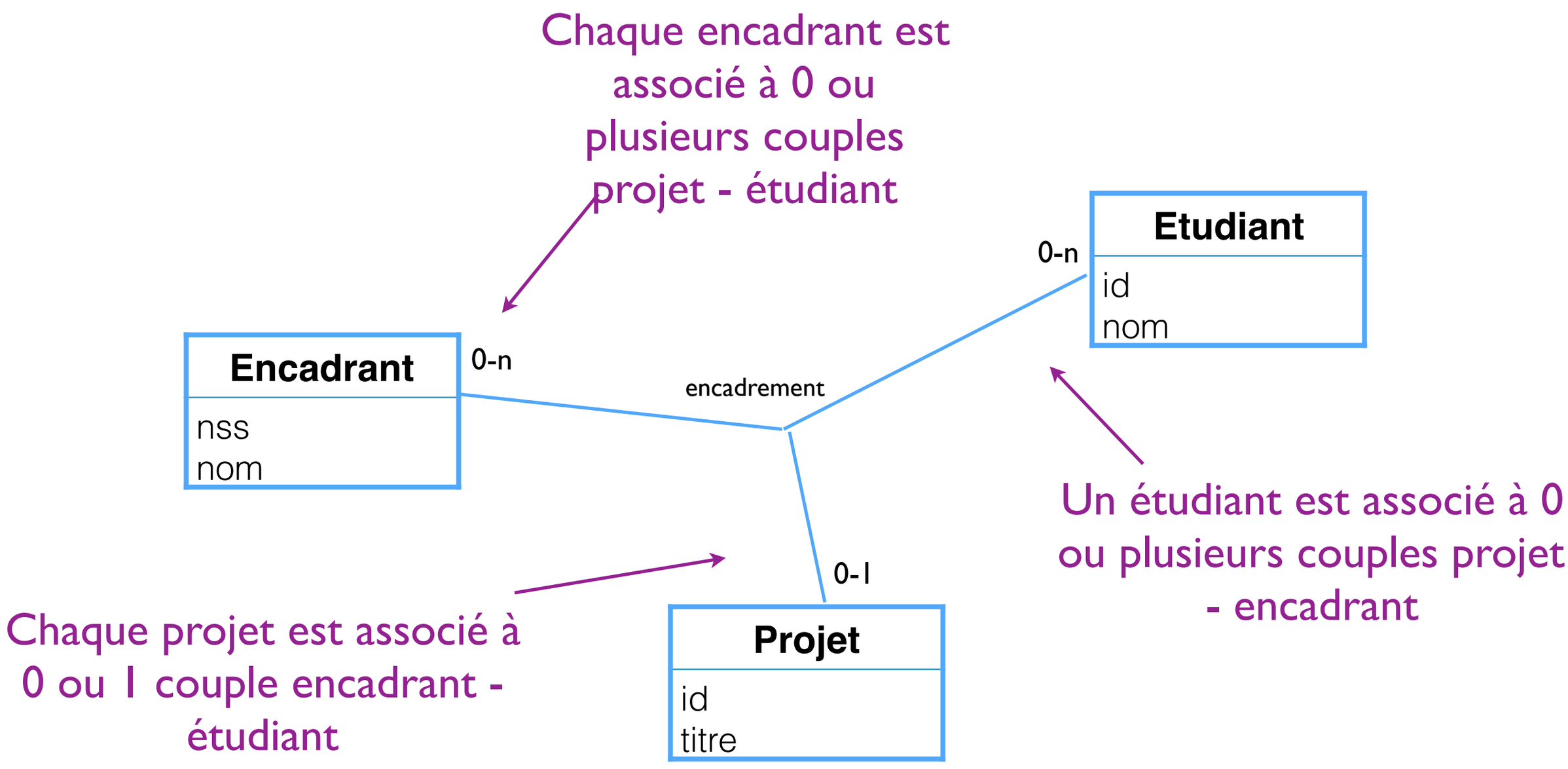


Contraintes de cardinalité dans les associations n-aires



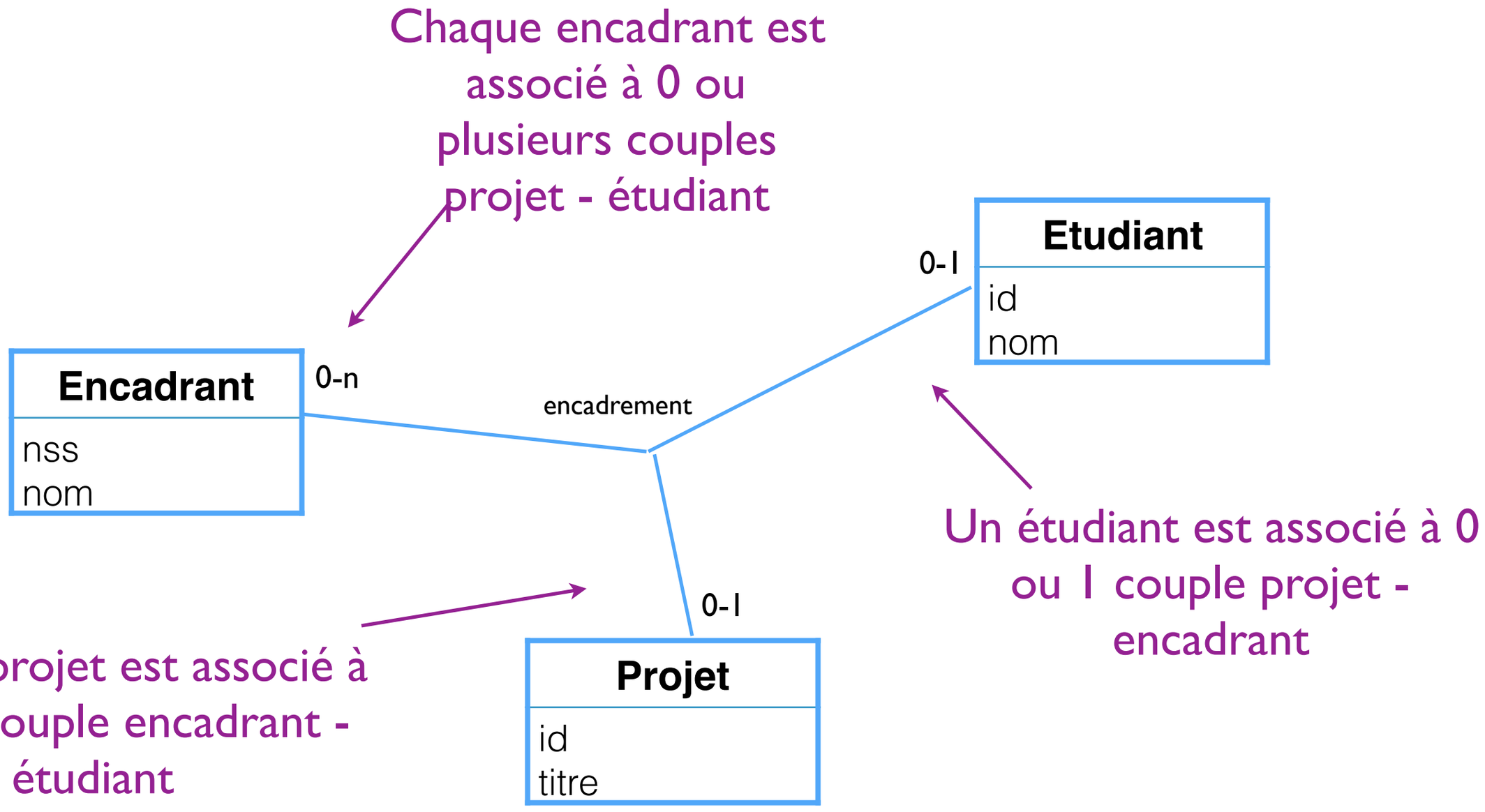
- Un encadrant peut encadrer plusieurs projets d'étudiants différents
- Un étudiant peut travailler sur des projets différents avec des encadrants différents
- Un projet peut être mené par des étudiants différents sous la direction de différents encadrants

Contraintes de cardinalité dans les associations n-aires



- Un encadrant peut encadrer plusieurs projets d'étudiants différents
- Un étudiant peut travailler sur des projets différents avec des encadrants différents
- Un projet est mené par au plus un étudiant et sous la direction d'un seul encadrant

Contraintes de cardinalité dans les associations n-aires

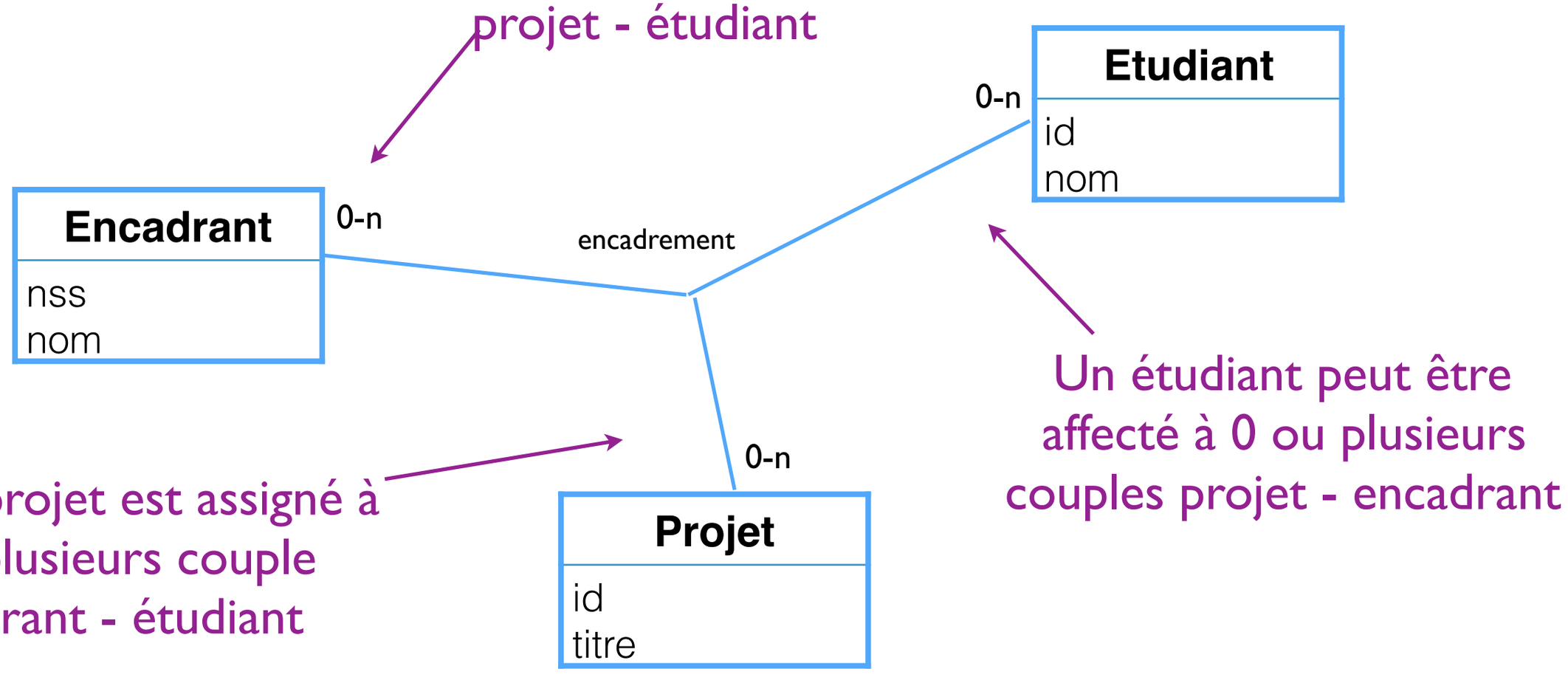


- Un encadrant peut encadrer plusieurs projets d'étudiants différents
- Un étudiant peut travailler sur un seul projet et avec un seul encadrant
- Un projet est mené par au plus un étudiant sous la direction d'un seul encadrant

Contraintes de cardinalité dans les associations n-aires

- Et si on voulait représenter aussi le fait qu'un étudiant ne peut pas travailler sur plus d'un projet avec un seul et même encadrant ?
  - » les cardinalités ne suffisent pas

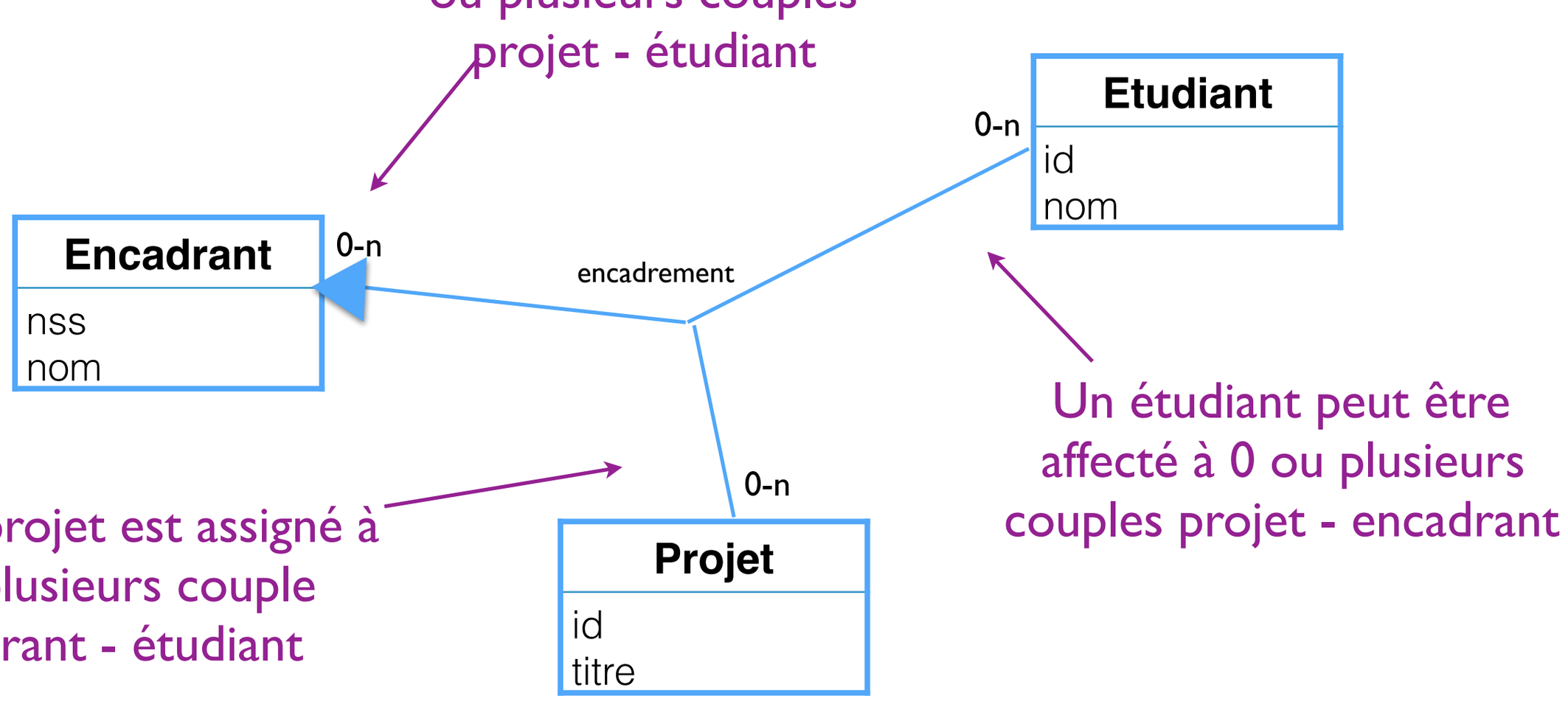
Chaque encadrant
peut être assigné à 0
ou plusieurs couples
projet - étudiant



Contraintes de cardinalité dans les associations n-aires

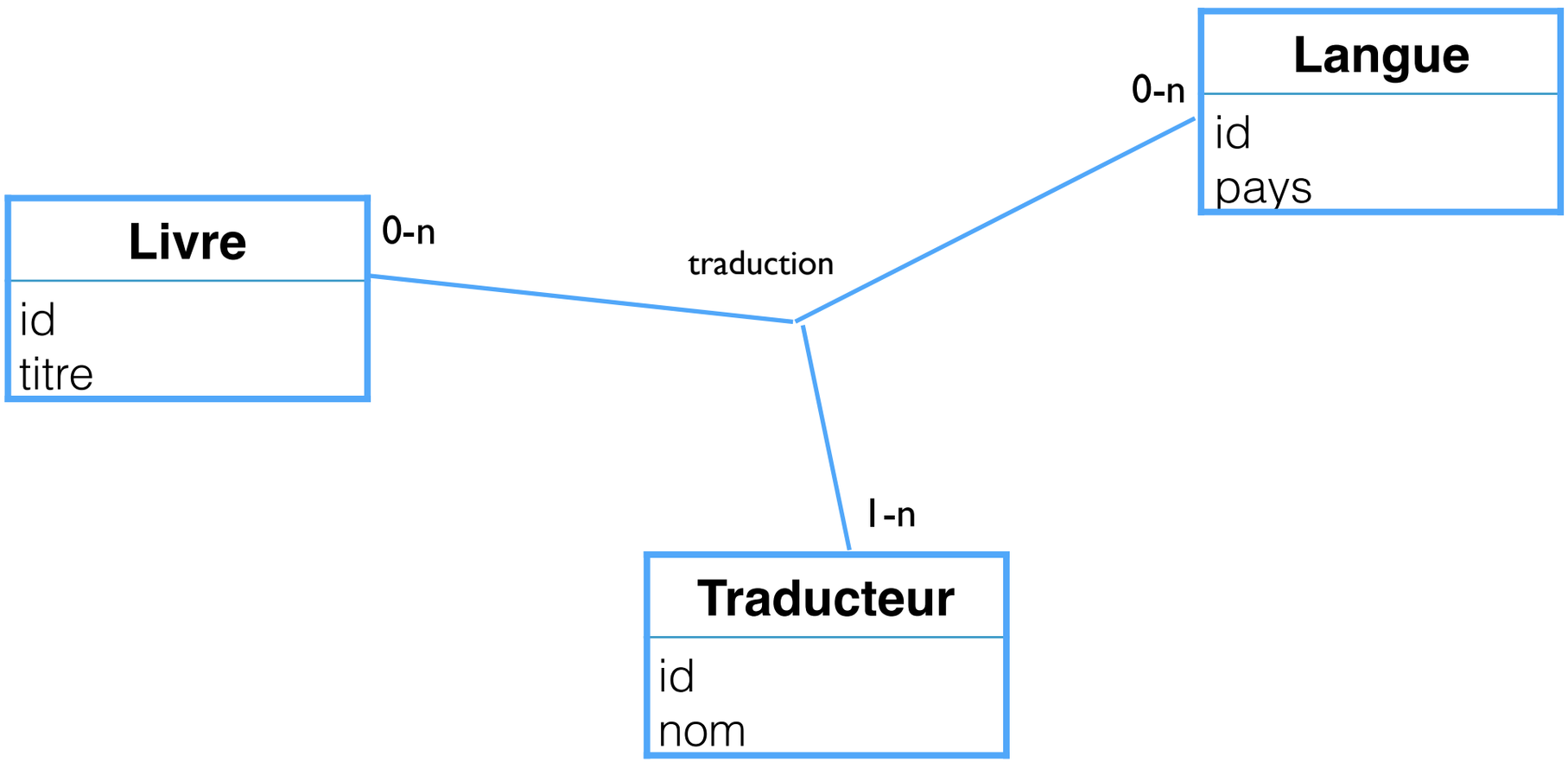
- Et si on voulait représenter aussi le fait qu'un étudiant ne peut pas travailler sur plus d'un projet avec un seul et même encadrant ?
  - » rajouter une flèche de étudiant et projet vers encadrant : leur valeur *détermine* celle de encadrant (contrainte d'intégrité « fonctionnelle »)

Chaque encadrant
peut être assigné à 0
ou plusieurs couples
projet - étudiant



Contraintes de cardinalité dans les associations n-aires

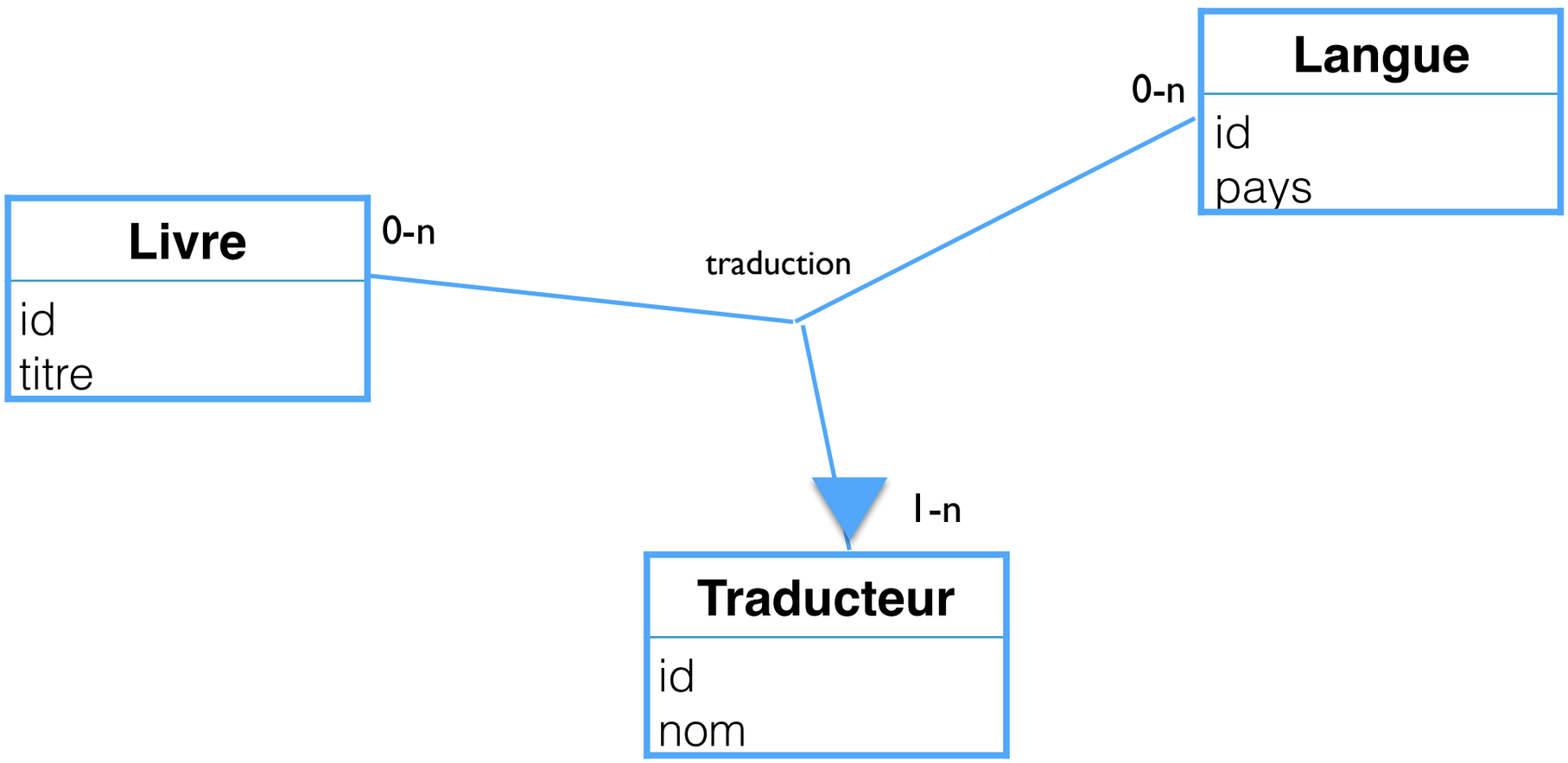
D'autres exemples : l'association Traduction



- Un livre est associé à 0 ou plusieurs couples langue traducteur
- Une langue est associée à 0 ou plusieurs couples livre traducteur
- Un traducteur est associé à 1 ou plusieurs couples livres langue

Contraintes de cardinalité dans les associations n-aires

D'autres exemples : l'association Traduction



- Un livre est associé à 0 ou plusieurs couples langue traducteur
- Une langue est associée à 0 ou plusieurs couples livre traducteur
- Un traducteur est associé à 1 ou plusieurs couples livres langue
- Un couple livre langue correspond au plus à 1 traducteur

Rappel : Contraintes dans le modèle E/R

- Le modèle E/R peut exprimer certaines contraintes sur les données :
  - Contraintes : conditions additionnelles pour que les instances d'un diagramme E/R soient valides

- Deux types de contraintes :
  - cardinalité
  - identification (clefs)

Clefs

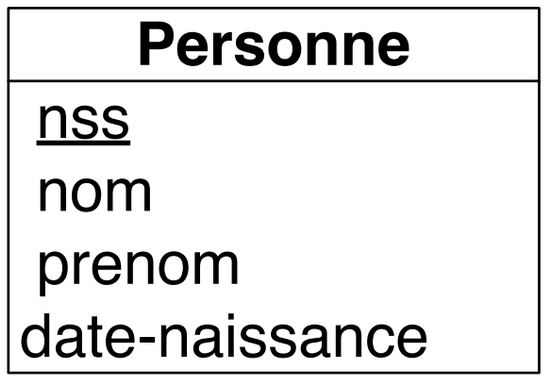
- Une **superclef** d'une entité est un ensemble de un ou plusieurs attributs tel que: il n'existe pas deux instances de l'entité avec la même valeur de tous ces attributs
  - Exemples :
    - nss est une superclef pour l'entité Personne
    - (titre, réalisateur) est une superclef pour l'entité Film
    - (nss, nom) est une superclef pour l'entité Personne
    - (ville, rue, numéro) est une superclef pour l'entité Bâtiment

- Une **clef** (ou **clef candidate**) d'une entité est une superclef minimale
  - nss est une clef pour l'entité Personne
  - (titre, réalisateur) est une clef pour l'entité Film
  - (ville, rue, numéro) est une clef pour l'entité Bâtiment
- Plusieurs clefs candidates peuvent exister pour une entité, mais pour chaque entité une seule clef est choisie comme **clef primaire**.

