



Haute Ecole de Namur - Liège - Luxembourg
Département économique
Implantation IESN



Bachelier en Informatique de Gestion
2^e année

Conception de bases de données

Syllabus

Françoise Dubisy

Chapitre 1. Qu'est-ce qu'une base de données?

Un ordinateur est une machine à traiter des informations. D'où l'importance des données. Les données doivent être organisées et stockées en vue de leur manipulation et interrogation ultérieures.

1.1. Historique

1.1.1 Fichiers séquentiels

A l'origine, les données sont stockées sur des cartes perforées. Ensuite, apparaissent les bandes magnétiques. Ces deux types de supports ne permettent l'exploitation des données que de façon séquentielle.

De plus, un fichier ne reprend que les données *relatives à une seule application*.

Si plusieurs applications utilisent les mêmes informations, celles-ci se voient dupliquées entièrement ou partiellement dans plusieurs fichiers, ce qui entraîne une **redondance** d'information.

Le principal inconvénient de cette duplication d'information est le **risque d'incohérence**. Une donnée peut être modifiée dans un fichier, sans que la modification soit répercutée correctement dans tous les fichiers contenant l'information. Cela vient du fait qu'*un fichier est dépendant d'une application*; c'est donc via cette application que toute modification d'information sera répercutée dans le fichier.

En outre, il est **impossible d'interroger** ces données **dispersées** dans différents fichiers **sans aucun lien entre eux**.

1.1.2. Evolution des supports

L'organisation des fichiers évolue: le **temps d'accès** aux données **diminue** (pointeurs, index,...).

Mais chaque application gère toujours ses propres fichiers, d'où toujours le risque de *redondance* et d'*incohérence* des données.

1.1.3. Evolution des systèmes d'exploitation

Les systèmes d'exploitation ont ensuite mieux géré les ressources physiques.

A. Mémoires et périphériques d'entrée-sortie

Le programmeur est de plus en plus déchargé des tâches de gestion du matériel périphérique.

Ex: imprimantes, disques
adresse physique des fichiers

B. Multi-tâches

Le **partage de l'unité centrale** (processeur) par **plusieurs utilisateurs** ("time-sharing") permet une meilleure utilisation du processeur.

Non seulement le processeur peut être partagé, mais également **un même programme**. Dans ce cas, une seule copie du code est présente en mémoire centrale, mais elle est exécutée par plusieurs utilisateurs.

En parallèle avec les systèmes d'exploitation, ont évolué les méthodes d'accès aux données et leur organisation.

1.1.4. Evolution des méthodes d'accès aux données et de leur organisation

Un même fichier de données peut être partagé par plusieurs utilisateurs. Un gestionnaire de fichier est alors indispensable pour éviter tout conflit d'accès.

Apparaissent alors les **systèmes transactionnels** reposant sur la notion de transaction. Une transaction est une suite logique d'instructions considérée comme formant un tout.

Il y a différentes façons de gérer l'accès à un fichier par un utilisateur:

- ① soit bloquer tout le fichier
- ② soit bloquer un seul enregistrement à la fois
- ③ soit adopter le principe de la transaction:

Scénario de mise à jour d'un enregistrement:

Début transaction

<i>lire</i> l'enregistrement	→	enregistrement bloqué uniquement le temps d'en faire une copie (copie1)
<i>modifier</i> l'enregistrement	→	création d'une seconde copie (copie2); puis appliquer les modifications à la copie2

Fin transaction

→ pour s'assurer que l'enregistrement du fichier n'a pas été modifié entre temps, une troisième copie (copie3) est faite:
si copie1 = copie3 alors écriture dans le fichier de copie2
sinon avertir l'utilisateur

1.1.5. Systèmes documentaires

Il s'agit d'un type de gestionnaire de fichiers plus élaboré.

En 1969, apparaît le concept de **banque de données**. Il s'agit de données mises à la disposition d'un grand nombre d'utilisateurs. Ces données sont cependant *faiblement structurées*. La consultation se fait uniquement via des **mots-clé** et des **dictionnaires**.

ex: répertoires de références bibliographiques

1.1.6. Systèmes de gestion de bases de données

En 1970, IBM propose un système *qui dispense le programmeur de naviguer jusqu'à l'information désirée*. C'est le **système de gestion de base de données** qui se charge de trouver le *chemin d'accès optimal*.

ex.: systèmes relationnels (langage SQL)

1.2. Inconvénients des systèmes de fichiers

① Dépendance des applications

Chaque application possède ses propres fichiers. Chaque fichier est généralement mis à jour par l'application qui l'a créé.

② Redondance des données

Même si deux applications ont besoin des **mêmes informations**, celles-ci se trouveront **dans deux fichiers** différents souvent *de structures différentes*.

③ Risque d'incohérence des données

La **duplication** des données dans des fichiers différents entraîne un risque d'**incohérence** si la modification d'une donnée n'est pas répercutée dans tous les fichiers.

④ Impossibilité d'interroger les données

Les données sont dispersées dans différents fichiers qui n'ont aucun lien entre eux.

⑤ Sécurité

Chaque utilisateur sauve ses propres fichiers avec une fréquence qui lui est propre.

⑥ Pas de modification de structure possible

Une fois un fichier créé, il est **impossible de modifier la définition des types d'enregistrement**.

ex.: ajouter/retirer un champ à tous les enregistrements d'un fichier.

Si une telle modification doit cependant avoir lieu, le seul recours pour le programmeur est de modifier le programme et de copier le fichier dans un nouveau fichier ayant la structure voulue.

Mais l'inconvénient principal est le suivant.

●* ⑦ Pas de relations entre fichiers

Il est difficile de mettre en relation les données de deux fichiers différents. On ne peut établir de relations logiques entre fichiers en vue de les exploiter via des interrogations (cfr section 1.3).

1.3. Liens entre les données

Quelques définitions:

DONNEE: enregistrement dans un **code** convenu d'une observation, un fait, un objet, une image, un son, un texte, un concept, ...
qui convient à une *communication*, une *interprétation* ou un *traitement* par l'homme ou la machine
en vue de *transmettre* ou de *stocker* de l'information, ou encore d'en *déduire* de nouvelles informations.

INFORMATION: sens, signification que l'on attache à ou que l'on déduit d'un ensemble de données ou d'associations entre elles.

L'aspect **subjectif** de la notion d'information est à souligner.

FICHER: ensemble de données utilisables en général par une seule application.

Supposons que l'on doive gérer une bibliothèque et qu'on nous demande de pouvoir répondre aux questions ci-dessous. Comment structurer les données?

① Etant donné un ouvrage, trouver le nom de l'auteur

⇒ Prévoir un ensemble de fiches ouvrage

Ex:

<i>Titre:</i>	Le lotus bleu
<i>Catégorie:</i>	BD
<i>Editeur:</i>	Casterman
<i>Auteur:</i>	Hergé

Il suffit de prévoir la caractéristique *Auteur* sur la fiche de chaque ouvrage.

② Etant donné un ouvrage, retrouver toutes les caractéristiques de l'auteur

☹ *Recopier toutes les caractéristiques de l'auteur sur les fiches de chacun de ses ouvrages!*

Cela induit de la redondance. En effet, les informations concernant un même auteur seront **dupliquées**. Par exemple, les informations concernant Hergé se retrouveront enregistrées autant de fois dans la base de données qu'il y a d'ouvrages de Hergé dans la bibliothèque.

Cette redondance est difficile à gérer; en effet, la moindre modification concernant un auteur devra être effectuée à plusieurs endroits dans la base de données. Le moindre oubli placera la base de données dans un état incohérent.

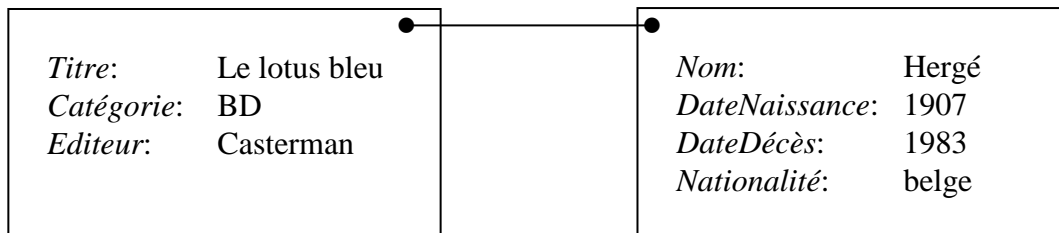
⇒ à éviter!

☺ *Prévoir une fiche par auteur et relier l'ouvrage à son auteur*

La recherche s'effectue en deux étapes:

- rechercher la bonne fiche ouvrage
- retrouver la fiche de l'auteur correspondant.

Ex:



On ne stocke qu'une fois les caractéristiques d'un auteur: il n'y a donc pas de problèmes de redondance. On relie ensuite la fiche auteur à une fiche ouvrage.

Il serait intéressant de disposer d'un système qui permette de gérer ce type de lien, à savoir:

A l'encodage :

A la création d'un nouvel ouvrage, il faudrait pouvoir préciser quel en est l'auteur, c'est-à-dire établir/mémoriser le lien entre la nouvelle fiche ouvrage et la fiche auteur correspondante.

