Conception de Base de Données

interro shema entite-association : 25%
clé cours : DB18

Contents

1 Qu'est ce qu'une base de donnée ?

1.1 Historique

Le stockage de données n'est pas nouveau en informatique. A l'origine, on utilisait des cartes perforées pour stocker de l'information. Ensuite, on utilisa des bandes magnétiques. Le problème avec ce type de fichiers est que les accès ne peuvent ce faire que séquentiellement. Même si les évolutions techniques ont permis de réduire les temps d'accès, les accès concurents aux fichiers pour différentes applications posent toujours un soucis majeur. Ce besoin d'accès concurrent à permis l'élaboration de nouveaux concepts et nottament les **systèmes transactionnels** (*suite d'instructions qui forment un tout*).

69' : Apparition de banques de données (faiblement structurées, classés par mots clés et dictionnaires)
70' : IBM Lance le SGBD (Système de gestion de Bases de Données)

Accès concurrents

différentes facon de gérer un fichier par un user

- Bloquer un fichier
- bloquer l'enregistrement sur lequel on écrit
- adopter le principe de transactions
  - suite logique d'instructions considérée comme formant un tout (soit toutes les instructions s'effectuent soit aucune).

Système transactionnels

Au début de la transaction on effectue une première copie des donées. Cette copie est ensuite elle même copiée et on effectue les modifications sur cette copie 2. Enfin, on effectue une copie 3 des données. Cette copie3 est comparée à la copie1 et si ces deux copies sont identiques, celà signifie que entre temps le fichier de données n'a pas été modifié. Si il y a eu modification, on préviendra l'utilisateur. Dans le cas contraire, on recopie la copie 2.

1.2 inconvénients des systèmes de fichier

- Dépendance des applications (Chaque application doit gérer ses fichiers)
- Redondance des données (si il y a deux programmes, il y aura deux fichiers avec les mêmes données)
- Risques d'incohérences des données
- Impossibilité d'interroger les données(il n'y a pas de lien entre les différents fichiers)
- Sécutité (auto-sauvegarde)
- Pas de modification de structure possible

- Pas de relation entre les fichiers (et les données dans ces fichiers)

1.3 Lien etntre les données

Une **Donnée** est un enregistrement dans un code en vue de transmettre ou stocker de l'information.
Une **information** (Subjectif) : sens ou signification que l'on attache ou déduit d'un ensemble de données. Un **Fichier** est un ensemble de données.

Liens Objets

Dans une BDD, on lie deux objets entre eux par l'intermédiaire de liens.

(Exemple : Un **objet** du *type livre* à un **lien** avec un **objet** du *type auteur*.)

1.4 Caractérisation d'une Base de Données

- **Relation** entre les données
- **Sauvegarde** des données sur un support (disque)
- **Partage** des données entre plusieurs utilisateurs (+gestion des droits d'accès)
- **Indépendance** des données par rapport aux applications
- **Sans redondance** inutile (sauf raisons de performances et de sécurité)
- Contrôle de **cohérence** (si il y a de la redondance, le système s'assure pour mettre à jour les données)
- Exploitation des donneés par **interrogations**
- **Longueur** d'enregistrement **variable** (NULL est différent de " " et 0)
- **Modification** possible de la structure des enregistrement

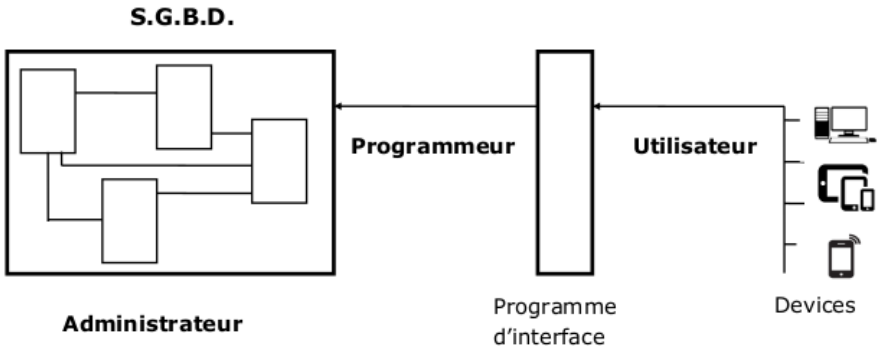
Une Base de Données est un ensemble de données en relation, indépendantes des applications, sans redondance inutile, partageable entre plusieurs utilisateurs et dont on peut accéder à n'importe quel contenu en réponse à une question

1.5 Difficultées d'identifiant des concepts et liens

Les types d'objets ont des caractéristiques et donnent des types d'enregistrements.

1.6 Système de gestion de base de données (S.G.B.D.)

Un SGBD ou DBMS(en anglais) est un outil permettant la gestion d'une base de donnée en permettant la création, la modification et la suppression d'un fichier. Cet outil permets ausi de rechercher des données et de les modifier dans les enregistrements faits dans les fichiers déjà existants.



Les Différents Roles

Le programmeur Un programmeur écrit son programme dans un language "classic". Il peut interragir avec la base de donnée, en utilisant le language SQL pour poser des questions (Querry):

```
* DQL : Data Querry Language : requête d'acces sous la forme déclarative
* DDL : Data Definition Language : ajoute et modifie des shémas de BDD
* DCL : Data Control Language : Traite les permissions
* DML : Data Manipulation Language : Ajoute et modifie des données
```

Les Utilisateurs Lambda Un utilisateur lambda est amené à utiliser l'application et interagit avec la base de données par le biais de l'application programmée par le programmeur.

L'Administrateur de Base de Donnée Une DataBase est gérée par un Administrateur. Il est responsable de l'ensemble du système :

- Création de la structure de originale de la BDD et organisation fichiers (DDL)
- Modifiaction de la structude de la BD
- Gestion des accès
- Backup et entretien de la BD

Le SGBD Le rôle d'un SGBD :

- **Accès optimal** à toute donnée
- **Traitement simultané** des données
- **Validité et cohérence** des données
- **Sécurité** (droit d'accès)
- **Sauvegarde et Récupération** # 2 Le modèle entités-associations

2.1 Introduction

Il existe 3 étapes dans la création d'une base de donées.

- *Shéma conceptuel* (**Entité-association**).
- *Shéma logique* (**Shéma conforme ou SGBD**).
- *Shéma physique* (script **SQL** si relationnel).

Le shéma conceptuel est indépendant de tout SGBD. Son but est de mettre en évidence les concepts évidents et les relations qu'ils ont les uns avec les autres. Pour la création d'un shémas conceptuel on peut utiliser comme outil le modèle entité-association. Il exprime sa séantique de données sur le principe des notions **d'entitées**, **d'associations**, **d'attributs** et le **mécanisme de contraintes d'intégrité**.

Sémantique : sens, signification attribuée aux données.