Clefs primaires

- Dans un diagramme E/R une clef primaire est spécifiée pour chaque entité

Syntaxe :



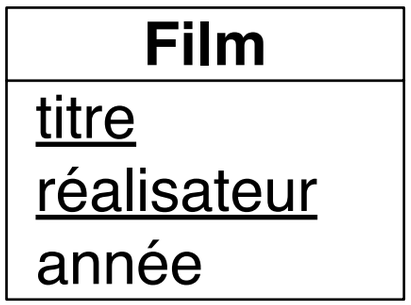
Sémantique :

Il n'existe pas deux personnes différentes p1 et p2 dans Instances(Personne) telles que: nss(p1) = nss(p2)

Clefs primaires

- Une clef avec deux attributs :

Syntaxe :



Sémantique :

Il n'existe pas deux films différents f1 et f2 dans Instances(Film) tels que : titre(f1) = titre(f2) **et** réalisateur(f1) = réalisateur(f2)

Clefs primaires

- Une clef avec trois attributs

Syntaxe :

Bâtiment
numero
rue
ville
nombre-étages
année-construction

Sémantique :

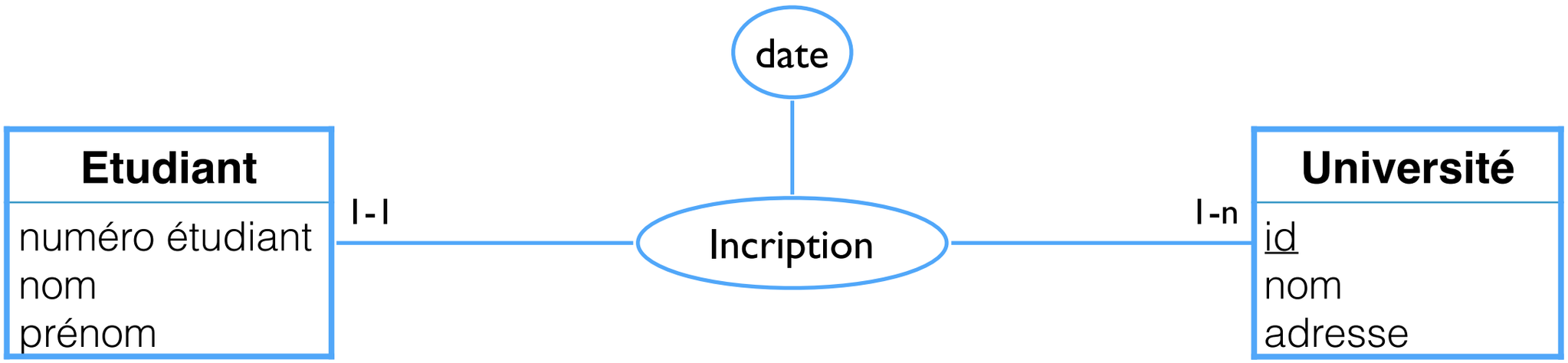
Il n'existe pas deux bâtiments différents b1 et b2 dans Instances(Bâtiment) tels que :
numero(b1) = numero(b2) **et** rue(b1) = rue(b2) **et** ville(b1) = ville(b2)

- **Remarque** : il peut exister deux bâtiments avec le même numéro et le même nom de rue (dans des villes différentes), mais pas avec le même numéro, la même rue et la même ville.

Entités faibles

- Pour certaines entités, aucun ensemble d'attributs ne forme une clef

  Exemple :

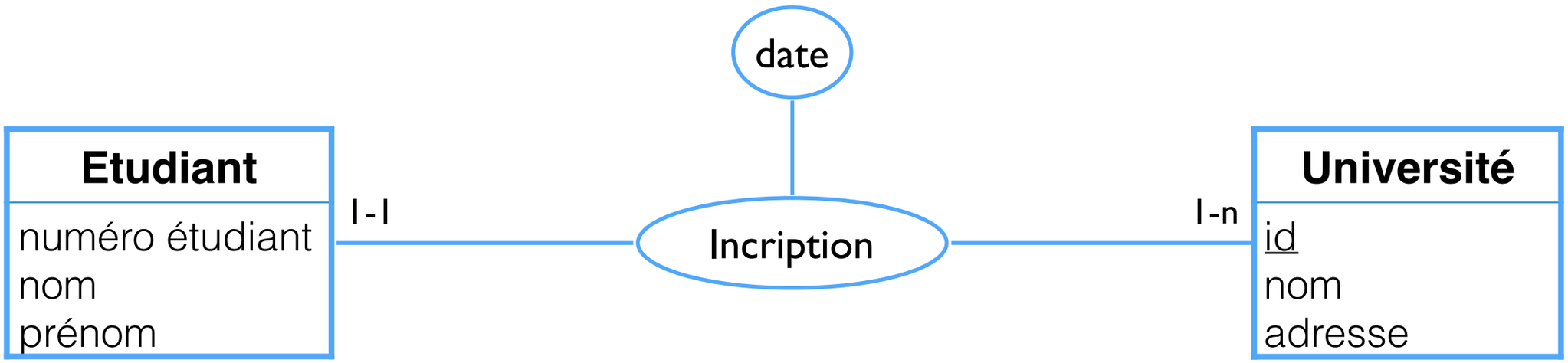


Quel ensemble d'attributs est une clef pour l'entité Étudiant ?

Entités faibles

- Pour certaines entités, aucun ensemble d'attributs ne forme une clef

  Exemple :



- **Un étudiant est identifié par son numéro étudiant au sein de son université.**

  En d'autres termes une clef pour Étudiant est donnée par le couple

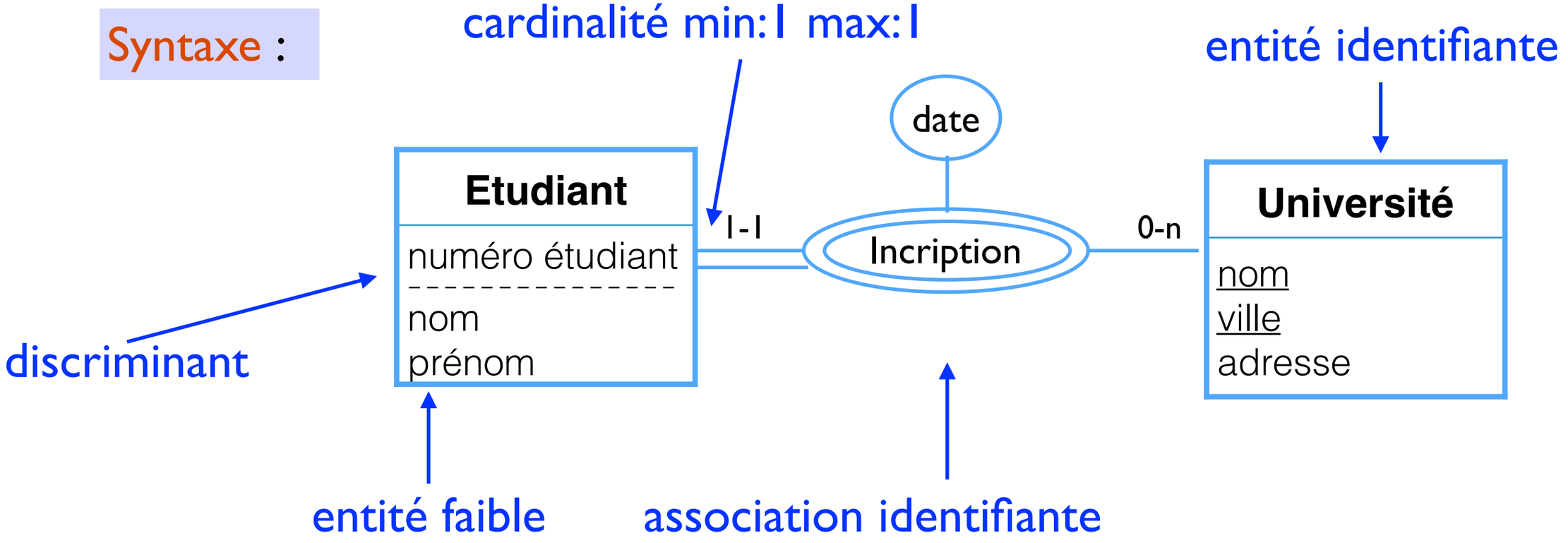
  - numéro étudiant

  - université d'inscription

Entités faibles

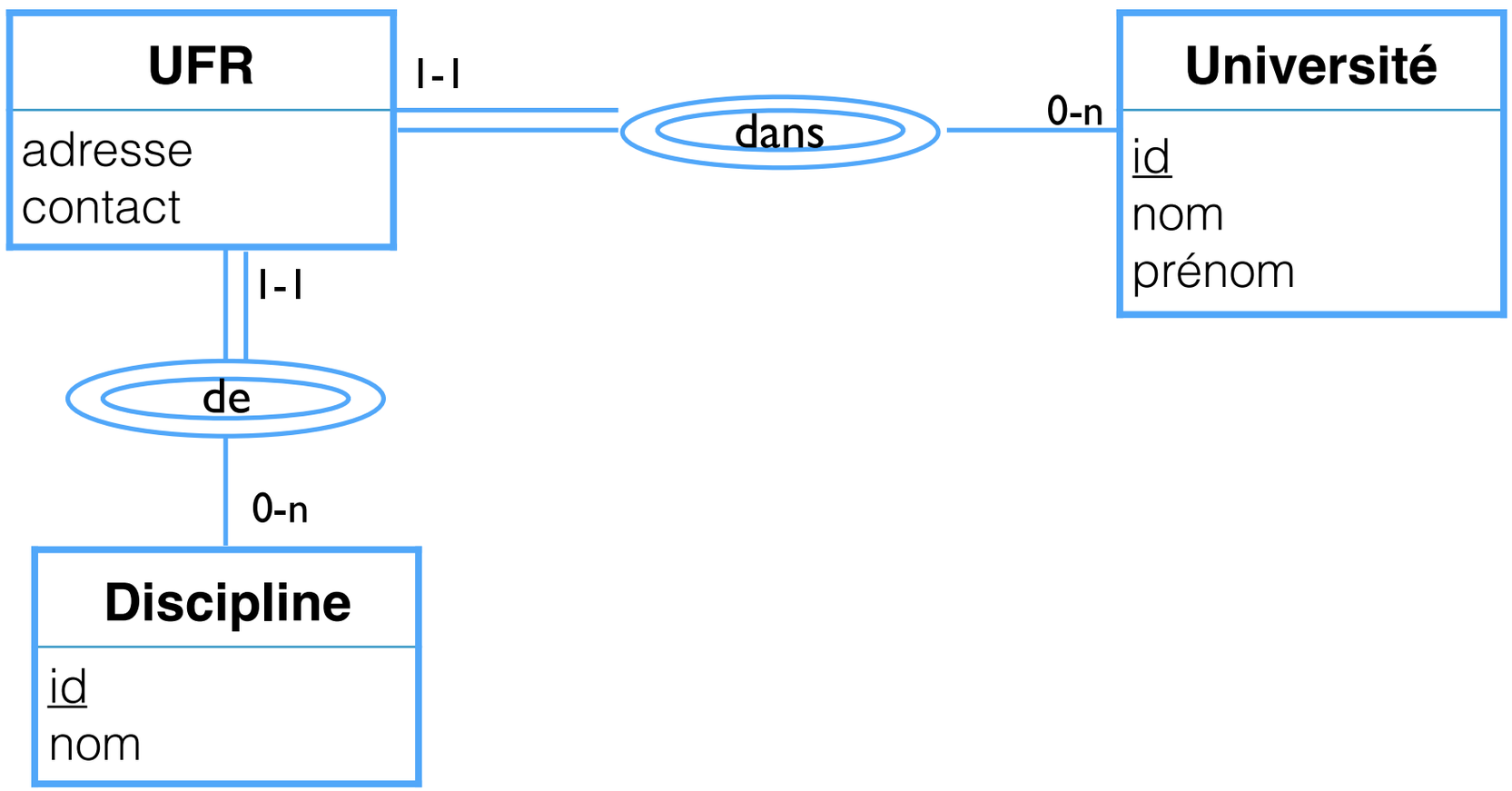
- Une entité faible est une entité identifiée par
  - un ensemble d'attributs internes appelés *discriminant* et
  - un ensembles d'entités appelées entités identifiantes
- Chaque entité identifiante doit être reliée à l'entité faible par une association binaire appelée association identifiante
- l'entité faible doit participer à l'association identifiante avec une cardinalité min:1 max:1

Syntaxe :



Entités faibles

- Un exemple avec plusieurs entités identifiantes :

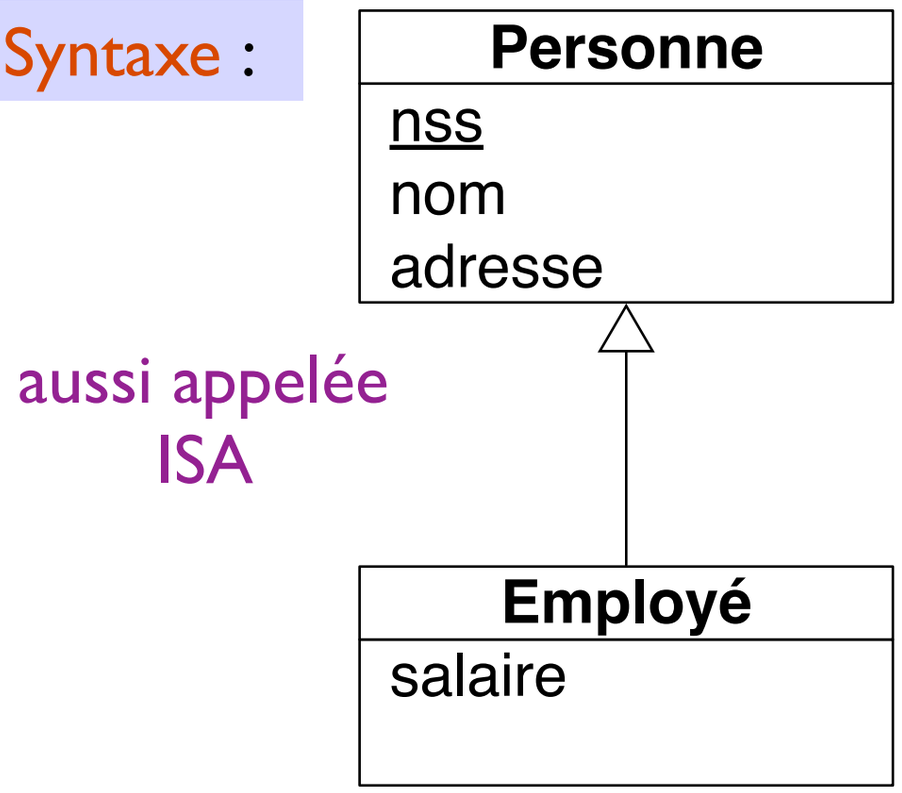


Remarque : dans cet exemple il n'y a pas d'attributs discriminants :
la clef de l'entité faible est totalement externe : Université et Discipline

Spécialisation

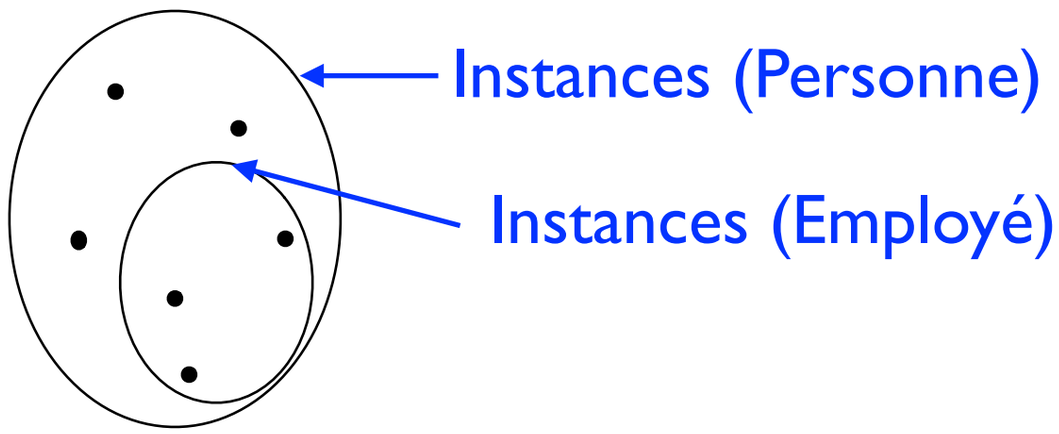
- Dans un diagramme E/R les instances de toutes les entités et toutes les associations sont disjointes entre elles par défaut
- La spécialisation (aussi appelé héritage ou généralisation) est un mécanisme pour spécifier des relations de sous-ensemble entre les instances de différentes entités

Syntaxe :



Sémantique :

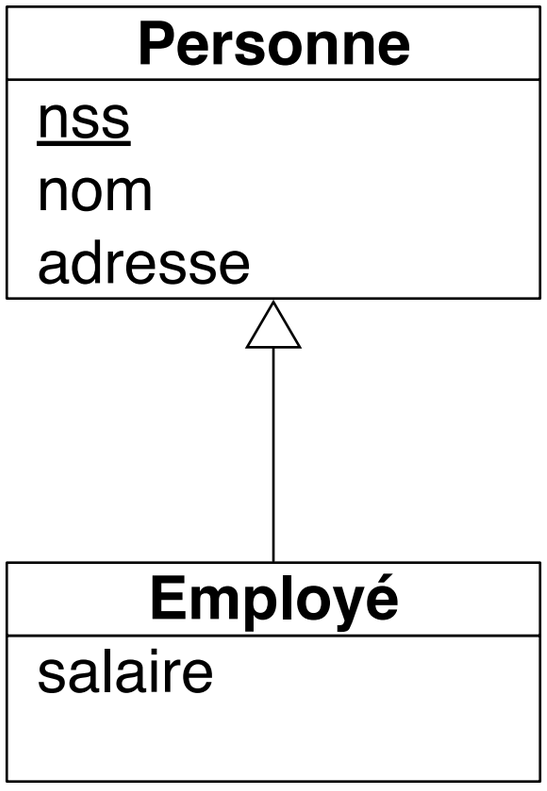
Instances (Employé) ⊆ Instances (Personne)

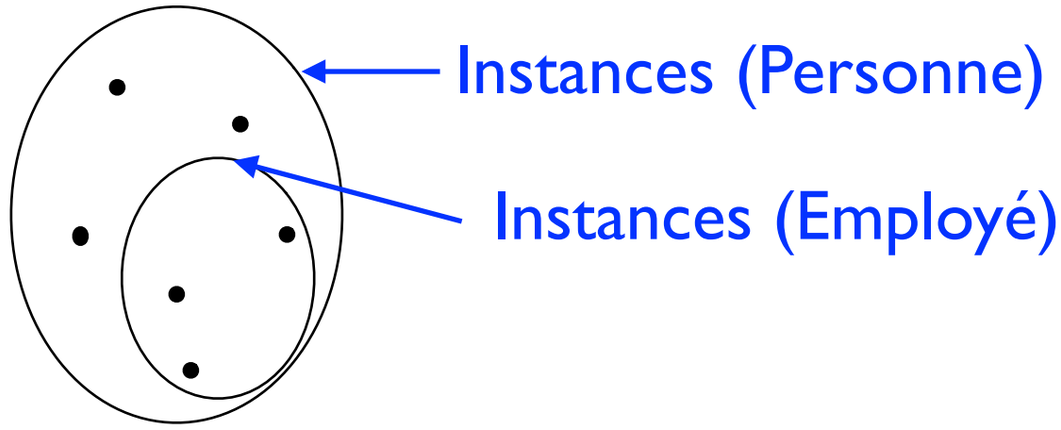


- Utilité : identifier un sous-ensemble des instances d'une entité avec des propriétés (attributs) spécifiques (i.e qui ne concernent pas les autres instances de la même entité)
  - Un employé est une personne qui a en plus un salaire

Spécialisation : héritage des attributs

- La spécialisation introduit un mécanisme de classification des instances des entités, similaire à l'héritage des classes d'objets dans les types de données
  - spécialisation ↔ relation classe / sous-classe
- De façon similaire aux classes, la spécialisation implique l'héritage des attributs :





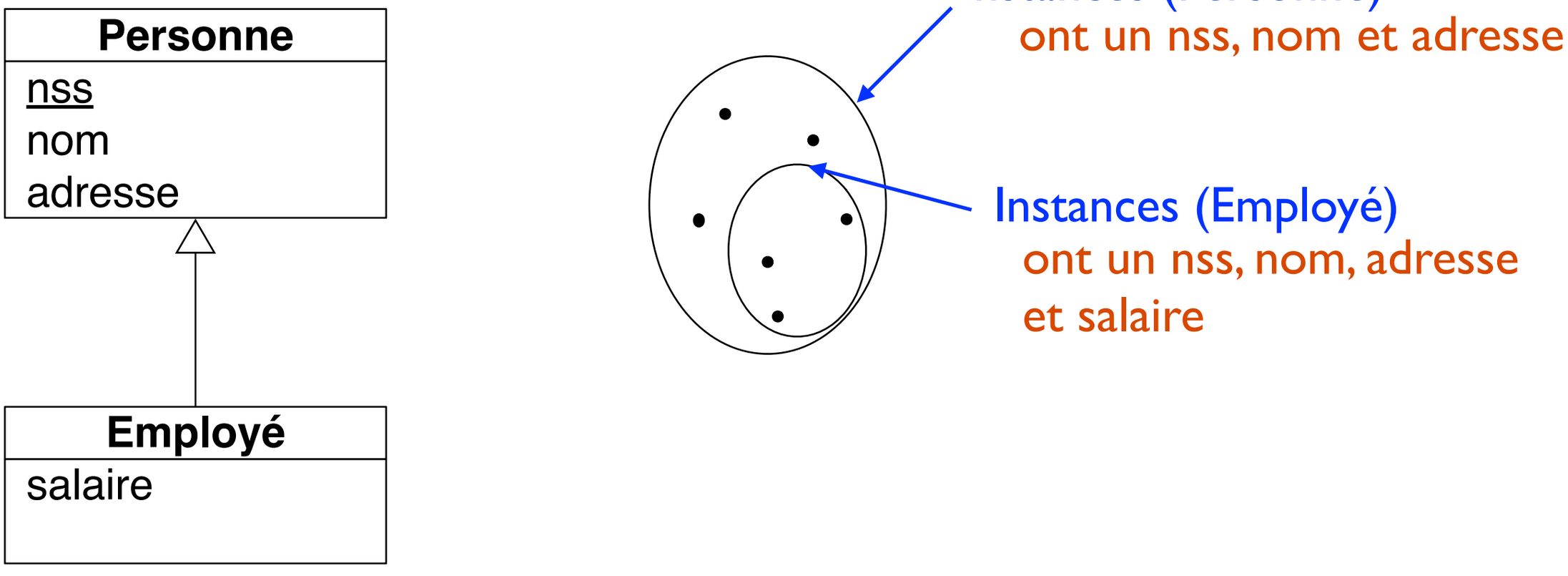
Pour chaque attribut A de Personne

A: Instances(Personne) → D

alors A est défini également sur les employés
alors A est un attribut de l'entité Employé
en revanche salaire : Instances(Employé) → R

Spécialisation : héritage des attributs

- En présence de spécialisation tous les attributs de l'entité mère sont aussi attributs de l'entité fille (ont dit qu'ils sont hérités par l'entité fille)
  - i.e. un employé est une personne donc il a un nss, nom et adresse
- En plus l'entité fille peut avoir d'autres attributs spécifiques

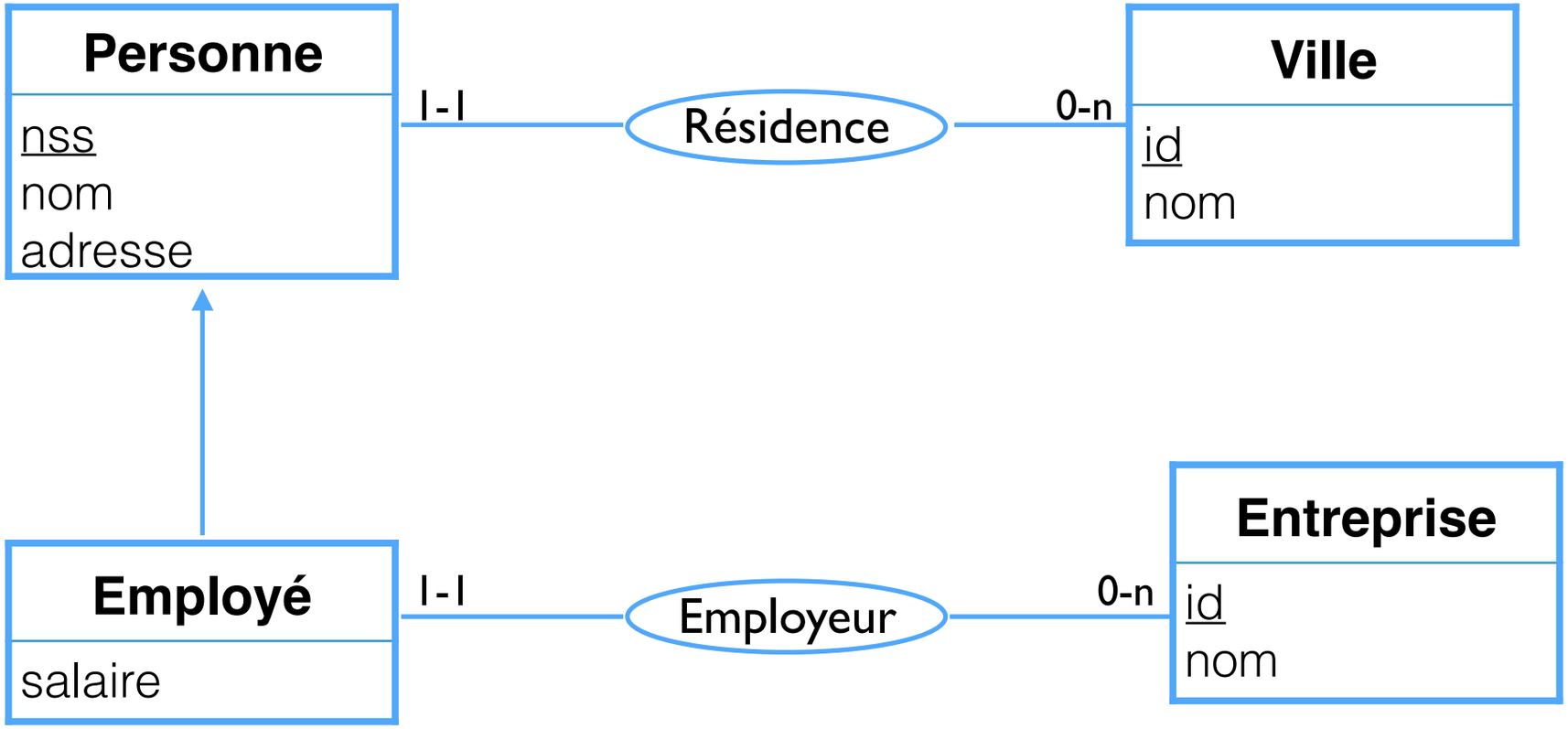


- Conséquence : l'entité fille hérite de la clef primaire de l'entité mère (**on ne spécifie donc pas de clef primaire pour les entités filles**)