☛ De plus, il serait intéressant que dans un souci de cohérence, le système **refuse** de relier le nouvel ouvrage à un auteur **dont la fiche n'existe pas dans la base de données**.

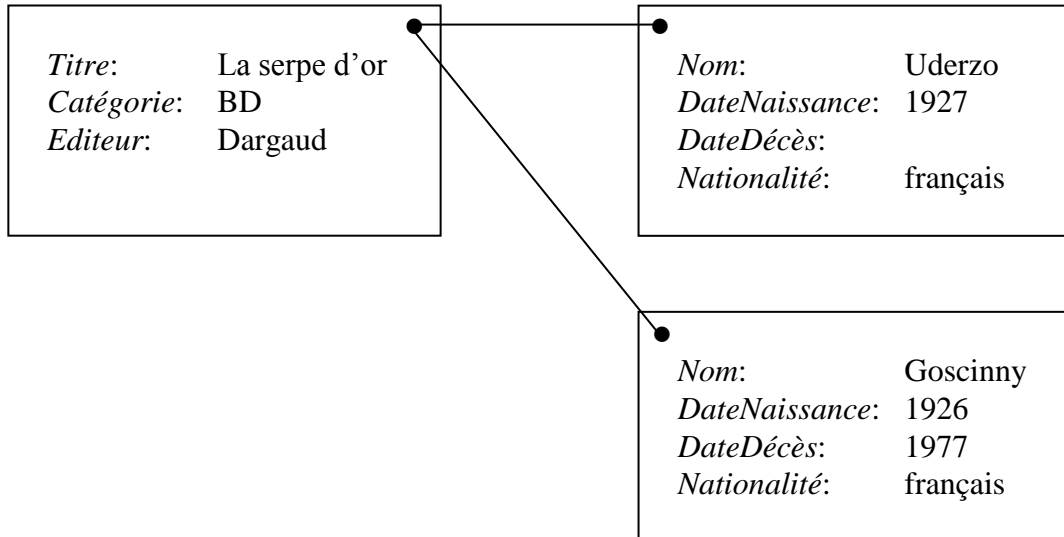
A l'utilisation (lors de la recherche d'information) :

Il faudrait un système capable, lors de l'interrogation de la base de données, d'exploiter ce lien, c'est-à-dire capable de retrouver toutes les caractéristiques de la fiche auteur liée à un ouvrage donné.

③ Etant donné un ouvrage, retrouver toutes les caractéristiques de ses auteurs

Le principe reste le même: on relie la fiche ouvrage *aux fiches* correspondant aux auteurs de l'ouvrage.

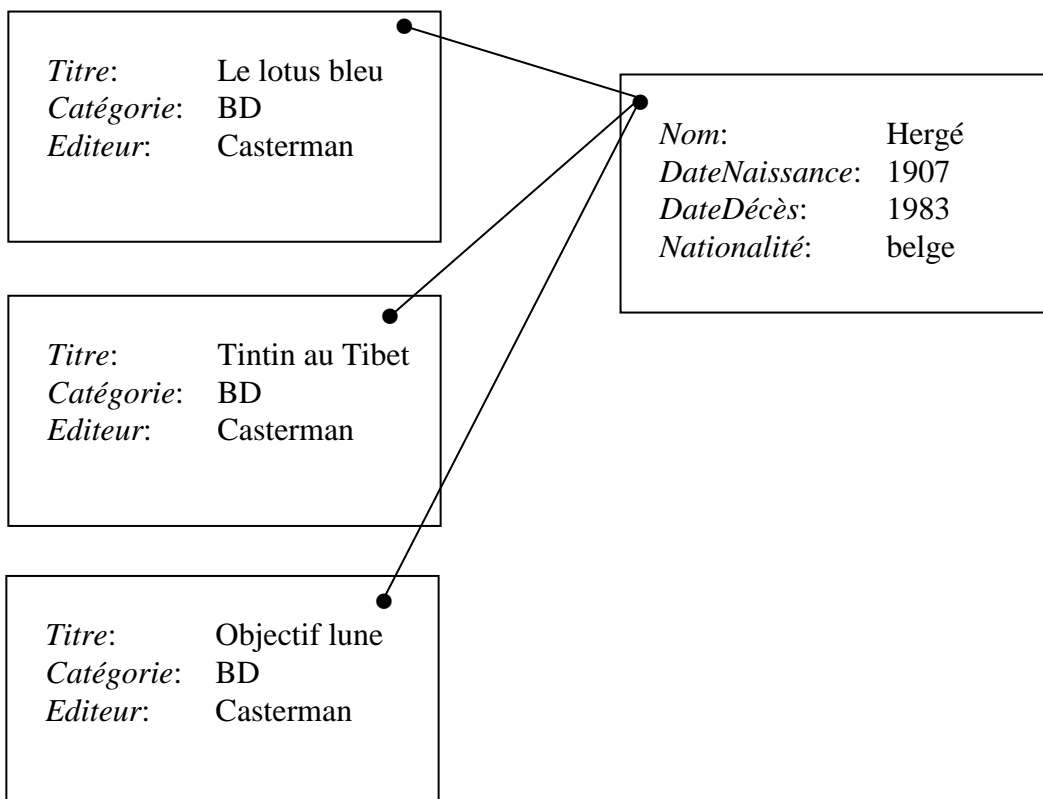
Ex :



④ Etant donné un auteur, retrouver tous ses ouvrages

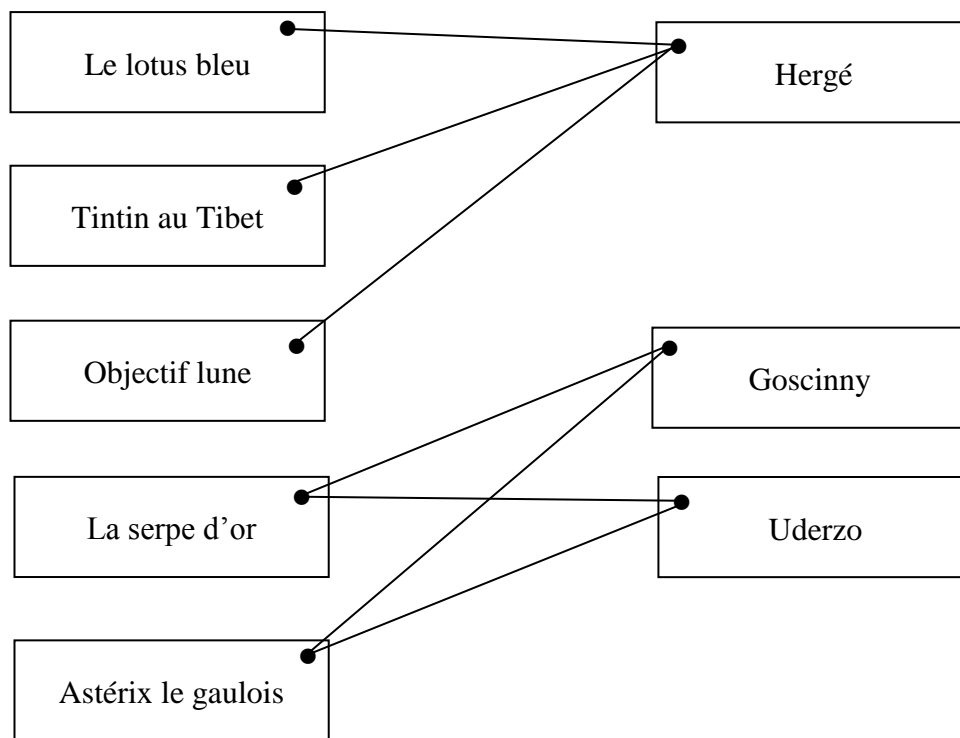
Il n'y a rien de plus à prévoir. Cette situation est une généralisation des points ① et ②.

Ex:

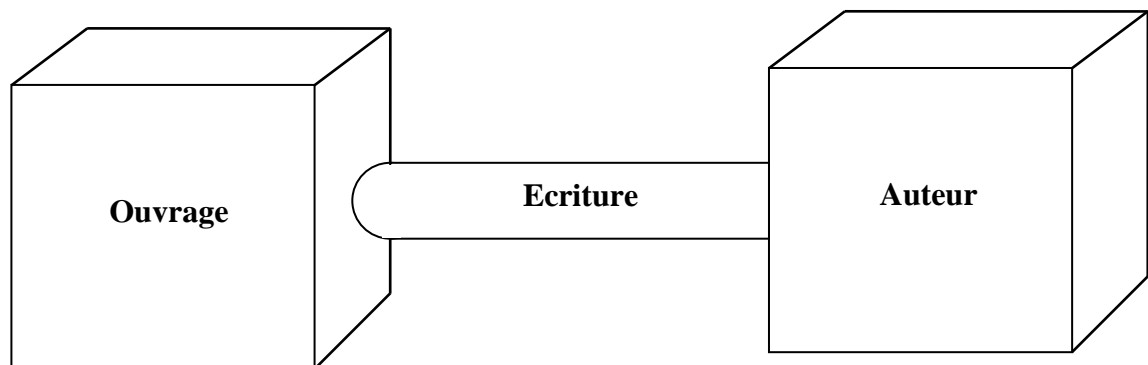


Par conséquent, une base de données est un ensemble de données reliées.