2.2 Entités et types d'entités(T.E.)

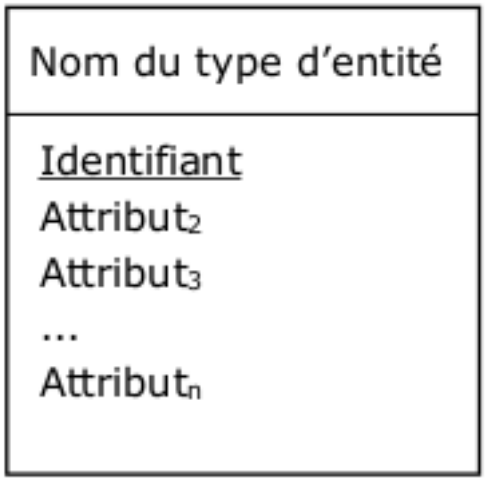
Une **Entité** est une chose qui existe dans le monde réel, à propos de laquelle on veut enregistrer des informations.

Un **Type d'entité** est une abstraction générale de ce qui caractérise plusieurs entités communes.

Ainsi, une entité deviendra une occurence d'un type d'entité. (ex : Bob est une ocurence de Humain)

> un Type d'entité ne prends pas de 's' (ex : nom et ~~noms~~) le s désigne l'ensemble de ses occurences.

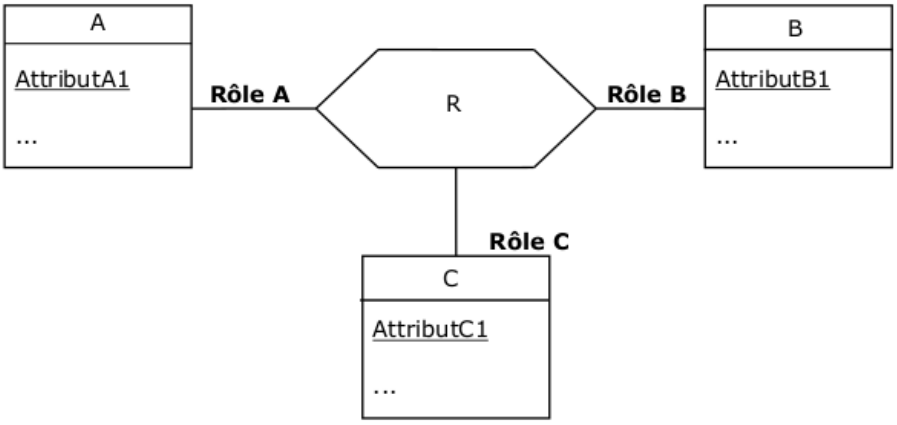
On définira donc un Type d'entité par son **Nom** et sa **Liste d'attributs**



L'**identifiant** possède une valeur distincte pour chaque occurrence du T.E.

2.3 Associations-Type d'associations (T.A)

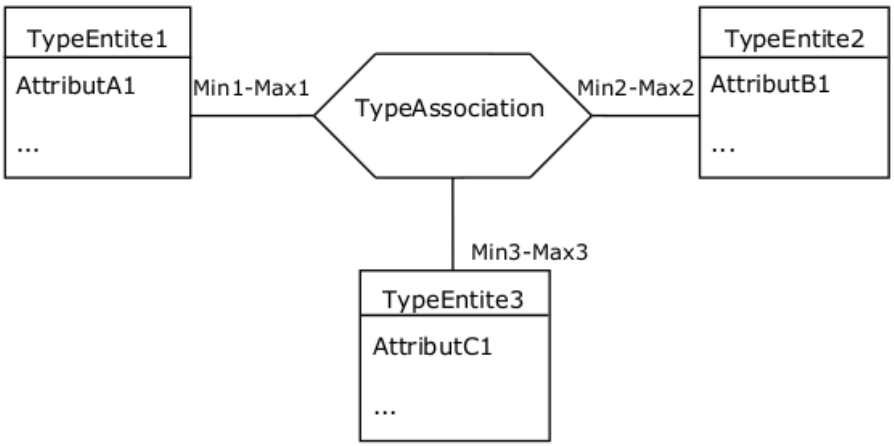
Une **association** est une correspondance, un lien entre 2 ou plusieurs entités.



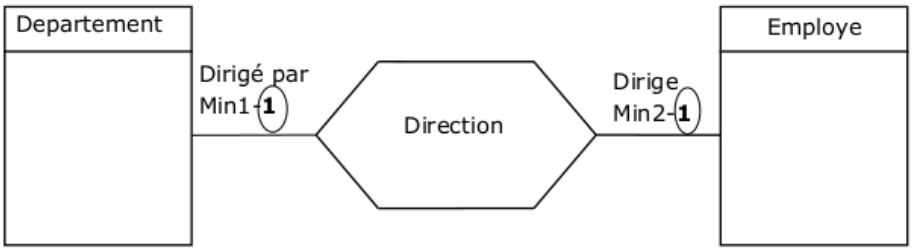
> Une Occurence d'un T.A. est toujours reliée à une et une seule occurence de chaque T.E. associé. (ex : une seule consultation pour un medecin et un client)

- 1 à 1
- 1 à N
- N à N

> L'emplacement des cardinalités est l'inverse de celui utilisé pour le diagramme de classes



cas d'un 1 à 1



Une cardinalité est le nombre de liens entre 2 types d'entités.

Obligatoire et Facultatif

La cardinalité minimum d'un type d'entités donne son caractére:

- obligatoire(**1**)
- facultatif(**0**).

2.4 Construction d'un Shéma conceptuel

Pour former un shémas, on cherche 2 concepts et un Lien qui les unis

2.5 Interêt de prévoir un type d'association plutôt qu'un atribut

Lors de la création d'un shémas, on pourait penser qu'utiliser un attribut plustôt qu'un type d'association peut être une bonne idée. En pratique, ce n'est jamais le cas. En effet, si vous laissez à vos utilisateurs le soin de rentrer (par exemple leur code de cours) ils se peut que ceux-ci se trompent et indiquent **n'importe quelle chaine de caractère**. Il n'y a pas non plus de **vérification de cohérence** que peut offir un SGBD par l'intermédiaire de **contraintes**.

2.6 Attributs

Un **Attribut** est défini par un *nom*, un *type* et le *domaine de valeurs admises*.

> Un Attribut Booleen possède 3 valeurs possibles : Vrai, Faux **ET NULL** quand on ne sait pas.

2.6.1 Identifiant

Un **identifiant** est un type particulier d'*attribut* qui prends une valeur différente pour chaque entité dans la Base de Données.

> Il est représenté sous-ligné dans le modèle entité-association

2.6.2 Mono/Multi-Valués

Un *attribut* peut être :

- **Mono-valué (ou simple)** (ex : nom)
- **Multi-valué (ou répétitif)** (ex : prenoms[1..6])
- **Obligatoire** prenoms[1..3]
- **Facultatif** prenoms[0..3]

2.6.3 Atomique ou Décomposable

Atomique : Attribut non décomposable
Décomposable : Attribut que l'on peut décomposer (ex : une adresse)