Spécialisation : héritage des associations

- Si l'entité mère participe à des associations, ses entités filles y participent automatiquement :

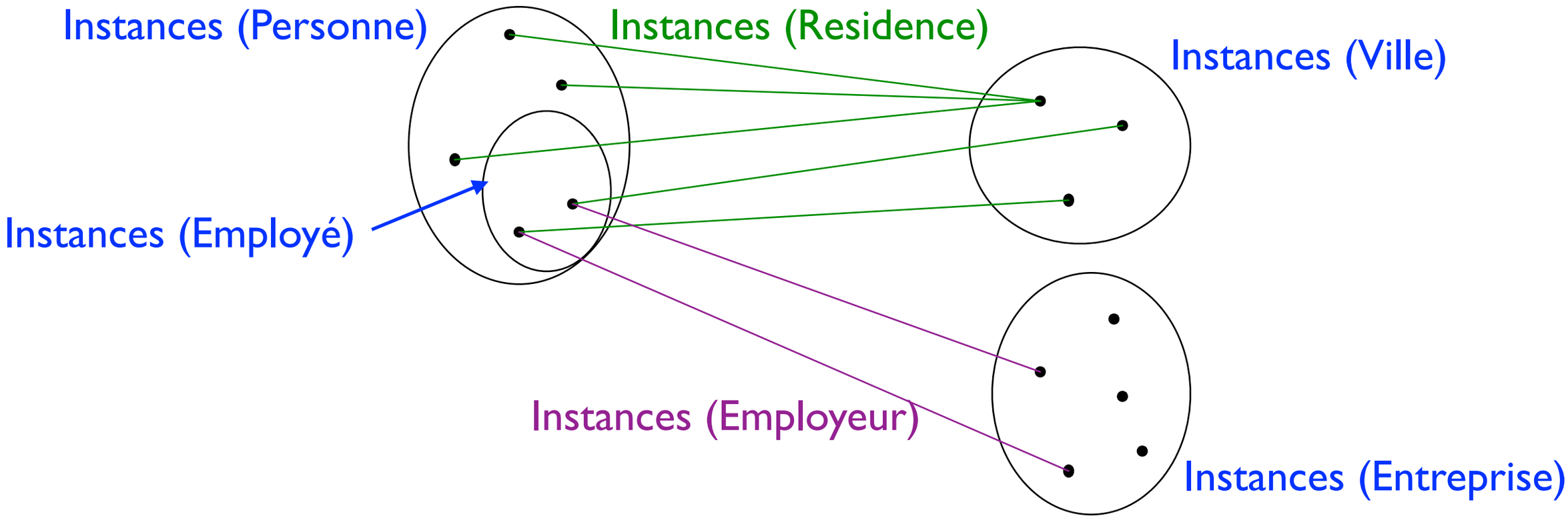
Exemple



- Chaque personne réside dans une ville
- un employé est une personne, donc chaque employé réside dans une ville
- en plus chaque employé travaille dans une entreprise

Spécialisation : héritage des associations

- Du point de vue des instances :

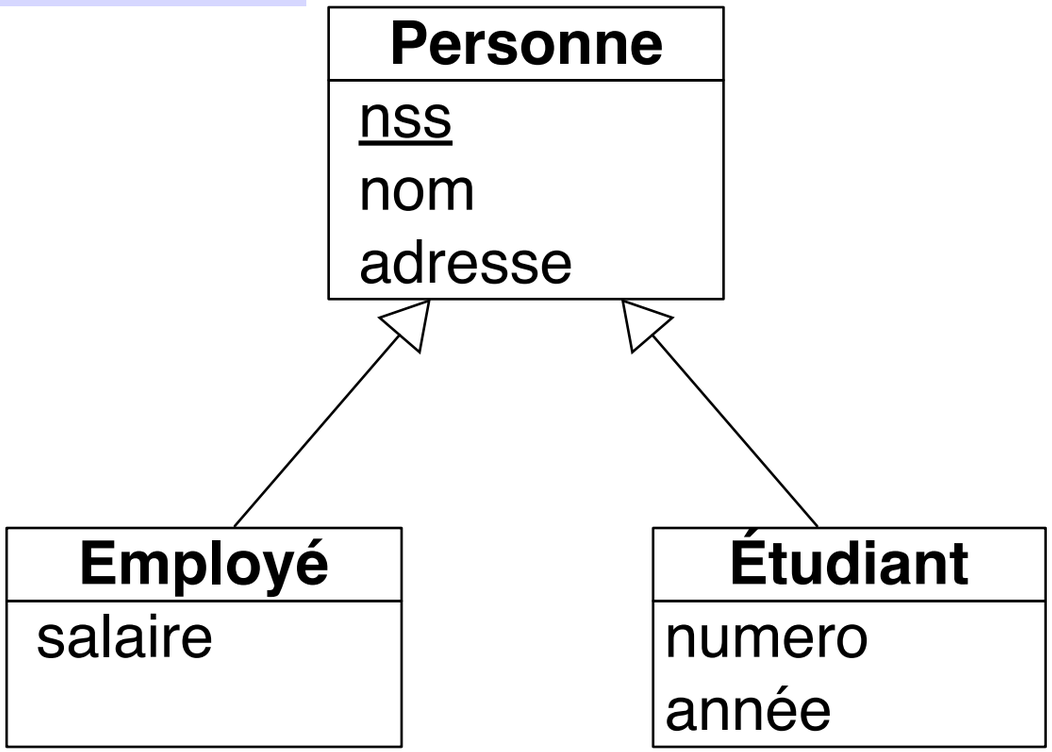


- Chaque personne réside dans une ville
- un employé est une personne, donc chaque employé réside dans une ville
- en plus chaque employé travaille dans une entreprise

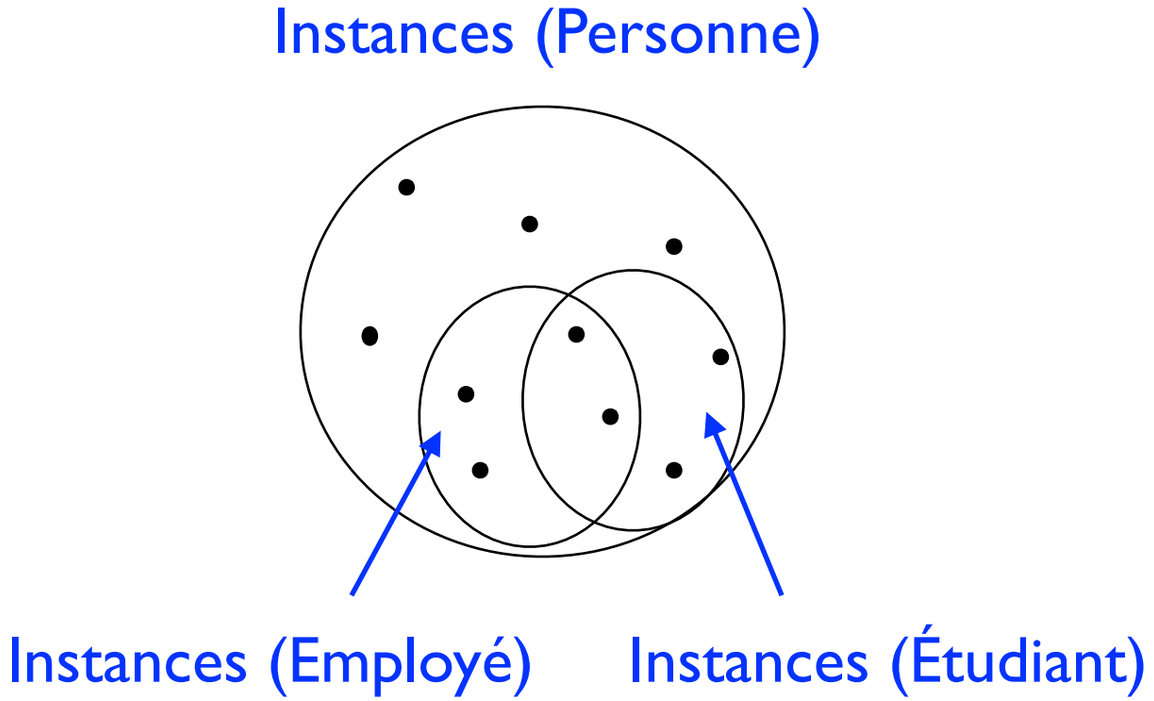
Spécialisation

- Un même entité peut avoir plusieurs spécialisations (pas de limite sur le nombre)

Syntaxe :



Sémantique :

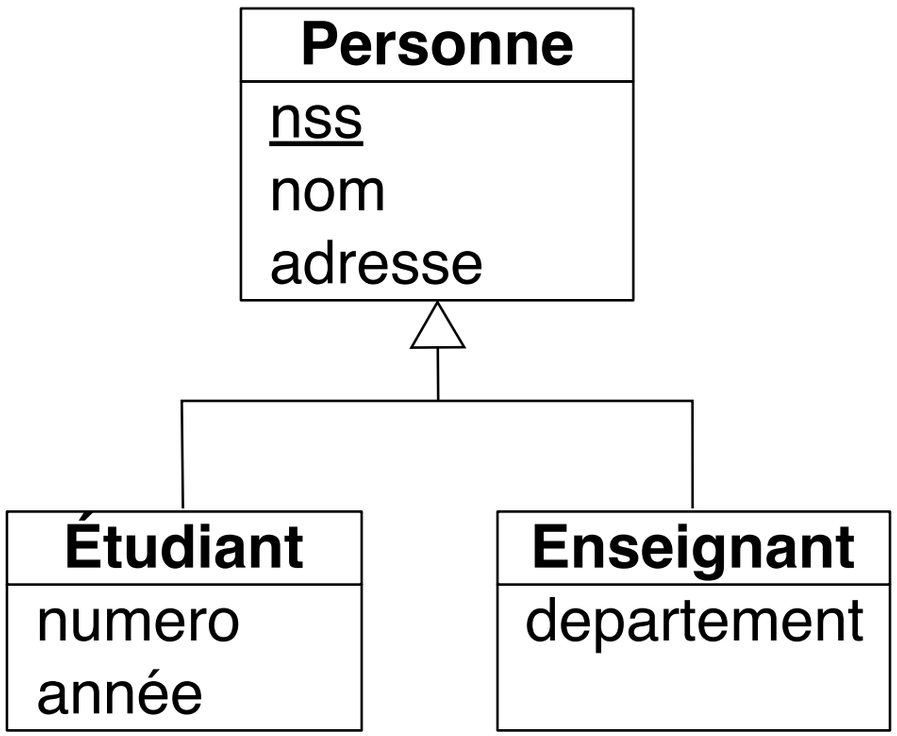


- Remarque : aucune contrainte sur les sous-ensembles :
  - ils peuvent avoir une intersection (les étudiants qui travaillent) ou pas
  - Ils peuvent être des sous-ensembles propres (il existe des personnes qui ne sont ni étudiants ni employés) ou pas

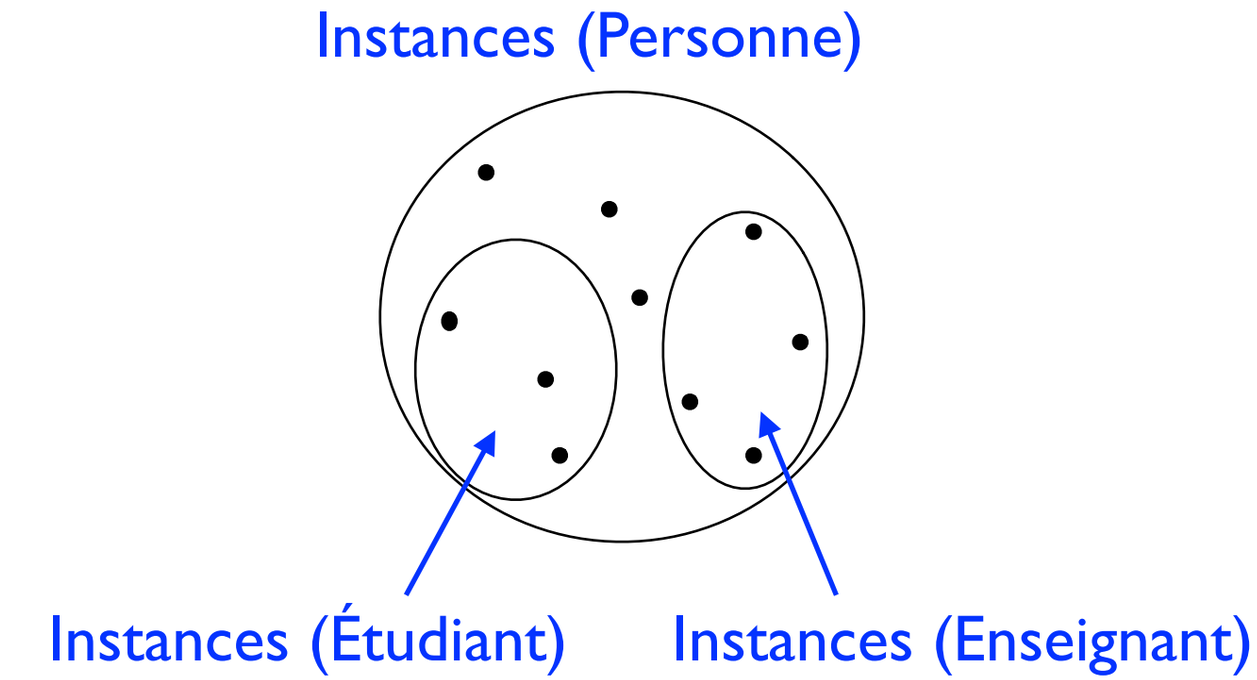
Spécialisation avec contraintes

- Des formes particulières de spécialisation permettent d'ajouter des contraintes
  - Spécialisation disjointe

Syntaxe :



Sémantique :



Instances (Étudiant) ⊆ Instances (Personne)

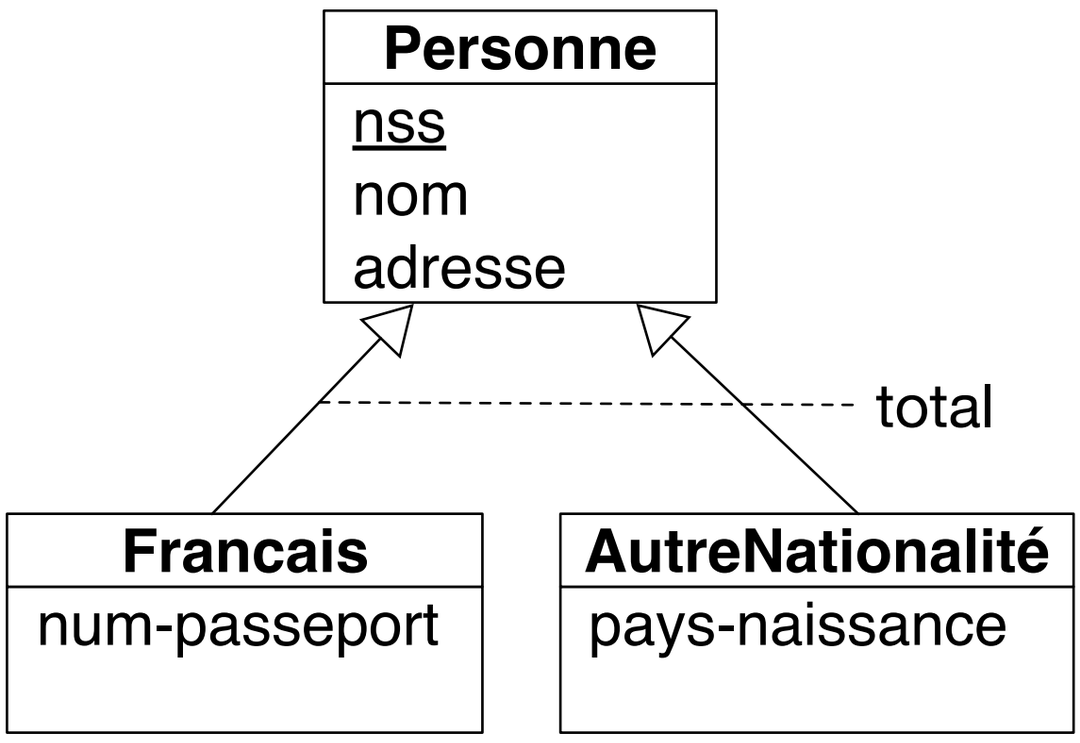
Instances (Enseignant) ⊆ Instances (Personne)

Instances (Étudiant) ∩ Instances (Enseignant) = ∅

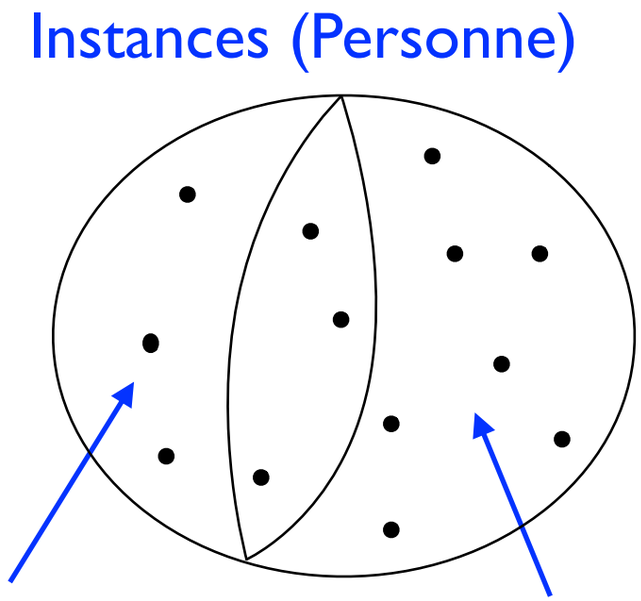
Spécialisation avec contraintes

- Des formes particulières de spécialisation permettent d'ajouter des contraintes
  - Spécialisation totale

Syntaxe :



Sémantique :

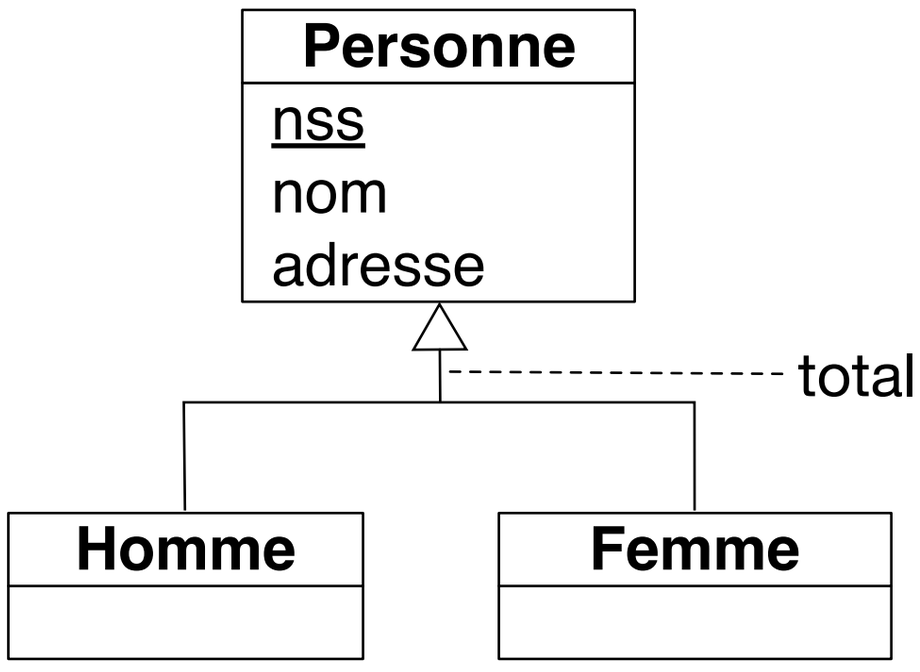


Instances (Francais) ⊆ Instances (Personne)
Instances (AutreNat) ⊆ Instances (Personne)
Instances (Francais) ∪ Instances (AutreNat) = Instances (Personne)

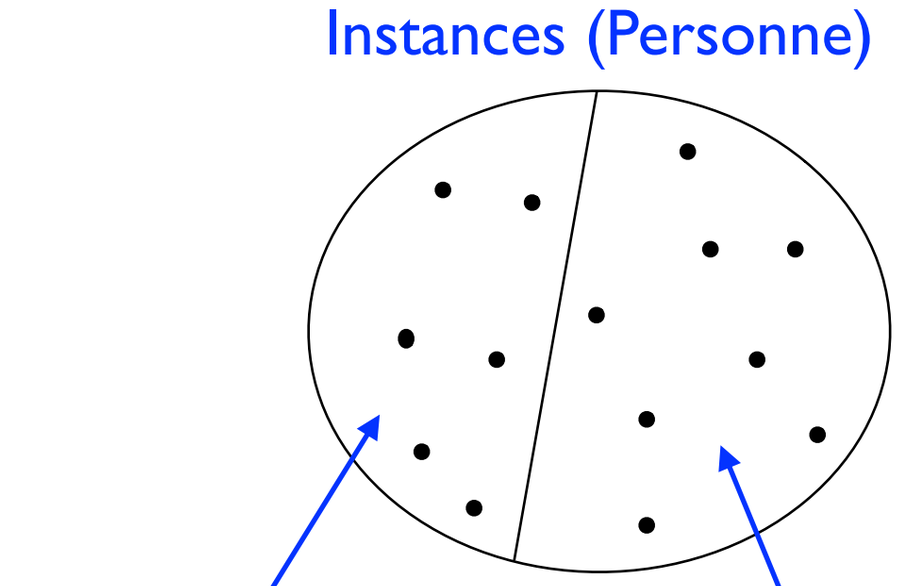
Spécialisation avec contraintes

- Des formes particulières de spécialisation permettent d'ajouter des contraintes
  - Spécialisation disjointe totale

Syntaxe :



Sémantique :



Instances (Homme) ⊆ Instances (Personne)

Instances (Femme) ⊆ Instances (Personne)

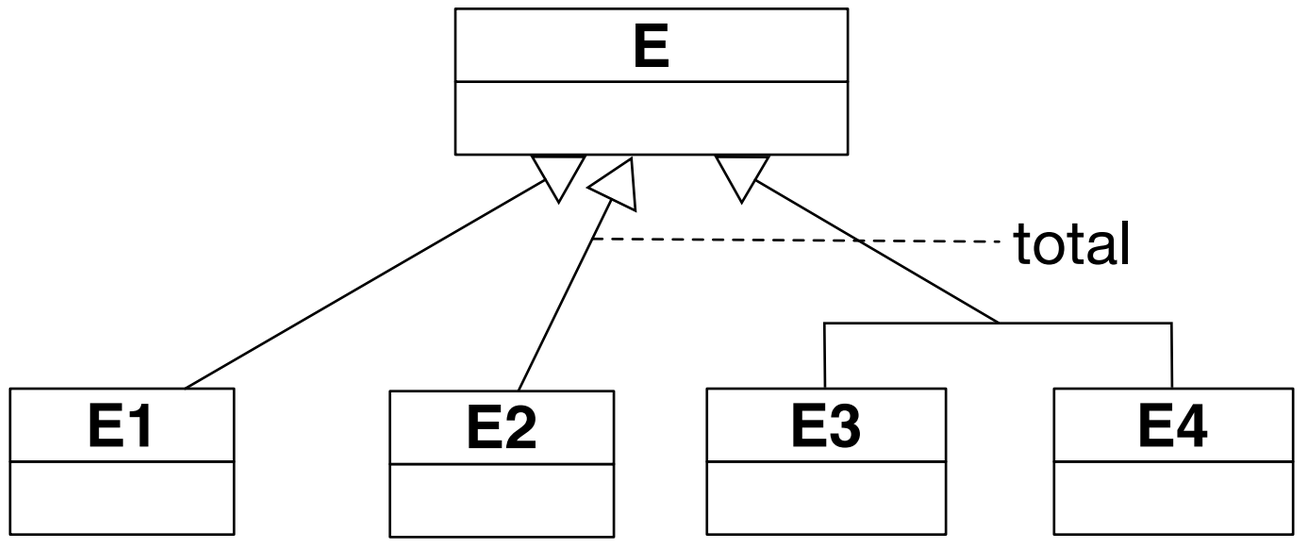
Instances(Homme) ∩ Instances (Femme) =∅

Instances (Homme) ∪ Instances (Femme) = Instances (Personne)

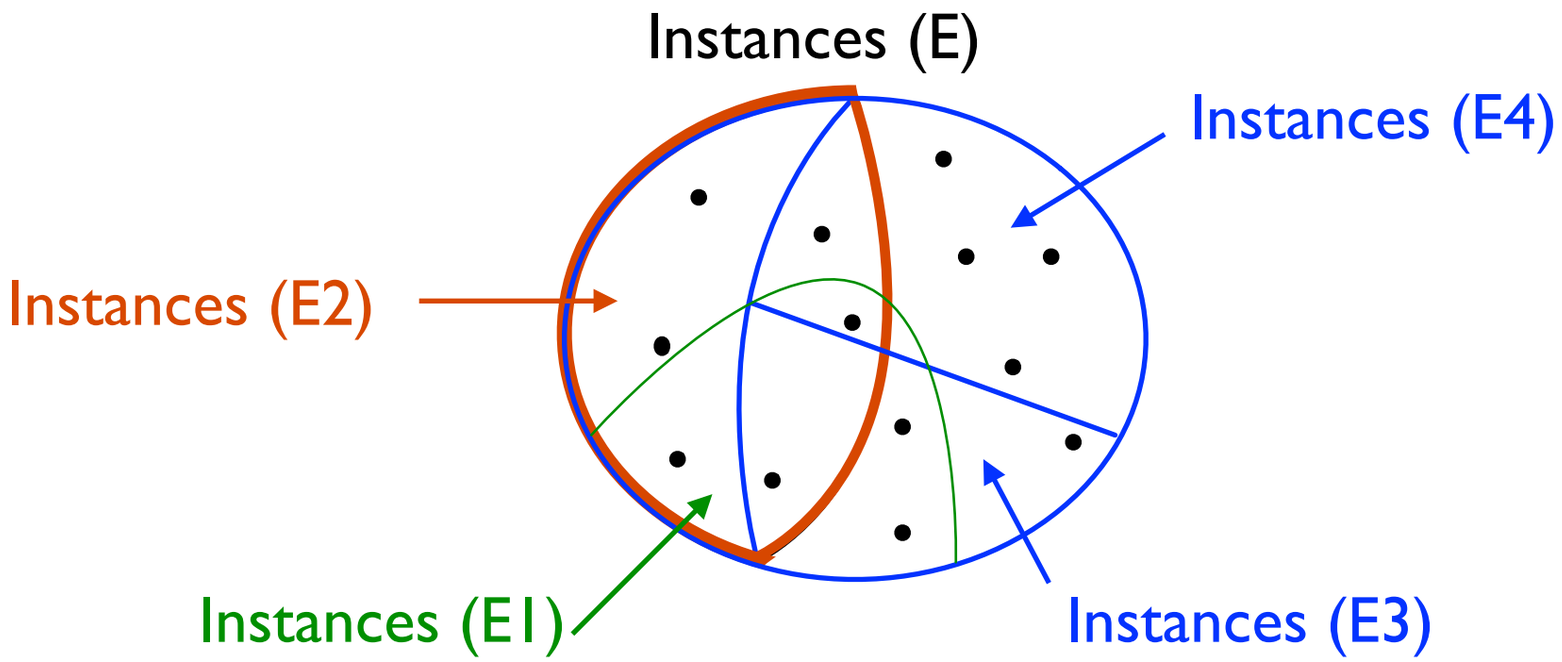
Spécialisation avec contraintes

- Des formes particulières de spécialisation permettent d'ajouter des contraintes
  - la contrante de totalité peut s'appliquer à un nombre arbitraire de spécialisations de la même entité

