INFO0009-2
Bases de données
(organisation générale)

Samuel Hiard

S.Hiard@uliege.be

URL :
https://services.montefiore.uliege.be/verif/cours/bd/exercices.html

Chapitre 1

Introduction

Définition

Base de données (BD) : ensemble de données conservées à long terme sur un ordinateur.

Caractéristiques :

- Grandes quantités et persistance ;
- La base de données doit pouvoir être interrogée et modifiée aisément ;
- Gestion sûre d'accès fréquents et multiples (transactions).

Système de gestion de base de données ou SGBD (DBMS) : programme *générique* qui permet la définition, la mise en œuvre et l'exploitation d'une base de données.

Les données

La réalisation de systèmes génériques de bases de données nécessite que l'on fixe un *cadre* (ou *modèle*) permettant de définir le *type* ou *schéma* (par opposition à l'*instance*) des données à traiter.

Modèle (cadre de définition) : concepts utilisés pour structurer et définir les données.

Schéma (type, plan) : schéma d'organisation de la BD ; description de l'organisation des données et de leur type. Il ne varie pas au cours de l'utilisation de la base de données.

Instance (extension) : contenu réel de la base de données à un moment fixé.

Analogie (données utilisées dans les langages de programmation)

Cadre de définition : Déclarations de type de données permises dans le langage utilisé.

Schéma : Ensemble particulier de déclarations utilisé dans un programme.

Exemple : `integer array A[1..n, 1..m]`

Instance : Valeurs des données à un moment donné de l'exécution du programme.

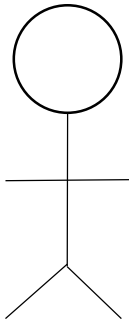
Exemple : $A : \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$

Le modèle entité-relation (ou entité-association)

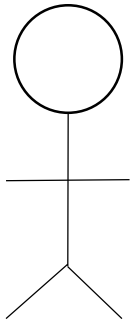
Le modèle entité-relation (*entity-relationship*) est un cadre de définition général du type du contenu potentiel d'une base de données. Il utilise deux types génériques de base :

- les *ensembles d'entités* : ensembles d'objets de même structure (à propos desquels on enregistre les mêmes informations),
- les *relations* (ou *associations*) entre ces ensembles.

Le modèle entité-relation: pourquoi?

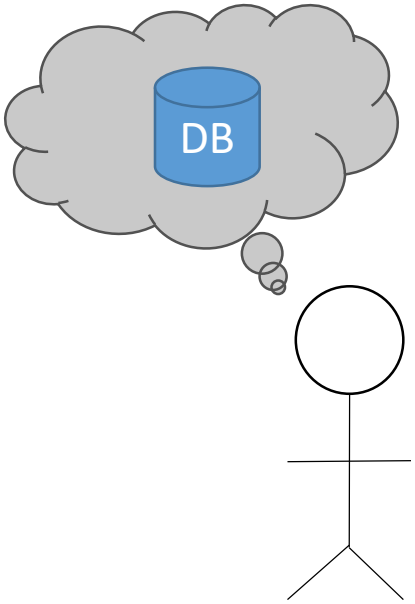


Client

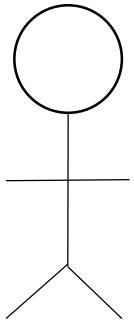


Commercial
Chef de projet
Chef d'équipe
Développeur

Le modèle entité-relation: pourquoi?

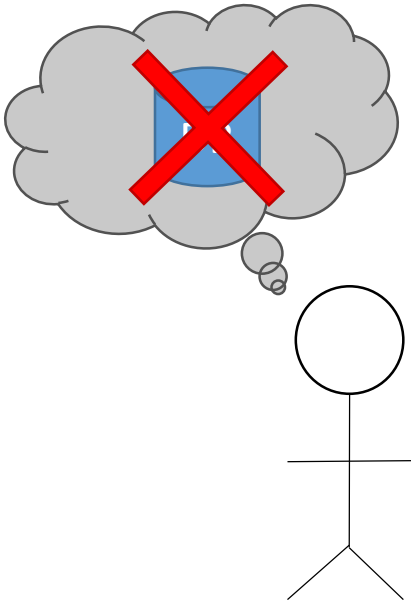


Client

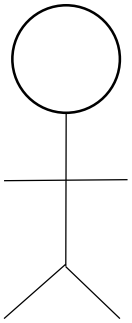


Commercial
Chef de projet
Chef d'équipe
Développeur

Le modèle entité-relation: pourquoi?

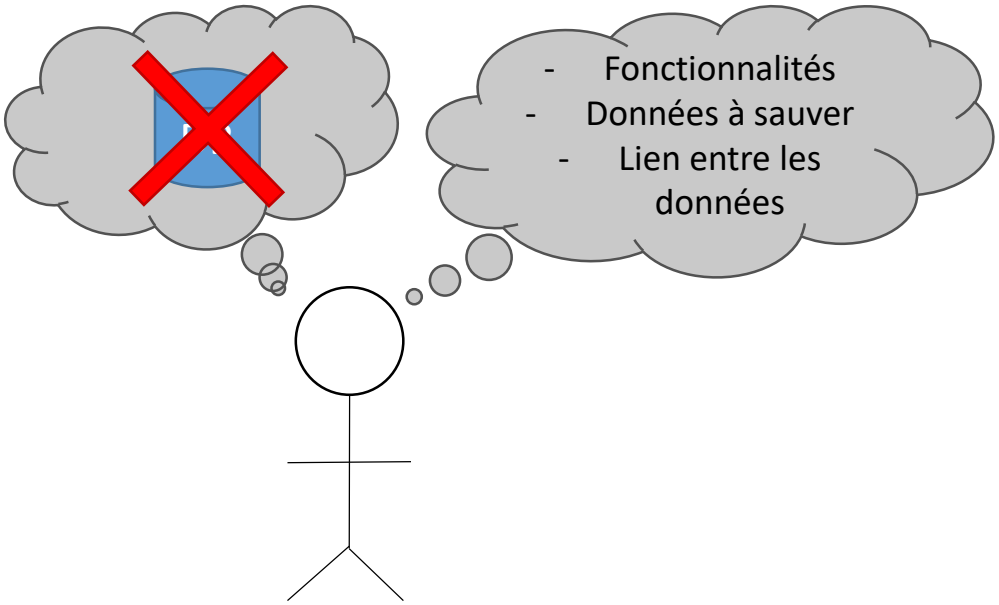


Client



Commercial
Chef de projet
Chef d'équipe
Développeur

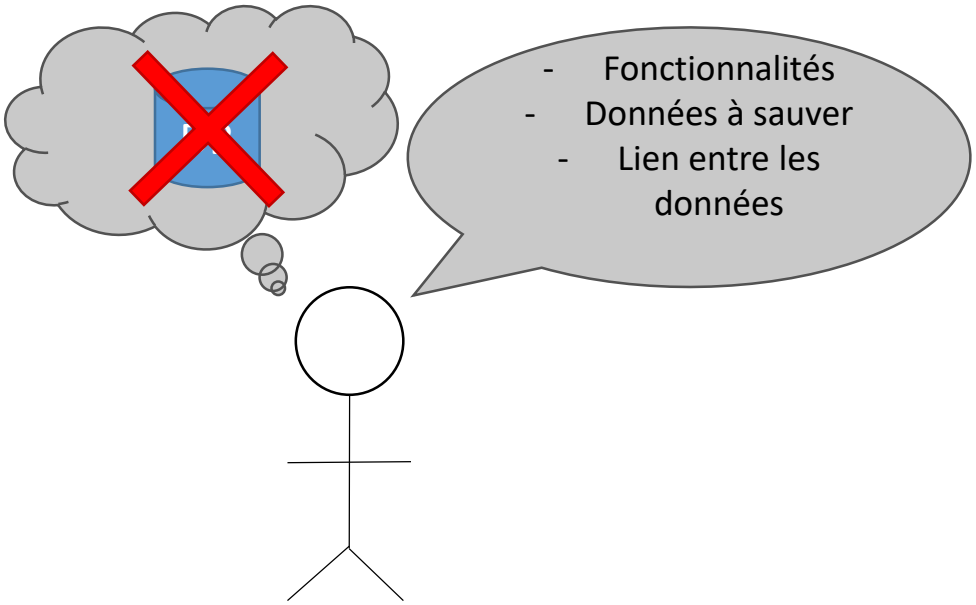
Le modèle entité-relation: pourquoi?



Client

Commercial
Chef de projet
Chef d'équipe
Développeur

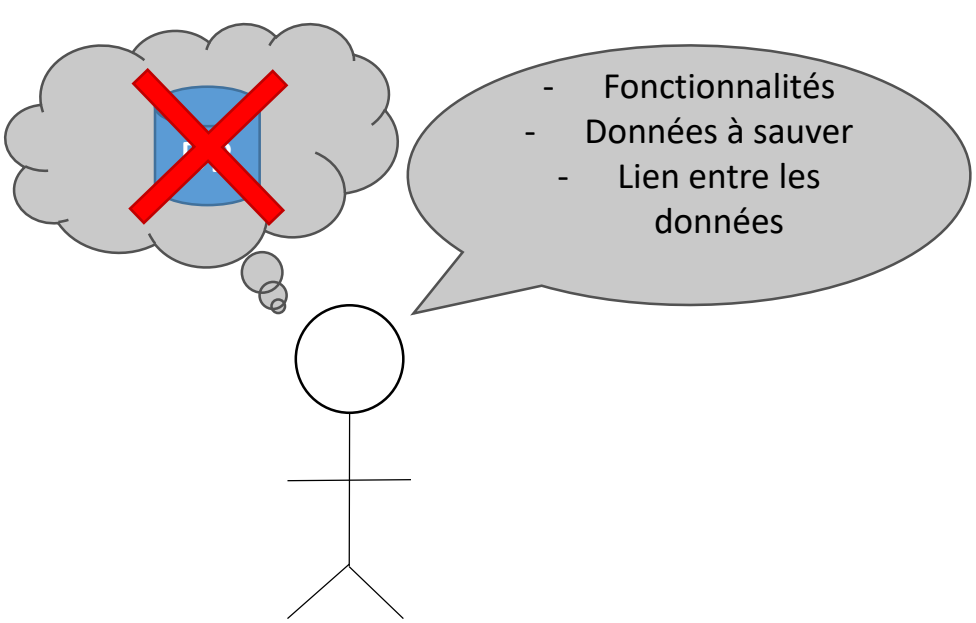
Le modèle entité-relation: pourquoi?



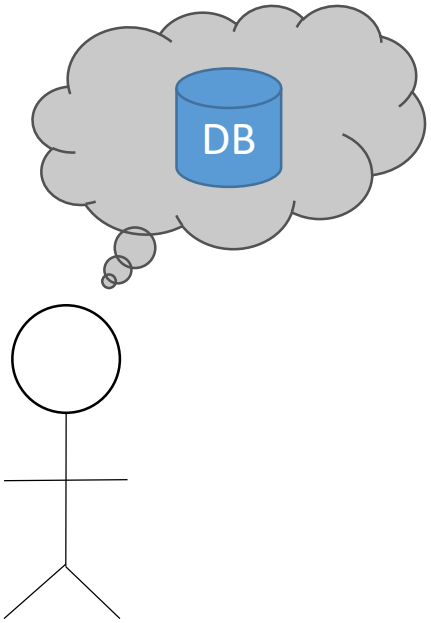
Client

Commercial
Chef de projet
Chef d'équipe
Développeur

Le modèle entité-relation: pourquoi?