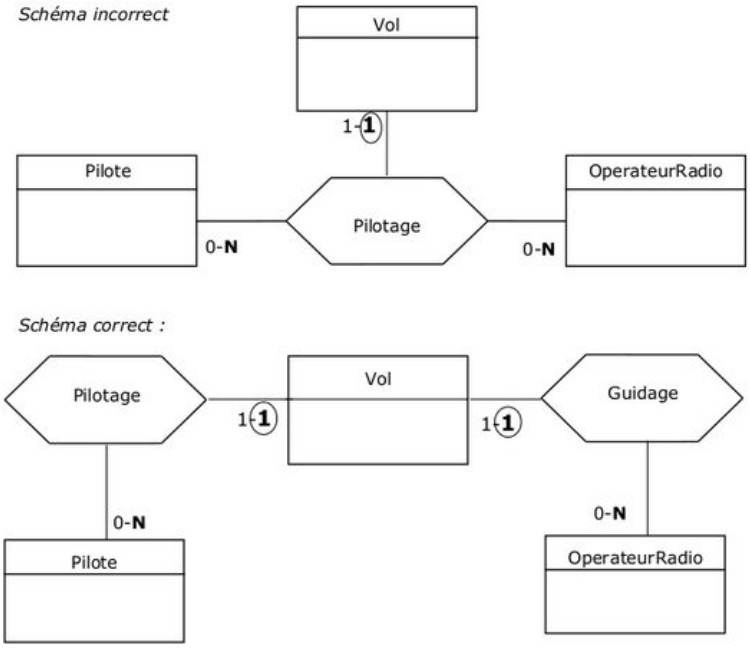
2.7 Non Redondance dans le shémas entités-association

Pour des raisons de cohérence et d'économie sur les disques on évitera toute forme de redondance pour toutes les **données déjà présentes** ou les **données calculables**

2.9 . . .

. . .

Un mauvais ternaire :



2.9.2 T.A. Cyclique

=**T.A. Réflexifs** ou **T.A.Récursifs**

2.9.3 T.A. Avec des Attributs

On peut ajouter à un T.A. N à N un attribut. Cette caractéristique décrira alors le lien qui unit les deux T.E. .

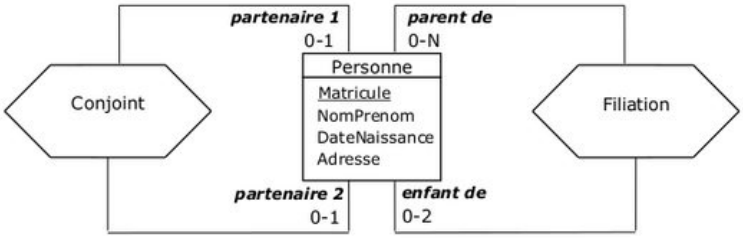
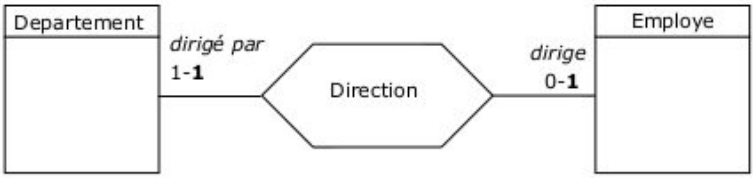


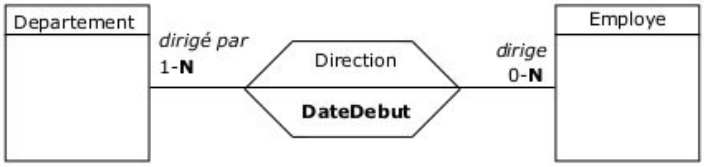
Figure 1: T.A. Cyclique

Ici le prix varie en fonction de l'article et du magasin.

On peut ainsi retenir une évolution. En transformant le shéma en un N à N.



On obtiens :



Si on à un T.A. 1 à N, il est inutile d'associer une caractéristique au T.A. Il peut être associé au T.E. avec la cardinalité maximum de 1.

Si le T.A. est facultatif du côté de la cardinalité maximum = 1, l'attribut placé dans le T.E. correspondant est alors facultatif.

2.10 Contraintes additionnelles

Le shéma est limité et ne suffit pas à lui seul, c'est pourquoi on complète le shéma avec de la documentation annexe. Cette documentation comprend la **définition**

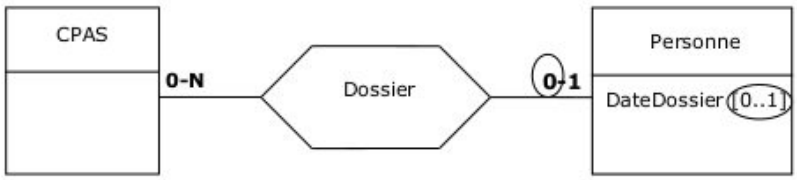


Figure 2: cardinalité minimum

des termes (T.E., T.A., Attribut) et des **contraintes additionnelles**.

On peut exprimer des *contraintes additionnelles* en Français ou en pseudo-language.

2.10.1 Identifiant avec des attributs

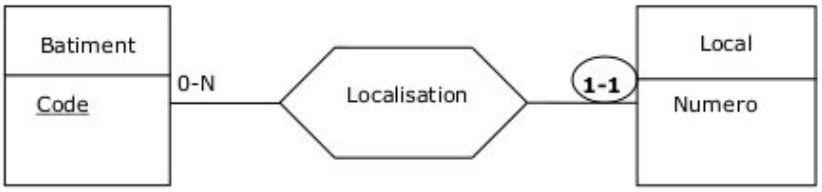
La valeur d'un identifiant est **fixe** et ne peut pas changer au cours du temps.

- Identifiant avec un seul attribut (soulignés)
- Identifiant composé d'Attributs (soulignés et reliés par une ligne verticale)
- Plusieurs identifiants

On préfère utiliser un identifiant technique qu'un identifiant composé en entreprise (Ex : ID007)

2.10.2 Identifiant Hybride

Un **identifiant hybride** est un identifiant composé de 0, 1 ou plusieurs attribut(s) **ET** un ou plusieurs rôle(s) via un ou plusieurs T.A.(s).



Le formalisme des schémas entité-association ne permet pas de représenter ce type d'identifiant **hybride**. Il faut donc avoir recours à une **contrainte additionnelle** exprimée en **pseudo-langage**.

L'identifiant du T.E. Local est spécifié via la contrainte additionnelle suivante :

id (Local) : Batiment, Numero.

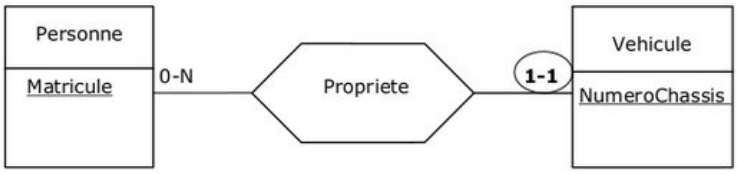
Un identifiant hybride n'est possible que s'il y a des cardinalités 1-1.

S'il existe plusieurs T.A. entre Batiment et Local, il est indispensable de préciser dans la contrainte via quel T.A. le T.E:

id (Local) : Localisation.Batiment, Numero

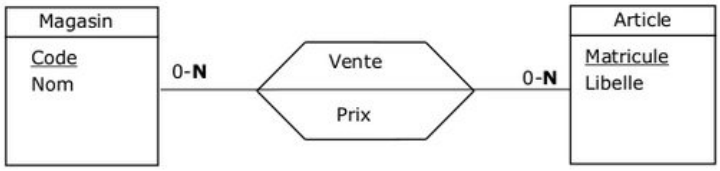
2.10.3 Identifiant de type d'association

1 à N Même idée que dans le cas d'un identifiant hybride. Un type d'association 1 à N est identifié par l'identifiant du type d'entité qui joue le rôle dont la cardinalité maximum est 1.