Au niveau fiches:



Au niveau types d'enregistrement:



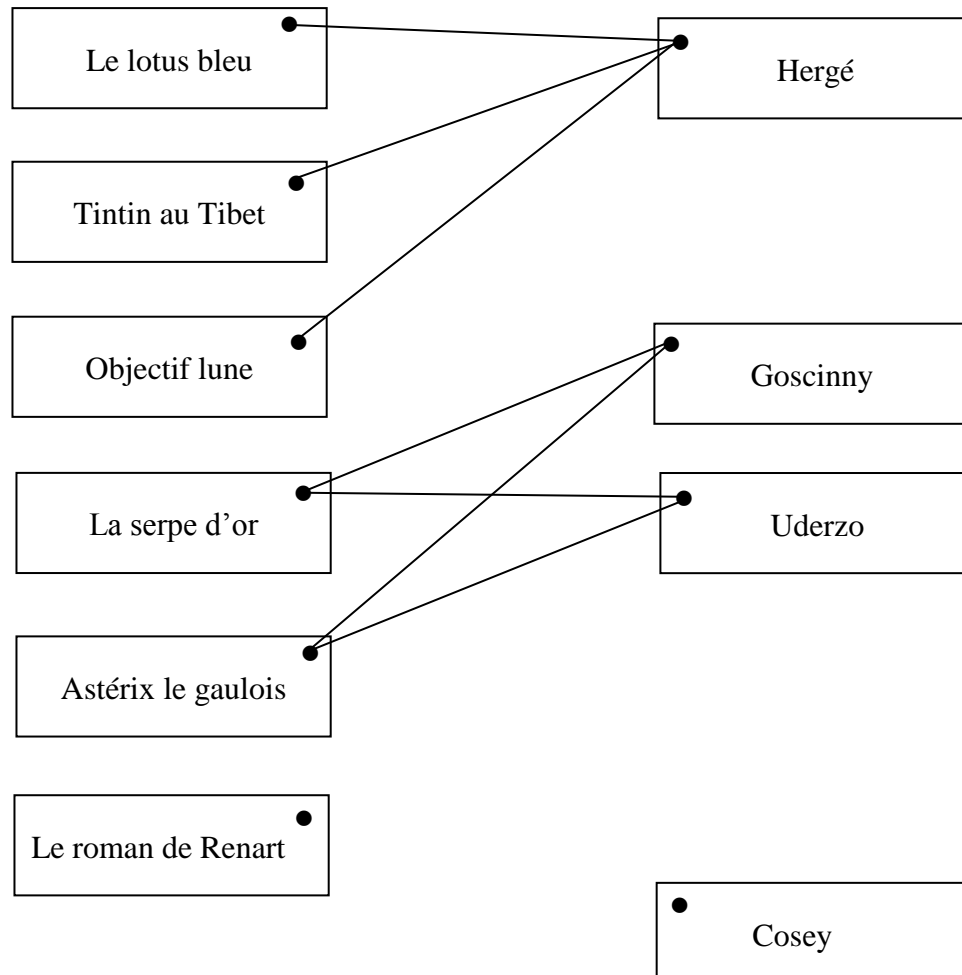
Lors de la création d'une base de données, on définit les liens **possibles** entre les objets.

Définir un lien *écriture* entre le type d'objets *ouvrage* et le type d'objets *auteur* c'est préciser qu'il est *possible* que des liens d'écriture existent entre des objets (fiches) qui sont du type *ouvrage* et des objets (fiches) qui sont du type *auteur*.

N.B. Cela ne signifie pas que tout objet du type ouvrage (càd toute fiche ouvrage) a forcément au moins un lien avec un objet du type auteur (càd une fiche auteur), et inversement. Par conséquent, pourrait être repris dans la base de données un ouvrage dont l'auteur est inconnu (ex: le roman de Renart) ou encore, pourrait être repris dans la base de données un

auteur ainsi que ses coordonnées, alors que la bibliothèque ne possède (encore) aucun de ses ouvrages (ex: Cosey).

La base de données pourrait contenir alors l'ensemble des fiches suivant:



1.4. Caractéristiques d'une base de données (BD)

Quels sont les objectifs d'une structuration des données sous forme de base de données? Une base de données aura les caractéristiques suivantes.

① Relations entre les données

(cfr précédemment)

② Sauvegarde des données sur un support (disque)

Les données doivent pouvoir être **mémorisées** en vue d'une réutilisation future. Elles doivent également pouvoir être **facilement modifiables**.

③ Partage des données entre plusieurs utilisateurs

Les données sont **centralisées** et partageables (multi-users). Cela a pour conséquence une gestion de la **sécurité** et de la confidentialité des données via les mécanismes *des droits d'accès*.

④ Indépendance des données par rapport aux applications

Les données ne sont plus envisagées que pour une application déterminée.

⑤ Sans redondance inutile

Toute redondance d'information doit être évitée, **sauf** pour des raisons de **sécurité** et de **performance**.

⑥ Contrôle de cohérence

La nécessité d'un contrôle de cohérence est *évidente en cas de redondance* (une modification doit être répercutée automatiquement à plusieurs endroits).

De plus, des **contraintes** additionnelles peuvent accompagner la définition d'une base de données. Ces contraintes devront être respectées lors de toute modification de données de la B.D. C'est l'administrateur du système qui définit les contraintes additionnelles, mais c'est le système de gestion de la base de données qui se charge de les vérifier.

⑦ Exploitation des données par interrogation

Toutes les données d'une B.D. sont "directement accessibles"; une B.D. est interrogeable sur **n'importe quel champ**. La recherche du *chemin d'accès optimal* n'est plus à la charge du programmeur.

⑧ Longueur des enregistrements variable

Dans un enregistrement, il peut y avoir des informations manquantes ou inconnues (valeur NULL).

*N.B. Ne pas confondre valeur **null** et*

valeur "blanc "	pour un champ alphanumérique
valeur "0 "	pour un champ numérique

⑨ Modification possible de la structure des enregistrements

Il est possible **d'ajouter, modifier ou supprimer un champ** d'un type d'enregistrement existant.

Une définition générale d'une base de données pourrait alors être:

Une BD est

"une collection de données en relation, indépendantes des applications et sans redondance inutile, partageable entre plusieurs utilisateurs, dont n'importe quel contenu peut être retrouvé en réponse à une question. "

1.5. Difficultés d'identification des concepts et des liens

Reprenons la gestion d'une bibliothèque. Envisageons un ou l'autre scénario typique.

① Des emprunteurs empruntent des exemplaires d'ouvrages

On ne considère que les emprunts en cours.

Il faut par exemple pouvoir gérer les cas suivants:

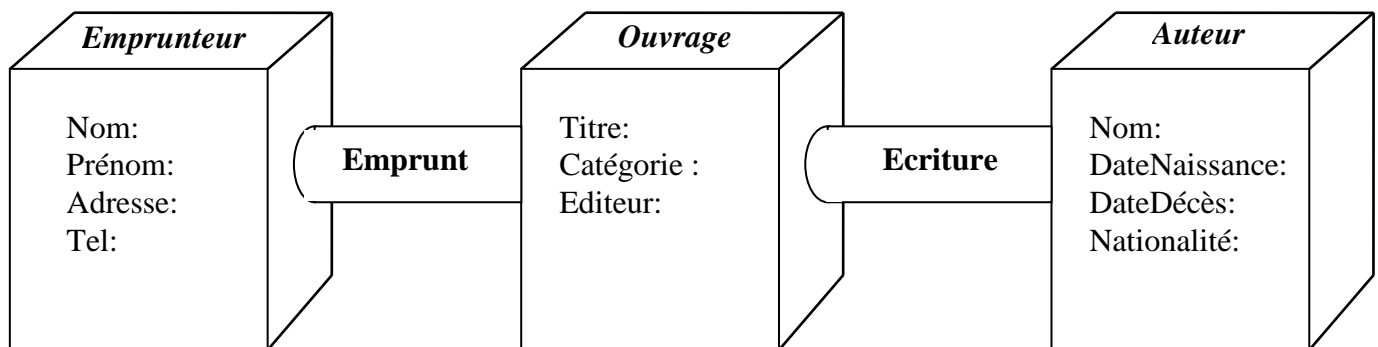
- Jules Dupond a emprunté l'unique exemplaire de la Serpe d'or et un exemplaire de Tintin au Tibet
- Marie Martin a emprunté un autre exemplaire de Tintin au Tibet
- John Leroy est inscrit comme emprunteur, mais n'a rien emprunté pour l'instant.

On doit pouvoir mémoriser ces emprunts et retrouver qui a emprunté quel ouvrage. On doit également pouvoir retrouver les caractéristiques des auteurs de chaque ouvrage.

Combien de types d'objet (càd de types d'enregistrement) faut-il prévoir? Lesquels?

Quels liens prévoir entre eux?

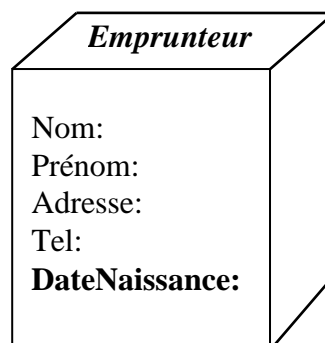
Quelles caractéristiques prévoir pour chaque type d'objet?