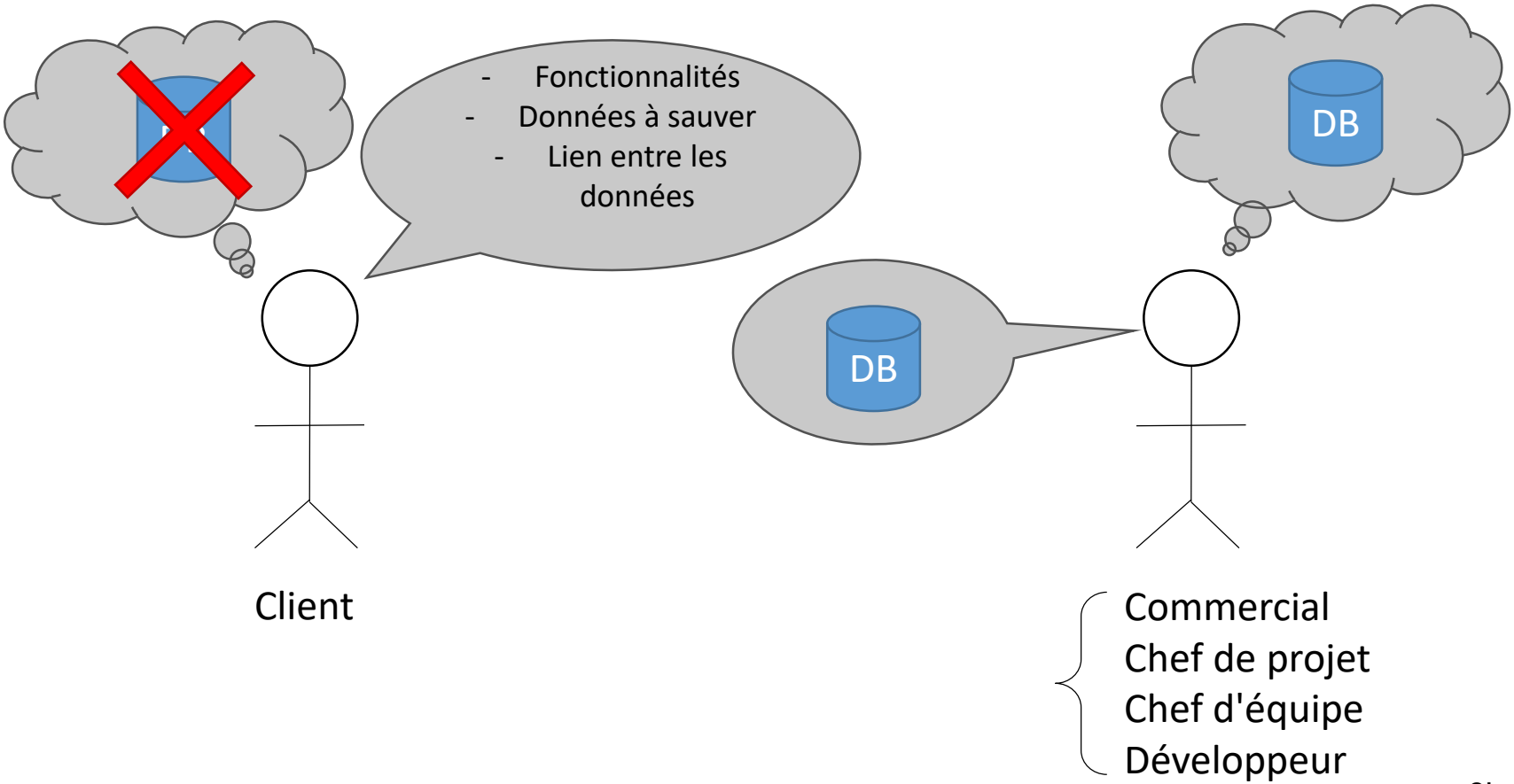


Client



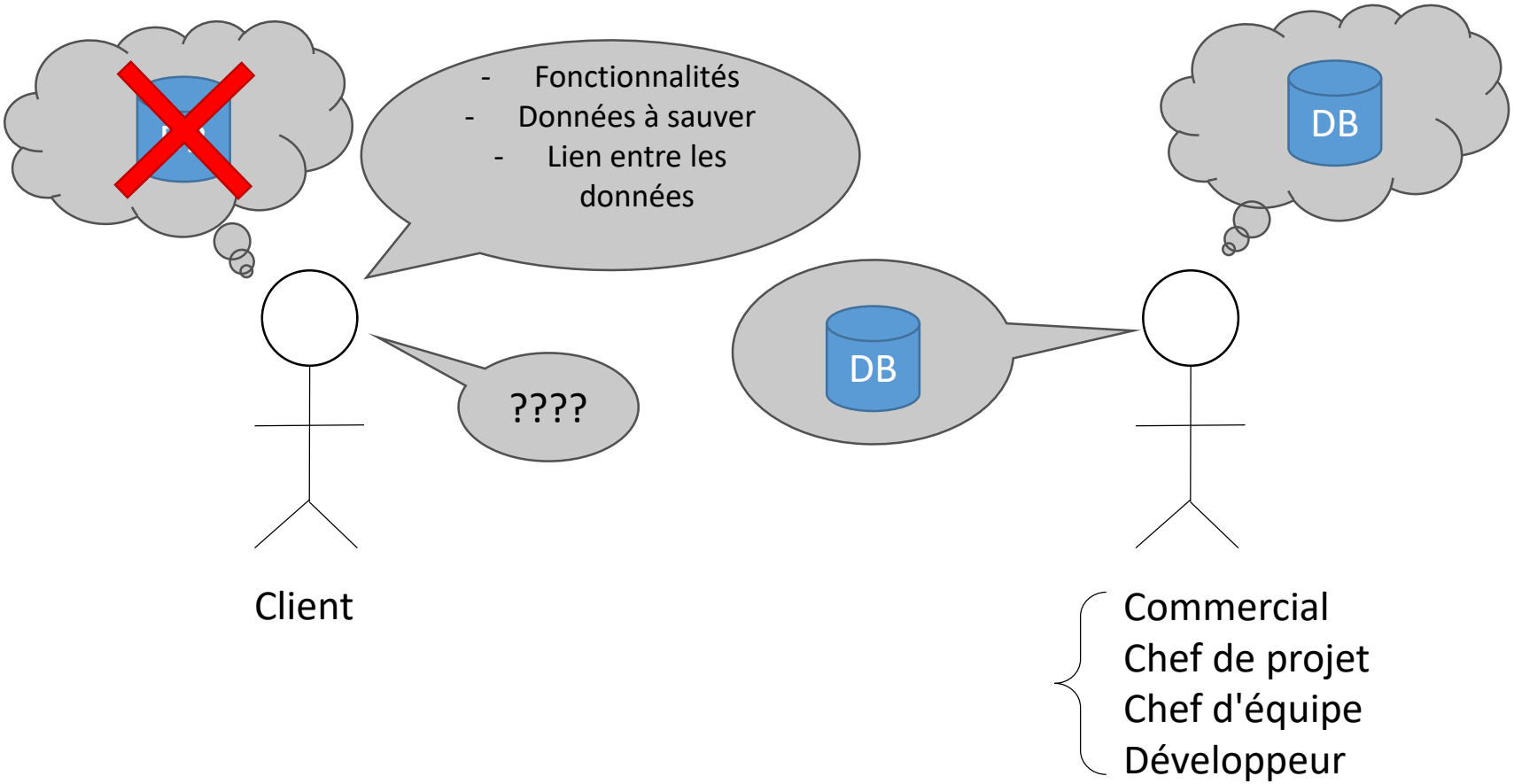
Commercial
Chef de projet
Chef d'équipe
Développeur

Le modèle entité-relation: pourquoi?

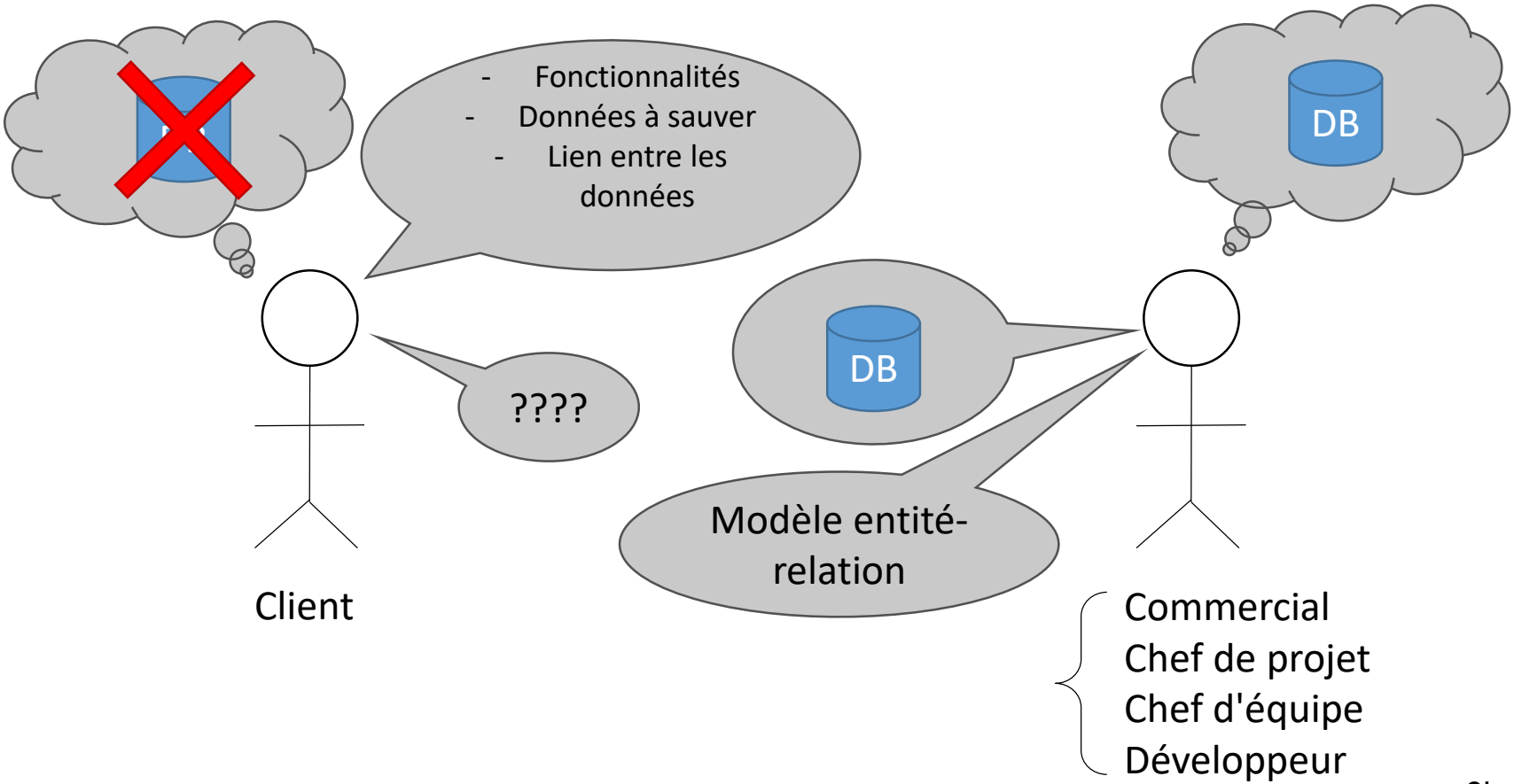


6h

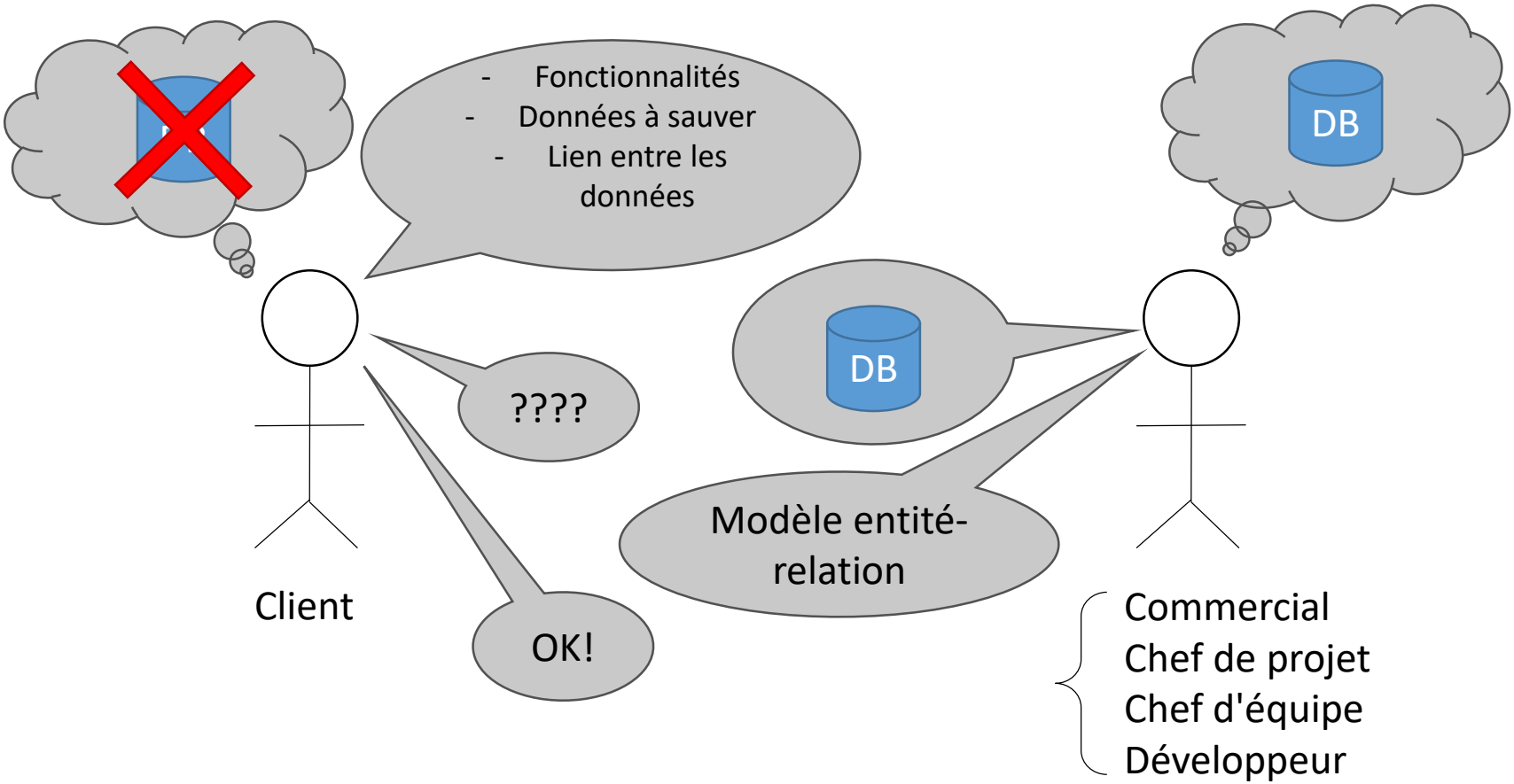
Le modèle entité-relation: pourquoi?



Le modèle entité-relation: pourquoi?



Le modèle entité-relation: pourquoi?



Les entités

Une *entité* (*entity*) est un objet (abstrait ou concret) au sujet duquel on conserve de l'information dans la base de données. Il faut qu'une entité soit individualisable (qu'elle puisse être distinguée d'une autre entité).

Exemples :

- entités concrètes : une personne, une voiture, une maison, un local, ...

- entités abstraites : un trajet, un cours, un horaire, un nombre, ...

Note : Le modèle entité-relation est général, il laisse toute liberté sur ce que peut être une entité. Toutefois, du point de vue des base de données, une entité n'a de sens que si l'on peut identifier les informations qui seront conservées à son sujet.

Ensembles d'entités
(entity sets)

Les entités sont regroupées en *ensembles d'entités* semblables (de même *type*), c'est-à-dire au sujet desquelles on veut conserver la même information.

Exemples :

- les employés d'une firme
- les cours de la faculté des sciences appliquées.

Note : Le choix de ce qui constitue un ensemble d'entités est laissé au concepteur de la base de données. La conséquence de ce choix est que les informations conservées dans la base de données au sujet d'entités d'un même ensemble doivent avoir la même forme (le même type).