Syntaxe (exemple):

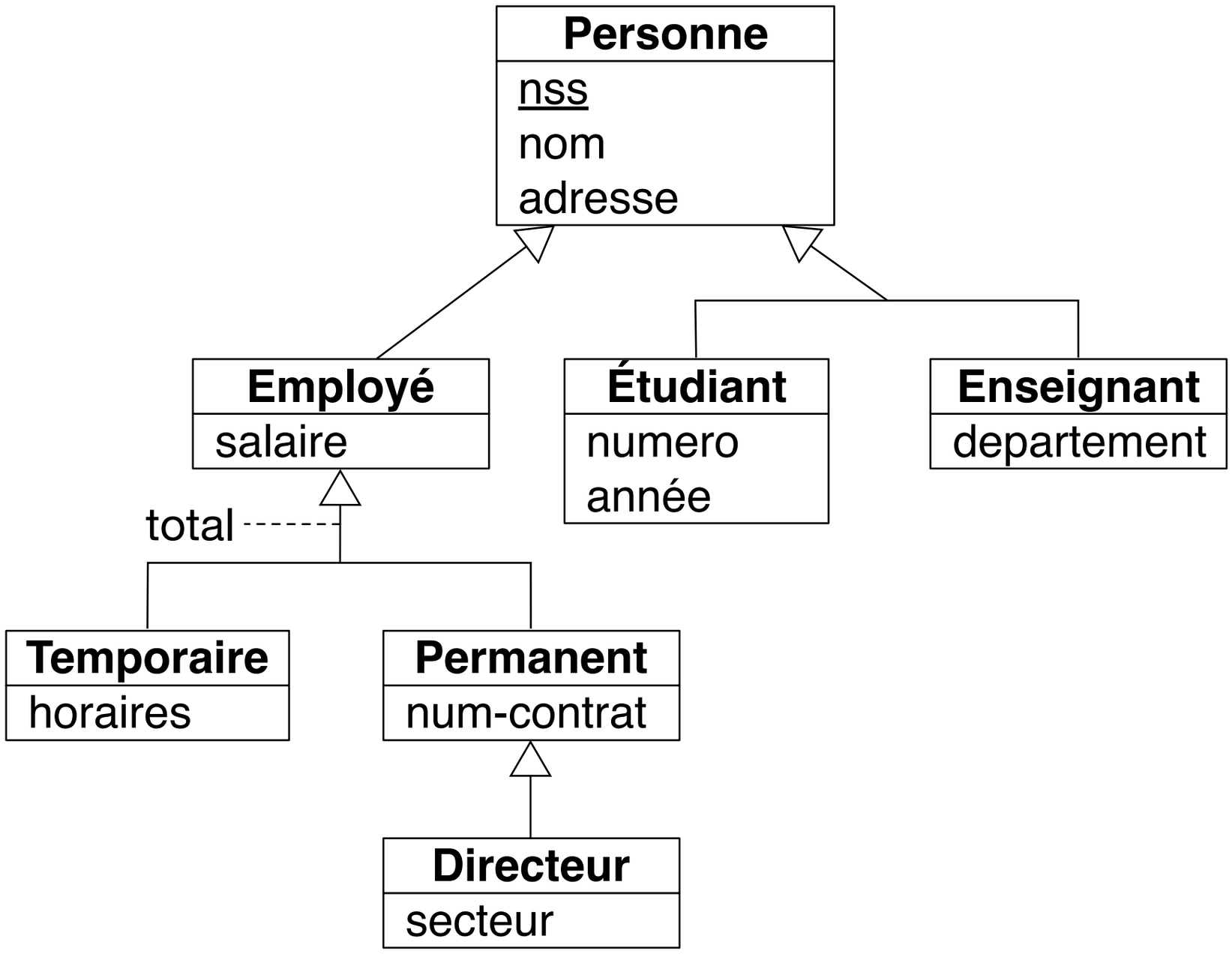


Sémantique :



Hiérarchies de spécialisations

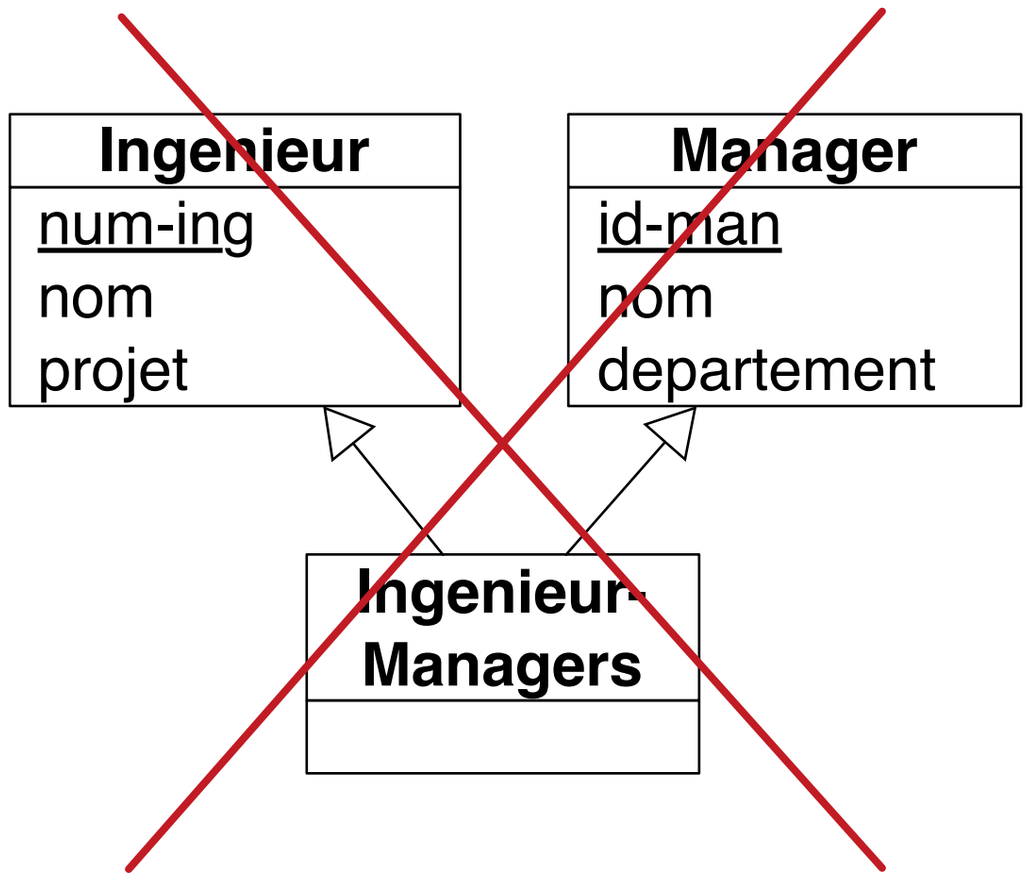
- Les spécialisations peuvent former une hiérarchie



- L'héritage s'applique tout au long de la hiérarchie. E.g. chaque directeur a un nss, nom, adresse, salaire, numéro-contrat et secteur

Hiérarchies de spécialisations

- Dans la hiérarchie l'héritage multiple est interdit : chaque entité peut avoir au plus une entité mère



Passage au modèle relationnel

- Un schéma relationnel des données peut être obtenu du diagramme E/R

- Deux phases :
  - Restructuration du diagramme E/R
  - Traduction dans un schéma relationnel

Restructuration du diagramme E/R

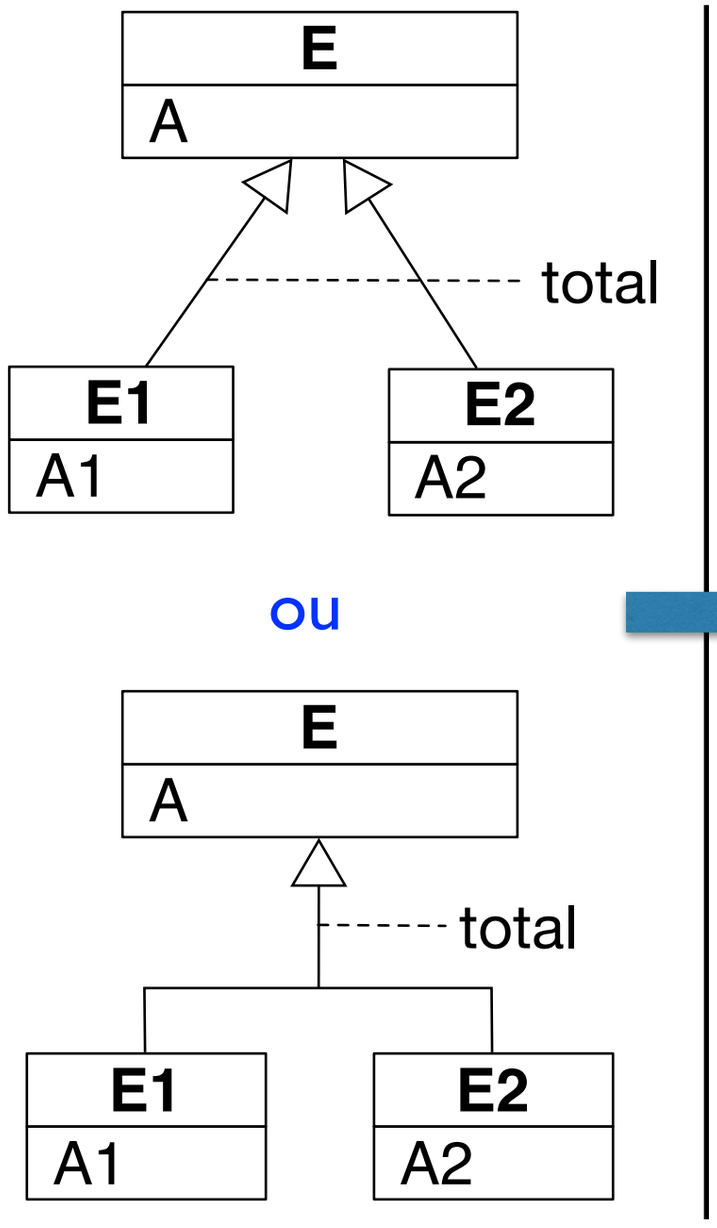
- Les spécialisations ne peuvent pas être traduites directement dans le modèle relationnel
- Le diagramme E/R doit donc être restructuré avant la traduction en relationnel
- D'autres restructurations, motivées par une estimation du coût d'accès aux données peuvent être effectuées dans cette phase :
  - Exemples
    - introduction d'attributs redondants
    - partition d'une entité (ou association) en deux
    - fusion de plusieurs entités (ou associations)

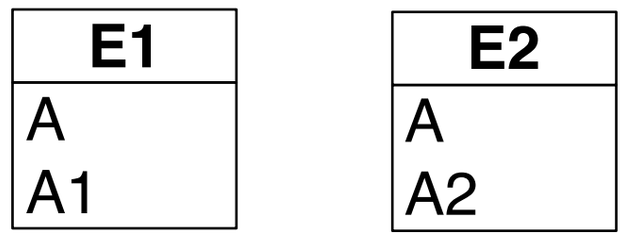
Restructuration des spécialisations

- Chaque spécialisation doit être éliminée du diagramme et remplacée par une re-formulation en termes d'entités et associations sans héritage
- Certaines des contraintes exprimées par la spécialisation seront perdues dans cette phase
- Il y a plusieurs possibilités de restructuration et il faut choisir la plus appropriée.
- Trois restructuration typiques

Restructuration des spécialisations

1) Eliminer l'entité mère (possible uniquement si la spécialisation est totale)





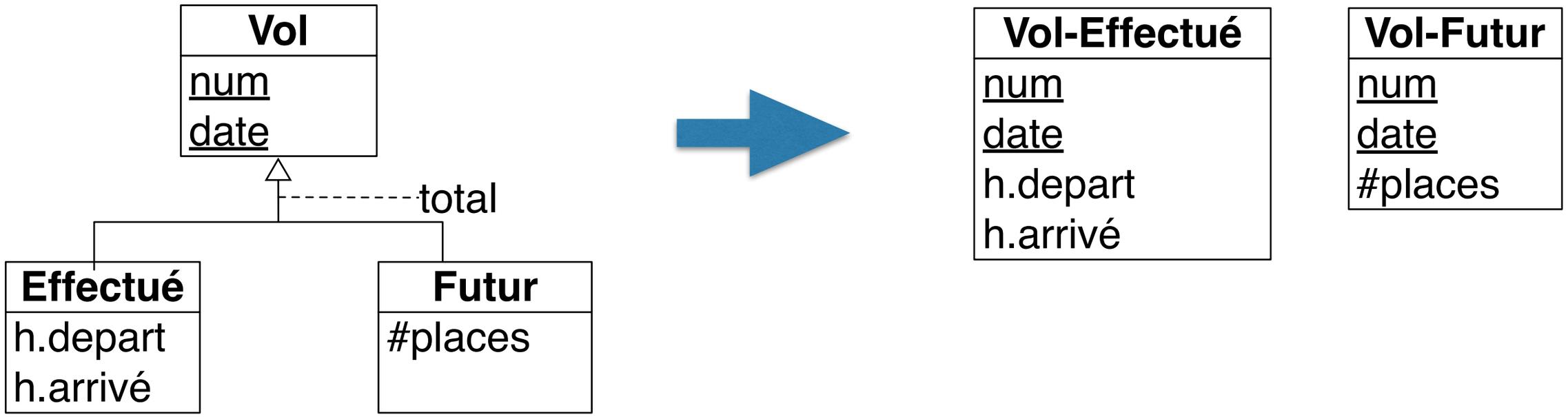
À préférer si l'entité mère
- a peu d'attributs
- participe à peu d'associations

et si, dans l'utilisation prévue des données,
les opérations qui accèdent uniquement à E sont peu fréquentes

Restructuration des spécialisations

1) Eliminer l'entité mère (possible uniquement si la spécialisation est totale)

Exemple

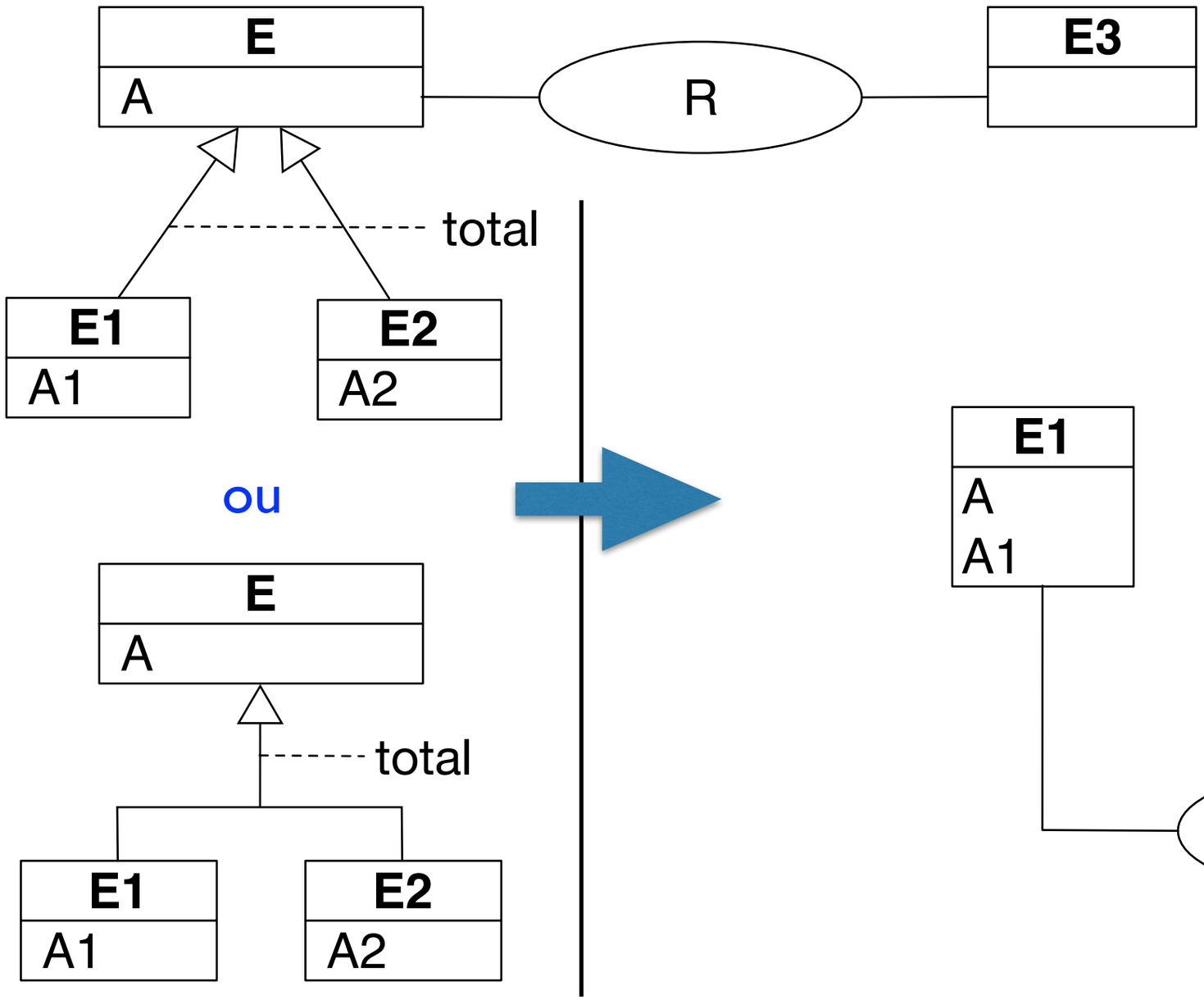


- On peut raisonnablement supposer que les opérations accèdent soit aux vols effectués (e.g. pour connaitre leur horaires effectifs) soit aux vols futurs (e.g. pour vérifier la disponibilité de places)

Restructuration des spécialisations

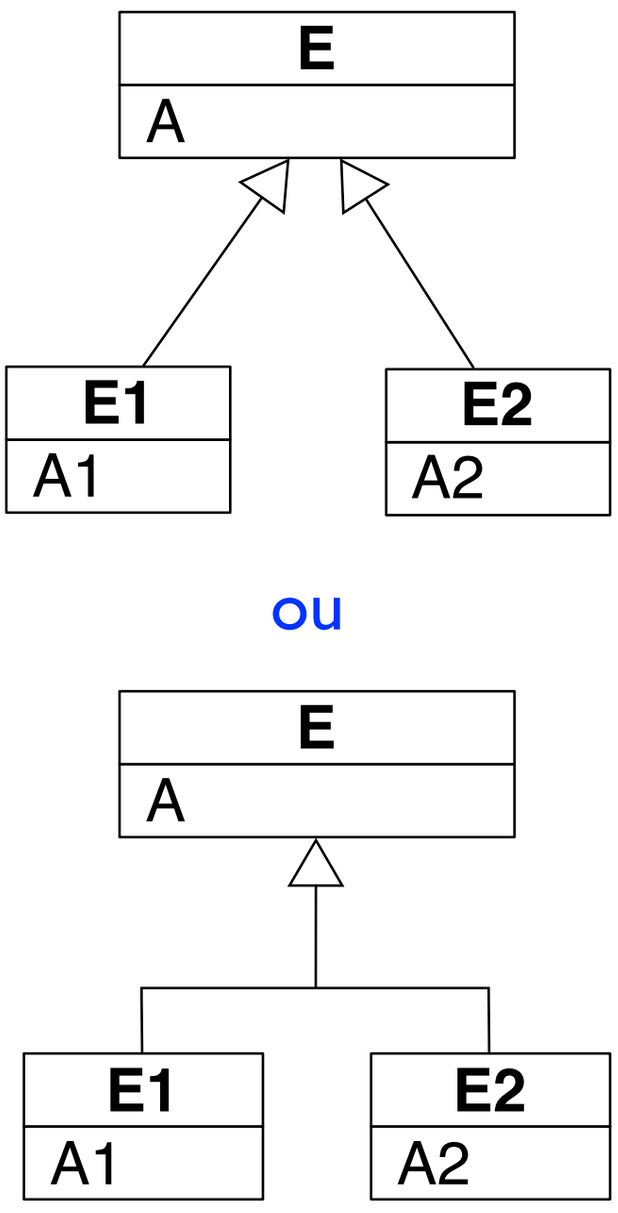
1) Eliminer l'entité mère (possible uniquement si la spécialisation est totale)

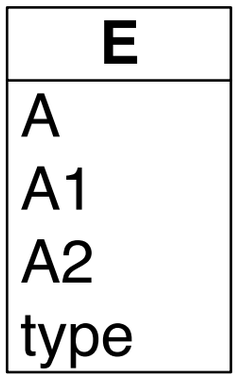
attention! les associations de E doivent être dédoublées



Restructuration des spécialisations

2) Eliminer les entités filles





type a valeurs :

- {E1, E2, E12, E}
  si la spécialisation est non-disjointe
- {E1,E2,E}
  si la spécialisation est disjointe

E n'est pas parmi les valeurs en cas de spécialisation totale

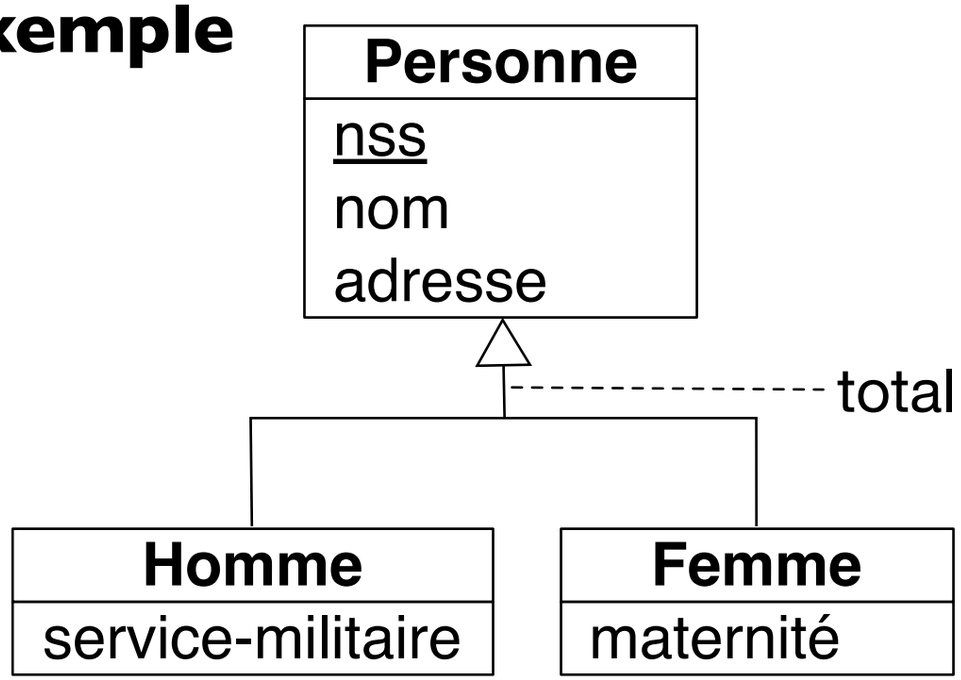
À préférer si les entités filles
- ont peu d'attributs
- participent à peu d'associations

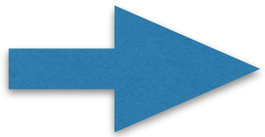
et si, dans l'utilisation prévue des données,
les opérations qui accèdent uniquement à E1 ou uniquement à E2 sont peu fréquentes