② Réduction de tarif selon le type d'emprunteur

On propose une réduction du tarif de 50% pour les plus de 60 ans, et de 30% pour les moins de 25 ans.

⇒ il suffit de prévoir une caractéristique supplémentaire pour le type d'objet emprunteur, à savoir la date de naissance.

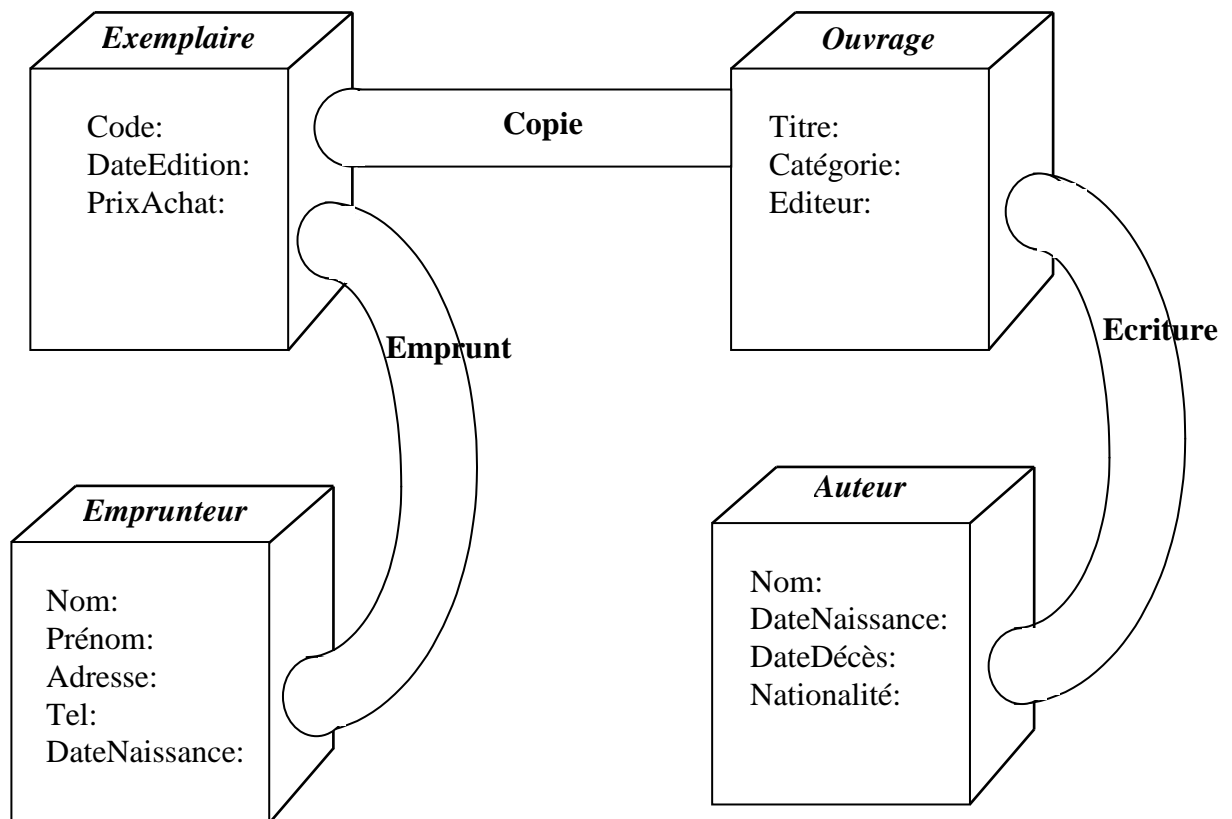


③ Retrouver la date d'édition et le prix d'achat des exemplaires empruntés

Il faut donc pouvoir identifier non plus uniquement l'ouvrage emprunté, mais quel exemplaire est emprunté. En effet, un même ouvrage peut être présent en plusieurs exemplaires dans la bibliothèque.

Contrairement au point ①, on ne mémorise plus quel ouvrage a emprunté un emprunteur, mais bien quel exemplaire de l'ouvrage ce dernier a emprunté. Il faut donc prévoir un type d'objet supplémentaire, à savoir *Exemplaire*.

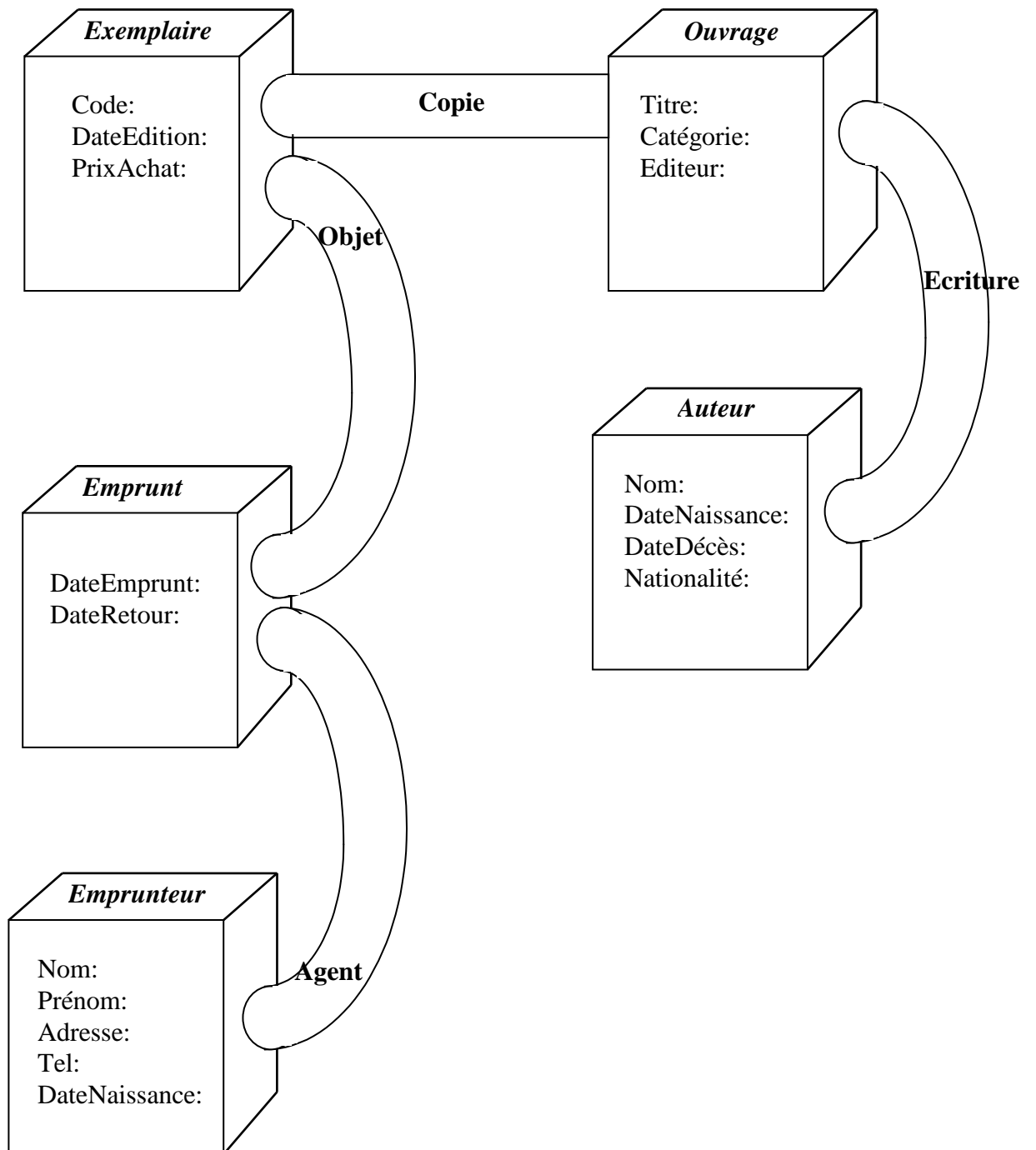
Le type d'objet *Exemplaire* est relié au type d'objet *Ouvrage*. Et le type d'objet *Emprunteur* n'est plus relié au type d'objet *Ouvrage*, mais bien au type d'objet *Exemplaire*.



④ Historique des emprunts

Supposons que, à des fins statistiques, on désire garder trace de tous les emprunts, c'est-à-dire les emprunts en cours mais également les emprunts antérieurs.

Il faut prévoir un type d'objet supplémentaire qui représentera cette notion d'emprunt. Les caractéristiques de ce nouveau type d'objet devront au moins reprendre la date d'emprunt et la date de retour. Cette dernière date permettra de distinguer les emprunts en cours des anciens emprunts : en effet, une date de retour sans valeur (inconnue) signifie qu'il s'agit d'un emprunt en cours. La date d'emprunt quant à elle permet d'identifier parmi les emprunts en cours ceux qui sont en retard.