Les (ensembles d') entités

- Contiennent de l’information propre à un objet, qu’il soit concret ou abstrait.
- Exemple : Une personne, un livre, une commande, une idée, ...
- Représentation :

Relations

(relationships)

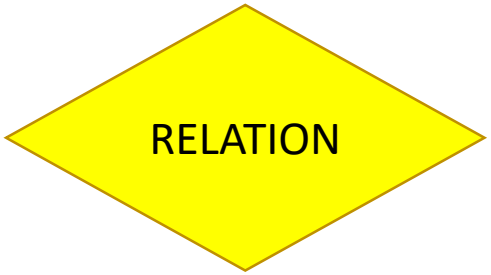
Une *relation* entre ensembles d'entités $E_1, E_2, \ldots, E_k$ est un sous-ensemble du produit cartésien $E_1 \times E_2 \times \cdots \times E_k$.

C'est donc un ensemble de k-tuples $(e_1, \ldots, e_k)$ tels que $e_1 \in E_1$, $e_2 \in E_2$, ..., $e_k \in E_k$.

k est le *degré* de la relation ; si $k = 2$, la relation est dite *binaire* et si $k = 3$, elle est dite *ternaire*.

Les relations

- Lient plusieurs entités et peuvent éventuellement contenir de l'information supplémentaire.
- Exemple : Passe_commande, A_écrit,...
- Représentation :



La description et le schéma des ensembles d'entités et des relations

Un *ensemble d'entités* est décrit par un *nom* et un *type* (la liste de ses attributs – cf. plus loin).

Exemples :

- A l'ensemble d'entités "les employés de l'ULg", on associe le nom EMPLOYÉ_ULg
- A l'ensemble d'entités "les cours de l'ULg", on associe le nom COURS_ULg

Une *relation* est décrite par un *nom* et une liste ordonnée de noms d'ensembles d'entités (son *type*).

Exemples :

- ENSEIGNE : EMPLOYÉ_ULg, COURS_ULg
- PATERNITÉ : PERSONNE, PERSONNE

Rôles

La participation d'un ensemble d'entités à une relation est son *rôle*.
On donne un *nom* au rôle d'un ensemble d'entités dans une relation.

Exemples :

- Dans la relation ENSEIGNE : EMPLOYÉ_ULg, COURS_ULg
  - le rôle de l'ensemble d'entités EMPLOYÉ_ULg est "enseignant"
  - le rôle de l'ensemble d'entités COURS_ULg est "cours enseigné"

- Dans la relation PATERNITÉ : PERSONNE, PERSONNE
  l'ensemble d'entités PERSONNE a deux rôles distincts : "père" et "enfant"

Contraintes de cardinalité

Pour chaque ensemble d'entités participant à une relation, c'est-à-dire pour chaque rôle d'un ensemble d'entités, on précise dans combien de tuples de la relation chaque entité peut apparaître.

En fait, on ne donne pas un nombre, mais on donne un intervalle : (*min*, *max*) où

- *min* est en général 0 ou 1,
- *max* est en général 1 ou ∞ (noté N ou $*$)

Exemple : Quelle est la participation de l'ensemble d'entités EMPLOYÉ_ULg dans la relation ENSEIGNE ?

En d'autres termes, quelle est la cardinalité du rôle "enseignant" dans la relation ENSEIGNE ?

Ou encore, combien de fois un employé particulier de l'ULg peut-il apparaître dans une quelconque extension de la relation ENSEIGNE ?

- *min* : 0
- *max* : N

Diagrammes entité-relation
(entity-relationship diagrams)

Un diagramme entité-relation est une description graphique des ensembles d'entités et des relations qui seront représentés dans une base de données.

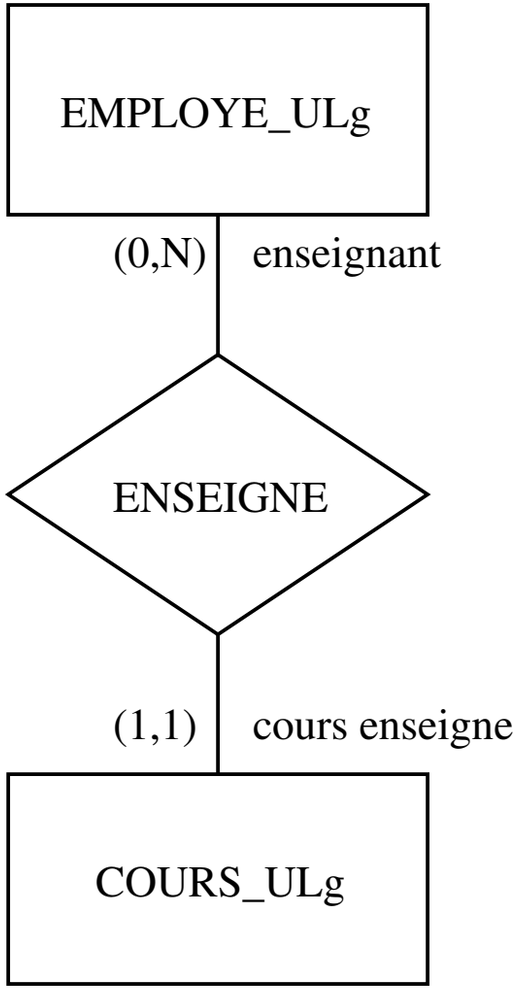
Ces diagrammes représentent donc le *schéma* des données qui seront représentées, par opposition à leur *extension* qui sera le contenu de la base de données.

Notation :

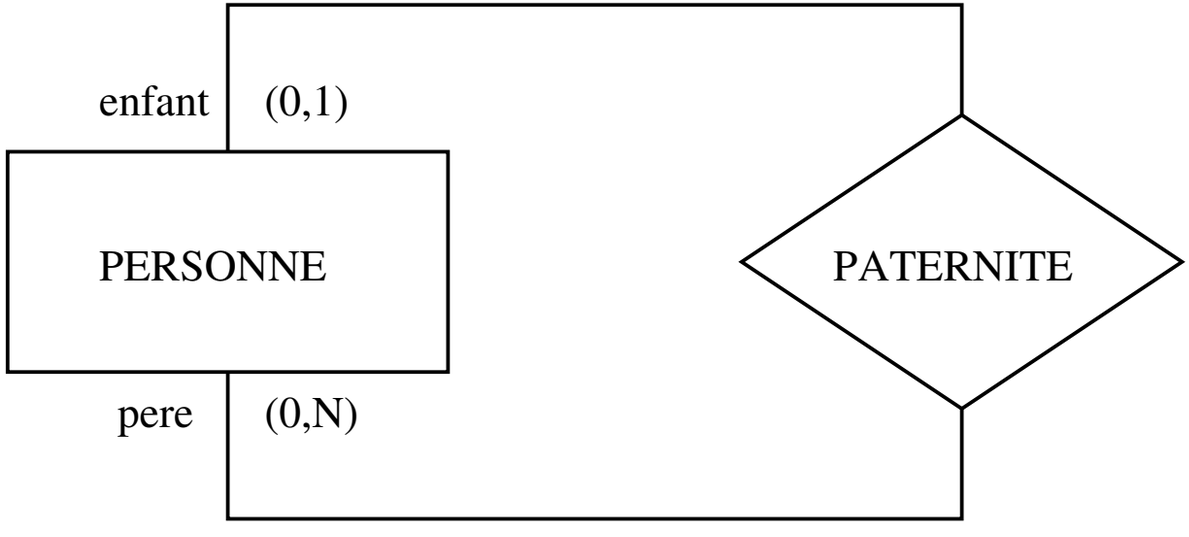
- Ensemble d'entités : nom dans un rectangle
- Relation : nom dans un losange ou un hexagone avec traits vers les ensembles d'entités participant à la relation (chaque trait est étiqueté par le nom du rôle et la contrainte de cardinalité correspondants)

Représentation graphique

Exemple 1 :

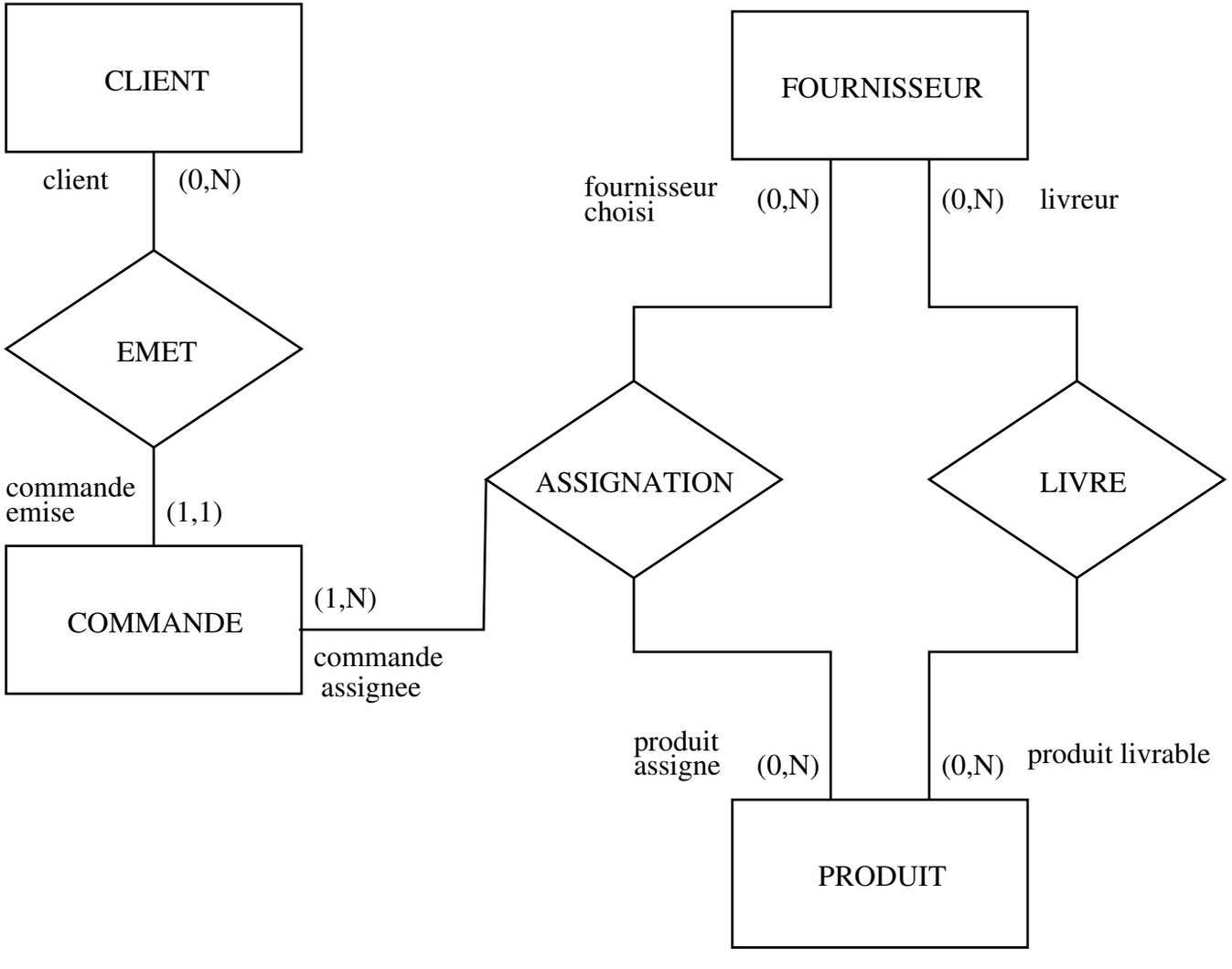


Exemple 2 :



Autre Exemple :

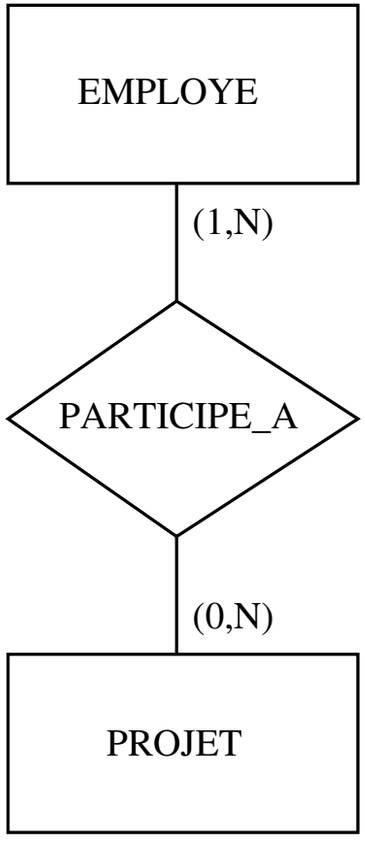
On veut modéliser le fait que des clients peuvent passer des commandes à une centrale d'achat qui a des fournisseurs habituels pour les différents produits possibles et qui décide par quel fournisseur un produit commandé doit être livré.



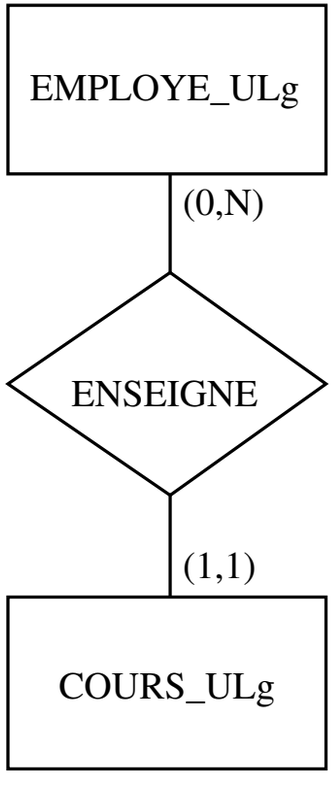
Remarque : Pour un rôle,

- si *min* > 0, la participation de l'ensemble d'entités à la relation est dite *totale* dans la relation

  Exemple : Une entité employé ne peut exister dans la base de données que si elle est associée à au moins un projet.