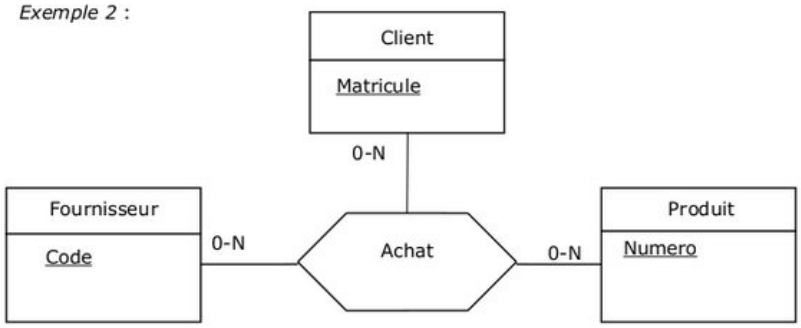
N à N Par défaut, un type d'association N à N est identifié par la combinaison des identifiants des types d'entité reliés.

Cependant on précise dans le cas d'un T.A. avec des caractéristiques son origine :

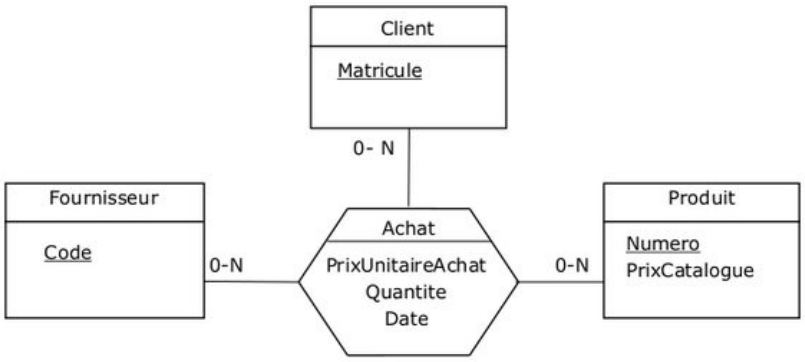


Id(Vente) : Magasin, Article

Exemple 2 :



id(Achat) : Client, Produit, Fournisseur



Cas un peu particulier :
id(Achat) : Client, Produit, Fournisseur, Date > > Il ne faut pas souligner l'attribut Date sur le schéma ; il n'est en effet pas à lui seul identifiant du T.A.

Résumé En résumé, l'identifiant à préciser pour un T.A. doit être l'identifiant **minimum**. Par conséquent, tous les attributs d'un T.A. n'interviennent pas forcément dans l'identifiant (cf quantité).
Si cet identifiant est **implicite**, on ne le précise pas sur le schéma.
Si ce n'est pas le cas (identifiant **explicite**), il s'agit d'une contrainte additionnelle qu'il faut rajouter au schéma.

2.10.4 Autres contraintes additionnelles

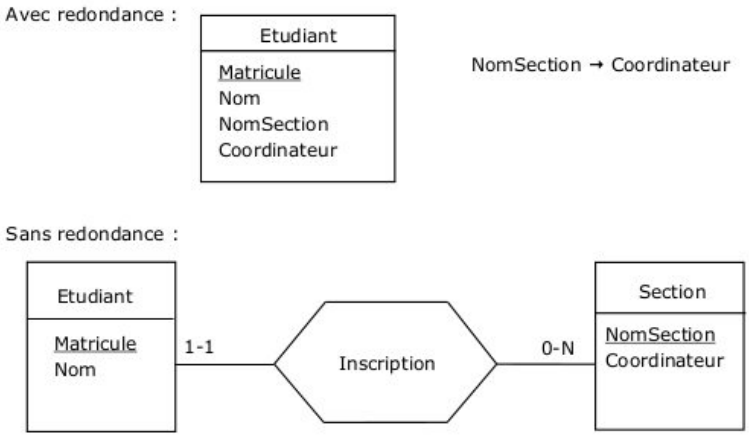
Toutes les contraintes additionnelles liées au métier doivent bien évidemment être identifiées lors de l'étape d'analyse conceptuelle.

Cependant, seules les contraintes additionnelles liées aux identifiants seront exigées lors de l'évaluation du cours de Conception de bases de données du bloc 2.

2.11 Dépendances fonctionnelles

l y a une dépendance fonctionnelle entre un groupe d'attributs dits déterminés et un attribut déterminant. Identifiant -> attributs

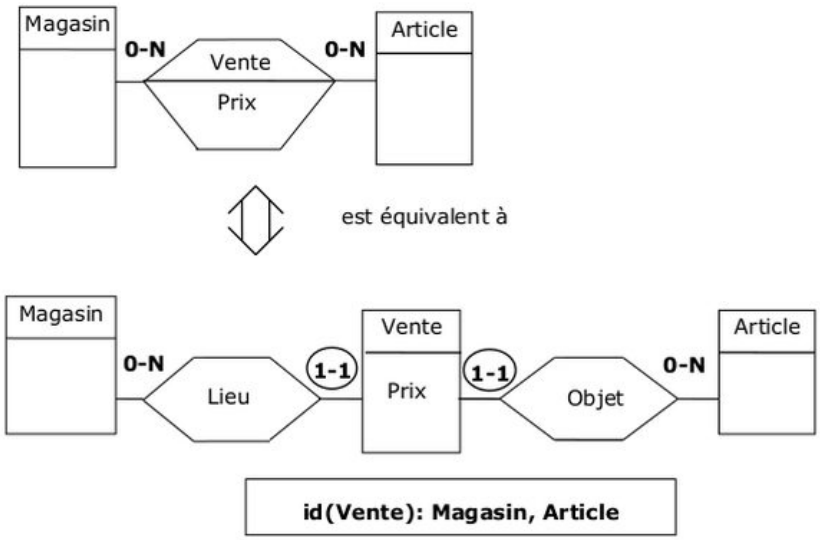
Les **dépendances fonctionnelles** entre l'identifiant du T.E. et d'autres attributs du même T.E. sont des **dépendances fonctionnelles normales**s qui n'engendrent **aucune redondance**.