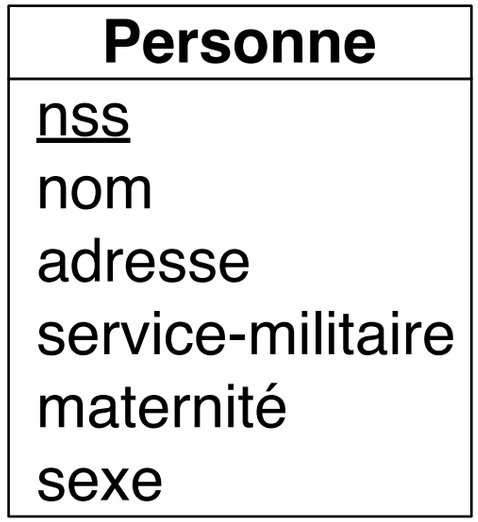
Restructuration des spécialisations

2) Eliminer les entités filles

Exemple







sexe a valeurs {Homme, Femme}

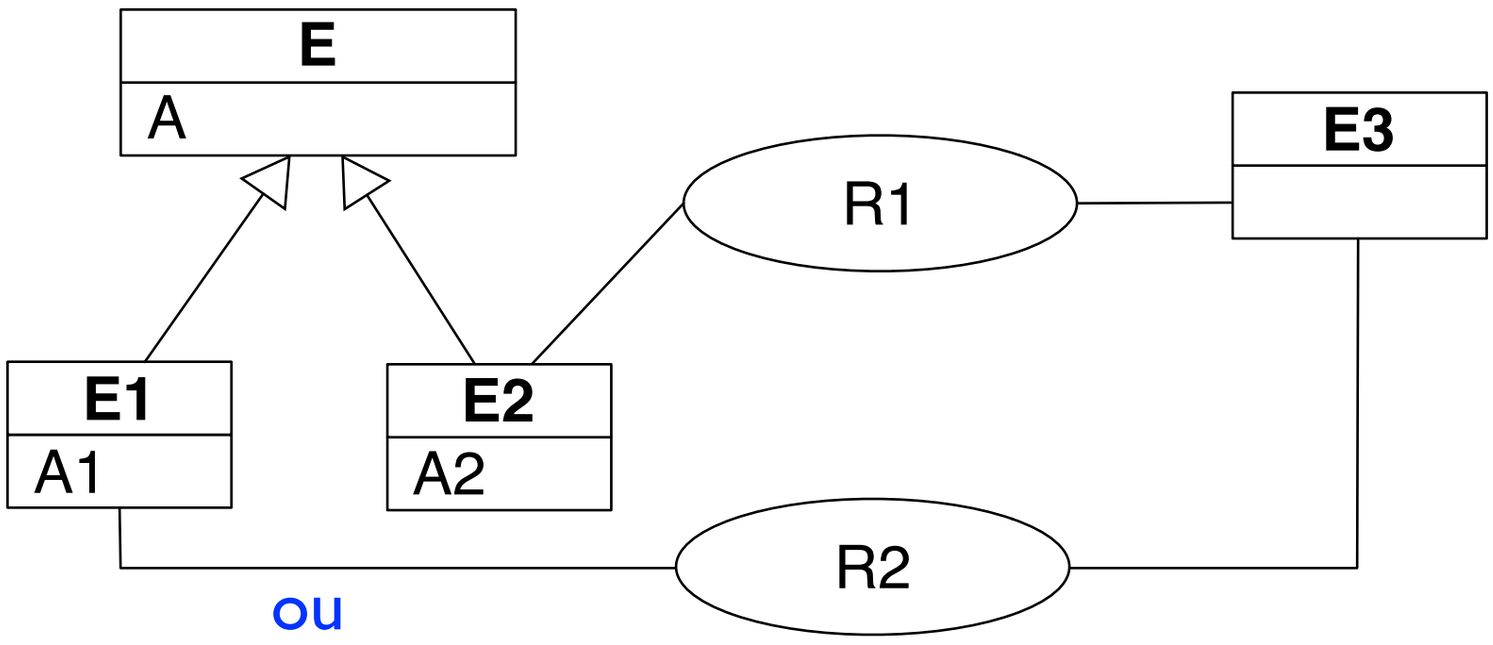
Assomption : la plupart du temps les opérations accèdent aux personnes indépendamment du fait qu'ils soient homme ou femme

Inconvénient de l'alternative 2):
les attributs spécifiques des entités filles (service militaire, maternité) ont des valeurs non significatives (NULL une fois traduits dans le modèle relationnel) pour une partie des instances

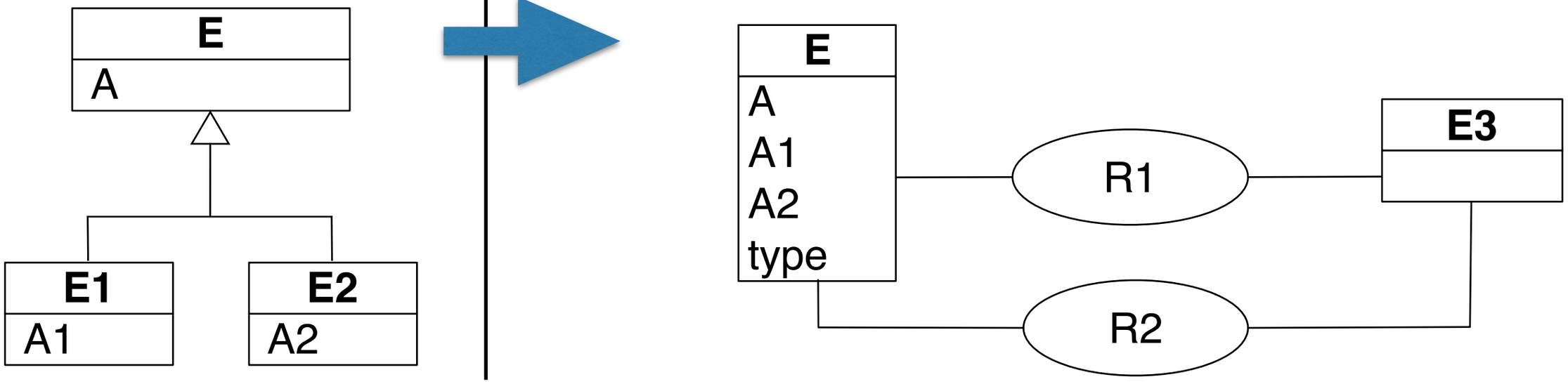
Restructuration des spécialisations

2) Eliminer les entités filles.

attention! les associations de E1 et E2 doivent être reportées sur E

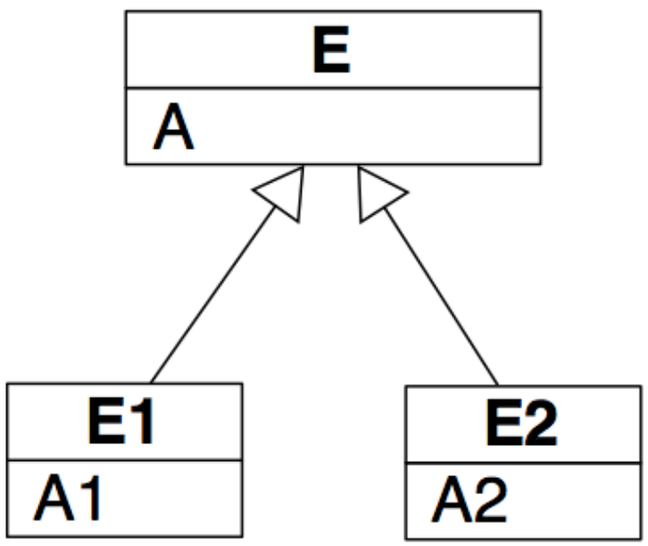


ou

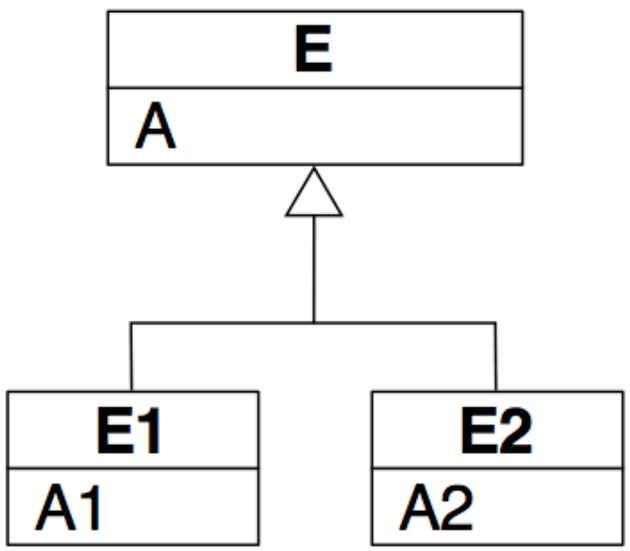


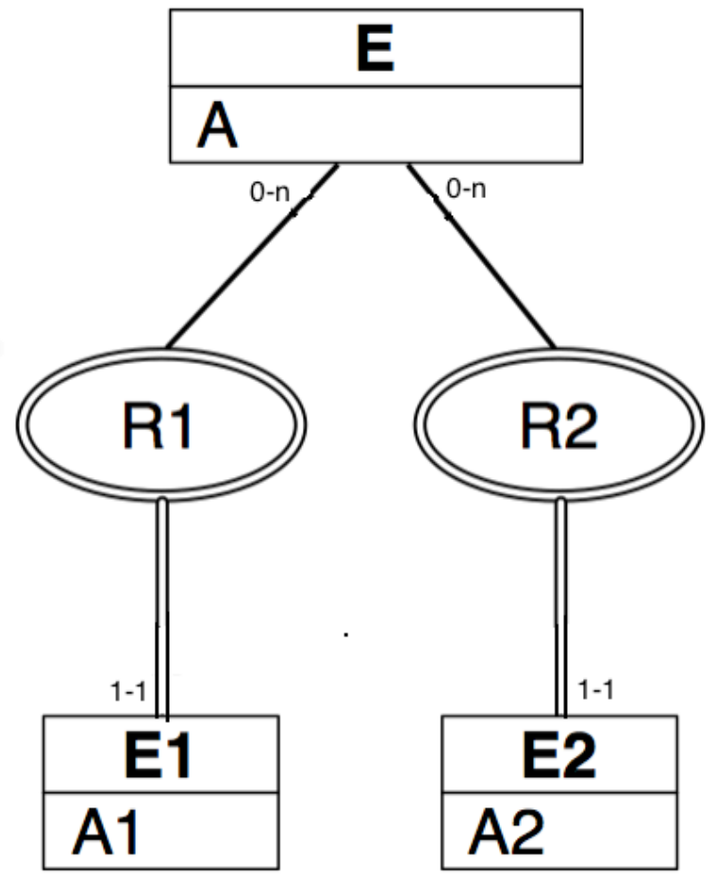
Restructuration des spécialisations

3) Maintenir toutes les entités, simuler la spécialisation avec des associations



ou





contrainte externe en cas de spécialisation totale:
chaque instance de E participe à R1 ou à R2

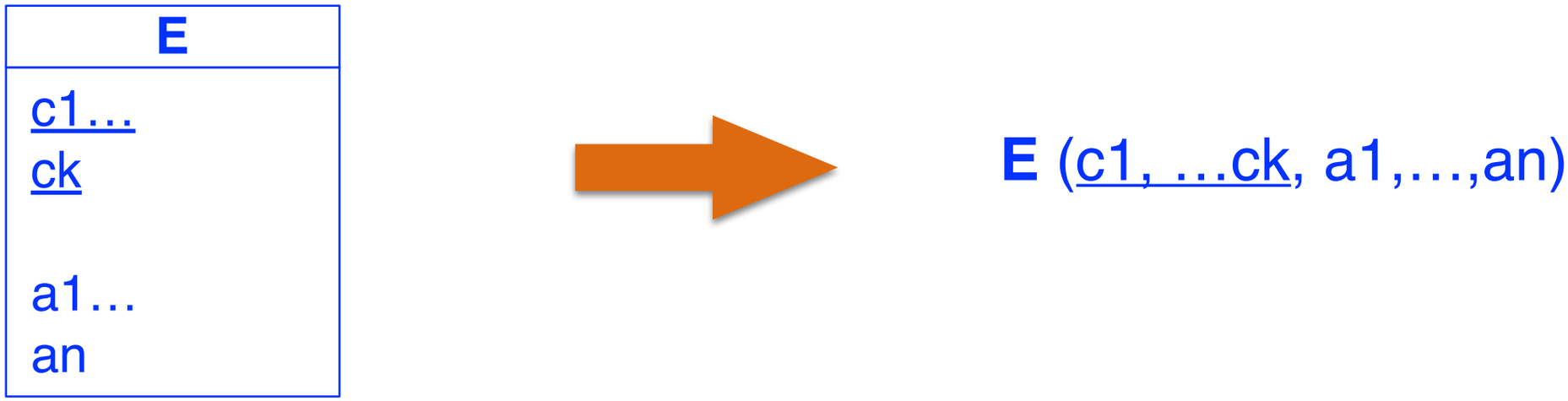
~~**contrainte externe en cas de spécialisation disjointe :**~~
aucune instance de E participe à la fois à R1 et à R2

Option la plus générale : à utiliser quand les conditions d'applicabilité de 1) ou 2) ne sont pas satisfaites

Traduction : diagramme E/R → schéma relationnel

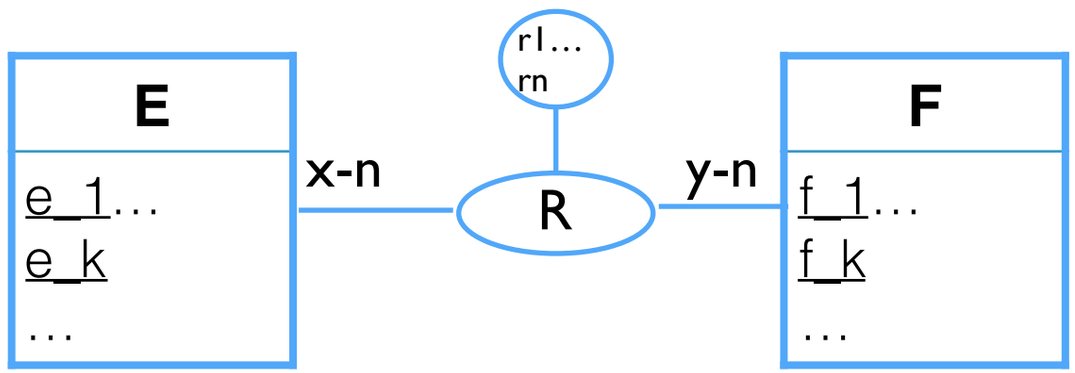
- Règle générale
  - chaque entité est traduite en un schéma de relation
  - chaque association est traduite en
    - (parfois) une schéma de relation
    - des contraintes de clefs étrangère
  - d'autres contraintes du schéma relationnel (inclusion, NOT NULL, ..) peuvent être déduites des contraintes de cardinalité du diagramme E/R
  - les contraintes externes plus complexes du diagramme E/R sont traduites par des assertions ou conditions de CHECK

Entités



- Cette traduction peut être ensuite modifiée au fur et à mesure que les associations auxquels E participe sont traduites, cf. plus loin
  - ceci est vrai en particulier si E est une entité faible et c1..ck son discriminant

Associations plusieurs à plusieurs



où x,y ∈ {0,1}

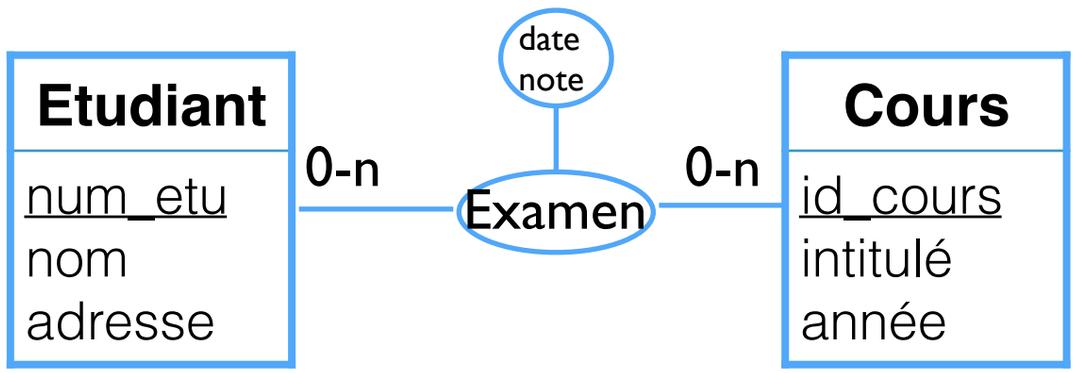
R (e1,…ek, f1,…,fm, r1,…,rn)

+ contraintes de clef étrangère :

- **R** [e1,…ek] ⊆ **E** [e1,…ek]
- **R** [f1,…fm] ⊆ **F** [f1, …,fm]

- Association n-n : seul cas où l'association binaire devient toujours une table
  - » devient une table dont la clef primaire est la combinaison des clefs des entités participantes
- Remarque. Seule la traduction de R est représentée ici (E et F doivent être traduites indépendamment)

Associations plusieurs à plusieurs - exemple



Examen (num-etu, id-cours, date, note)

+ contraintes :

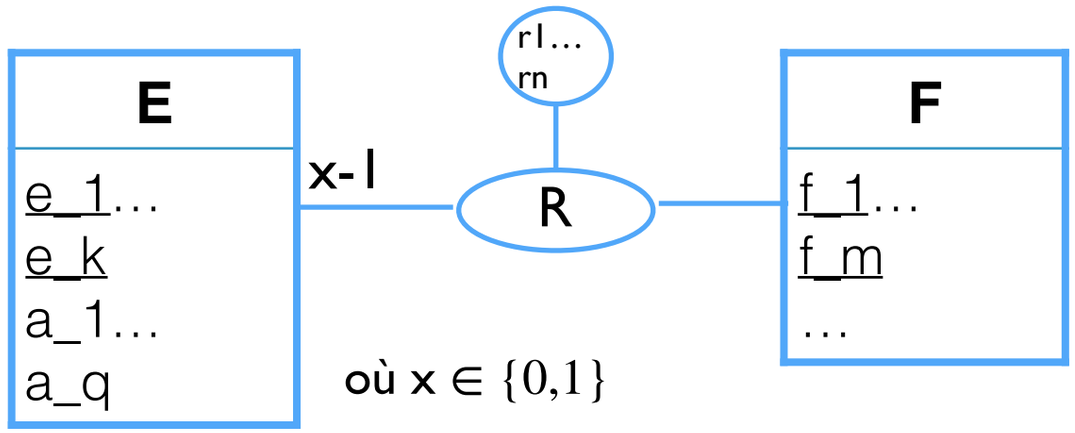
Examen [num-etu] ⊆ **Etudiant** [num-etu],

Examen[id-cours] ⊆ **Cours**[id-cours]

- Remarque sur la clef de la relation produite. La relation Examen représente les instances de l'association Examen, i.e. des couples <étudiant, cours>
  - un tel couple est identifié par <clef de l'étudiant, clef du cours>
  - cet identifiant est minimal pour Examen car un étudiant peut être associé à plusieurs cours et un cours à plusieurs étudiants

Associations un à plusieurs ou un à un

- Premier cas : R n'est pas identifiante pour l'entité avec participation max 1 (E)



E (e1..ek, a1..aq, f1..fm, r1..rn)

+ contraintes :

- **E** [f1,…fm] ⊆ **F** [f1, …,fm]
- Si x est 1 :

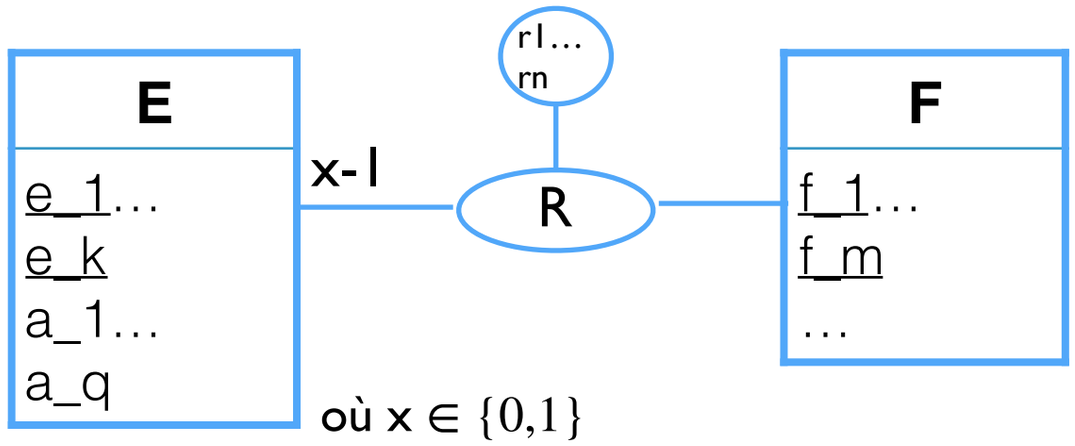
  E [f1,…fm] NOT NULL

Remarques :

- on modifie la traduction de E en ajoutant à ses attributs l'identifiant de F, et les attributs de R
- correct puisque chaque instance de E est associée par R à au plus une instance de F
- Dans le cas de cardinalité 0-1 ceci pourrait introduire des NULL dans les valeurs de f1…fm (instances de E qui ne participent pas à R)

Associations un à plusieurs ou un à un

- Premier cas : R n'est pas identifiante pour l'entité avec participation max 1 (E)



E (e1..ek, a1..aq, f1..fm, r1..rn)

+ contraintes :

- **E** [f1,…fm] ⊆ **F** [f1, …,fm]
- Si x est 1 :

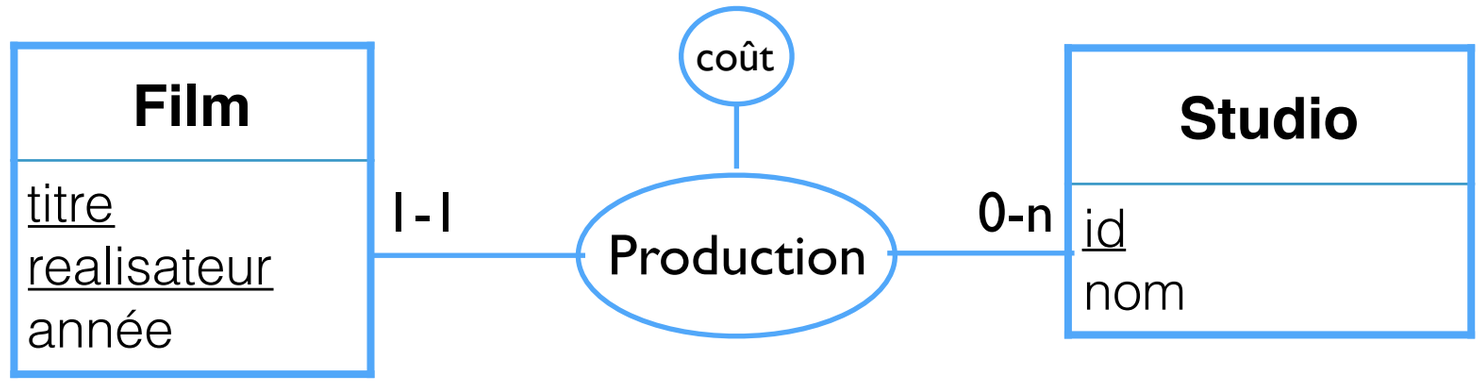
  E [f1,…fm] NOT NULL

Encore des remarques :

- F doit être traduite indépendamment
- l'effet de la traduction de plusieurs associations de ce type peut se cumuler en E
- si la cardinalité max est 1 des deux cotés, on modifie E ou F, **pas les deux**

Associations un à plusieurs ou un à un

- **Exemple**



Film (titre, réalisateur, année, id-studio, coût-production)

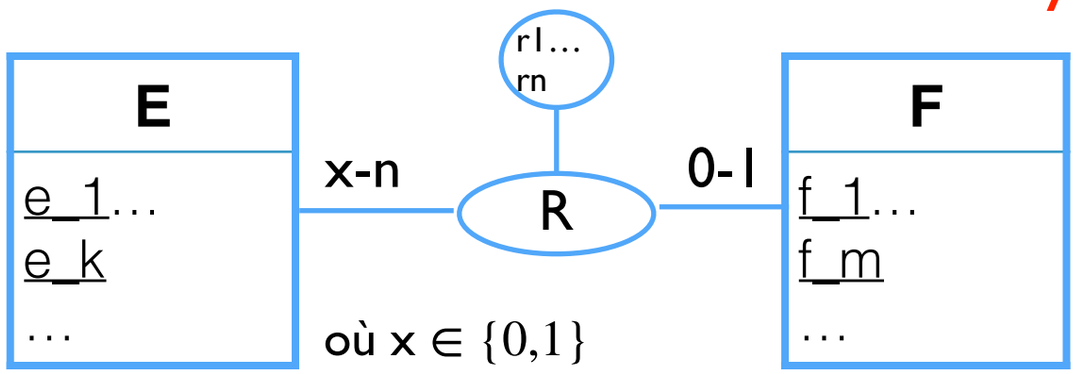
+ contrainte de clef étrangère :
Film [id-studio] ⊆ **Studio** [id]
Film [id-studio] NOT NULL

- Remarque. On peut renommer les concepts (entités, associations, attributs) en passant au schéma relationnel

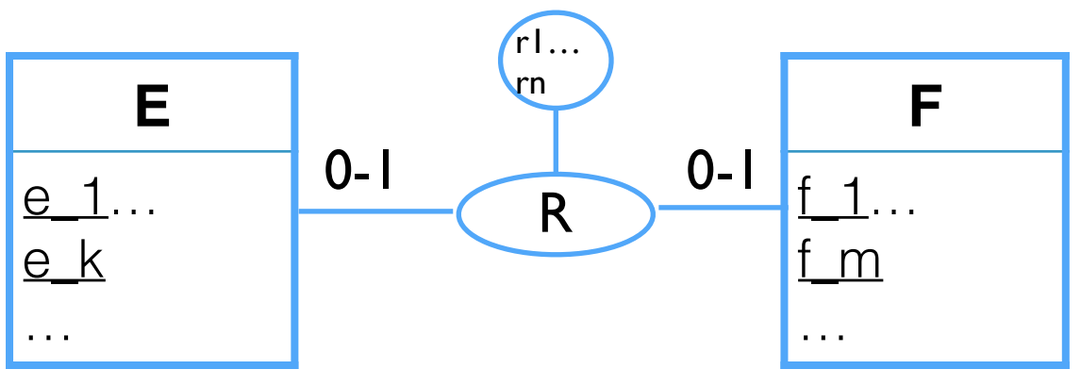
Associations un à plusieurs ou un à un

- La traduction de l'association peut entrainer des NULL (Si aucun coté n'a cardinalité 1-1)
- Si on souhaite l'éviter, une alternative est de traduire l'association comme une association plusieurs à plusieurs
- Mais la clef contient moins d'attributs

ALTERNATIVE (si aucun coté n'a cardinalité 1-1)



R (e1,…ek, f1,…,fm, r1,…,rn)