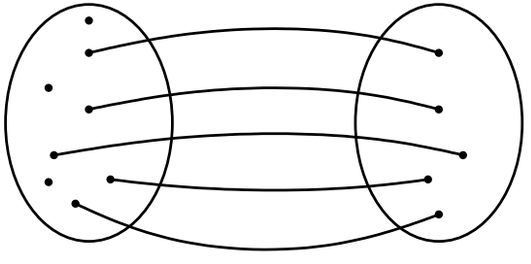
– si *min* $= 0$, la participation de l'ensemble d'entités à la relation est dite *partielle*.
Exemple : Une entité employé de l'ULg peut exister dans la base de données même si elle n'enseigne aucun cours.



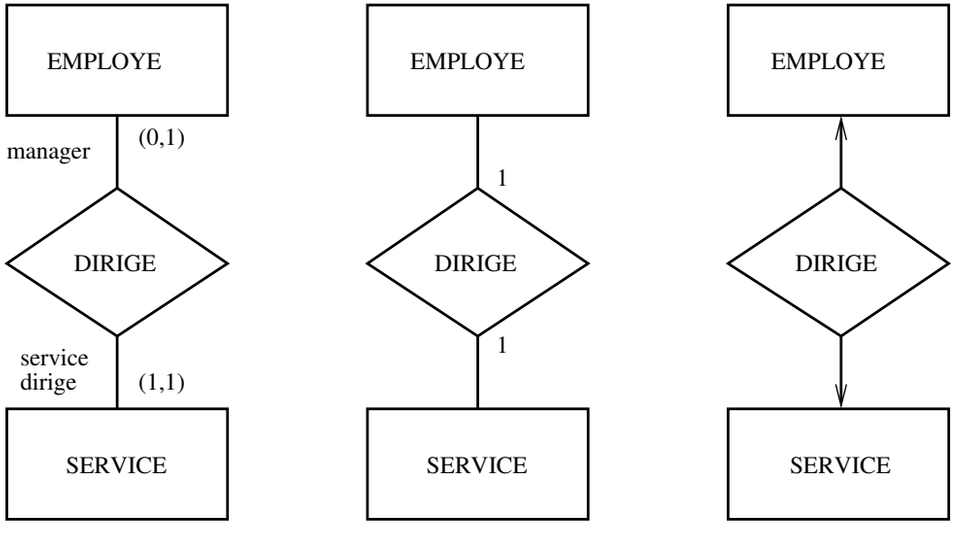
Autres conventions possibles

Plutôt que d'utiliser les contraintes de cardinalité telles que décrites, on utilise parfois des *quotients de cardinalité (cardinality ratios)*. Les plus courants correspondent à distinguer des relations

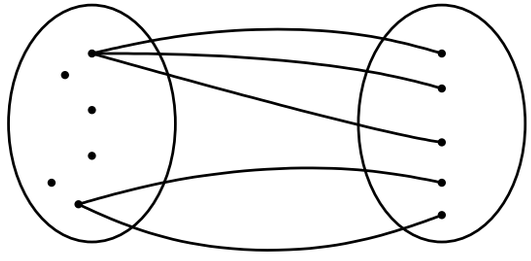
– *one-one (biunivoques)*, en abrégé *1 :1*



Exemple : DIRIGE : EMPLOYÉ, SERVICE

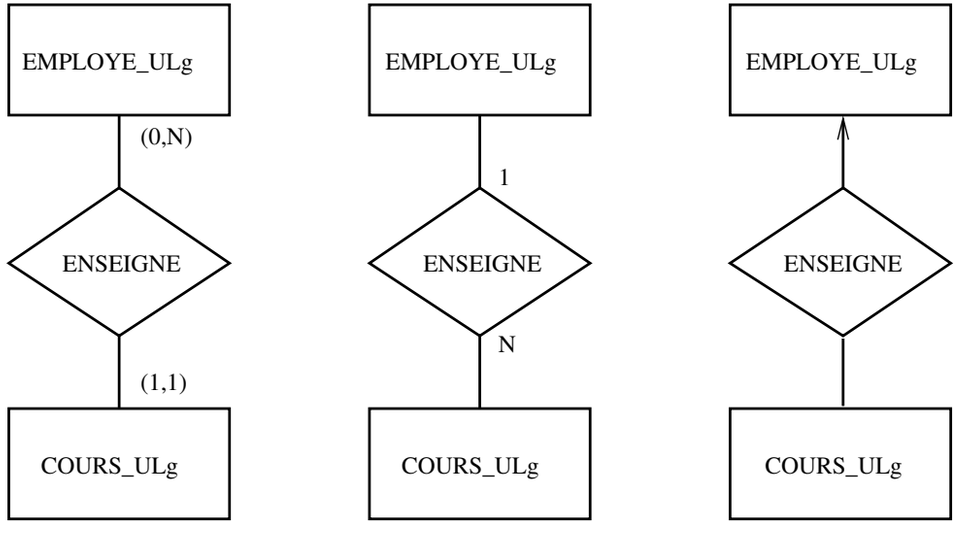


– *one-many*, en abrégé *1 :N*

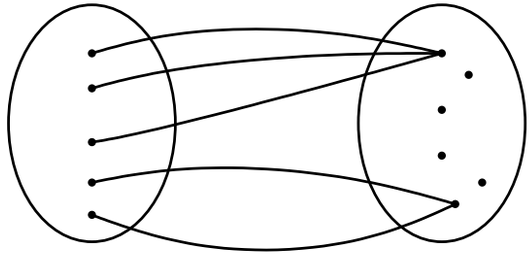


Exemple :

ENSEIGNE : EMPLOYÉ_ULg, COURS_ULg

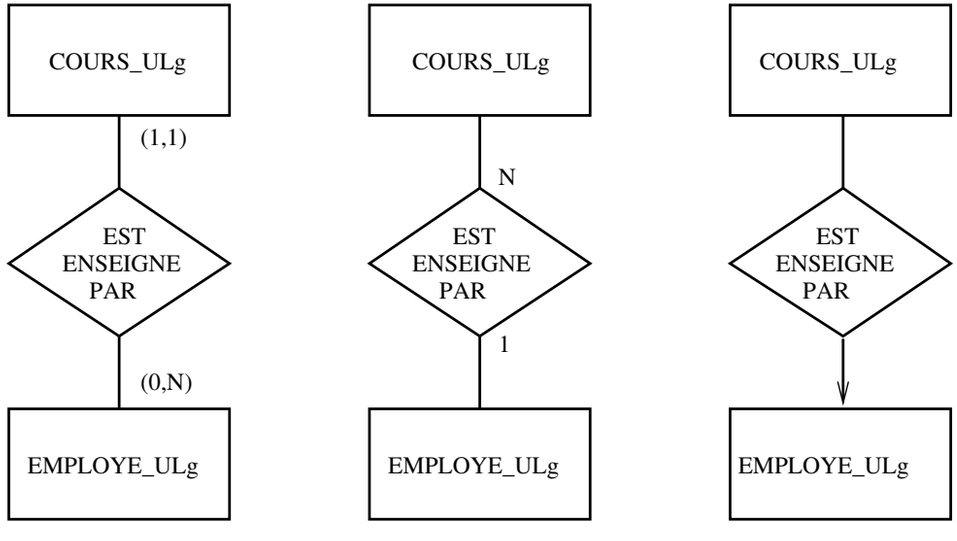


– *many-one (univoques)*, en abrégé *N :1*
Ce sont en réalité des fonctions.

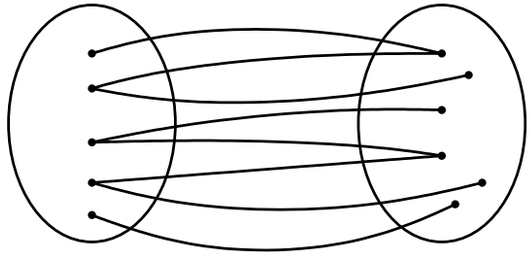


Exemple :

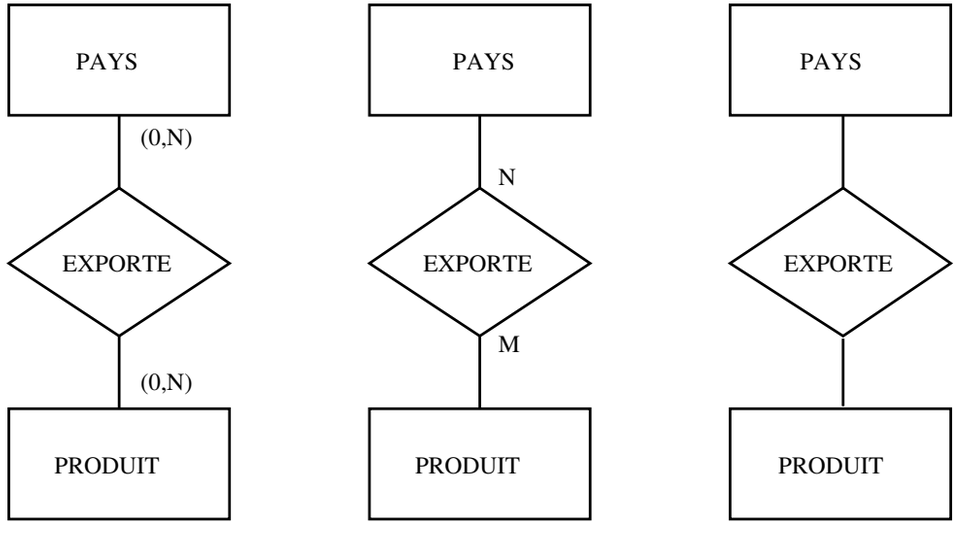
EST_ENSEIGNÉ_PAR : COURS_ULg, EMPLOYÉ_ULg



– *many-many*, en abrégé *N :N* (ou *M :N*)



Exemple : EXPORTE : PAYS, PRODUIT



Attention aux confusions possibles ! Il faut adopter *une* notation ! Nous utiliserons les contraintes de cardinalité.

Attributs

Les informations conservées au sujet des entités d'un ensemble sont leurs *attributs*.

Exemple :

- le nom, l'adresse, la date de naissance d'une personne
- l'intitulé, la charge horaire d'un cours

Toutes les entités d'un ensemble ont exactement les mêmes attributs.

Chaque attribut d'un ensemble d'entités

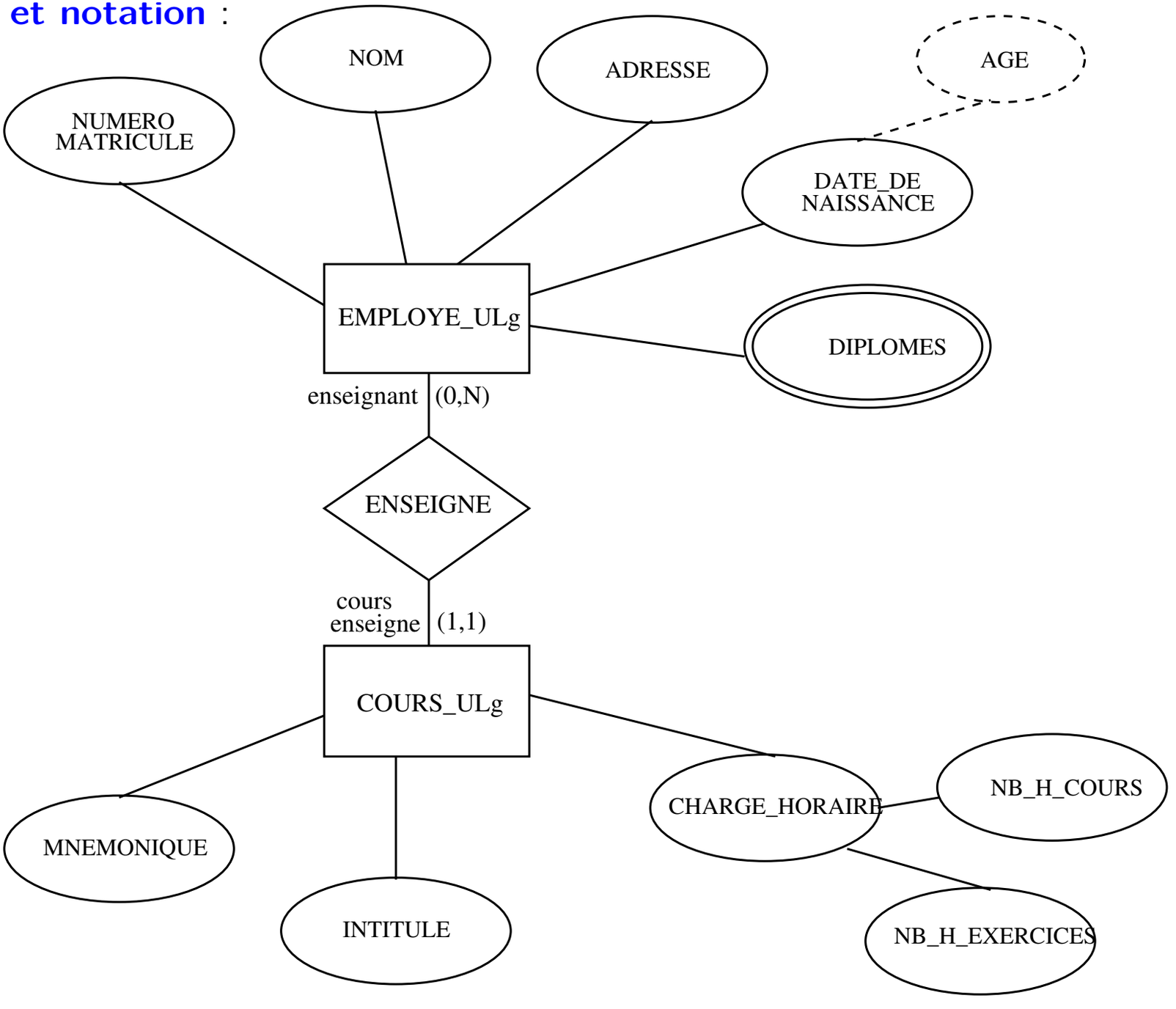
- a un *nom* unique dans le contexte de cet ensemble d'entités,
- prend ses valeurs dans un domaine spécifié (*type* de l'attribut).