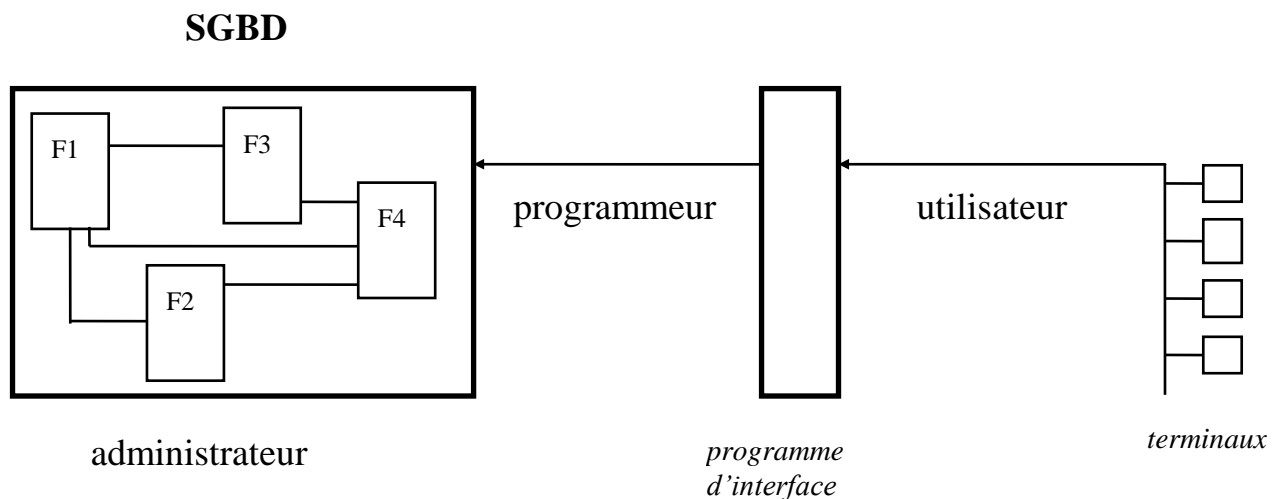
1.6. Systèmes de gestion de bases de données (SGBD)

Un SGBD est un outil permettant de gérer une base de données, à savoir, principalement:

- créer et supprimer des fichiers
- insérer, effacer et modifier des enregistrements dans des fichiers existants
- rechercher des données

Toute base de données gérée par un SGBD doit (devrait) présenter les différentes fonctionnalités décrites au point 1.4.

La figure 1.5. illustre l'interaction d'un SGBD avec les différents agents.



Passons en revue les rôles des différents intervenants.

1.6.1. Programmeur

C'est l'informaticien qui interagit directement avec le SGBD au moyen de deux types d'outils:

① Langage de programmation classique (Cobol, C, ...)

Le programme d'application (interface utilisateur: écrans, menus, ...) sera écrit au moyen de langages de programmation dits PROCEDURAUX. Les langages procéduraux spécifient le **comment faire** à travers un algorithme détaillant à l'ordinateur la marche à suivre.

Ces programmes dépendent du domaine d'application. Par exemple, dans le domaine bancaire, il s'agira de programme de transfert d'argent entre deux comptes, de commandes de chèques, ...

② Langages d'exploitation de bases de données

Le langage SQL (**S**tructured **Q**uery **L**angage) est un langage d'accès normalisé aux bases de données relationnelles. Ce n'est pas un langage de programmation à proprement parlé (SQL est NON PROCEDURAL), mais un langage de requêtes basé sur la théorie des ensembles et exprimé sous forme **déclarative**. Il ne faut plus spécifier le comment faire, mais on se contente de dire **quoi faire**. En d'autres termes, on se contente de spécifier les données que l'on désire obtenir, sans plus se soucier du chemin à parcourir pour y arriver. Par exemple, ne sera plus à la charge du programmeur l'écriture de la boucle qui passe en revue séquentiellement les enregistrements d'un fichier jusqu'à trouver ceux qui répondent aux critères de recherche. Le programmeur se contentera de spécifier au SGBD les critères de recherche.

1.6.2. Utilisateur

Il s'agit de non informaticiens qui vont effectuer des opérations élémentaires de routine sur la BD via des terminaux. Ils sont donc utilisateurs des programmes d'interface créés par les programmeurs. Dans le domaine bancaire, les utilisateurs seront par exemple, le gérant d'une agence qui effectue le transfert d'un compte client vers une autre agence, ou du client bancontact qui demande le montant de son compte.

L'utilisation se fait essentiellement au moyen de menus.

L'utilisateur a accès à une partie restreinte de la BD. Ainsi, le gérant de la banque a accès aux informations concernant tous les clients de l'agence, et le client bancontact aux informations relatives uniquement à son compte.

La sécurité et la confidentialité des données sont assurées par des codes d'accès (ex: carte et mot de passe bancontact).

1.6.3. Administrateur

L'administrateur est la personne responsable de l'ensemble du système.

Ses fonctions sont les suivantes:

① Création de la structure originale de la BD et organisation des fichiers

Via le **Data Definition Language** (DDL) (cfr: *create table*, *drop table* en SQL)

② Modification de la structure de la BD

Ex: ajout/ suppression d'un champ dans un fichier

également via le **Data Definition Language** (cfr *alter table* en SQL)

③ Gestion des accès

C'est l'administrateur qui donne les droits d'accès à la BD aux utilisateurs finaux et programmeurs.

④ Backup et entretien de la BD

1.6.4. S.G.B.D.

Le système de gestion de bases de données s'occupe de la gestion physique des fichiers. Ses fonctions sont les suivantes:

① Accès optimal à toute donnée

Le SGBD tente de répondre aux demandes des utilisateurs, en recherchant le chemin optimal.

② Traitement simultané des données

Les données sont partagées par plusieurs utilisateurs. La gestion des accès concurrents est à la charge du SGBD.

③ Validité et cohérence des données

Les contraintes éventuelles définies par l'administrateur (à la demande des concepteurs de la BD) doivent être vérifiées à tout moment. Des exemples de telles contraintes pourraient être: "Si l'attribut *état* d'un exemplaire d'ouvrage a pour valeur "*en réparation*", l'exemplaire correspondant ne peut être emprunté", "L'âge d'un étudiant ne peut être inférieur à 17 ans", "Aucun compte ne peut avoir un solde inférieur à -200.000.",... C'est le SGBD qui se charge de les vérifier à chaque modification des données. *Toute demande de modification de données qui ne satisferait pas ces contraintes est rejetée.*

④ Sécurité des données

L'administrateur définit les droits des utilisateurs. A chaque demande d'accès (en lecture ou en écriture) de la part d'un utilisateur, le SGBD vérifie ses droits.

⑤ Sauvegarde et récupération

En cas de pannes ou d'erreurs (logiciel ou matériel), le SGBD doit pouvoir garantir de restaurer les données dans un état **antérieur cohérent**. Par exemple, en cas de panne au milieu d'une transaction, le SGBD doit pouvoir annuler la transaction et donc "replacer" la BD dans l'état (cohérent) précédant la transaction.

Partie 1

Analyse conceptuelle

Chapitre 2. Le modèle entités-associations

2.1. Introduction

Face à une application à créer, il faut en établir les spécifications exactes, c'est-à-dire, définir le cahier des charges, à partir de l'étude du domaine d'application et de l'interview des utilisateurs.

L'analyse d'une application peut être divisée en deux parties majeures: *l'analyse des traitements* (analyse fonctionnelle) et *l'analyse des données*.

La conception d'une base de données se fera en trois étapes:

- la définition des concepts et leurs liens
- la définition d'un schéma conforme au S.G.B.D. choisi
Par exemple, la transposition des concepts en tables, si le S.G.B.D. choisi est de type relationnel.
Un schéma, dit schéma *logique*, est alors produit par l'administrateur du système en collaboration avec les analystes.
- la production du schéma physique correspondant
(en S.G.B.D. relationnel, il s'agit de la création physique des tables en SQL)

Il serait intéressant de disposer d'un outil permettant de construire des schémas de données indépendamment de tout S.G.B.D.. Ce type de schéma, appelé schéma **conceptuel**, devrait mettre en évidence les concepts importants et leurs relations les uns avec les autres. Un schéma conceptuel sert de support commun aux différents intervenants: analystes, administrateurs de B.D, programmeurs, et, dans une moindre mesure, aux utilisateurs interviewés (cfr fig. 1).

Un des outils les plus utilisés pour établir un schéma conceptuel est le **modèle entités-associations**. Il trouve son origine dans les travaux de Bachman (1969), et s'est développé pendant les années 70.

Le modèle entités-associations est “*un modèle qui permet d'exprimer la «sémantique» des données mémorisées et/ou véhiculables à l'aide des concepts **d'entités**, **d'association**, **d'attributs** et des mécanismes des **contraintes d'intégrité**.*”

Rappel:

La notion de donnée est différente de la notion d'information.

- *Donnée*: représentation (enregistrement) codée des propriétés d'un concept, d'un objet, d'un fait.
- *Information*: signification (potentielle) attachée à une donnée.

La “*sémantique*” est le *sens*, la *signification* attribuée aux données.

Le but d'un schéma conceptuel est de représenter facilement, sous forme d'un schéma plus lisible, les objets au sens large et leurs relations (liens, associations).

Un tel schéma peut également servir de documentation d'une B.D en vue de sa maintenance. Il sera traduisible facilement en un schéma logique conforme à un S.G.B.D. et enfin en un schéma physique.