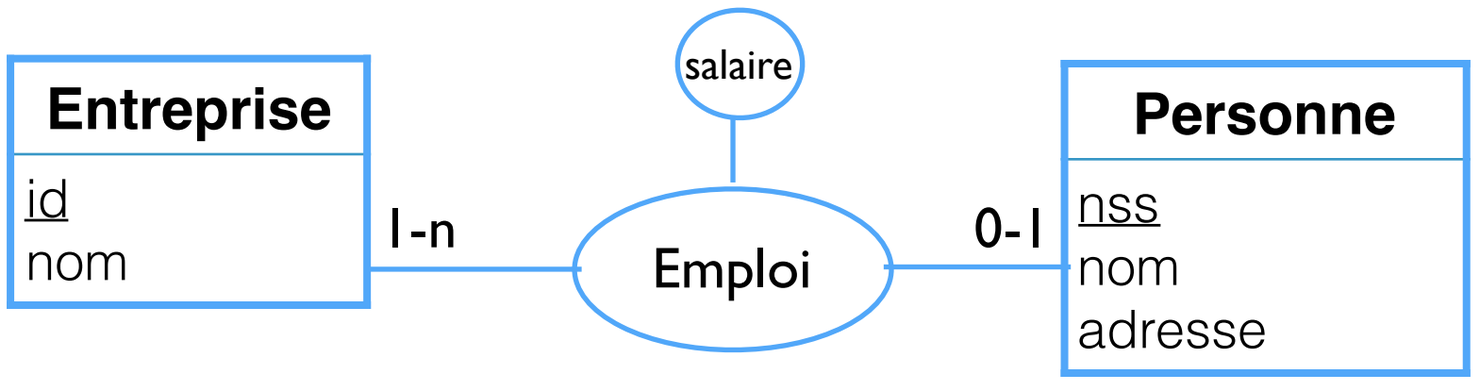
R (e1,…ek, f1,…,fm, r1,…,rn)

ou

R (e1,…ek, f1,…,fm, r1,…,rn)

Associations un à plusieurs ou un à un

- **Exemple de traduction alternative :** Association un à plusieurs



Emploi (nss, id-entreprise, salaire)

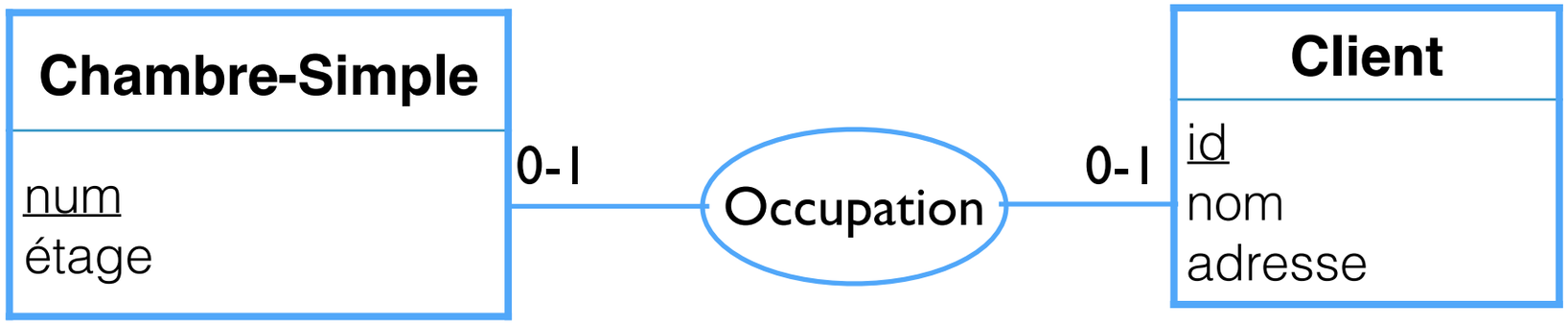
+ contraintes :

Emploi[nss] ⊆ Personne[nss]

Emploi[id-entreprise] ⊆ Entreprise[id]

Associations un à plusieurs ou un à un

- **Exemple de traduction alternative :** Association un à un



Occupation (num-chambre, id-client)

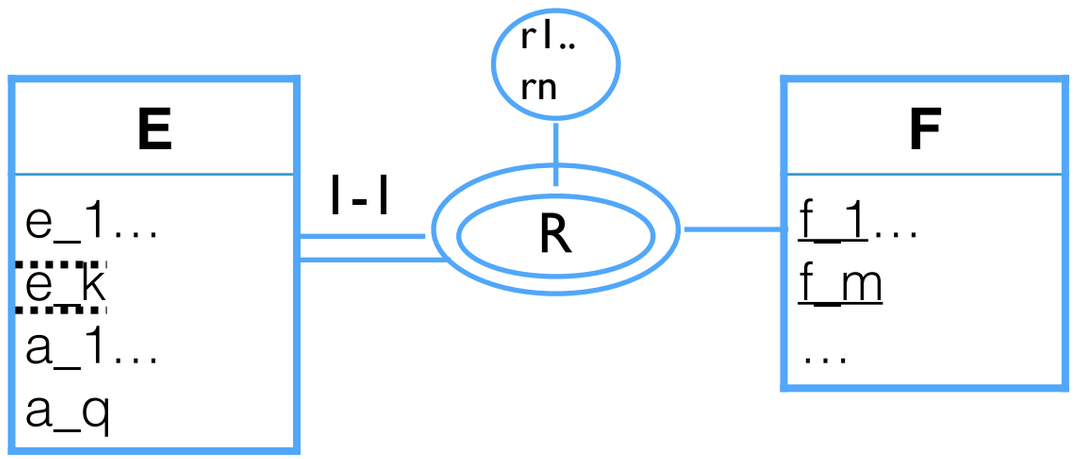
+ contraintes :

Occupation[num-chambre] ⊆ Chambre[num]

Occupation[id-client] ⊆ Client[id]

Associations un à plusieurs ou un à un

- Deuxième cas : R est identifiante pour l'entité avec participation max 1 (E)



E (e1..ek, a1..aq, f1..fm, r1..rn)

+ contraintes :

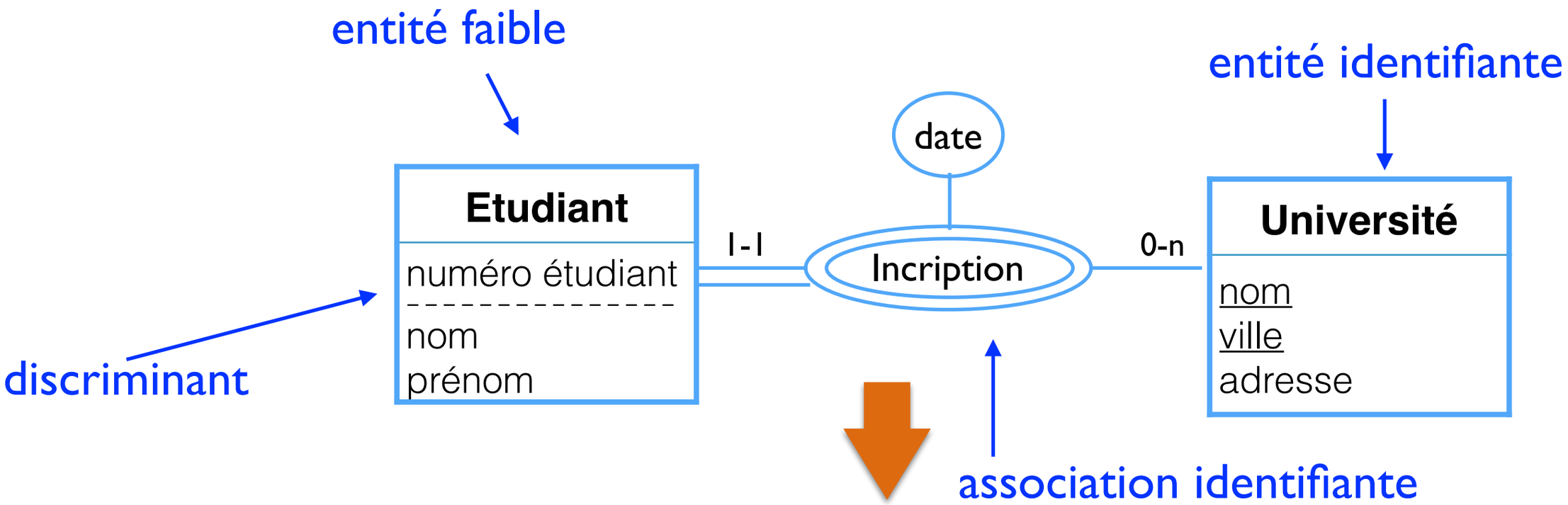
- **E** [f1…fm] ⊆ **F** [f1…fm]

Remarques :

- même traduction que dans le premier cas, mais la clef de E est étendue avec celle de F
- **E** [f1…fm] NOT NULL est impliquée par la contrainte de clef dans ce cas
- l'effet de la traduction de plusieurs associations de ce type (et du premier type) peut se cumuler en E
- si la cardinalité est 1-1 des deux cotés, R doit être traduite dans E (dont R est identifiante), pas dans F

Associations un à plusieurs ou un à un

- **Exemple d'association identifiante**

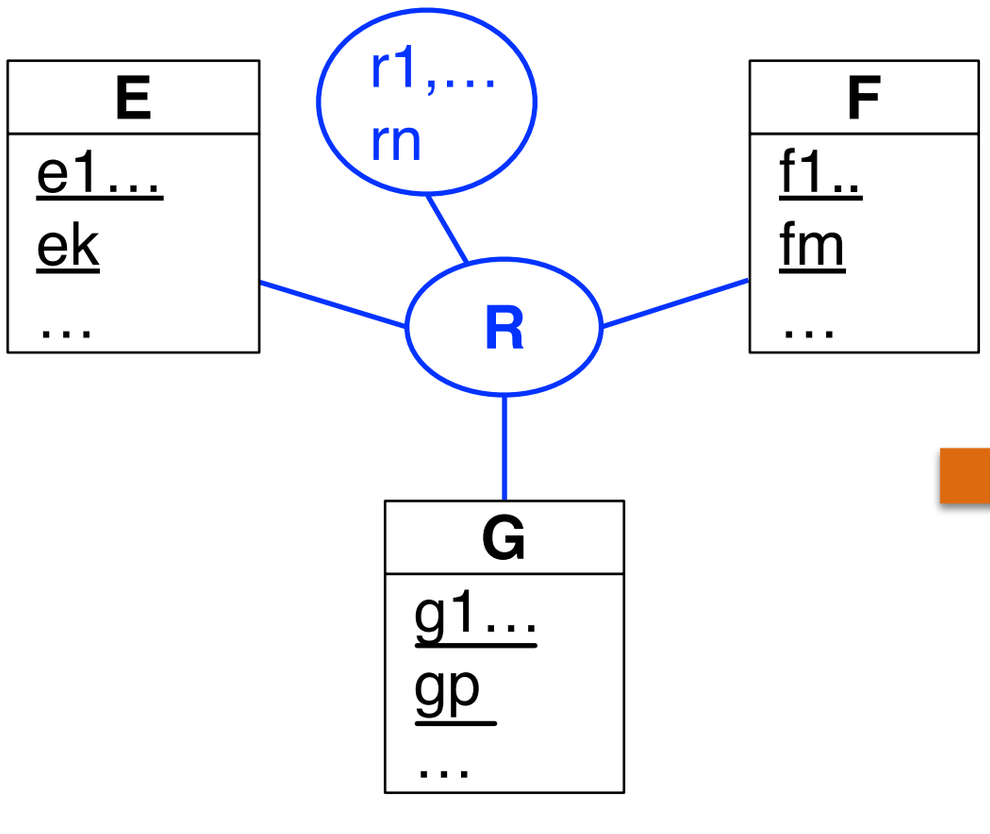


Étudiant (num-etu, nom-univ, ville-univ, nom, adresse, date-inscription)

+ Contrainte de clef étrangère :

Étudiant [nom-univ, ville-univ] ⊆ Université[nom, ville]

Traduction des associations n-aires : l'association devient une table



R (e1,…ek, f1,…,fm, g1,..,gp,r1,…,rn)

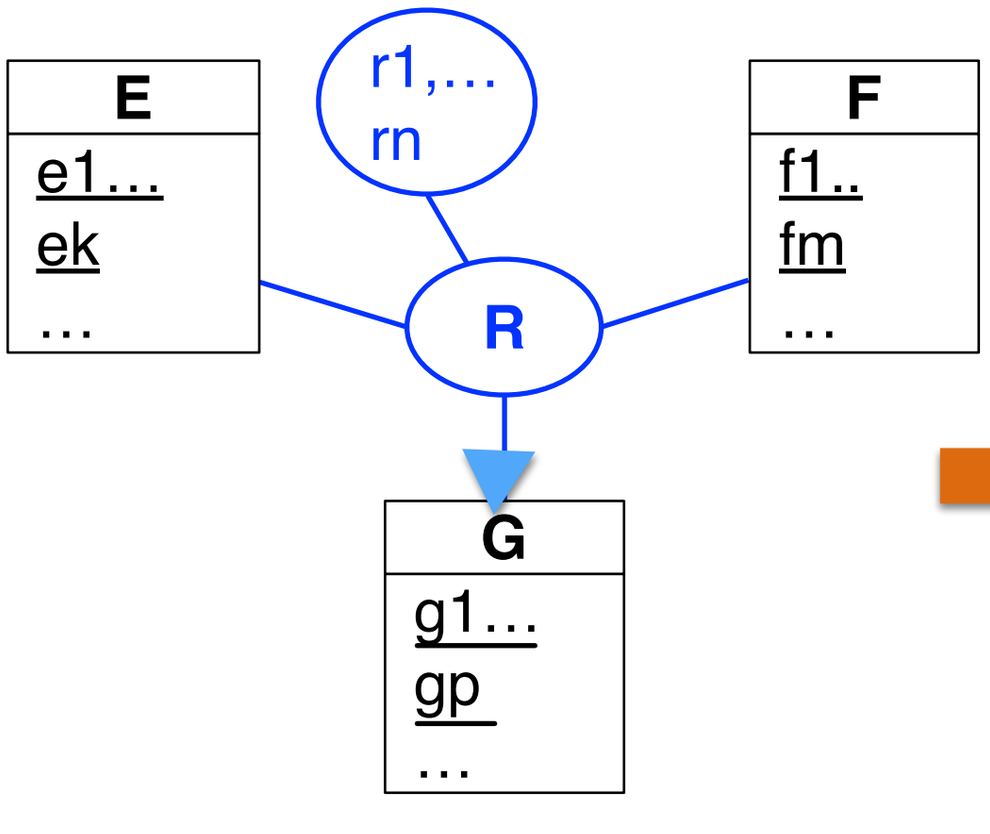
+ contraintes :

- **R** [e1,…ek] ⊆ **E** [e1,…ek]
- **R** [f1,…fm] ⊆ **F** [f1, …,fm]
- **R** [g1,…gp] ⊆ **G** [g1,…gp]

- **Clef de R :**

- (e1,…ek, f1,…,fm, g1,..,gp) est une superclef de R, mais pas nécessairement minimale
- superclef minimale : dépend des contraintes de cardinalité mais aussi de possibles contraintes externes ; à déterminer au cas par cas.

Traduction des associations n-aires : l'association devient une table



R (e1,…ek, f1,…,fm, g1,..,gp,r1,…,rn)

+ contraintes :

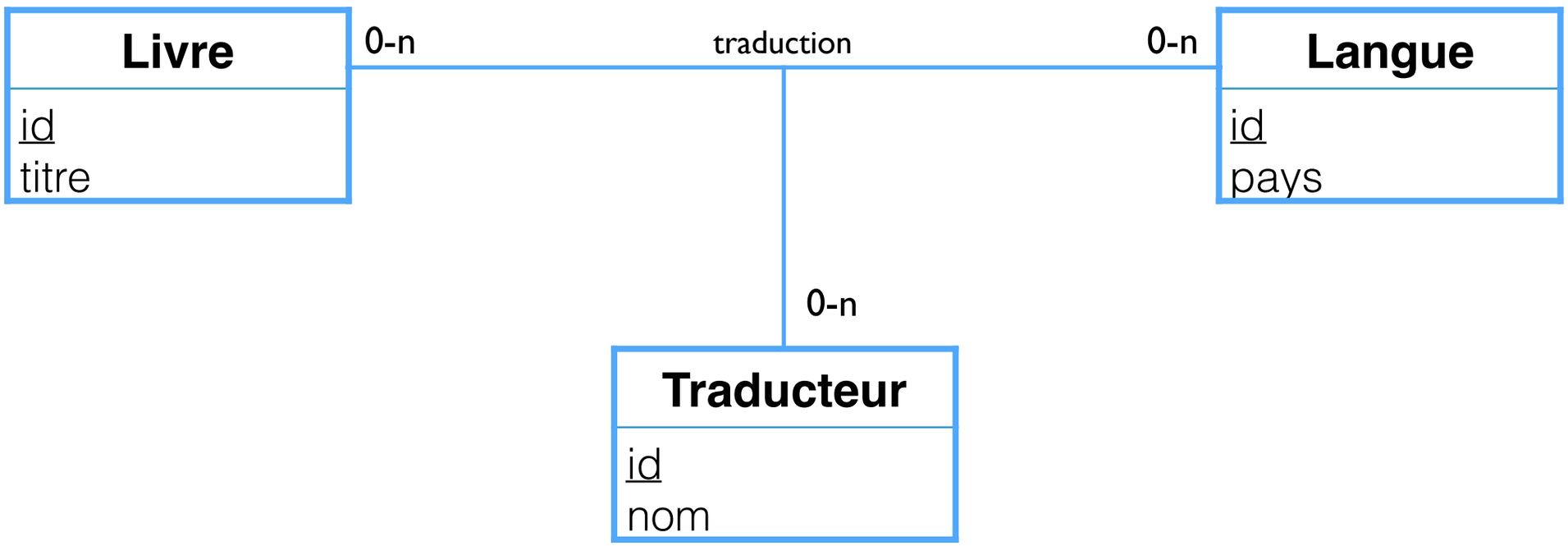
- **R** [e1,…ek] ⊆ **E** [e1,…ek]
- **R** [f1,…fm] ⊆ **F** [f1, …,fm]
- **R** [g1,…gp] ⊆ **G** [g1,…gp]

- **Clef de R :**

- (e1,…ek, f1,…,fm, g1,..,gp) est une superclef de R, mais pas nécessairement minimale
- superclef minimale : dépend des contraintes de cardinalité mais aussi de possibles contraintes externes ; à déterminer au cas par cas.
- Si une flèche pointe du coté de G alors (e1,…ek, f1,…,fm, g1,..,gp) n'est pas minimale et (e1,…ek, f1,…,fm) est une clef

Traduction des associations n-aires

- Exemple 1



Traduction (id-livre, id-langue, id-traducteur)

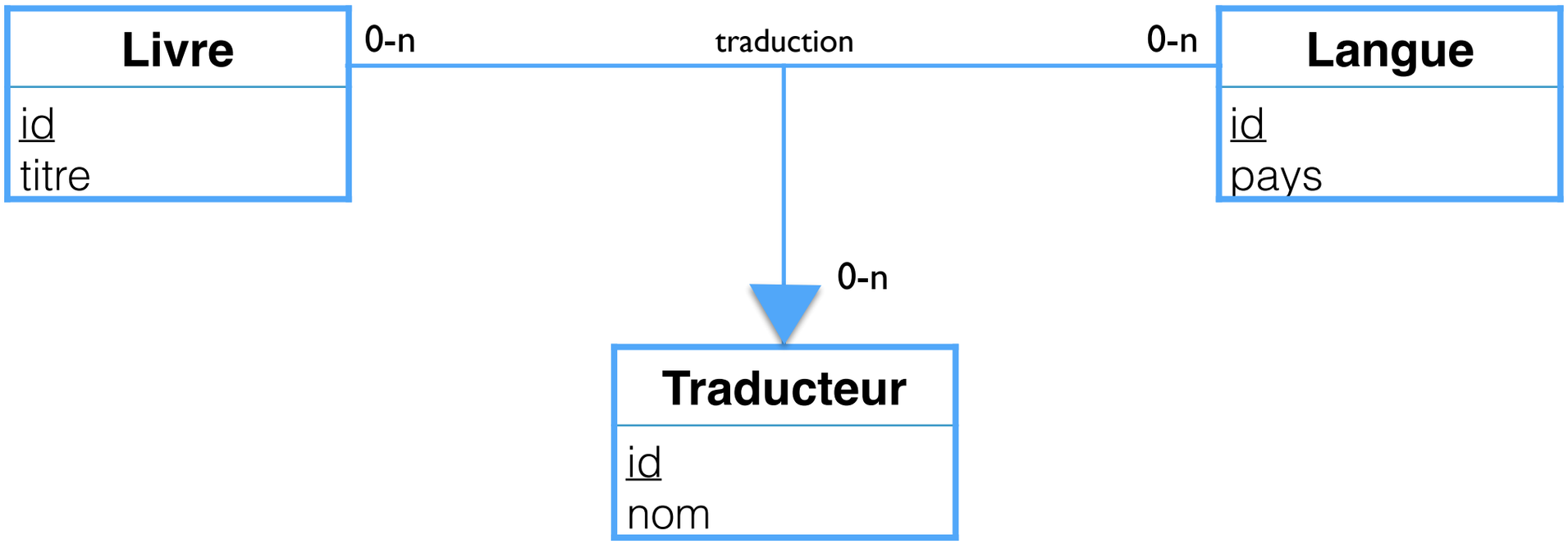
+ contraintes :
Traduction[id-livre] ⊆ Livre[id]

Traduction[id-langue] ⊆ Langue[id]

Traduction[id-traducteur] ⊆ Traducteur[id]

Traduction des associations n-aires

- Exemple 1 bis (avec une contrainte d'intégrité fonctionnelle)



Traduction (id-livre, id-langue, id-traducteur)

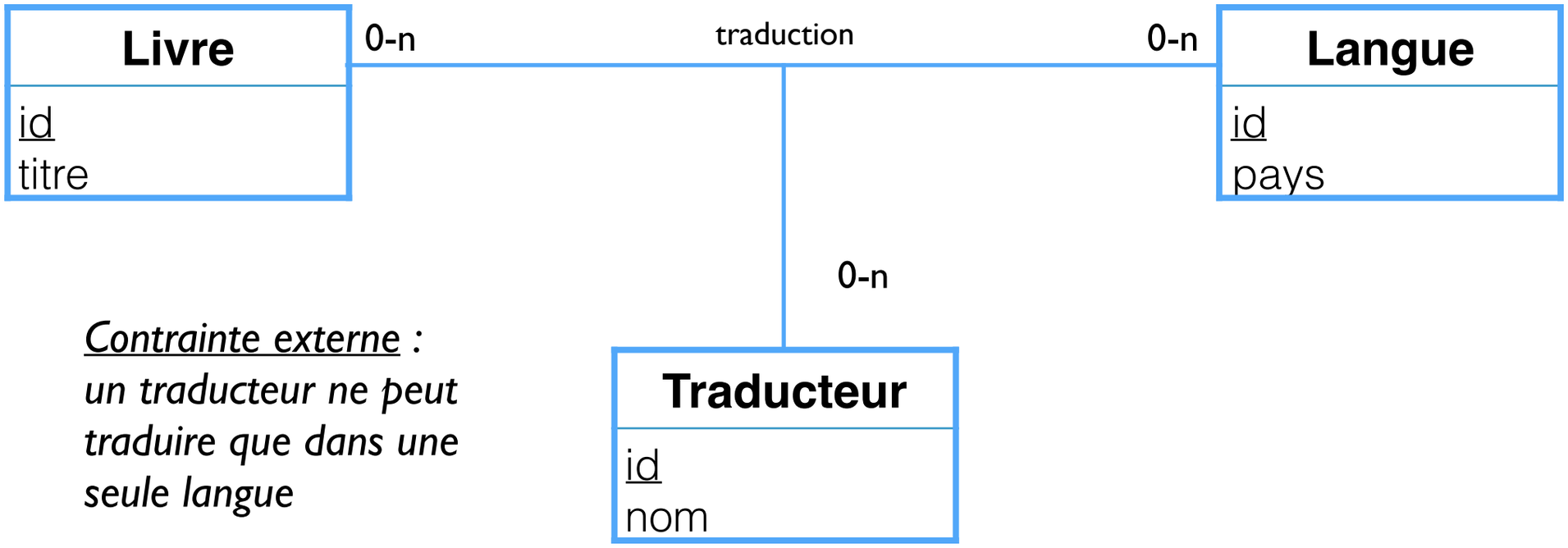
+ contraintes :
Traduction[id-livre] ⊆ Livre[id]

Traduction[id-langue] ⊆ Langue[id]

Traduction[id-traducteur] ⊆ Traducteur[id]

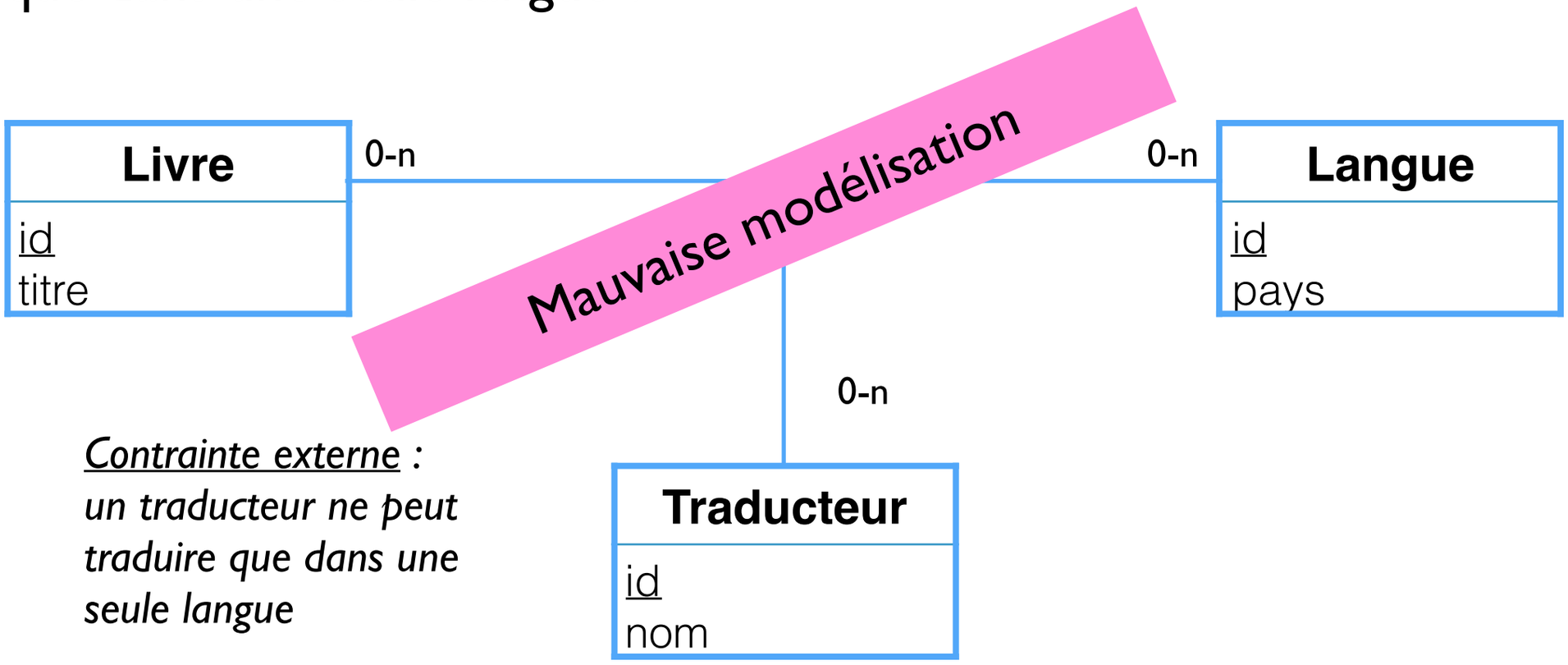
Contraintes de cardinalité dans les associations n-aires

- Et si on avait voulu représenter aussi le fait qu'un traducteur ne peut traduire que dans une seule langue ?