Différentes catégories d'attributs

Un attribut peut être

- *simple (atomique)* ou *composite* (décomposable en plusieurs attributs plus simples)
  Exemple : l'attribut ADRESSE peut être constitué des attributs plus simples RUE_ET_NUMERO, BOÎTE, VILLE, CODE_POSTAL, PAYS.
- *obligatoire* (une entité doit avoir une valeur pour cette attribut) ou *facultatif*
  Exemple : l'attribut BOÎTE intervenant dans ADRESSE est facultatif.
- *à valeur unique* ou *à plusieurs valeurs*
  Exemple : l'attribut DIPLÔMES (d'une personne) peut avoir plusieurs valeurs pour une même personne.
- *enregistré* ou *dérivé* (d'un autre attribut)
  Exemple : l'attribut ÂGE (d'une personne) peut être dérivé de l'attribut enregistré DATE_DE_NAISSANCE.

Exemple et notation :



Notion de clé

Une *clé* d'un ensemble d'entités est un ensemble (minimum) d'attributs qui identifient de façon unique une entité parmi cet ensemble.

Donc, deux entités distinctes d'un ensemble ne peuvent jamais avoir les mêmes valeurs pour les attributs de la clé.

Exemples :

- le numéro de matricule d'un étudiant
- la marque et le numéro de série d'une voiture

Il est courant d'associer aux entités d'un ensemble un champ spécifiquement destiné à servir de clé (numéro de matricule, numéro d'inscription, etc.)

Notation : Les attributs de la clé sont soulignés.

Ensembles d'entités faibles

Il peut arriver qu'un ensemble d'entités ne dispose pas d'attributs constituant une clé : c'est un *ensemble d'entités faible*.

Les entités d'un ensemble faible se distinguent les unes des autres par

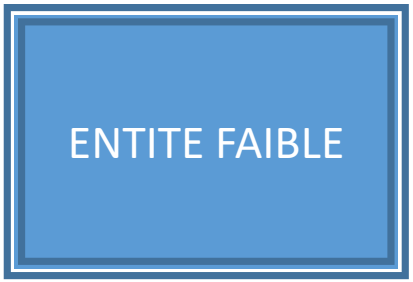
- (éventuellement) certains de ses attributs,
  et
- le fait qu'elles sont en relation avec d'autres ensembles d'entités.

On doit donc généraliser la notion de *clé* : la *clé* d'un ensemble d'entités est constituée

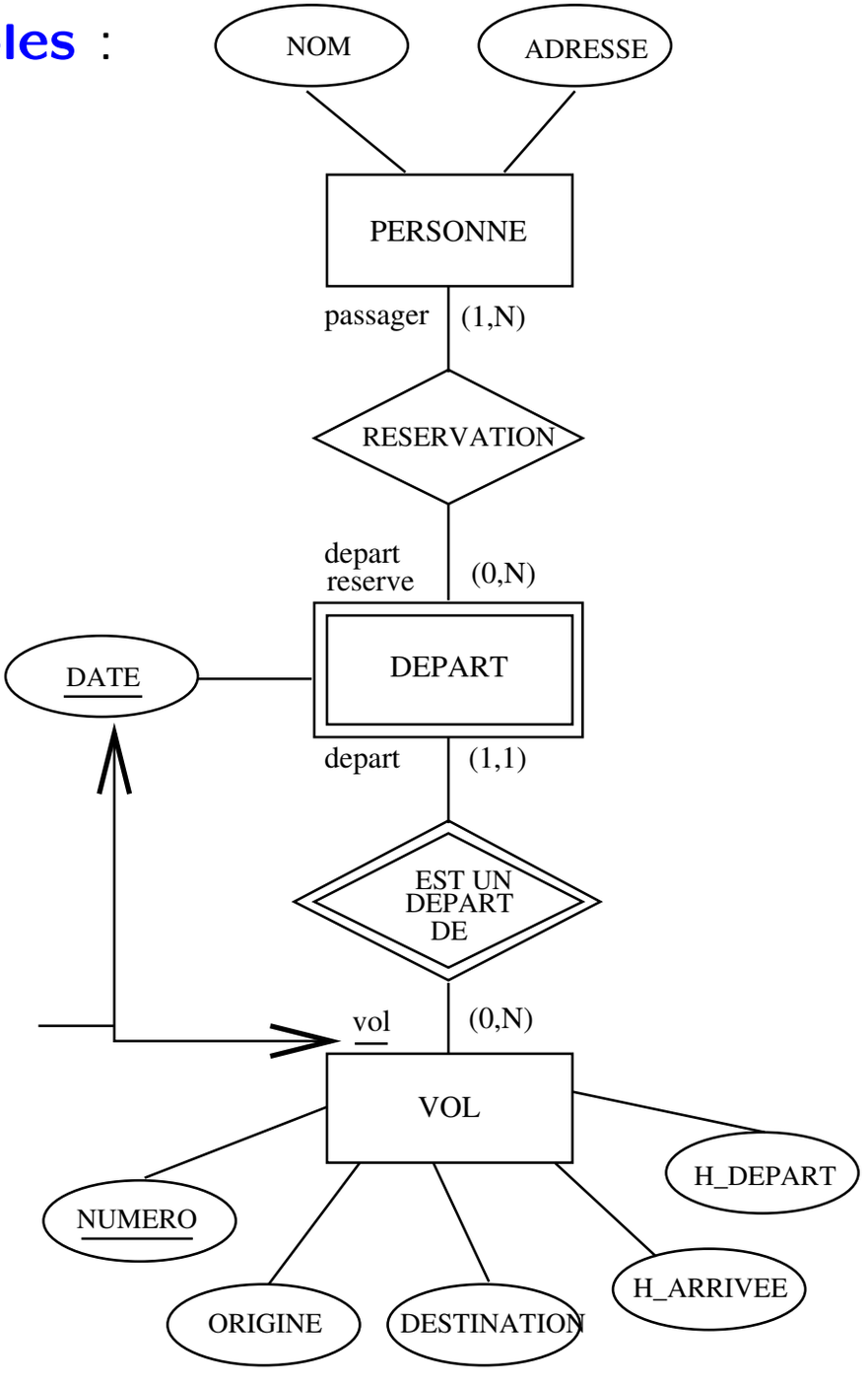
- d'attributs de l'ensemble d'entités
  et/ou
- de rôles joués par d'autres ensembles d'entités dans leur relation avec cet ensemble d'entités (*relations identifiantes*).

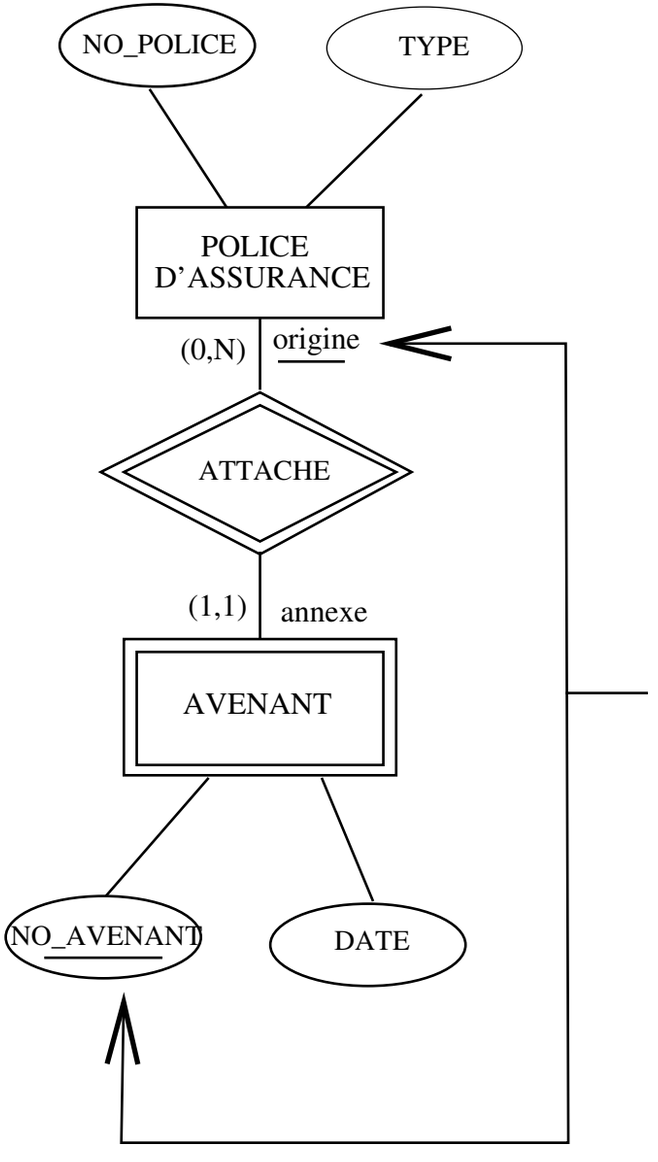
L'entité faible

- Se définit à partir de ses attributs-clés et du rôle d’une autre entité.
- Exemple : Joueur de football (avec entité équipe). Peut s'identifier à partir de son numéro sur le terrain + l'identifiant de l'équipe.
- Représentation :



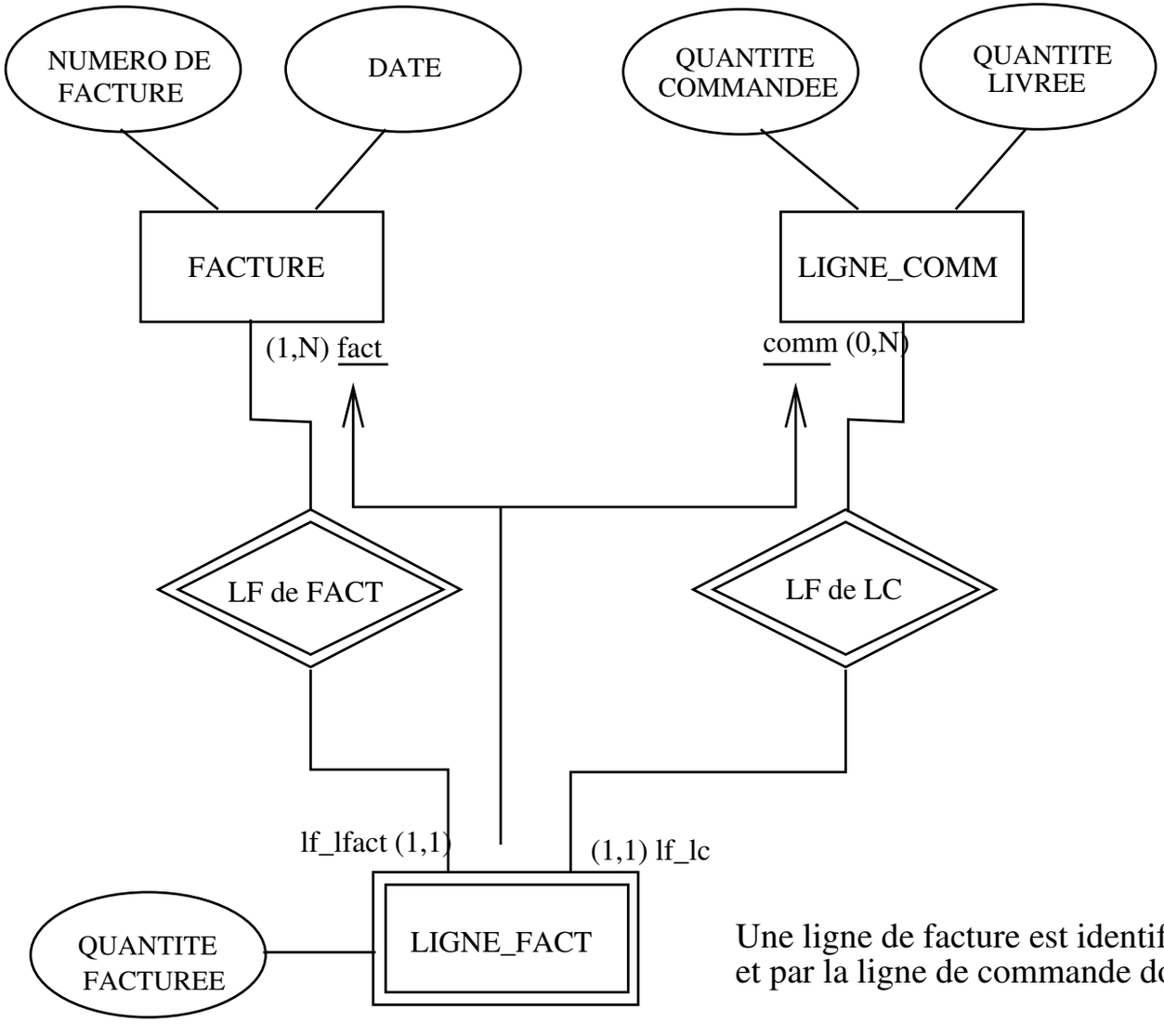
Exemples :





Un avenant est identifie par sa police d'origine et par son numero au sein de celle–ci.

Exemples (suite) :



Remarque : Le rôle de l'entité faible dans ses relations identifiantes doit avoir une cardinalité $(1,1)$.