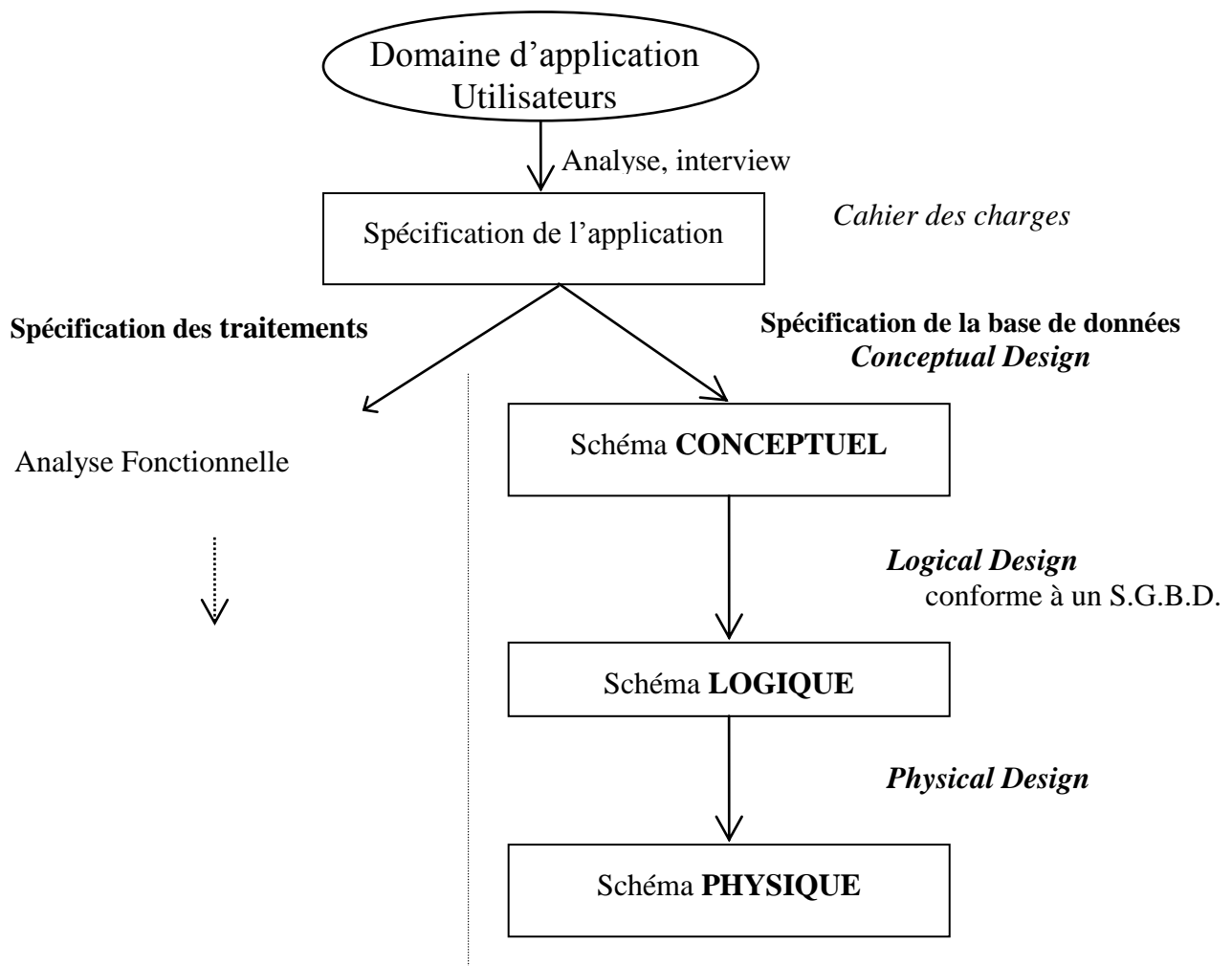


Figure 1: Analyse d'une application

2.2. Entités - Types d'entité (T.E.)

Il s'agit du concept de base du modèle entités-associations.

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

Une entité n'existe que par rapport à un individu (l'analyste) qui la considère comme importante, et donc digne d'être représentée.

Une entité peut aussi bien représenter une chose *concrète, physique* (Dupond, Tintin, mon bic, l'école (bâtiment), la voiture de M. Leroy,...) qu'une chose *abstraite* (L'IESN, la justice, ...).

Une entité sera caractérisée par des attributs et des valeurs d'attributs.

Exemple:

Employé n° 5 :

- Nom: *Dupond*
- Adr: *32, rue de Fer, 5000 Namur*
- DateNaissance: *02/12/63*
- Tel: *081 / 22.22.32*

Dans l'étape de conception d'une B.D, *on ne s'intéresse pas aux éléments individuels*, aux individus en particulier, *mais on s'intéresse aux types*.

Si l'on trace un parallèle avec la *technologie des fichiers*, on ne s'occupe pas de remplir les fichiers avec des *enregistrements*, mais on définit les **types d'enregistrement**.

Si l'on trace un parallèle avec la *programmation orientée objet*, on ne s'intéresse pas aux *objets* mais on s'intéresse aux **classes**.

On ne s'intéresse donc pas aux entités en particulier (occurrences), mais aux **types d'entités**.

Exemple: le type d'entité **Employé**

Tous les employés possèdent les mêmes attributs, mais chaque employé a ses propres valeurs d'attributs.

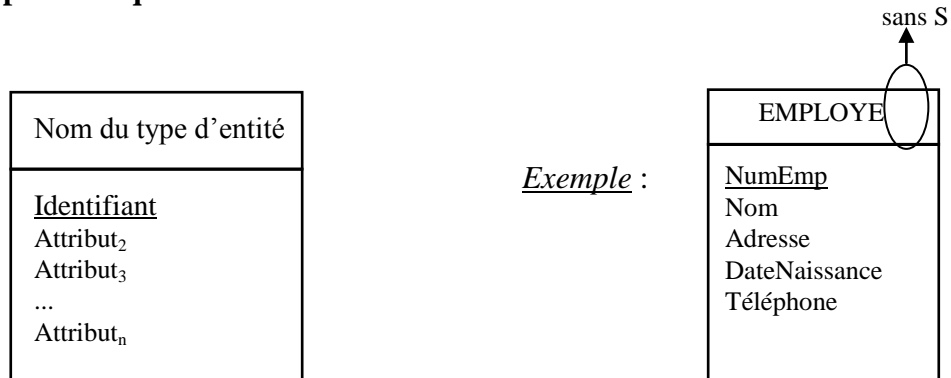
Un T.E. décrit le **schéma d'un ensemble d'entités**.

Un type d'entité (T.E.) est défini par :

- un nom
- une liste d'attributs

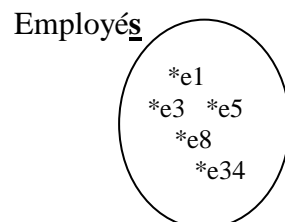
Graphiquement, un type d'entité sera représentée par une “*boîte*” reprenant les différents attributs, surmontée de son ***nom***.

L'attribut ***identifiant*** sera *souligné*. Un attribut est un identifiant pour un T.E. si sa valeur est **distincte pour chaque occurrence du T.E.**



La **durée de vie** d'une entité est **indépendante** de celle de son **type**: des entités seront créées et détruites régulièrement sans que l'existence du type n'en soit affectée.

☛ Il ne faut pas confondre un **type d'entité** (ex: *Employé*) et l'**ensemble de ses occurrences** (qui sera noté *Employés*).



2.3. Associations - Types d'association (T.A.)

2.3.1. Qu'est-ce qu'un type d'association ?

Une association est une **correspondance**, un **lien** entre deux ou plusieurs entités où chacune assume un **rôle** donné.

Exemple 1 : Consultation du docteur Leplat par Dupond

Rôles :

- Dupond *est* le patient *ausculté*
- le docteur Leplat *ausculte* Dupond

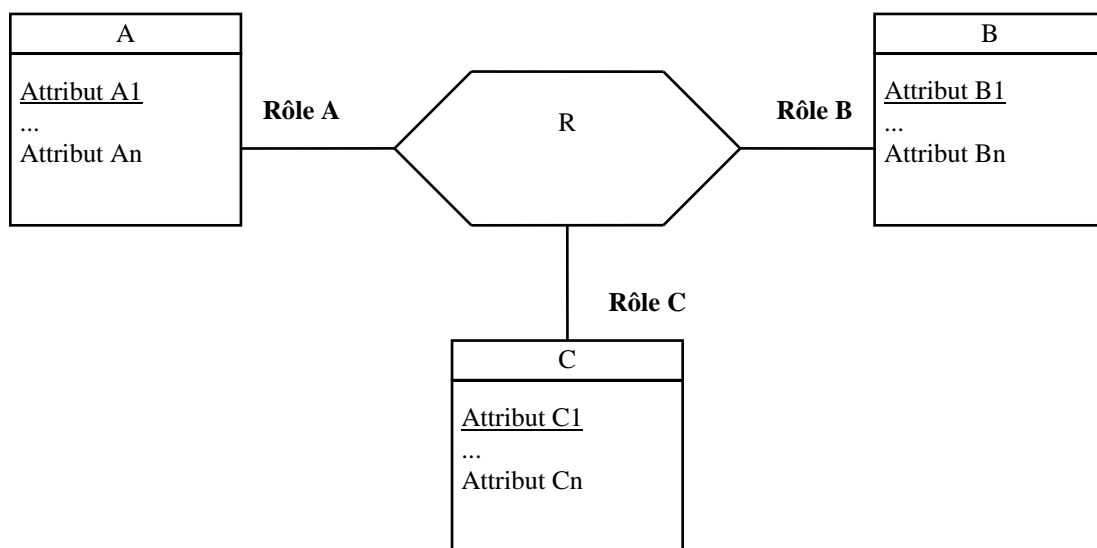
Exemple 2 : Prescription d'aspirine à Pierre par le docteur Lesage.