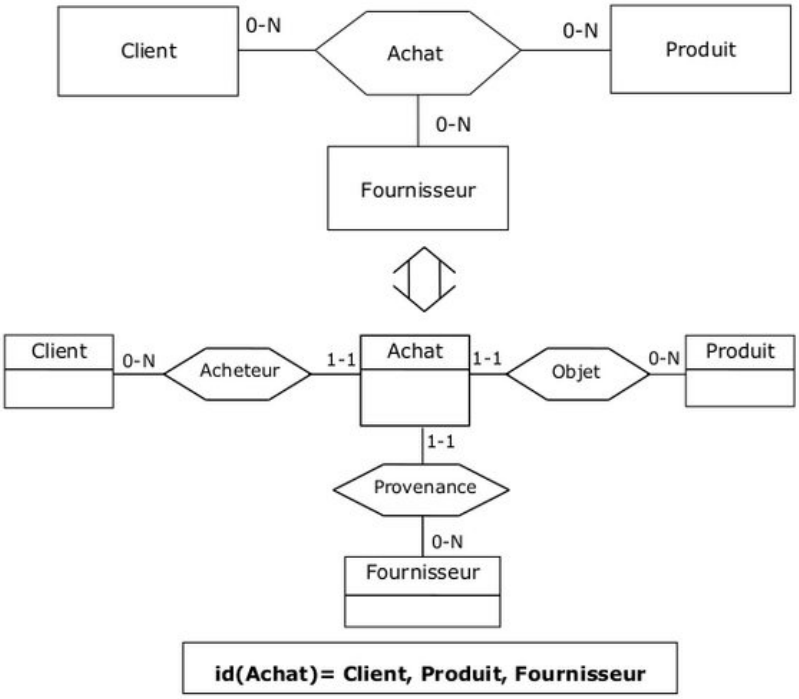


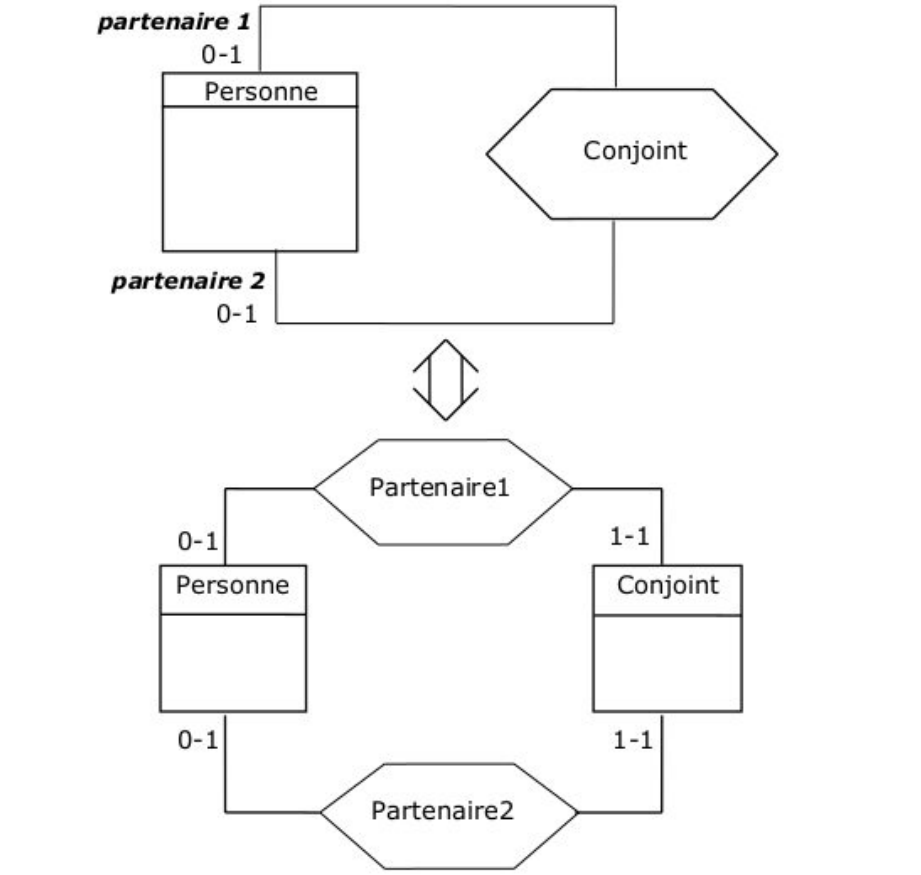
2.12 Transformation de schéma

On peut transformer un shéma sans pour autant en modifier la sémantique.

2.12.1 Transformation d'un type d'association en un type d'entité

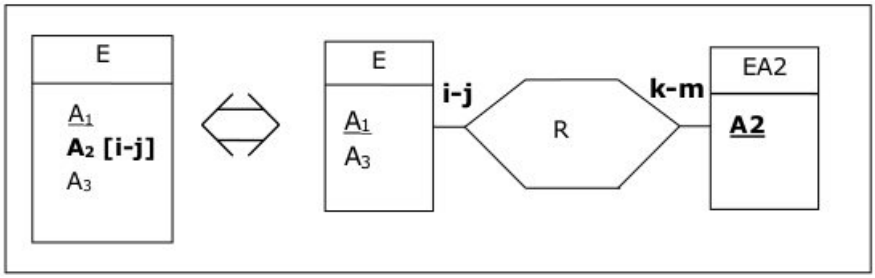



Client
0-N
Achat
0-N
Produit
0-N
Fournisseur
Client
0-N
Acheteur
1-1
Achat
1-1
Objet
0-N
Produit
1-1
Provenance
0-N
Fournisseur
id(Achat)= Client, Produit, Fournisseur



id(Conjoint) = Partenaire1.Personne, Partenaire2.Personne

2.12.2 Transformation d'attribut en type d'entité



3 Bases de données relationnelles Beaucoup de bases de donées sont de type relationnelles. Le language le plus connu pour communiquer avec ces base est le

SQL (Structured Querry Language). C'est un langage dit déclaratif (il est non procédural)

3.1 Structure de base : les tables

Les bases de données relationnelles tiennent leur nom de la structure de base qu'est la **relation**. Une relation est une structure qui rassemble **des données reliées** entre elles.

Une *table* représente un **concept**.
Une *ligne* de cette table est une **occurence**.
Une *Colonnne* est un **attribut** du concept T.E.. On en définit un *nom*, un *type*, une *longueur* et éventuellement un *domaine*. Tous les attributs sont atomiques et monovalués. La *valeur* est donnée par l'intersection d'une ligne et d'une collonne.

La valeur **NULL** peut signifier :

- Valeur de l'attribut inconnue
- L'attribut ne s'applique pas
- Certaines occurences ne possèdent pas de valeurs pour cet attribut

3.2 Ordre des colonnes

Aucun ordre n'existe. Le résultat dépendra de la requête : (ex : *select Nom, Prenom, DateNaissance, Statut, Telephone from Personne*)

3.3 Identifiant ou clé primaire

une **primary key** en SQL est un identifiant. une clé primaire est composé d'**un ou plusieurs attributs**.

Par convention, la/les colonnes seront soulignés. Et aucune clé primaire ne peut prendre de valeur NULL. Si une table possède plusieurs identifiant, on en choisit un qui sera utilisé comme clé primaire. les autres sont appelées **clé candidates** ou **alternatives**

3.4 Traduction d'un schéma conceptuel en relationnel

- Un attribut décomposable doit être éclaté.
- Les attributs multis-valués doivent être représentées sous la forme d'une autre T.E.

Exemple :

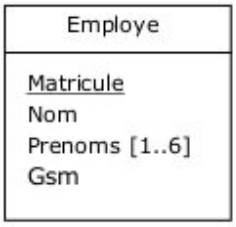


Schéma (1)

L'attribut *Prenoms[1..6]* est extrait du type d'entité *Employe* et fait l'objet d'un second type d'entité *Prenom*. Ce dernier est relié au type d'entité *Employe* par un nouveau type d'association. Le type d'entité *Employe* (schéma (1)) et le schéma (2) sont équivalents du point de vue sémantique (se référer à la section 3.4.2. pour la représentation des types d'association). Un employé peut avoir de 1 à 6 prénoms et un prénom peut être porté par plusieurs employé(s).