Attributs et clés de relations

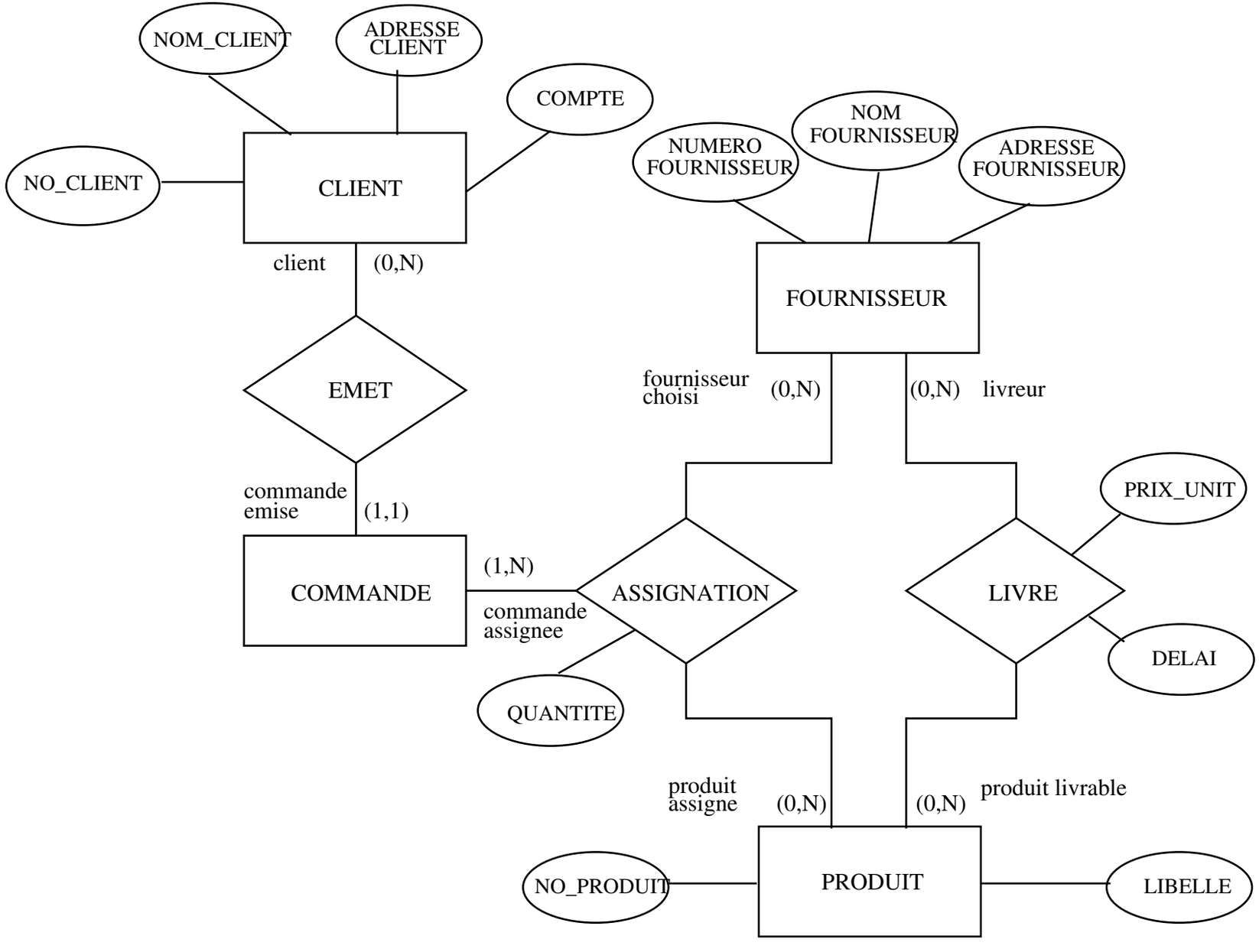
Les relations peuvent avoir des attributs, exactement comme les ensembles d'entités.

Une *clé* d'une relation identifie de façon unique un tuple parmi tous ceux de la relation.

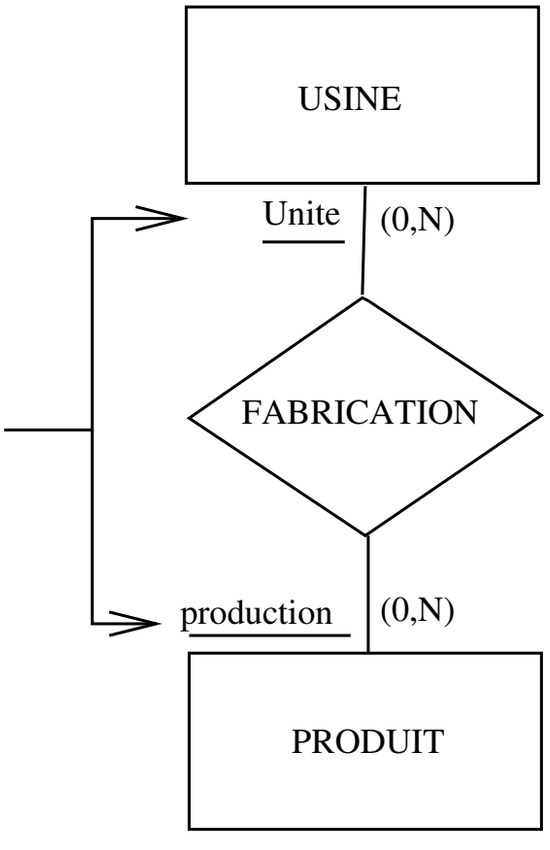
La *clé* d'une relation est constituée

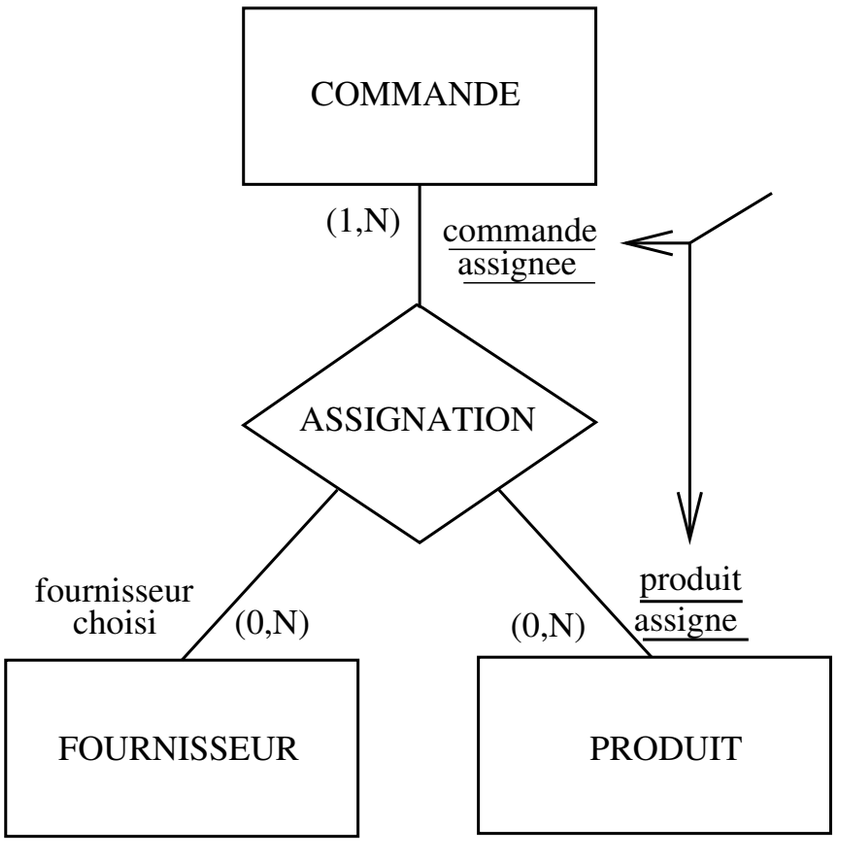
- de rôles joués par des ensembles d'entités dans cette relation.
  et/ou
- d'attributs de la relation

Exemple (attributs de relations) :

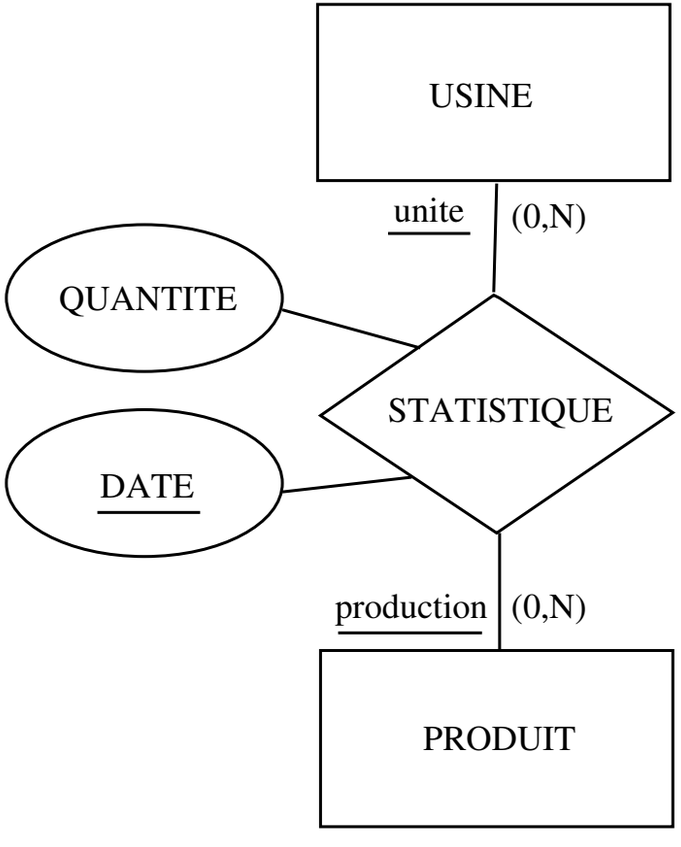


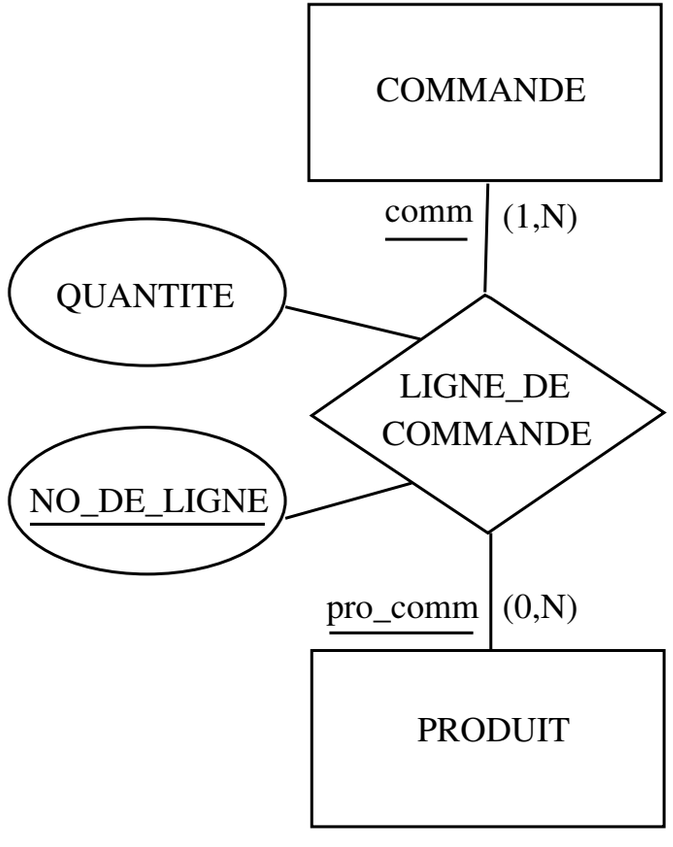
Exemples (clés de relations) :



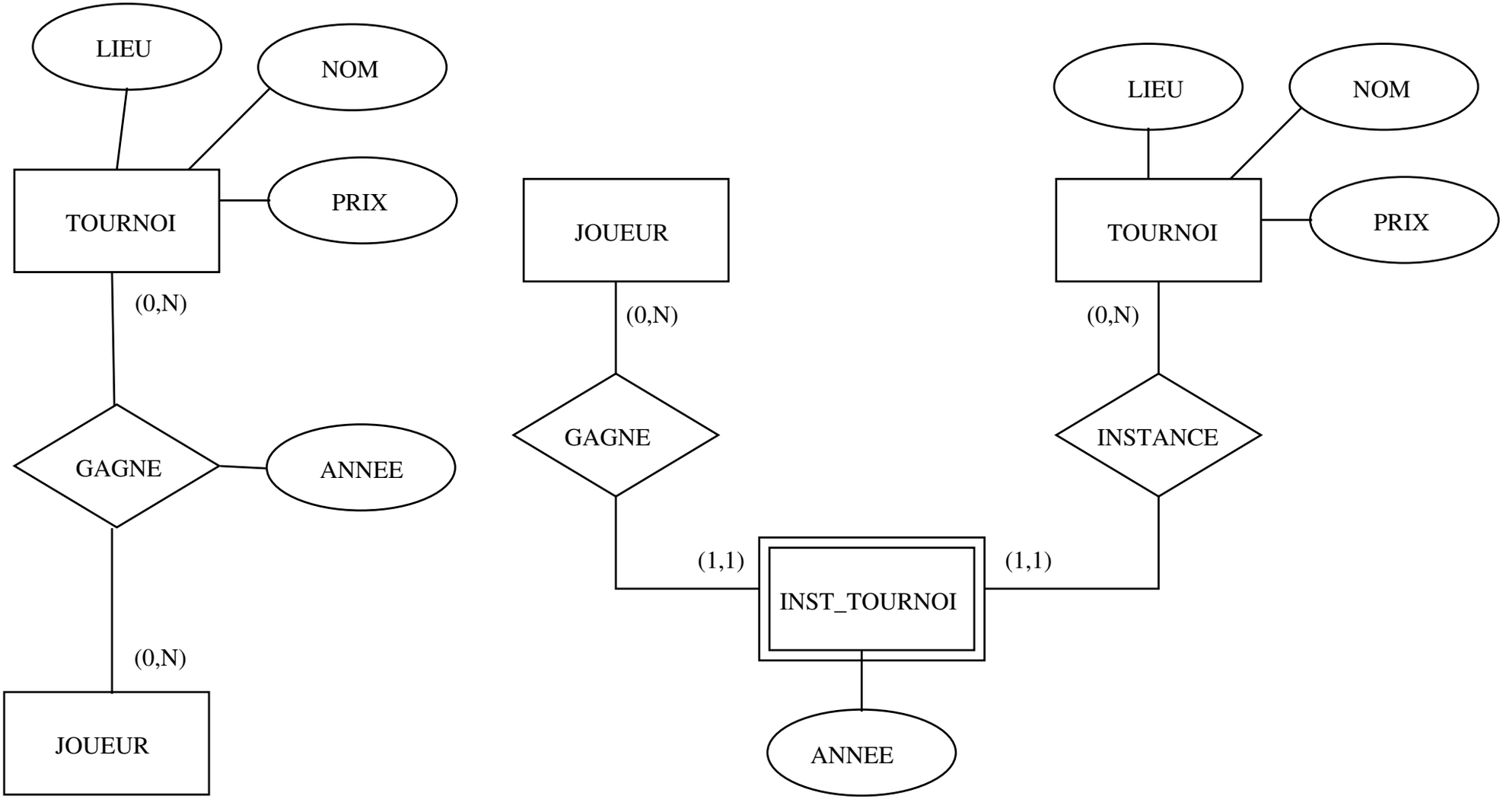


Exemples (clés de relations) :





Observer la correspondance entre ensemble d'entités faible avec attribut et relation avec attribut :



Contraintes d'intégrité

Contraintes que doivent satisfaire les données d'une base de données à tout instant. Elles sont exprimées au niveau du schéma.