Rôles :

- l'aspirine *est prescrite*
- le docteur Lesage *prescrit*
- Pierre *bénéficie (d'une prescription)*

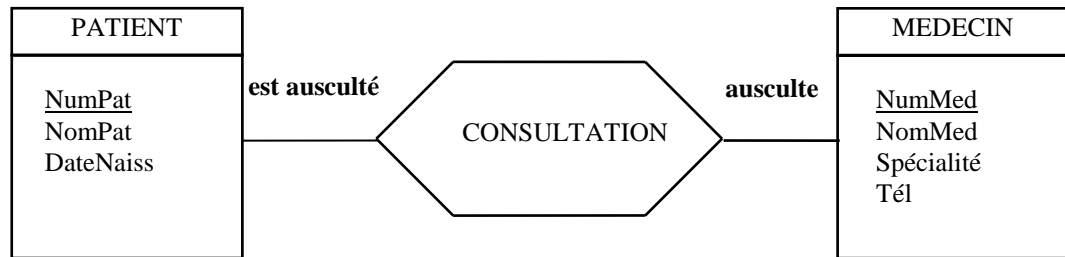
Comme pour les entités, on ne s'intéresse pas aux associations particulières entre des entités particulières, c'est-à-dire aux *instances* ou aux *occurrences* d'associations. Dans l'étape de conception d'une base de données, on s'intéresse aux **types d'associations entre des types d'entités**.

Graphiquement, un type d'association (**T.A.**) sera représenté par un "hexagone aplati" reliant les types d'entité concernés. Les rôles joués par les types d'entité peuvent également apparaître sur le schéma.

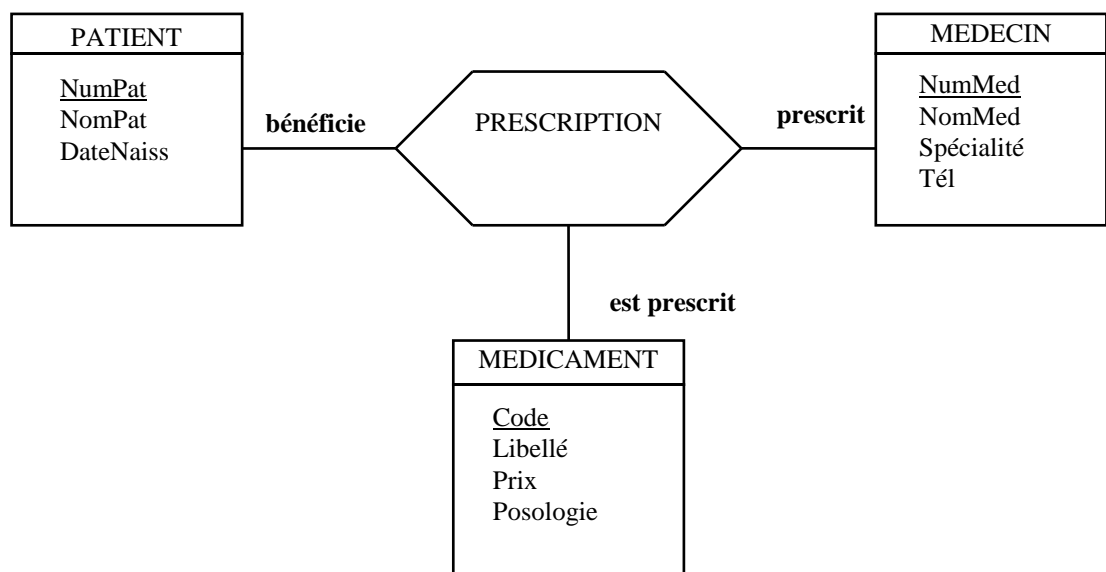


Les schémas entités-associations correspondant aux exemples sont:

Exemple 1:

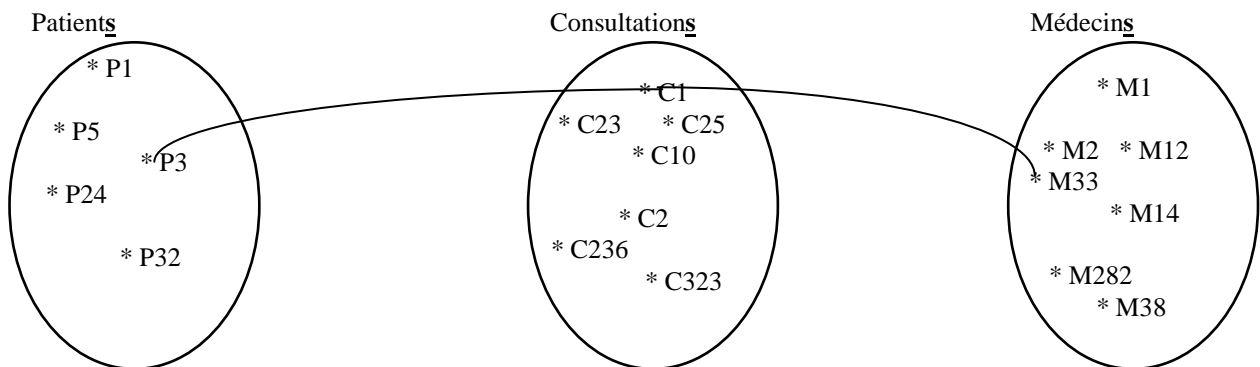


Exemple 2 :



L'ensemble de toutes les occurrences d'un type d'association forme un ensemble mathématique au même titre que l'ensemble des occurrences d'un type d'entité.

Reprenons l'exemple 1



Une occurrence particulière de l'ensemble des consultations (soit l'occurrence C1) est un lien entre un patient particulier (P3) et un médecin particulier (M33).

Quelles sont exactement les relations (traits) entre ces ensembles ?

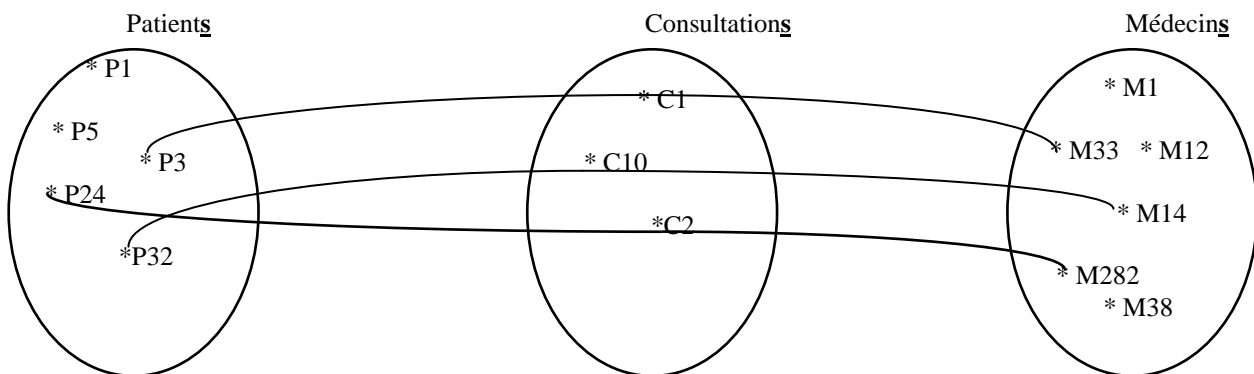
A combien de relations (liens/traits) au minimum et au maximum participe un élément donné?

- Un élément de l'ensemble des "Patients" *peut-il* avoir **plusieurs** liens avec des éléments de l'ensemble des "Consultations"?
- Un élément de l'ensemble des "Patients" *doit-il obligatoirement* avoir **au moins un** lien avec un élément de l'ensemble "Consultations"?
- Un élément de l'ensemble des "Médecins" *peut-il* avoir **plusieurs** liens avec des éléments de l'ensemble des "Consultations"?
- Un élément de l'ensemble des "Médecins" *doit-il obligatoirement* avoir **au moins un** lien avec un élément de l'ensemble "Consultations"?

- **Combien** de liens un élément de l'ensemble des "Consultations" a-t-il avec l'ensemble des "Patients" et avec l'ensemble des "Médecins"?

☛ Une occurrence d'un T.A. est toujours reliée à une et une seule occurrence de chaque T.E. associé.

Exemple :



Une occurrence de CONSULTATION ne peut être reliée qu'à une seule occurrence de PATIENT et à une seule occurrence de MEDECIN. En effet, il s'agit de la consultation d'un médecin particulier par un patient particulier; par exemple, la consultation du docteur Leplat par le patient Dupont.