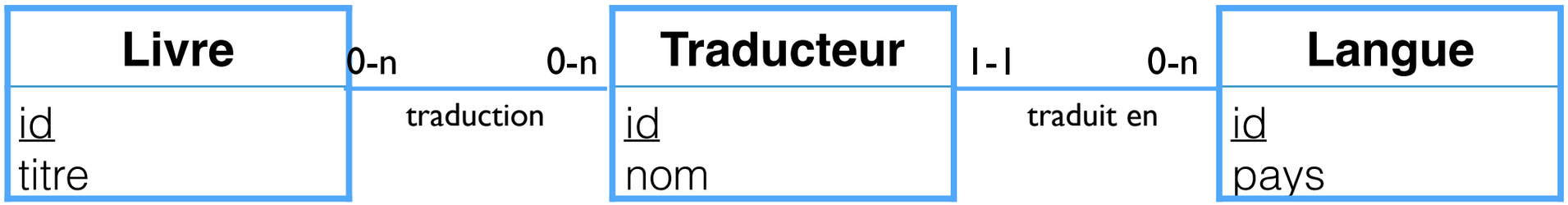
Contraintes de cardinalité dans les associations n-aires

- Et si on avait voulu représenter aussi le fait qu'un traducteur ne peut traduire que dans une seule langue ?



Contraintes de cardinalité dans les associations n-aires

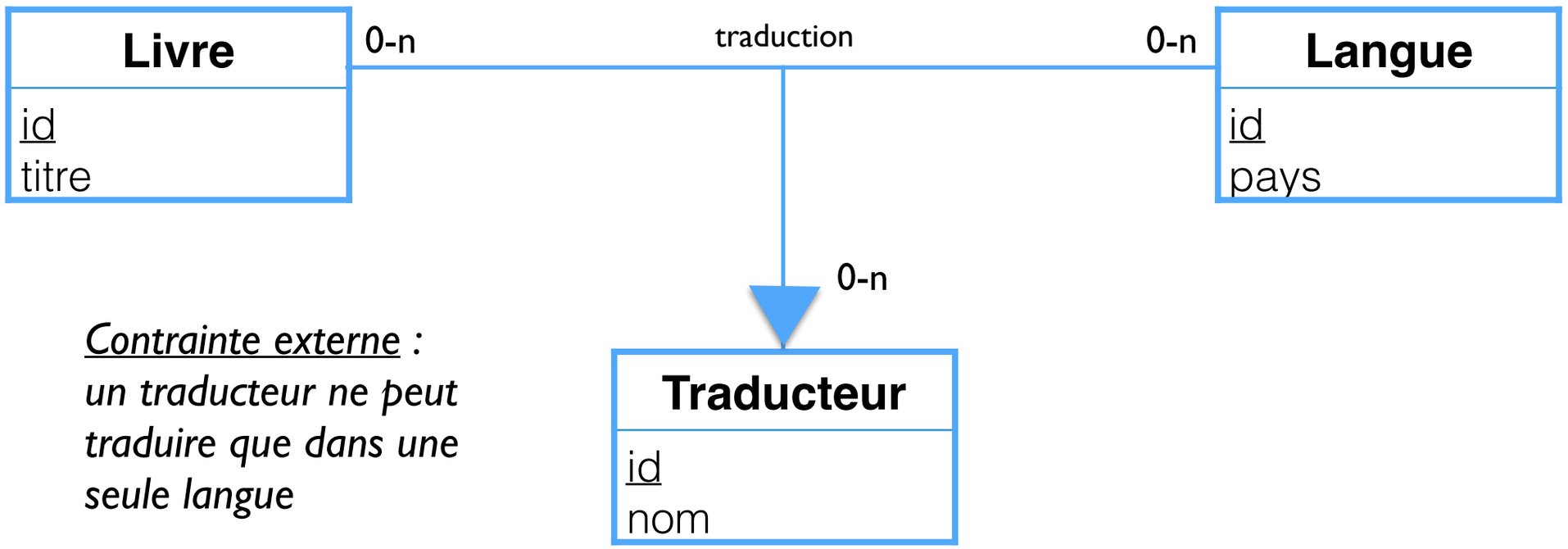
- Une modélisation avec association binaire était la seule correcte dans ce cas :



Conclusion : attention à bien exprimer les contraintes externes sur les associations n-aires : elles peuvent révéler que le choix d'association n-aire est mauvais

Contraintes de cardinalité dans les associations n-aires

- En revanche si on souhaite conserver la contrainte d'intégrité fonctionnelle :



Traduction (id-livre, id-langue, id-traducteur)

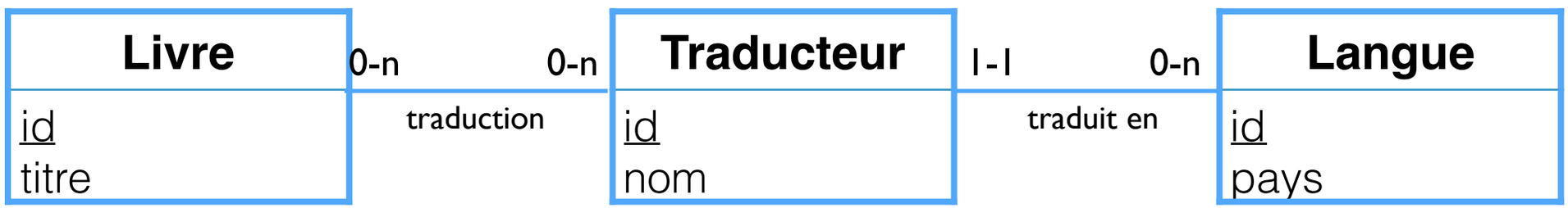
Traducteur(id, nom)

Langue(id, pays)

Livre(id, titre)

Contraintes de cardinalité dans les associations n-aires

- Alternative (équivalent) :



Contrainte externe :
un même livre ne peut être traduit en une même langue que par un seul traducteur

Traduction (id-livre, id-traducteur)

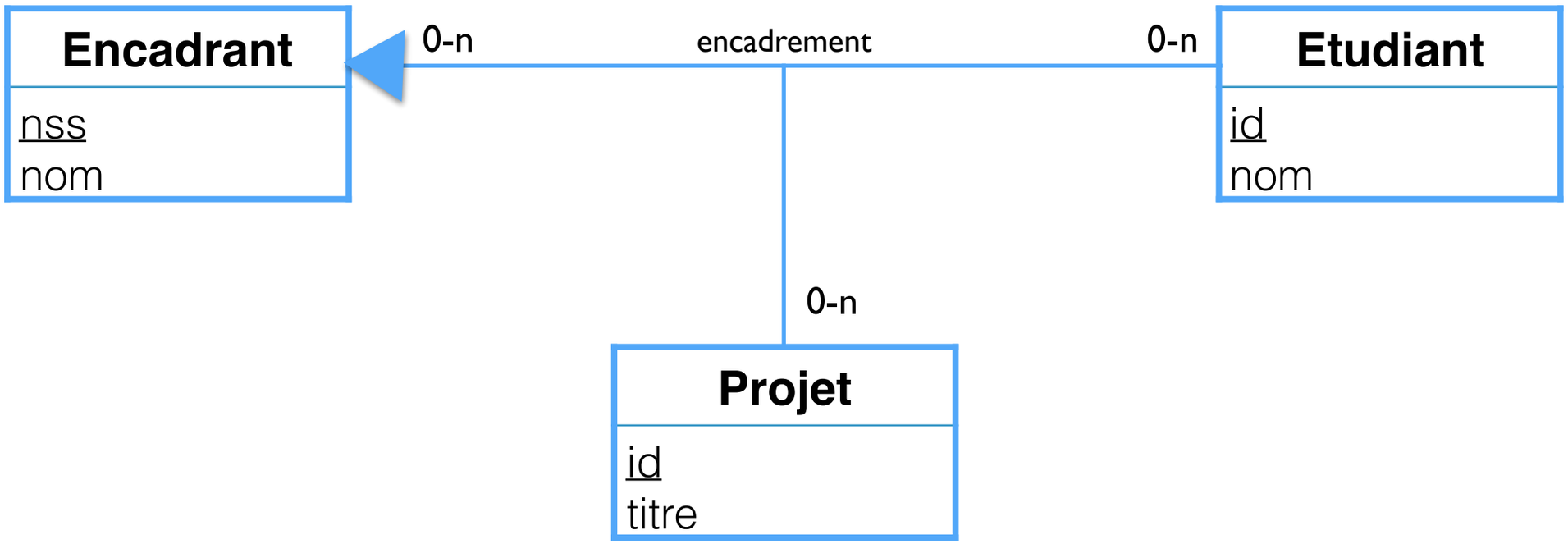
Traducteur(id, id_langue, nom)

Langue(id, pays)

Livre(id, titre)

Traduction des associations n-aires

- Exemple 2 (avec une contrainte d'intégrité fonctionnelle)



Encadrement (id-etu, id-projet, nss-enc)

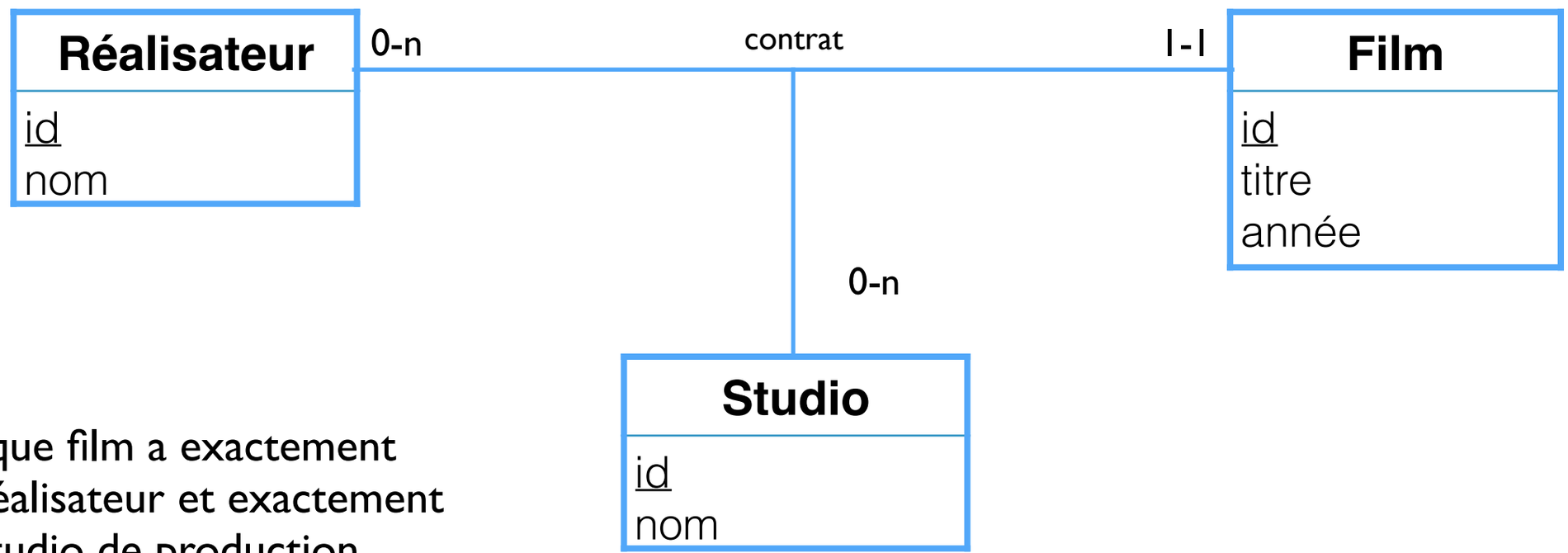
Encadrement [id-etu] ⊆ Etudiant[id]

Encadrement [id-projet] ⊆ Projet[id]

Encadrement [nss-enc] ⊆ Encadrant [id]

Traduction des associations n-aires

- Exemple 3



Chaque film a exactement
un réalisateur et exactement
un studio de production

Contrat (id_film, id-real, id_studio)

Contrat [id_film] ⊆ Film [id]

Contrat [id_studio] ⊆ Studio [id]

Contrat [id_real] ⊆ Réalisateur [id]

Traduction des associations n-aires

- En fait cette modélisation est meilleure (génère moins de redondance) :

Chaque film a exactement
un réalisateur et exactement
un studio de production

Contrat (id_film, id_studio)

Contrat [id_film] ⊆ Film [id]

Contrat [id_studio] ⊆ Studio [id]

Outils gratuits pour dessiner des diagrammes E/R

- Dia
- Calligra Flow

- Beaucoup d'autres outils permettent également la transformation automatique ER → Relationnel (MySQL Workbench, ER2SQL, …)
  - Attention : ils ont leur propre syntaxe et sémantique pour ER
  - on obtient souvent un meilleur modèle quand on traduit "à la main"

Dépendances fonctionnelles
et
formes normales

Modélisation de BD relationnelles

Conception du modèle relationnel (schéma) à partir du réel

Deux approches

Approche "brute - force" :

- Identifier des attributs d'intérêt
- repartir les attributs dans plusieurs relations

Approche modélisation conceptuelle :

- production d'un modèle conceptuel
- traduction en relationnel (automatique)
- potentiellement : ultérieur raffinement

Dans les deux cas on a besoin de :

- savoir détecter si un schéma relationnel a ou non de "bonnes propriétés"
- si ce n'est pas le cas :

  des techniques pour le reconduire à un "bon" schéma (*forme normale*)

Qualité d'un schéma relationnel

Quelles sont de "bonnes propriétés" d'un schéma relationnel?

Exemple:

Attributs relatifs à des vendeurs, produits, et fournitures

V#: numéro de vendeur

Vnom: nom du vendeur

Vville: ville du vendeur

P#: numéro du produit

Pnom: nom du produit

Pville: ville où le produit est stocké

Qte: quantité de produit fournie au vendeur

Qualité d'un schéma relationnel

- Un schéma relationnel possible : une seule relation "fourniture" avec tous les attributs

R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

- C'est une mauvaise modélisation! Pourquoi?

1) Redondance

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	...	...	...	...
3	MagicV	Paris	...	...	...	...
2	IdealB	Lyon	...	...	...	...
2	IdealB	Lyon	...	...	...	...

Ex: Vnom et Vville sont **déterminés** par V#, i.e.

si deux fournitures ont le même V# , elles ont aussi le même Vville et le même Vnom

On représente l'information que le vendeur 3 est MagicV et est basé à Paris, une fois pour chaque fourniture : **redondant**

Qualité d'un schéma relationnel

R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

C'est une mauvaise modélisation! Pourquoi?

1) Redondance

2) Anomalies de mise à jour

Vnom ou Vville pourrait être mis à jour dans une fourniture et pas dans une autre, ce qui donnerait une incohérence. Pour éviter cela : mise à jour plus couteuse.

3) Anomalies d'insertion

On ne peut pas stocker un vendeur s'il ne reçoit pas de fourniture

4) Anomalies de suppression

Si on supprime toutes les fournitures d'un vendeur, on perd toute l'info sur ce vendeur

Qualité d'un schéma relationnel

Solution : un "bon" schéma

Vendeur (V#, Vnom, Vville) Clef: V#

Produit (P#, Pnom, Pville) Clef: P#

Fourniture(V#, P#, Qte) Clef: V# P#

Plus d'anomalie! Comment y arriver?

La théorie de la normalisation des bd relationnelles nous donne:

- Des formes normales :

  - propriétés d'un schémas qui garantissent absence (ou réduction) de redondance, et des anomalies qui en dérivent

  - définies par rapport à un ensemble de contraintes (appelés dépendances)

- Des techniques de normalisation : passage d'un schéma arbitraire (mauvais) à un schéma en forme normale (typiquement par décomposition)

Contraintes d'intégrité et dépendances

Vers une definition formelle de “qualité” d’un schema relationnel

Contrainte d'intégrité sur un schéma

Une propriété que les instances du schéma sont censées satisfaire pour être valides

- e.g. contrainte de clef: NSS est une clef pour la relation Personne (NSS, nom, adresse)

- c’est la réalité qu’on modélise qui impose les contraintes

Le processus de modélisation doit identifier non-seulement les information à représenter, mais également les contraintes qui existent sur celles-ci

⇒ Notre point de départ : un schéma relationnel (potentiellement à raffiner) avec un ensemble de contraintes identifiées

Contraintes d'intégrité et dépendances

- Dépendances fonctionnelles : Une forme particulière de contraintes d'intégrité
- Exemple

  Schéma : R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

  Un ensemble de dépendances fonctionnelles qu'on peut raisonnablement supposer :

  V# → Vnom Vville
  P# → Pnom Pville
  V# P# → Qte

- Sémantique (intuition) : pour que une instance J de la relation R soit valide, J doit satisfaire :

  - si deux tuples dans J ont la même valeur de V#

    alors ils ont la même valeurs de Vnom et de Vville

  - si deux tuples dans J ont la même valeur de P#

    alors ils ont la même valeurs de Pnom et de Pville

  - si deux tuples dans J ont la même valeur de V# et la même valeur de P#

    alors ils ont la même valeurs de Qte

Dépendances fonctionnelles

Exemple

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	322	manteau	Lille	2
1	StarV	Rome	546	veste	Rome	1
3	MagicV	Paris	322	manteau	Lille	5
2	IdealB	Lyon	145	jupe	Paris	7
2	IdealB	Lyon	234	jupe	Lille	1

J satisfait V# → Vnom Vville et P# → Pnom Pville

Dépendances fonctionnelles

Exemple

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	322	manteau	Lille	2
1	StarV	Rome	546	veste	Rome	1
3	MagicV	Paris	322	manteau	Lille	5
2	IdealB	Lyon	145	jupe	Paris	7
2	IdealB	Lyon	234	jupe	Lille	1

J viole V# P# → Qte

Dépendances fonctionnelles (DF)

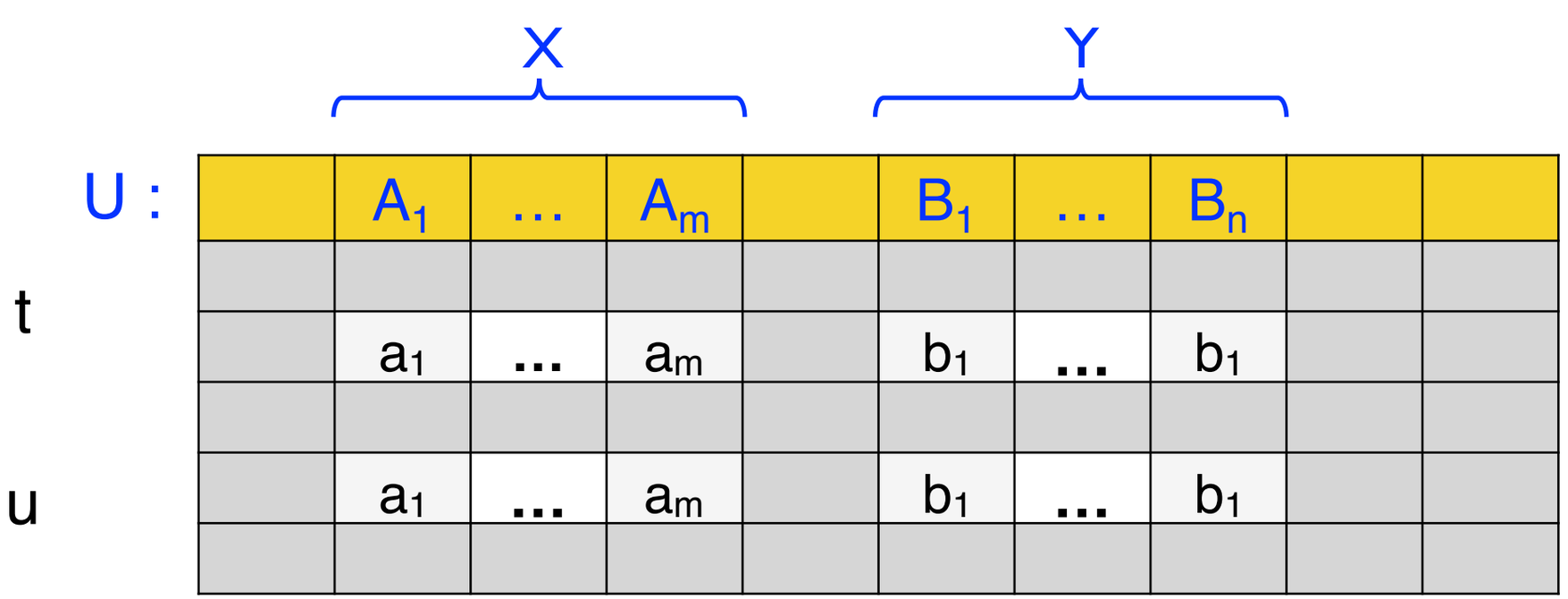
Soit R(U) un schéma de relation avec U : ensemble d'attributs

Une dépendance fonctionnelle est une expression : $X \rightarrow Y$, avec $X, Y \subseteq U$

Une instance J de R(U) satisfait $X \rightarrow Y$ si pour toute paire de tuples t, u dans J

$$t[X] = u[X] \Rightarrow t[Y] = u[Y]$$

(si t et u sont en accord sur X alors t et u sont en accord sur Y)



J satisfait un ensemble F de DF, si J satisfait chaque DF dans F

Dépendances fonctionnelles (DF)

Soit R(U) un schéma de relation avec U : ensemble d'attributs

Une dépendance fonctionnelle est une expression : $X \rightarrow Y$, avec $X, Y \subseteq U$

Une instance J de R(U) satisfait $X \rightarrow Y$ si pour toute paire de tuples t, u dans J

$$t[X] = u[X] \Rightarrow t[Y] = u[Y]$$

(si t et u sont en accord sur X alors t et u sont en accord sur Y)

Remarque sur la notation.