Nous en avons déjà rencontré deux formes :

- les contraintes de cardinalité (*min*, *max*),
- les clés.

On peut en donner d'autres :

- contraintes sur les attributs (lient les valeurs d'un attribut) ;
- contraintes référentielles (on ne fait référence qu'à une entité existante).
- etc.

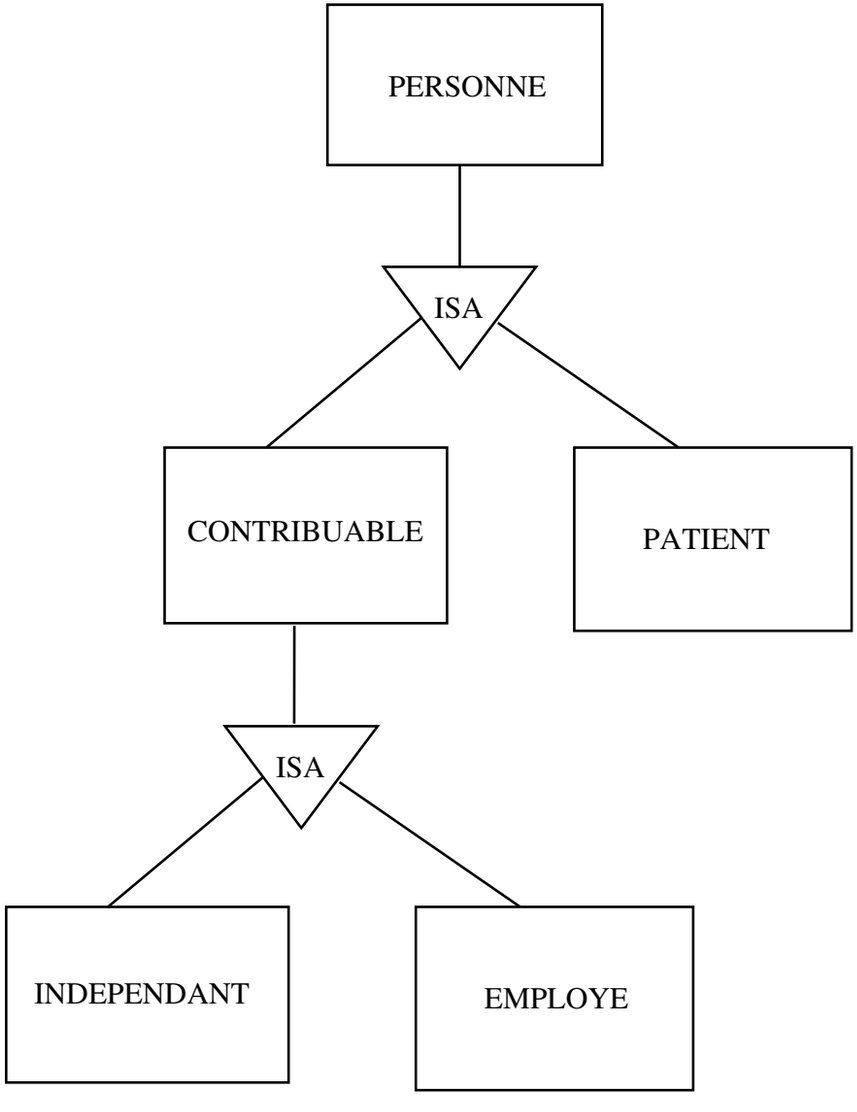
Une extension du modèle entité-relation : Relations IS-A

Il est parfois intéressant de définir des ensembles d'entités de façon hiérarchique, la hiérarchie correspondant à une structure d'inclusion.

Ensemble d'entités inclus : *sous-classe* (ou *ensemble d'entités spécifique*)

Ensemble d'entités en comprenant d'autres : *super-classe* (ou *ensemble d'entités générique*)

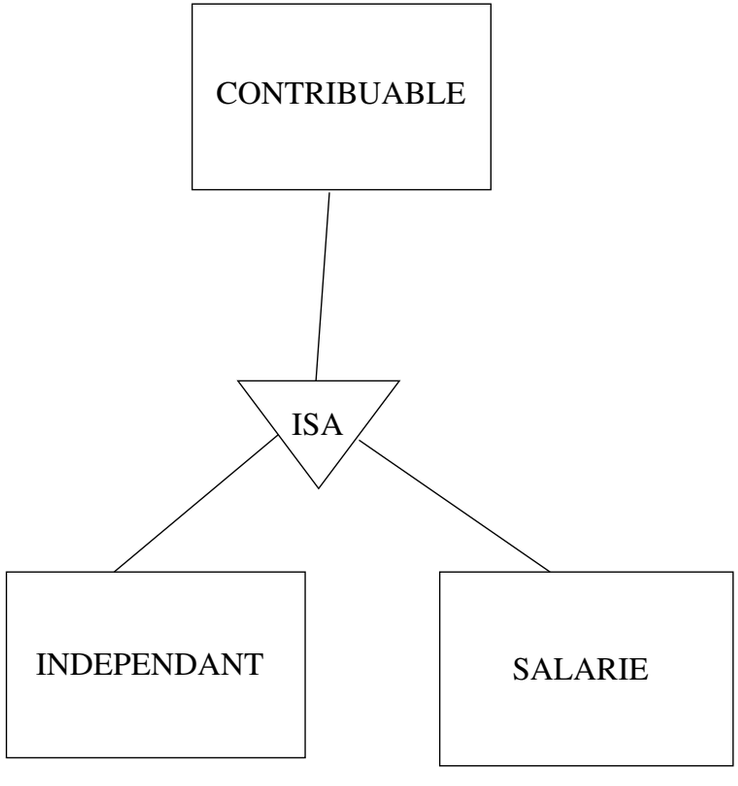
Exemple : Une personne peut être un contribuable et/ou un patient, et si cette personne est un contribuable, elle peut être soit indépendante soit salariée.



Spécialisation et généralisation

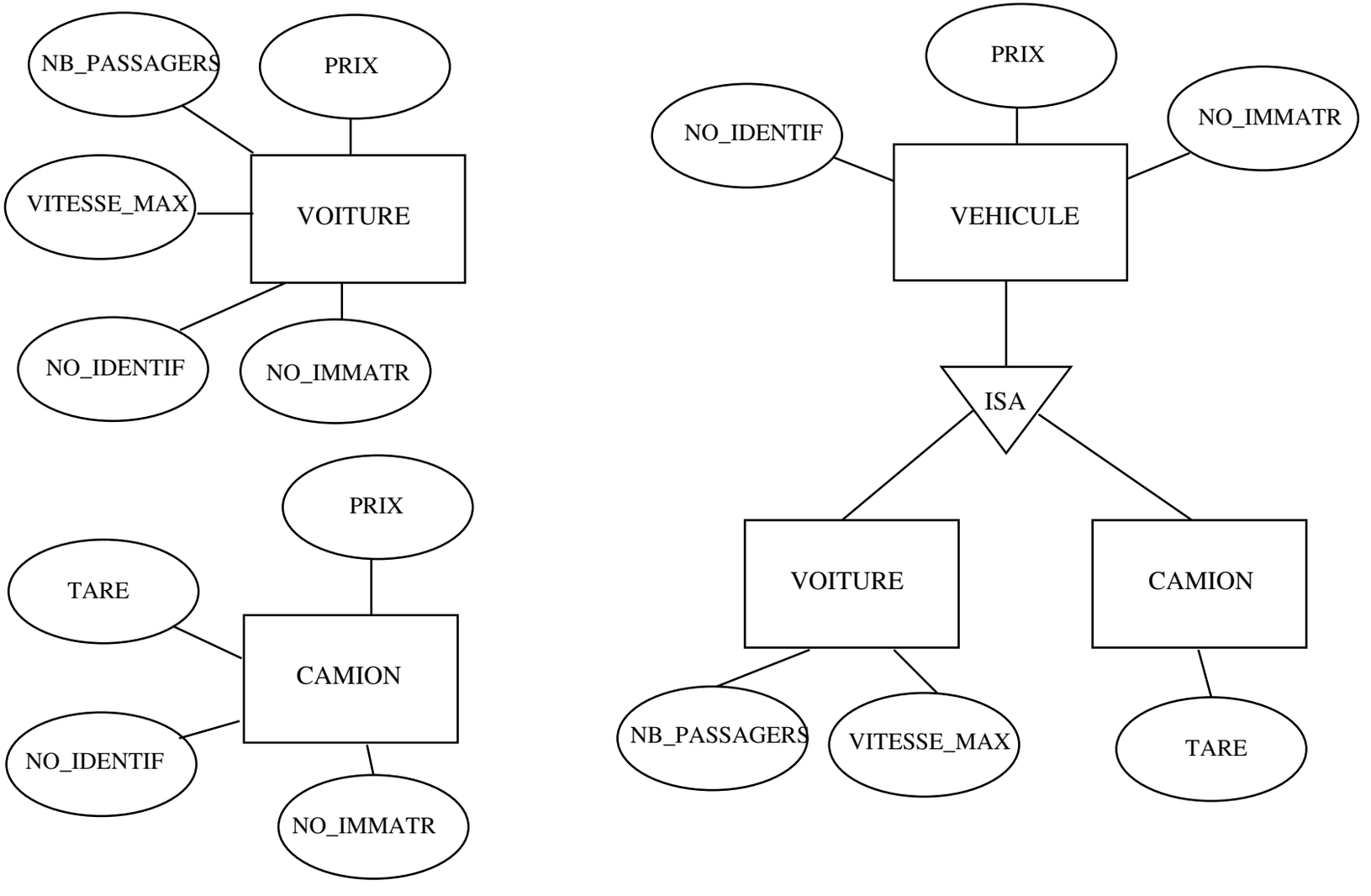
Spécialisation : division d'un ensemble d'entités en sous-classes

Exemple :



Généralisation : regroupement de plusieurs ensembles d'entités en une super-classe

Exemple :



Note : Souvent, la distinction est arbitraire.

Une classe peut être divisée selon plusieurs critères indépendants, donnant lieu à plusieurs spécialisations (taxonomies) différentes.

Exemple : L'ensemble d'entités EMPLOYÉ peut donner lieu à deux taxonomies basées sur des critères distincts :

- la fonction de l'employé donne lieu à une spécialisation en 3 sous-classes : SECRÉTAIRE, TECHNICIEN, CADRE
- le type de rémunération donne lieu à une autre spécialisation, complètement indépendante de la première, en 2 sous-classes : SALARIÉ, CONSULTANT

Une classe peut être la sous-classe de plusieurs superclasses, participant ainsi à plusieurs généralisations.

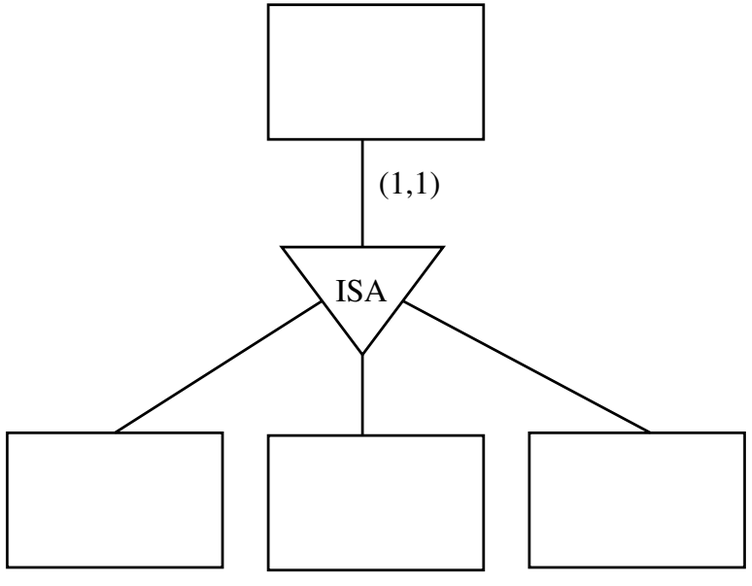
Exemple : Dans le schéma conceptuel des personnes associées à une université, l'ensemble d'entités ÉLÈVE-ASSISTANT est une sous-classe des ensembles d'entités EMPLOYÉ et ETUDIANT.

Contraintes sur les spécialisations/généralisations

Deux catégories de contraintes indépendantes :

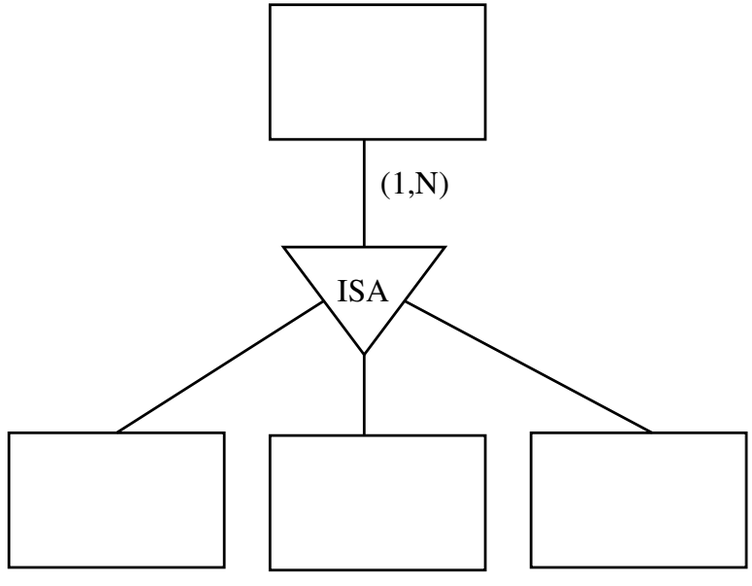
- Quand l'union des sous-classes donne la super-classe, les sous-classes constituent une *couverture* de la super-classe.
  Dans ce cas, la *cardinalité minimum* du rôle de la super-classe dans la relation IS-A est 1 s'il y a couverture, et est 0 sinon.
- Les sous-classes d'une super-classe peuvent être *disjointes* ou non.
  Dans ce cas, la *cardinalité maximum* du rôle de la super-classe dans la relation IS-A est 1 si les sous-classes sont disjointes, et N sinon.