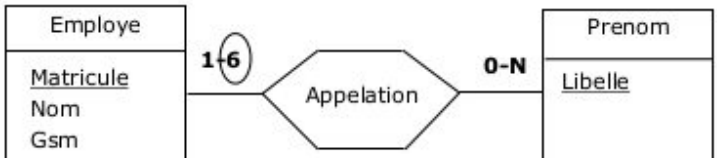
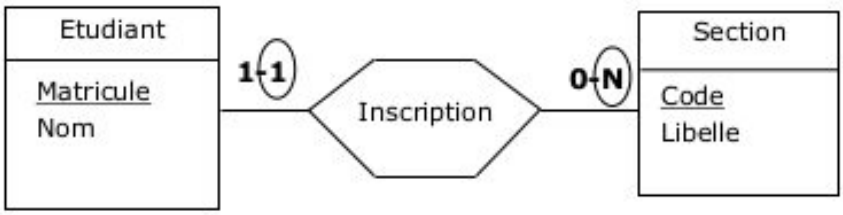
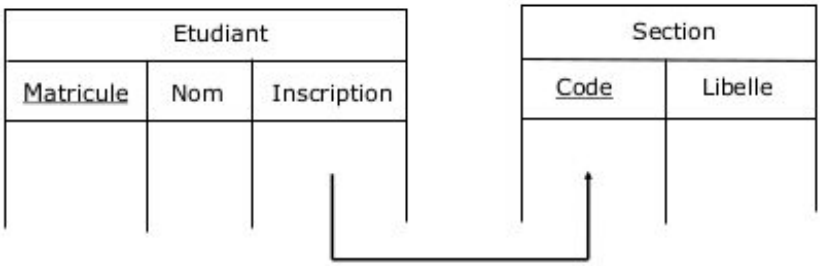


Schéma (2)

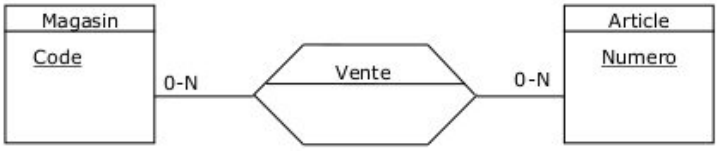
- La **clé étrangère** permets de représenter une T.A.

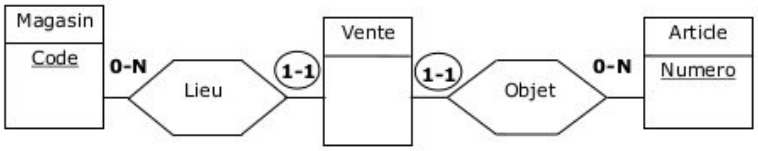




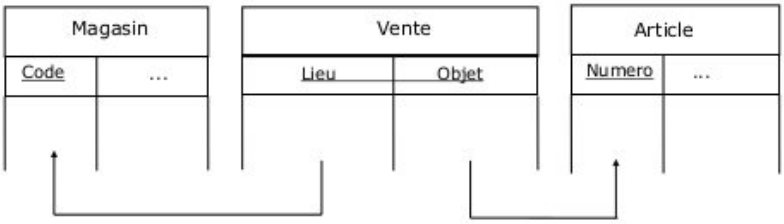
> Notons que si l'identifiant de la table reliée est composé de plus d'une colonne, la clé étrangère sera composée d'autant de colonnes.

- Il faut éclater tous les types N à N



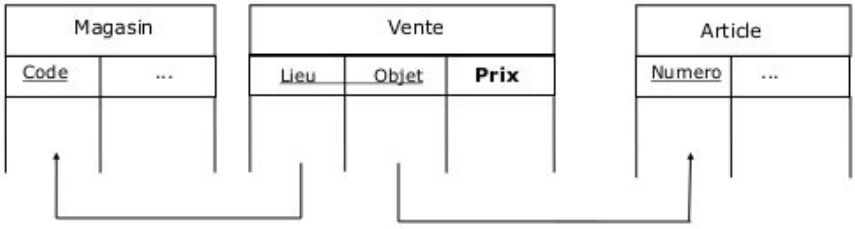


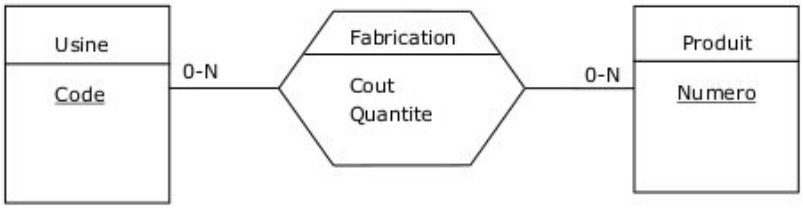
id(Vente): Magasin, Article



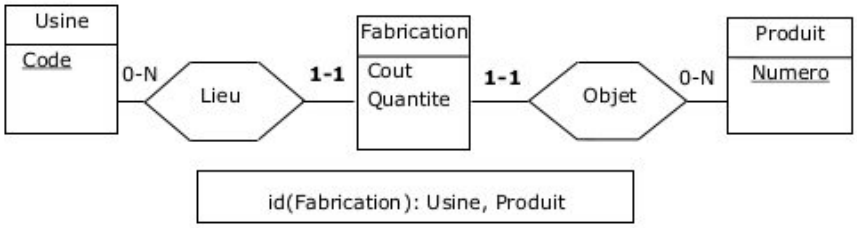
- Cas de caractéristique pour le T.A.

La traduction logique en base de données relationnelle du schéma conceptuel ci-dessus est la suivante

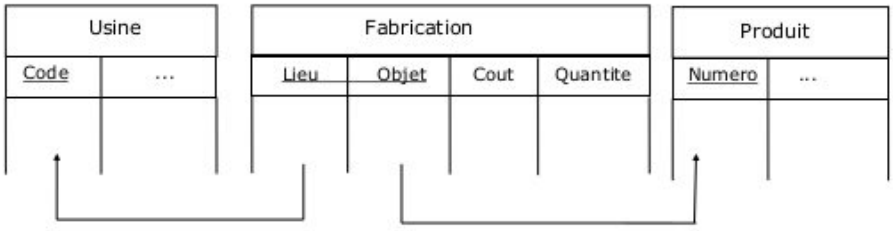




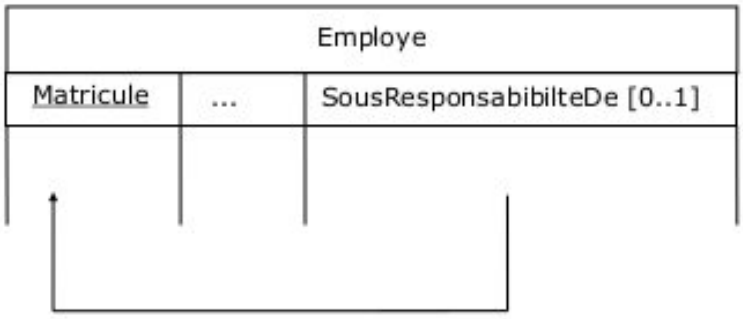
Ce schéma conceptuel doit d'abord être transformé.