Par la suite un ensemble d'attributs {A1,…,An} sera dénoté par A1…An

Donc A1…An → B1..Bn dénotera la DF {A1, …,An} → {B1, …, Bn}

Dépendances fonctionnelles et modélisation E/R

S'il y a eu un phase de modélisation E/R, les contraintes d'identification, les associations, les contraintes de cardinalité et les contraintes externes du schéma E/R impliquent des DF sur le schéma relationnel

Rappel : exemple de mauvaise modélisation

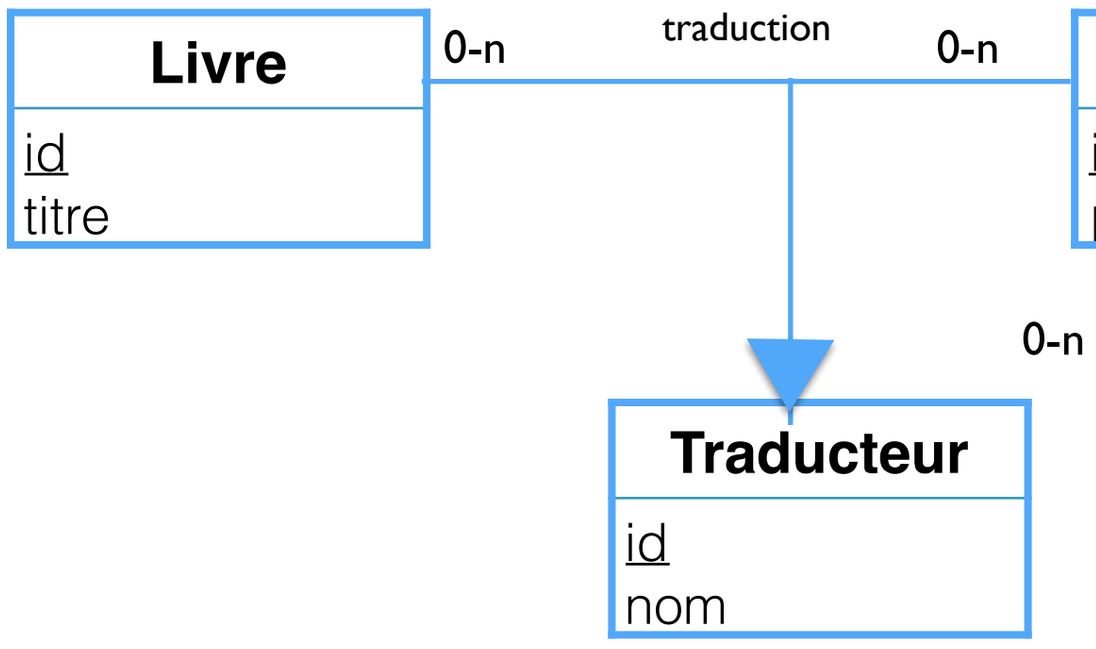


Schéma relationnel associé :

Livre (id-livre, titre)

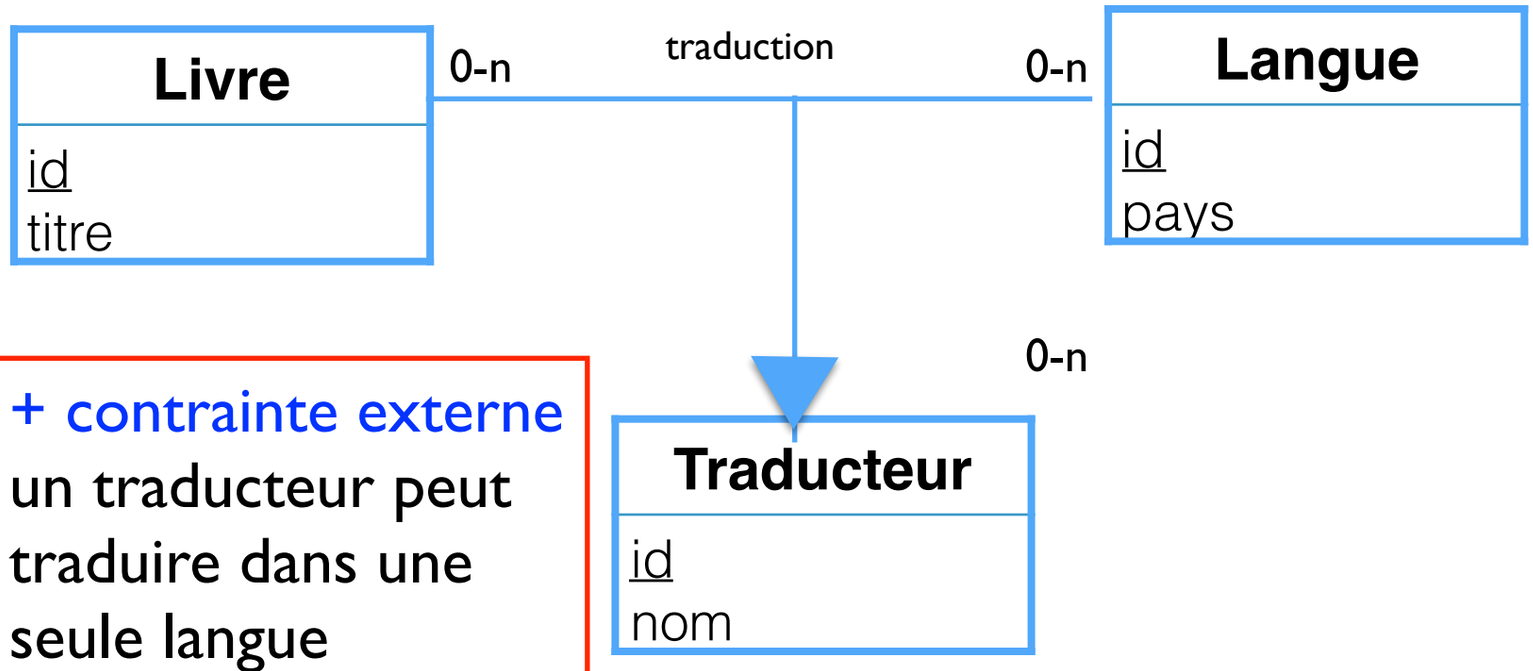
Langue (id-langue, pays)

Traducteur (id-trad, nom)

Traduction (id-livre, id-langue, id-trad, date)

+ contrainte externe
un traducteur peut traduire dans une seule langue

Dépendances fonctionnelles et modélisation E/R



+ contrainte externe
un traducteur peut traduire dans une seule langue

Schéma relationnel associé :

Livre (id-livre, titre)

Langue (id-langue, pays)

Traducteur (id-trad, nom)

Traduction (id-livre, id-langue, id-trad, date)

DF sur le schéma relationnel:

sur Livre : id-livre → titre — par la contrainte d'id. sur l'entité Livre

sur Langue : id-langue → pays — par la contrainte d'id. sur l'entité Langue

sur Traducteur : id-trad → nom — par la contrainte d'id. sur l'entité Traducteur

sur Traduction : id-livre id-trad id-langue → date — par l'association Traduction

id-livre id-langue → id-trad — par la contrainte d'intégrité fonctionnelle

id-trad → id-langue — par la contrainte externe

Dépendances fonctionnelles et qualité du schéma

- Un schéma relationnel est "bon" ou pas, selon les contraintes qui y sont associées

  – **Exemple 1 :** **R** (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

  redondances et anomalies dues par exemple à la dépendance fonctionnelle

  V# →Vnom, Vville

  – Exemple 2. Traduction (id-livre, id-trad, id-langue, date**)**

  redondances et anomalies dues à la dépendance fonctionnelle

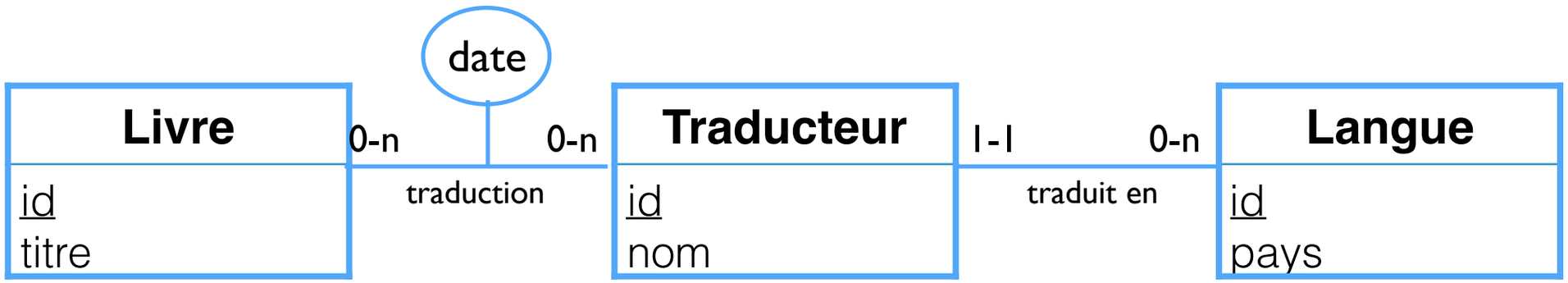
  id_trad → id_langue

Traduction

id-livre	id-trad	id-langue	date
3	5	44	02/2006
1	5	44	10/2001
2	5	44	03/1984
2	3	22	09/1998

Dépendances fonctionnelles et qualité du schéma

- Une meilleure modélisation conceptuelle produit un schema relationnel qui n'a pas ces problèmes :



Livre (<u>id-livre</u>, titre) id-livre → titre

Langue (<u>id-langue</u>, pays) id-langue → pays

Traducteur (<u>id-trad</u>, nom, id-langue) id-trad → nom , id-trad → id-langue

Traduction (<u>id-livre, id-trad</u>, date) id-livre id-trad → date

Traduction

id-livre	id-trad	date
3	5	02/2006
1	5	10/2001
2	5	03/1984
2	3	09/1998

Traducteur

id-trad	nom	id-langue
5	Dupont	44
3	Blanc	22

Qualité d'un schéma relationnel : formes normales

- *Formes normales* :
  - formalisent la notion de "qualité" d'un schéma par rapport à un ensemble de contraintes
  - viennent avec des techniques de *normalisation* : "corriger" un "mauvais" schéma pour le reconduire à un schema en forme normale
  - normalisation : transformation qui opère directement sur le modèle relationnel (sans revenir sur la modélisation conceptuelle)
  - *but : éliminer les problèmes de redondance et les anomalies*

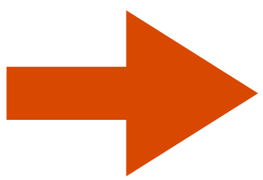
Livre (id-livre, titre)

Langue (id-langue, pays)

Traducteur (id-trad, nom)

Traduction (id-livre, id-langue, id-trad, date)

normalisation



Livre (id-livre, titre)

Langue (id-langue, pays)

Traducteur (id-trad, nom, id-langue)

Traduction (id-livre, id-trad, date)

Formes normales

- Plusieurs formes normales proposées pour les schéma relationnels :
  - première, deuxième, troisième, Boyce-Codd, ...

- *Forme normale de Boyce-Codd* (FNBC): la plus restrictive pour un schema relationnel par rapport à un ensemble de dependences fonctionnelles

- Sa definition nécessite d'introduire un peu de terminologie sur les dependences fonctionnelles…

Vers la definition des formes normales: implication de DF

- Soit R(U) un schema de relation (U ensemble d'attributs) et F un ensemble de DF sur U
  - ex. R(ABC), F= { A → B, B → C}
- Les DF données peuvent impliquer d'autre DF additionnelles

- Exemple: A → B et B → C impliquent A → C

C'est à dire
toute instance de relation qui satisfait A → B et B → C
satisfait également A → C

- Un autre exemple :

A→C, BC→D, AD→E implique AB → E

Implication de DF

Définition.

Un ensemble F de DF implique une autre DF $X \rightarrow Y$

si toute instance de relation qui satisfait F satisfait également $X \rightarrow Y$

Notation pour " F implique $X \rightarrow Y$" : $F \models X \rightarrow Y$

Toutes les DF impliqués par F: $F^+ = \{ X \rightarrow Y \mid F \models X \rightarrow Y \}$

Exemple : $\{ A \rightarrow B,\ B \rightarrow C \}^+$ inclut les dépendances suivantes :

$A \rightarrow B$, $B \rightarrow C$, $A \rightarrow C$ $AB \rightarrow C$, ...

mais aussi des DF "triviales" (i.e. satisfaites par toute instance)

$A \rightarrow A$, $AB \rightarrow A$, $ABC \rightarrow A$, $B \rightarrow B$, $AB \rightarrow B$, etc.

Clefs et super-clefs

Soit R(U) un schéma de relation et F un ensemble de DF sur U.

- Super-clef : $X \subseteq U$ tel que $F \models X \rightarrow U$

  X détermine tous les attributs de R

- Clef (ou clef candidate) : $X \subseteq U$ tel que X est une super-clef et il n'existe pas

  $Y \subsetneq X$ tel que Y est une super-clef

Exemple : R(ABC) $F = \{A \rightarrow B, B \rightarrow C\}$

Super-clefs : A, AB, AC, ABC

Clef : A (la seule)

Forme normale de Boyce-Codd (FNBC)

Forme normale de Boyce-Codd

Un schéma de relation R(U) est en FNBC par rapport à un ensemble de DF F sur R(U),

ssi pour tout $X \rightarrow A \in F^+$ tel que $A \notin X$, X est une super-clef pour R

C'est à dire les seules DF non-triviales sont celles induites par des super-clefs

FNBC : Exemple 1

Exemple des traductions :

schema 1 (venant de l'association ternaire) :

Livre (<u>id-livre</u>, titre) {id-livre → titre}

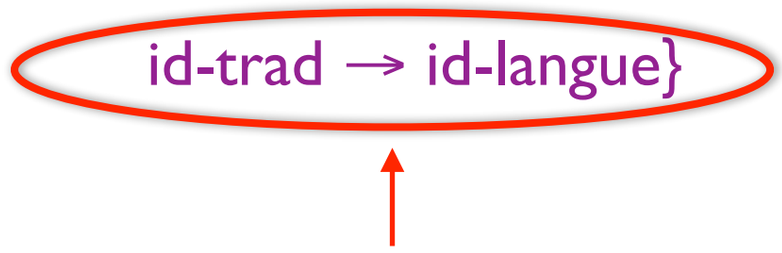
Langue (<u>id-langue</u>, pays) {id-langue → pays}

Traducteur (<u>id-trad</u>, nom) {id-trad → nom}

Traduction (<u>id-livre, id-trad</u>, id-langue, date) : {id-livre id-trad id-langue → date

id-livre id-langue → id-trad

id-trad → id-langue}



La relation Traduction n'est pas en FNBC par rapport à ses DF

FNBC : Exemple 1

Exemple des traductions

schéma 2 (venant des deux associations binaires) :

Livre (<u>id-livre</u>, titre) {id-livre → titre}

Langue (<u>id-langue</u>, pays) {id-langue → pays}

Traducteur (<u>id-trad</u>, nom, id-langue) {id-trad → nom , id-trad → id-langue}

Traduction (<u>id-livre, id-trad</u>, date) {id-livre id-trad → date}

Chaque relation est en FNBC par rapport à ses DF

FNBC : Exemple 2

schema 1

R(V#, P#, Vnom, Pnom, Vville, Pville, Qte)

F = V#→Vnom Vville

P#→Pnom Pville

V# P#→Qte

R n'est n'est pas en FNBC par rapport à F

En effet ni V# ni P# ne sont super-clefs pour R (ni $V\#^+$ ni $P\#^+$ ne déterminent Qte)

FNBC : Exemple 2

schema 2

- Vendeur (V#, Vville, Vnom) V#→Vnom Vville FNBC
- Produit (P#, Pville, Pnom) P#→Pnom Pville FNBC
- Fourniture (V# P# Qte) V# P# → Qte FNBC

(V# est une super-clef pour *Vendeur*

P# est une super-clef pour *Produit*

V# P# est une super-clef pour *Fourniture*)

FNBC =

absence de redondance (et anomalies associées)
dues au DF

Une vue (simplifiée) de la modélisation de schéma relationnels avec des DF

1. - Choisir les attributs d'intérêt U et produire un schéma de relation R(U)

- Alternative : utiliser une étape de modélisation conceptuelle (e.g E/R) et traduire en un schéma relationnel $R_1(U_1)$ ... $R_k(U_k)$.

2. Spécifier toutes les DF pour R (ou pour R_1.. R_k)

- rappel : un schéma E/R peut exprimer des DF par les contraintes d'identification, les associations, les contraintes de cardinalité et les contraintes externes

3. Si R n'est pas dans une forme normale souhaitée (FNBC par exemple)*

- normaliser R (ou chaque Ri)

- alternative :"corriger" la modélisation conceptuelle et revenir à l'étape 2.

* **Remarque.** Si on est passé par une étape de modélisation conceptuelle, R_1..R_k a des chances d'être déjà en forme normale, bien que ce ne soit pas garanti.

Normalisation

- Donné R(U) , F

  on souhaite “normaliser” R(U) par rapport à F
- L’algorithme de normalisation depend de la forme normale souhaitée
- Dans tous les cas : normalisation par decomposition
- Avant d’étudier ces algorithmes (ce que vous ferez en M1) on a besoin de comprendre l’implication de DF

Implication de DF : Axiomes de Armstrong

Trois règles d'inférence (dont la correction est facile à vérifier) :

Pour un schéma de relation R(U) , et $X, Y, Z \subseteq U$

1) *Transitivité :* $\{ X \rightarrow Y,\ Y \rightarrow Z \} \models X \rightarrow Z$

2) *Augmentation :* $X \rightarrow Y \models X\,Z \rightarrow Y\,Z$

3) *Réflexivité :* $\models XY \rightarrow X$ (appelée DF triviale)

Ces règles ne sont pas seulement correctes , il s'agit d'axiomes , i.e

$F \models X \rightarrow Y$ ssi

$X \rightarrow Y$ peur être dérivé de F par application successives des trois règles ci-dessus

Implication de DF : d'autres règles

Plusieurs autres règles correctes, mais pas nécessaires pour former des axiomes (dérivables des axiomes) :

Union : $\{ X \rightarrow Y,\ X \rightarrow Z \} \models X \rightarrow YZ$

Séparation : $X \rightarrow YZ \models X \rightarrow Y$

et d'autres encore ...

Remarque : pour simplifier la notation, on peut omettre {…} pour les ensembles de DF :

$X1 \rightarrow Y1, \ldots, Xn \rightarrow Yn$ dénote l'ensemble de DF : $\{ X1 \rightarrow Y1, \ldots, Xn \rightarrow Yn \}$

Implication de DF : équivalence

Ensembles équivalents de DF

Soit F et G deux ensembles de DF sur R(U)

F est équivalent à G si $F \models G$ et $G \models F$

(i.e. ssi $F^+ = G^+$)

On peut toujours remplacer un ensemble de DF avec un ensemble équivalent

Remarque : Par les règles d'*Union* , *Séparation* et *Réflexivité*:

- $X \rightarrow A_1, \ldots A_n$ équivalent à $X \rightarrow A_1, \ldots, X \rightarrow A_n$
- $XY \rightarrow YZ$ équivalent à $XY \rightarrow Z$

Implication de DF

Question principale d'un point de vu algorithmique:

Comment vérifier si un ensemble F de DF implique une DF $X \rightarrow Y$?

Ou bien, par les équivalences du slide précédent :

Comment vérifier si un ensemble F de DF implique une DF $X \rightarrow A$?

(X :ensemble, A: attribut)

Implication de DF : Clôture d'un ensemble d'attributs

Vérifier si X→A est impliqué par un ensemble F de DF :

- on pourrait utiliser les axiomes de Armstrong (et les autres règles dérivables) pour essayer de dériver X→A à partir de F
- souvent plus utile de penser en termes de clôture de X

Clôture de X (par rapport à F): l'ensemble d'attributs "déterminés" par X

Définition.

La clôture d'un ensemble d'attributs X par rapport à un ensemble F de DF est

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

Exemple.

R (ABCDE) F = { AB→C, C→ D, E→D } $(AB)^+$ = ABCD

Implication de DF : Clôture d'un ensemble d'attributs

Vérifier si $X \rightarrow A$ est impliqué par un ensemble F de DF :

- on pourrait utiliser les axiomes de Armstrong (et les autres règles dérivables) pour essayer de dériver $X \rightarrow A$ à partir de F
- souvent plus utile de penser en termes de clôture de X

Clôture de X (par rapport à F): l'ensemble d'attributs "déterminés" par X

Définition.

La clôture d'un ensemble d'attributs X par rapport à un ensemble F de DF est

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

Caractérisation : $F \models X \rightarrow A \quad \text{iff} \quad A \in X^+$

Vérifier si $X \rightarrow A$ est impliqué par F : se réduit à calculer une clôture

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples en accord sur X=AB

sont aussi en accord sur :

A B

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type “accessibilité”

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu’une instance de R satisfait F et que deux tuples en accord sur X=AB

alors sont aussi en accord sur :

Ⓐ B (A→C)

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples en accord sur X=AB

sont aussi en accord sur :

A B C

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples en accord sur X=AB

sont aussi en accord sur :

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples en accord sur X=AB

sont aussi en accord sur :

A B C D

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples en accord sur X=AB

sont aussi en accord sur :

Ⓐ B C Ⓓ (AD→E)

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples en accord sur X=AB

sont aussi en accord sur :

A B C D E

Exemple de calcul de la clôture

La clôture est calculée par un simple algorithme de type "accessibilité"

Exemple R(ABCDEF) F = {A→C, BC→D, AD→E} X = AB

On calcule X^+ de façon incrémentale.

Idée : supposer qu'une instance de R satisfait F et que deux tuples en accord sur X=AB

sont aussi en accord sur :

A B C D E

X^+ = ABCDE

Calcul de la clôture d'un ensemble d'attributs

Algorithme général:

Soit F un ensemble de DF sur R(U) et $X \subseteq U$.

L'algorithme suivant calcule la clôture X^+ de X par rapport à F

$X_c := X$

tant que il existe $V \rightarrow Z$ dans F tel que $V \subseteq X_c$ et $Z \not\subseteq X_c$

$X_c := X_c \cup Z$

renvoyer X_c

X_c grandit à chaque itération

Comme U est fini, l'algorithme termine en au plus |U| itérations

Décomposition d'un schéma de relation

L'outil indispensable pour arriver à une forme normale

- Soit R(U) un schéma de relation
- Une décomposition de R(U) est un ensemble { $R_1(S_1)$, …, $R_k(S_k)$ } de schémas de relation tels que:

$$U = \bigcup_{i=1}^{k} Si$$

- Exemple

{Vendeur (V#, Vnom, Vville),
Produit (P#, Pnom, Pville),
Fourniture(V#, P#, Qte) }

est une décomposition de
R (V#, Vnom, Vville, P#, Pnom, Pville, Qte)

Propriétés d'une décomposition

- On ne peut pas décomposer arbitrairement

- Conditions pour une décomposition "raisonnable" :

  - Décomposition sans perte d'information

  - Décomposition sans perte de dépendances fonctionnelles

Décomposition sans perte d'information

Idée : Si on remplace R (V#, Vnom, Vville, P#, Pnom, Pville, Qte) par {Vendeur, Produit, Fournitures}

notre BD, au lieu de stocker une instance J de R stockera ses projections

$\pi_{V\#, Vnom, Vville}(J)$ $\pi_{P\#, Pnom, Pville}(J)$ $\pi_{V\#, P\#, Qte}(J)$

J

V#	Vnom	Vville	P#	Pnom	Pville	Qte
3	MagicV	Paris	5	jupe	Paris	5
3	MagicV	Paris	6	veste	Lille	2
2	IdealB	Lyon	12	manteau	Lyon	1
2	IdealB	Lyon	13	jupe	Paris	1

$\pi_{V\#, Vnom, Vville}(J)$

V#	Vnom	Vville
3	MagicV	Paris
2	IdealB	Lyon

$\pi_{P\#, Pnom, Pville}(J)$

P#	Pnom	Pville
5	jupe	Paris
6	veste	Lille
12	manteau	Lyon
13	jupe	Paris

$\pi_{V\#, P\#, Qte}(J)$

V#	P#	Qte
3	5	5
3	6	2
2	12	1
2	13	1

Décomposition sans perte d'information

Idée

- La décomposition doit garantir que pour toute instance J de R, les projections de J contiennent la "même information" que J

- C'est à dire on doit pouvoir reconstruire une instance J de R à partir de ses projections

- Comment tenter de reconstruire l'instance à partir de ses projections?
  Jointure naturelle

$$\pi_{V\#,Vnom,Vville}(J) \bowtie \pi_{P\#,Pnom,Pville}(J) \bowtie \pi_{V\#,P\#,Qte}(J)$$

Décomposition sans perte d'information

Rappel. Jointure naturelle de deux instances de relation:
I avec ensemble d'attributs X, et J avec ensemble d'attributs Y

$I \bowtie J$

retourne l'ensemble des tuples t sur attributs $X \cup Y$ telles que $t[X] \in I$ et $t[Y] \in J$

X= {A, B}

Y= {B,C}

I

A	B
1	2
4	2
6	6
7	7

J

B	C
2	3
2	5
9	1
8	8

$I \bowtie J$

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

Décomposition sans perte d'information

Propriété souhaitée pour notre décomposition :

$$J = \pi_{V\#, Vnom, Vville}(J) \bowtie \pi_{P\#, Pnom, Pville}(J) \bowtie \pi_{V\#, P\#, Qte}(J)$$

pour toute instance valide J de R

Est cela vrais?

Dans l'exemple de J donné, oui. Mais pour d'autres J valides?

Intuitivement, Oui : puisque en partant de la fourniture (V#, P#, Qte)

- V# nous permet de récupérer toutes les infos sur un unique vendeur
  (grâce à la DF V# → Vnom Vville)
- P# nous permet de récupérer toutes les infos sur un unique produit

  (grâce à la DF P# → Pnom Pville)

(une procédure plus rigoureuse pour ce test plus loin)

la propriété de décompositions sans perte d'information dépend

des dépendances fonctionnelles

Décomposition sans perte d'information (*lossless join*)

Définition.

Soir R(U) une schéma de relation et F un ensemble de DFs sur R.

Une décomposition { $R_1(S_1)$, …, $R_k(S_k)$ } de R est

sans perte d'information par rapport à F

ssi, pour toute instance J de R qui satisfait F,

$$J = \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$$

Un exemple de décomposition avec perte d'information

R(A, B, C) décomposition : { R1(A, B), R2(B, C) }

F= {AB → C}

Il existe une instance J de R qui satisfait F, mais qu'on ne peut pas reconstruire à partir de ses projections:

J

A	B	C
1	2	3
4	2	5

$\pi_{AB}(J)$

A	B
1	2
4	2

$\pi_{BC}(J)$

B	C
2	3
2	5

$\pi_{AB}(J) \bowtie \pi_{BC}(J)$

A	B	C
1	2	3
4	2	5
1	2	5
4	2	3

Décomposition sans perte d'information (*lossless join*)

Pour une instance J arbitraire, quelle est la connexion entre

J et $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$?

Décomposition sans perte d'information (*lossless join*)

Pour une instance J arbitraire, quelle est la connexion entre

J et $\pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$?

- Pour tout J, $J \subseteq \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$

  Par la définition de jointure naturelle et projection :

  $t \in J \Rightarrow t[Si] \in \pi_{Si}(J)$ pour tout i $\Leftrightarrow t \in \pi_{S1}(J) \bowtie \pi_{S2}(J) \bowtie .... \bowtie \pi_{Sk}(J)$

- le seule problème est donc que les jointures peuvent générer des tuples en plus (voir exemple précédent)

- Mais J n'est pas arbitraire : J satisfait des DFs, cela peut garantir l'inclusions inverse dans certains cas

- Tester si un decomposition est sans perte d'information : un simple algorithme existe

Décomposition FNBC sans perte d'information

- Chaque schéma de relation a une décomposition en un ensemble de schémas de relations FNBC sans perte d'information

cette décomposition ne préserve pas toujours les DF

- **Exemple**

R (Ville, CP, Rue, Numero)

F = Ville, Rue, Numero → CP, CP → Ville

– R n'est pas en FNBC : CP n'est pas une super-clef

– R n'a pas de décomposition en schémas FNBC, qui préserve F :

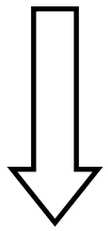
  - Soit ρ une decomposition qui ne contient pas R,
    aucune relation de ρ peut avoir une DF locale de la forme X → CP
    (sinon Ville, Rue, Numero $\subseteq$ X)
  - $\Rightarrow$ le DF locales n'impliquent pas Ville, Rue, Numero → CP

Troisième Forme Normale (3NF)

Problème avec FNBC:

Tous les schémas de relation ne peuvent pas être décomposés en un ensemble de schémas FNBC qui préserve *à la fois* l'information *et* les dépendances fonctionnelles

cf. exemple R(Ville, Rue, Numero, CP)



<u>Troisième forme normale (3NF)</u>

Un schéma de relation R est en troisième forme normale par rapport à un ensemble F de DF sur R si pour tout $X \rightarrow A \in F^+$ tel que $A \notin X$

soit X est-une super-clef de R **soit** A **appartient à une clef de** R

3NF est plus "faible" que FNBC :

FNBC implique 3NF mais pas vice-versa

3NF admet donc une certaine forme de redondance et des anomalies, mais considérées acceptables

Troisième Forme Normale - Exemple

Exemple

R (Ville, CP, Rue, Numero)

F = Ville Rue Numero → CP, CP → Ville

Violation de FNBC: CP → Ville

Néanmoins, *Ville* appartient à la clef *Ville Rue Numero*

donc R est en 3NF par rapport à F

R est en 3NF mais pas en FNBC

Conclusion

- Cours de BD avancé de M1 : algorithmes pour calculer l'implication des dépendances fonctionnelles, algorithmes de décomposition pour produire des schémas en forme normale (que nous ne verrons pas dans ce cours)
- En fait, ces algorithmes sont implémentés dans la plupart des outils d'aide à la conception de schéma

- Pour l'instant (e.g., pour le projet), vous pouvez vous contenter de vérifier que le schéma relationnel auquel vous arrivez est bien en FNBC, ou à défaut, en 3FN
- Pour cela, il est impératif de bien spécifier explicitement toutes les dépendances fonctionnelles impliquées par le cahier des charges que vous souhaitez modéliser (via le diagramme E-R ou bien à défaut, en tant que contraintes externes)