Contraintes sur les spécialisations/généralisations (2)



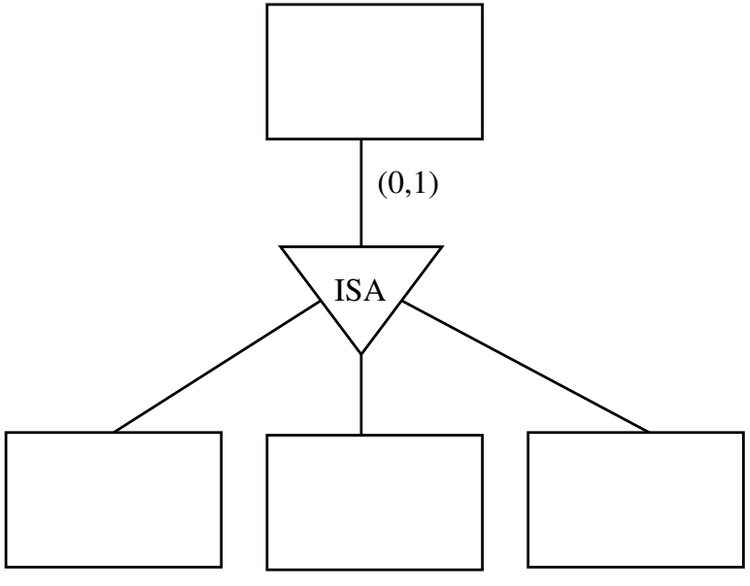
couverture
+
sous−classes disjointes

PARTITION



couverture
+
sous−classes non disjointes

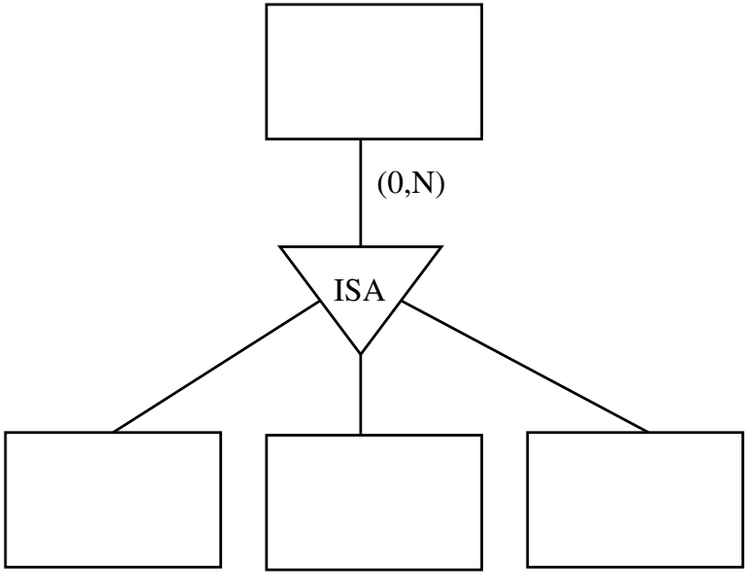
Contraintes sur les spécialisations/généralisations (3)



pas de couverture

+

sous−classes disjopintes



pas de couverture

+

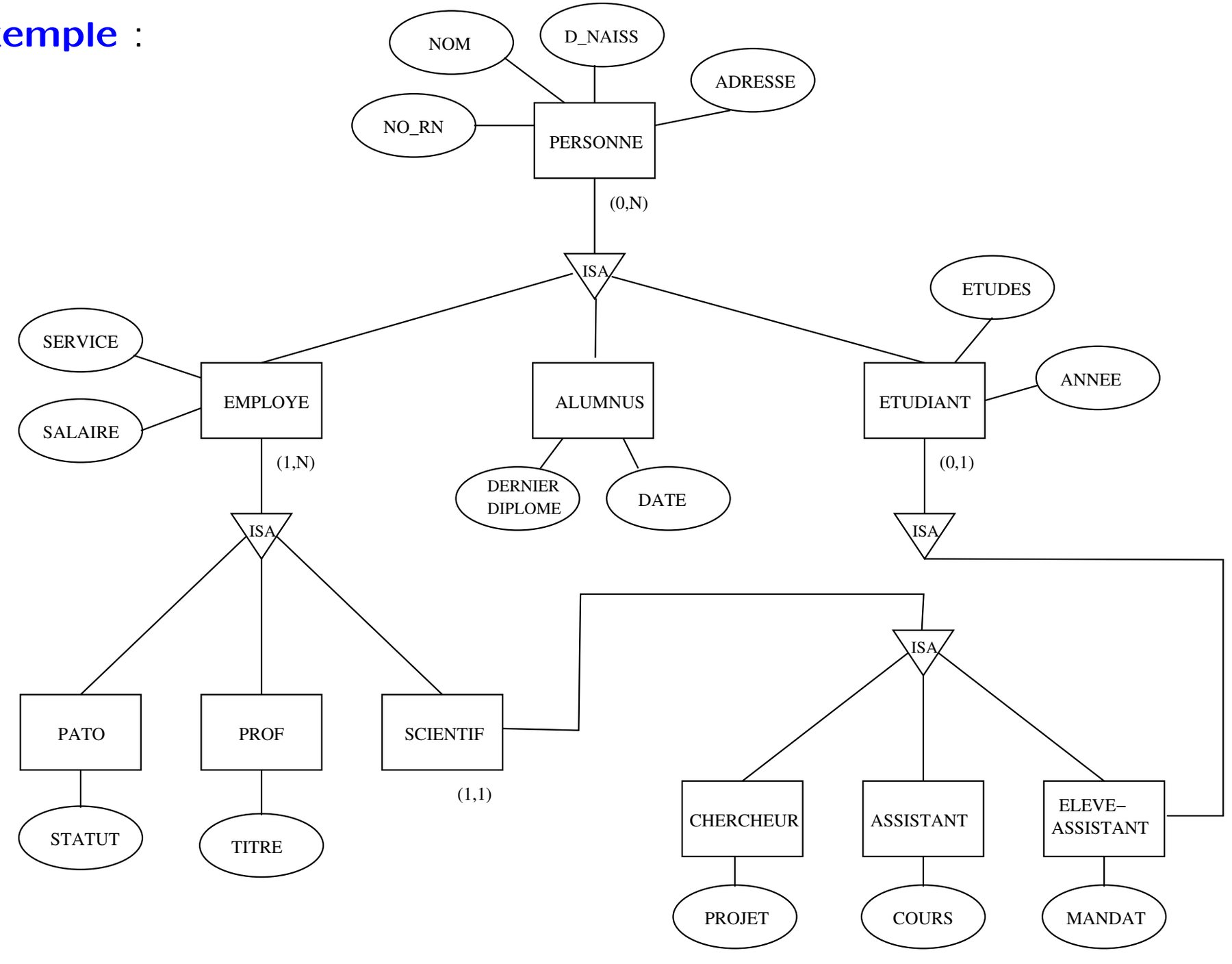
sous−classes non disjointes

Héritage

A l'inclusion entre ensembles d'entités correspond l'*héritage* des propriétés (attributs et rôles)

Remarque : Lorsque un ensemble d'entité est une sous-classe de deux classes différentes, on parle d'*héritage multiple*. On n'a alors plus une hiérarchie d'inclusion (donc un arbre), mais une structure d'inclusion plus générale : un graphe dirigé acyclique (DAG).

Exemple :


NOM
D_NAISS
ADRESSE
NO_RN
PERSONNE
(0,N)
ISA
ETUDES
SERVICE
EMPLOYE
ALUMNUS
ETUDIANT
ANNEE
SALAIRE
(1,N)
DERNIER DIPLOME
DATE
(0,1)
ISA
ISA
ISA
PATO
PROF
SCIENTIF
(1,1)
CHERCHEUR
ASSISTANT
ELEVE–ASSISTANT
STATUT
TITRE
PROJET
COURS
MANDAT

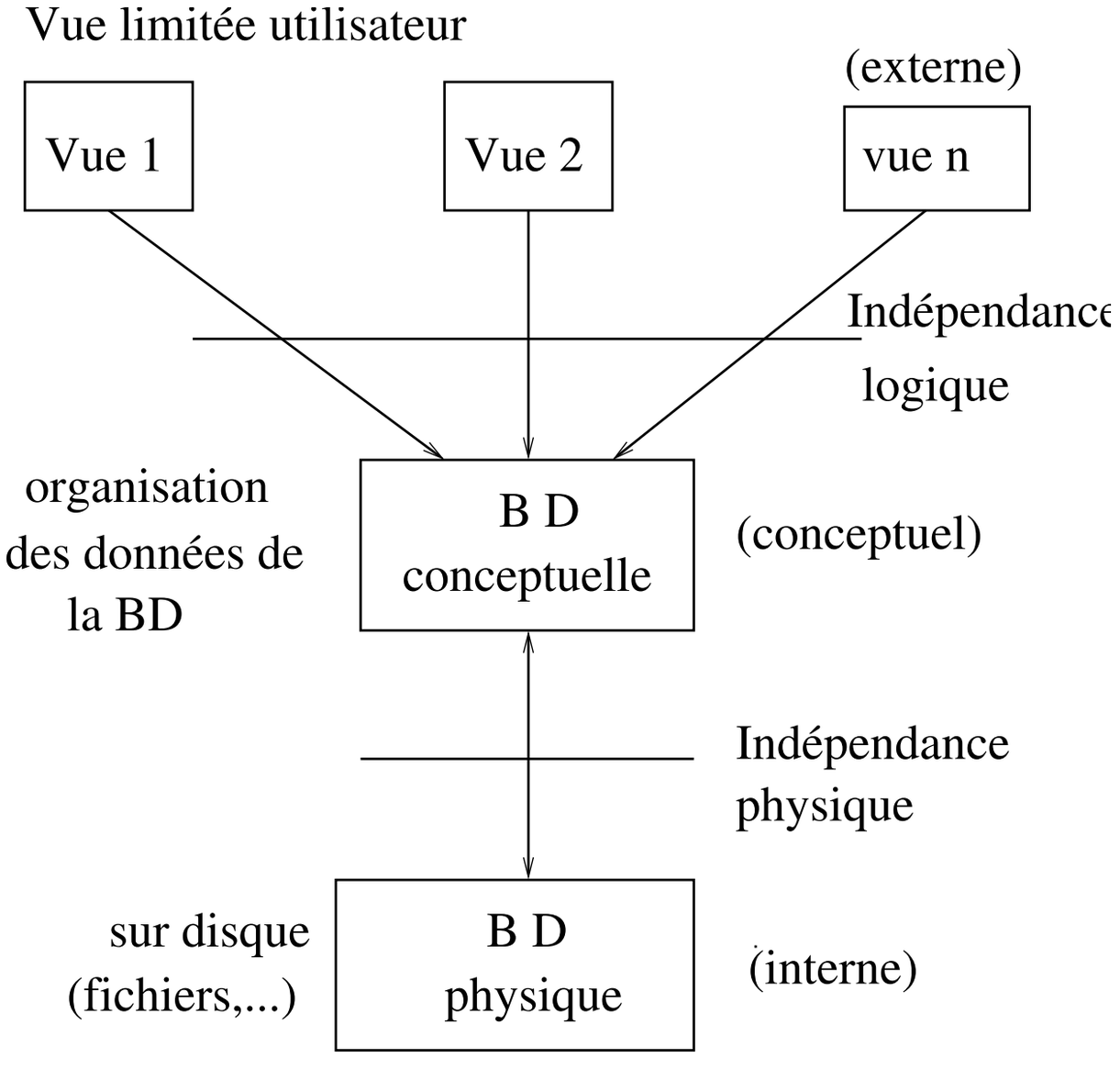
Que doit fournir un SGBD ?

La possibilité de définir, mettre en œuvre et exploiter une base de donnée à un *niveau suffisamment abstrait* (proche du modèle entité-relation).

Plus précisément, il doit offrir

- le support d'un *modèle de données*
- le support de langages de haut niveau pour la définition de la structure des données, l'accès et la manipulation des données
- la *gestion de transactions* : parallélisme des opérations sur la base de données
- le *contrôle d'accès* : restrictions d'accès et vérification de la validité des données
- la capacité de *récupération* en cas de panne

Niveaux d'abstraction
Indépendance



Analogie (langages de programmation)

Niveau externe (vue) : tableau à une dimension

```
integer array B[1..n]
```

où, par exemple, $B[i] = \sum_{j=1}^{m} A[i,j]$

Niveau conceptuel :

```
integer array A[1..n, 1..m]
```

Niveau physique : `A[i,j]` est enregistré dans un bloc commençant à l'adresse

$$a_0 + 4(m(i-1) + (j-1))$$

a_0

$A[1,1]$	$A[1,2]$	$\cdots$	$A[1,m]$	$A[2,1]$	$\cdots$

Indépendance entre niveaux

(Data independence)

Existe si une modification à un niveau (par exemple pour améliorer l'efficacité) ne nécessite pas une modification du niveau supérieur.

Deux types d'indépendance :

Physique : entre niveaux physique et conceptuel

Logique : entre niveaux conceptuel et vue

Modèles de données implémentés dans les SGBD

Le modèle entité-relation sert de référence dans la conception de schémas de bases de données.

Toutefois, il n'est pas implémenté dans les SGBD. On lui préfère le modèle relationnel qui est plus simple et plus facile à implémenter.