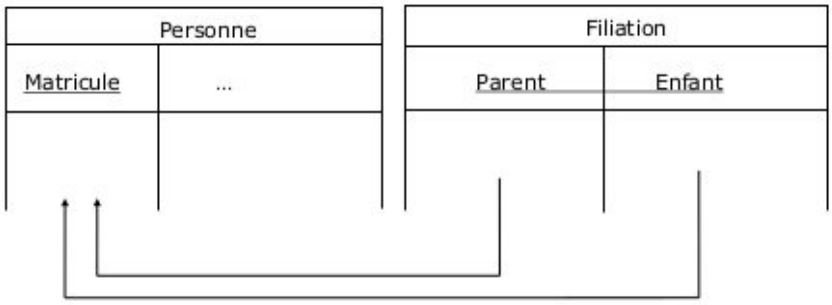


Le schéma conceptuel ainsi transformé est ensuite traduit en tables.

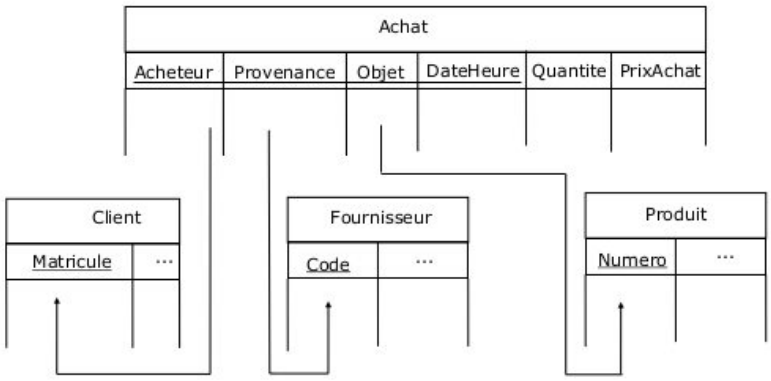


- Une T.A. Cyclique



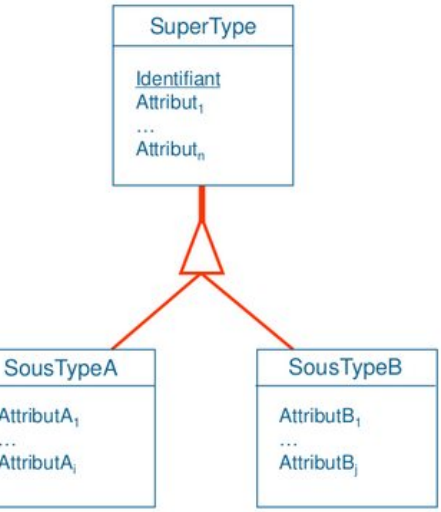


- Représentation des types d'association de degré supérieur à 2

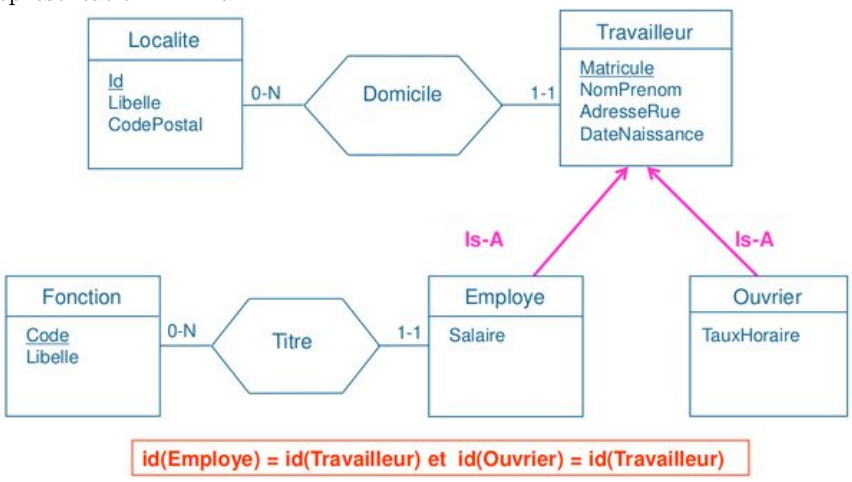


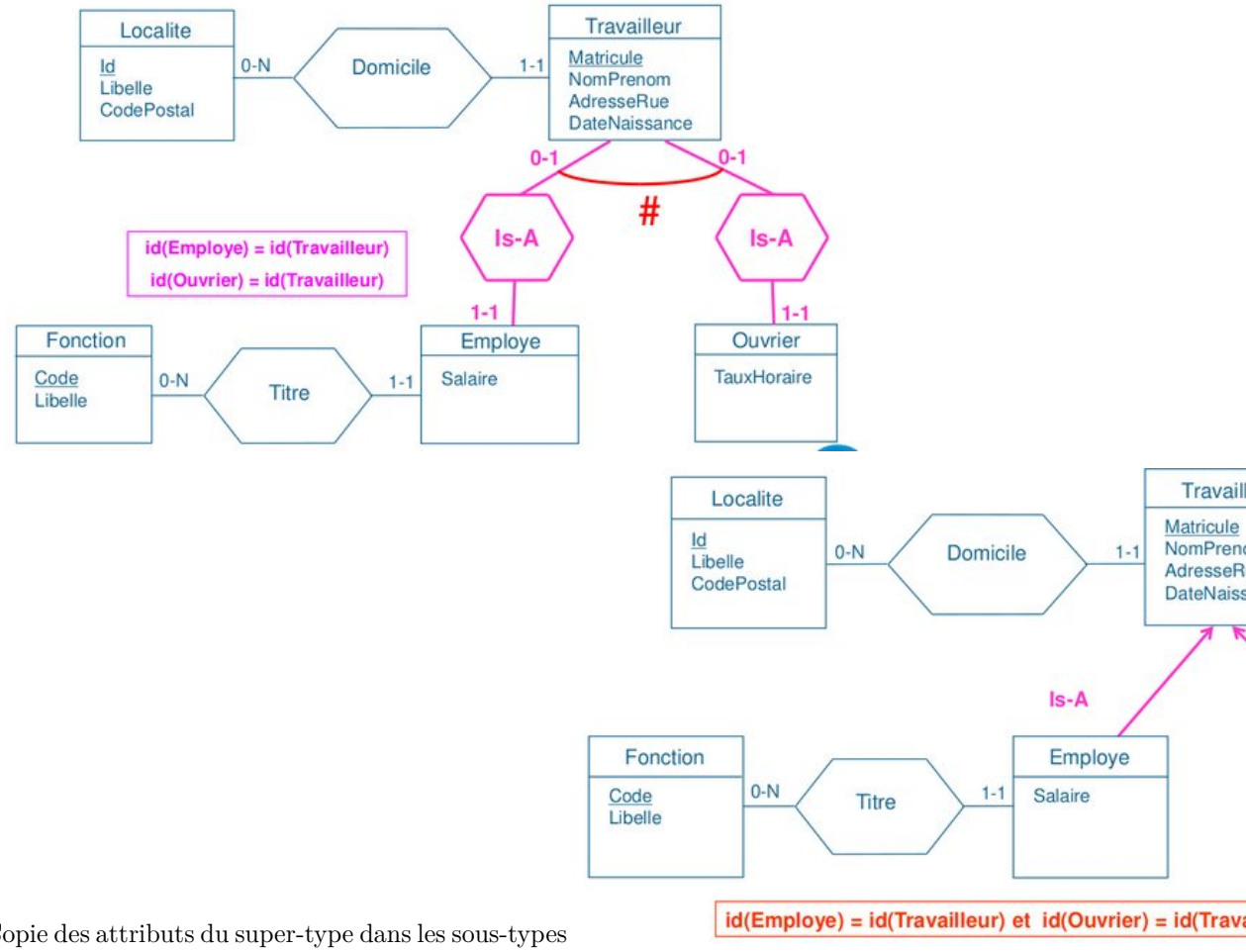
4 L'héritage dans les bases de données

Il y a héritage des attributs du super-type (héritage de l'identifiant, pas d'identifiant spécifique aux sous-types)

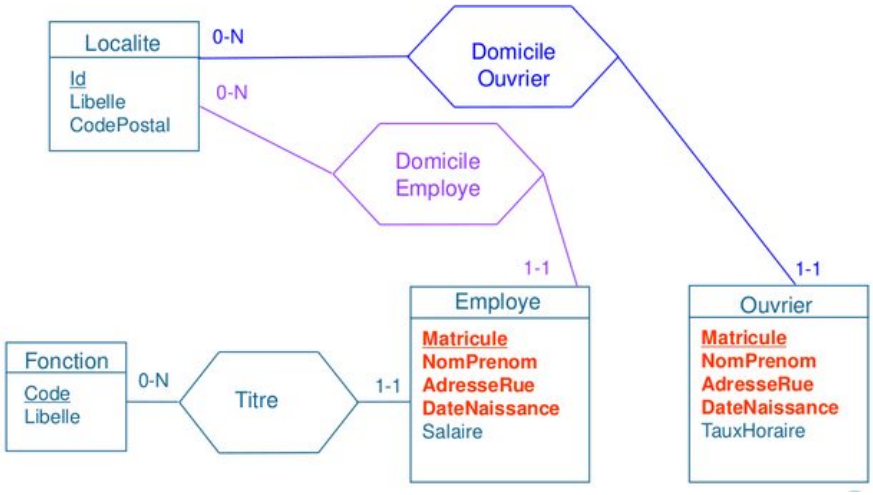


représentation DB main :

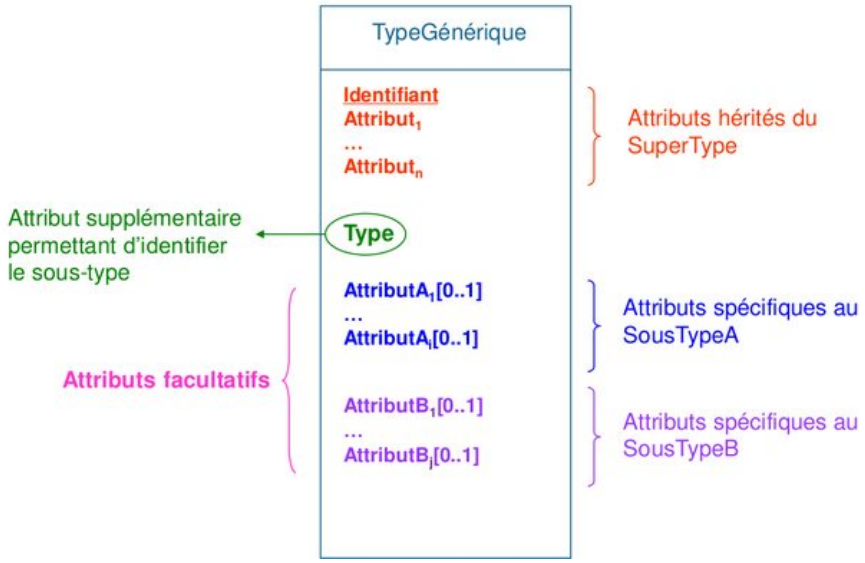




Copie des attributs du super-type dans les sous-types



4.1 utilisation d'un type générique

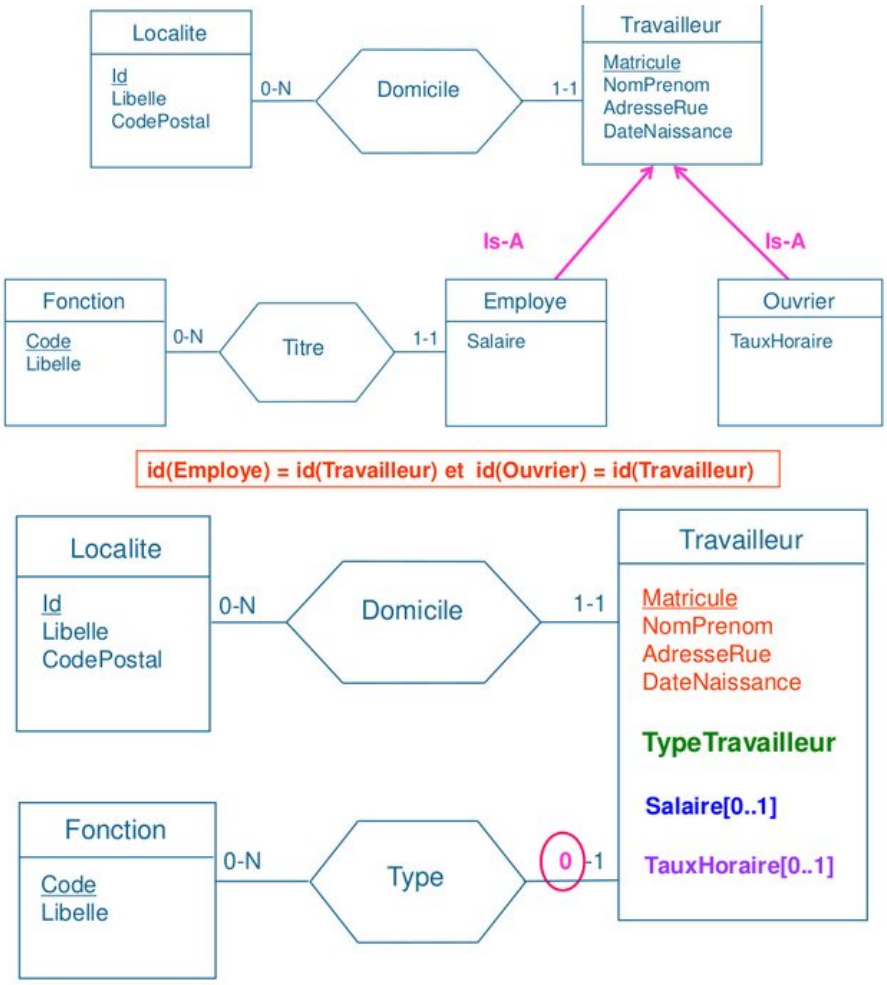


Si Type = "SousTypeA"

- Tous les AttributBi = null

Si Type = "SousTypeB"

- Tous les AttributAi = null



4.2 Quel type de transformation choisir?

Du nombre d'attributs dans

- le super-type : Si pas nombreux, représentation des types spécifiques
- les sous-types : Si pas nombreux, représentation du type générique

Du nombre de T.A. reliés

- au super-type : Si nombreux, pas de représentation des types spécifiques
- aux sous-types : Si nombreux, pas de représentation du type générique

S'il faut enregistrer des occurrences du super-type qui ne sont pas des spécialisations. La represetation d'un type spécifique ne va pas suffire.

Si contraintes de performance : bcp de tables, bcp de jointure, couteux. On

évite donc les Is-a