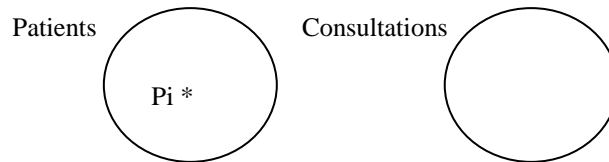
Mais, une occurrence particulière de PATIENT peut être reliée à **plusieurs** occurrences de CONSULTATION: *un patient peut consulter plusieurs médecins*. De même, une occurrence particulière de MEDECIN peut être reliée à **plusieurs** occurrences de CONSULTATION: *un médecin peut être consulté par plusieurs patients*.

Les questions intéressantes à se poser sont donc les suivantes :

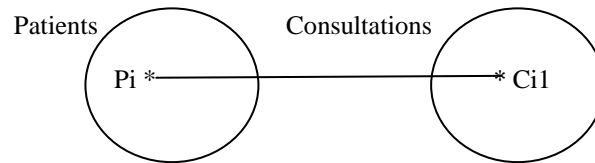
A) *A combien d'occurrences de CONSULTATION peut être reliée une occurrence de PATIENT?*

Trois possibilités: 0, 1 ou N.

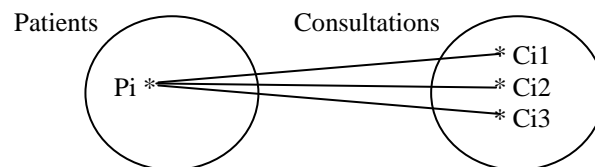
0: si le patient n'a (encore) consulté aucun médecin, il n'est relié à aucune consultation.



1: si le patient n'a consulté qu'un seul médecin, il est relié à une et une seule consultation



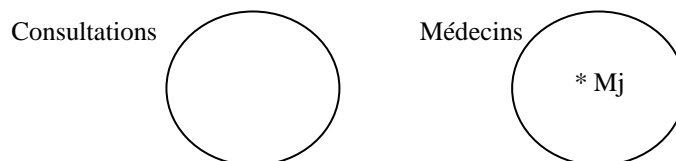
N: si le patient a consulté plusieurs médecins: il est relié à plusieurs consultations.



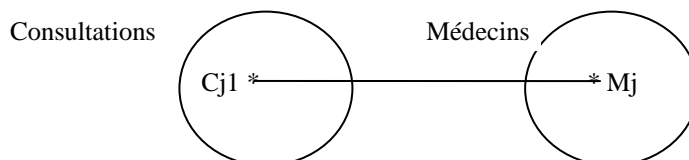
B) *A combien d'occurrences de CONSULTATION peut être reliée une occurrence de MEDECIN?*

Trois possibilités également: 0, 1 ou N.

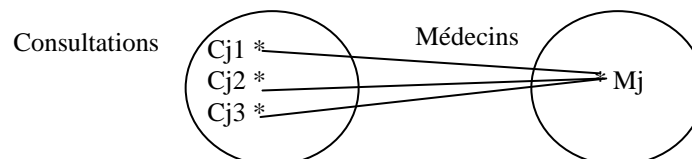
0: si le médecin n'a toujours pas été consulté.



1: si le médecin n'a donné qu'une seule consultation.



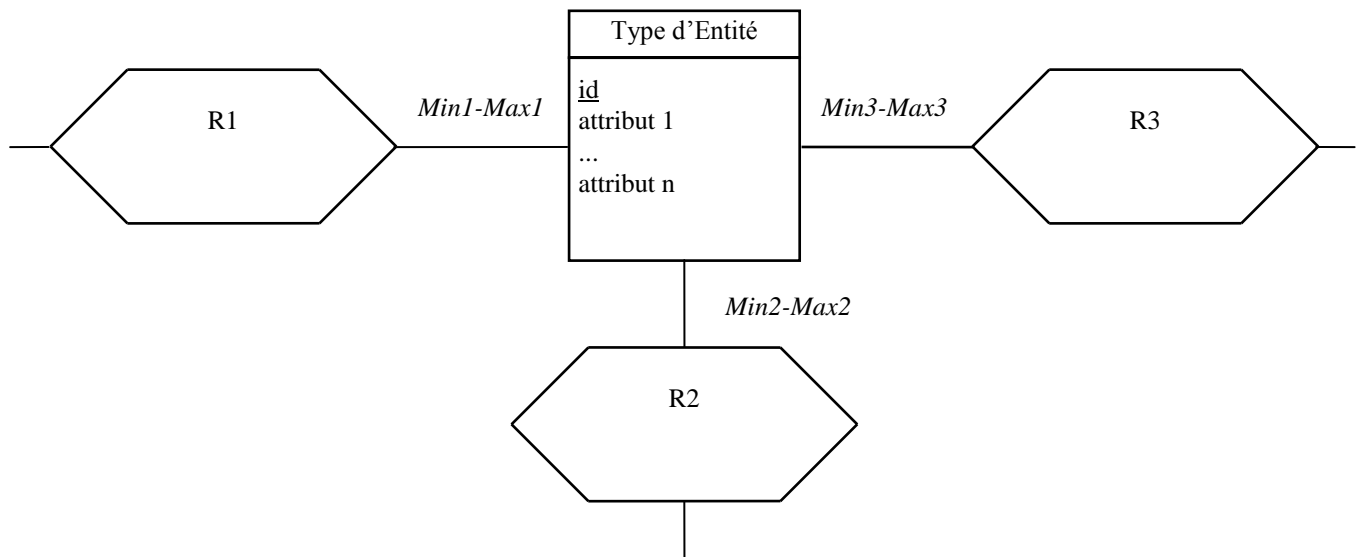
N: si le médecin a donné plusieurs consultations.



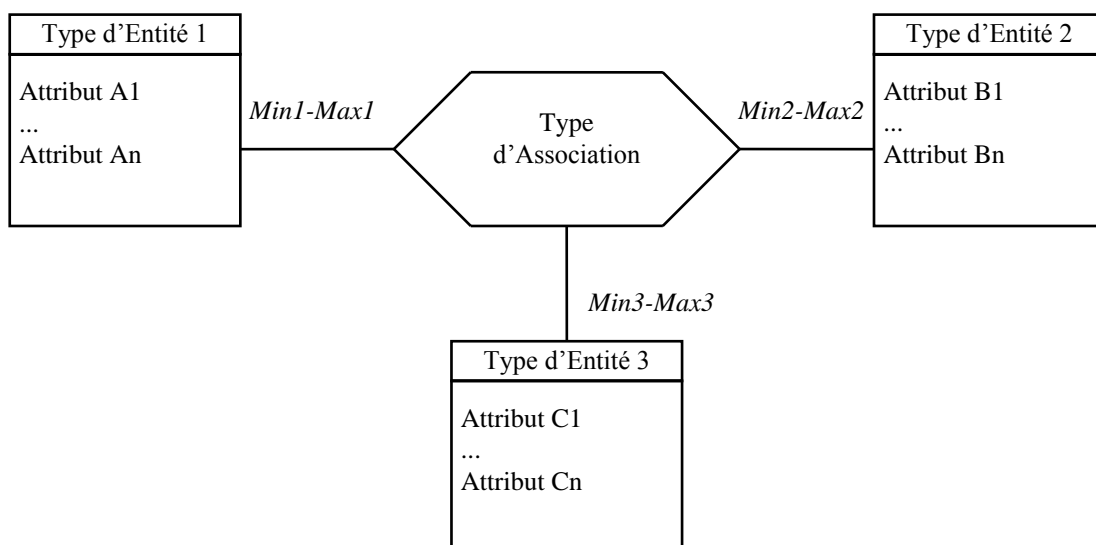
N'importe quelle occurrence d'un T.E. peut avoir au minimum *Min* et au maximum *Max* liens avec des occurrences de T.A.

On parlera de **cardinalités** ou **connectivités *minimum*** et cardinalités ou connectivités ***maximum***.

Des cardinalités sont associées au T.E. pour chaque lien avec un T.A.



Autrement dit, il faut spécifier les **cardinalités Min-Max pour chaque rôle d'un T.A.**



2.3.2. Cardinalités maximales

Un type d'associations est une **relation** entre des types d'entités.

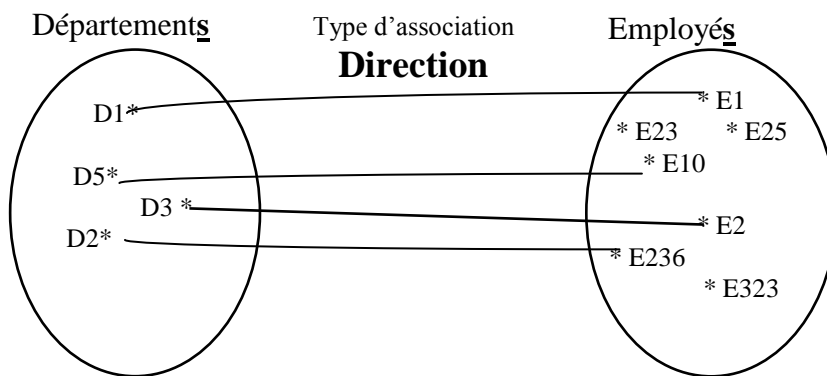
Les cardinalités maximum d'un T.A. dépendent du **type de cette relation**. Il y a trois types de relation :

- 1 à 1
- 1 à N
- N à N

La notion de **cardinalités** concerne les **types d'associations**, tandis que la notion de **type de relation** concerne les **ensembles**.

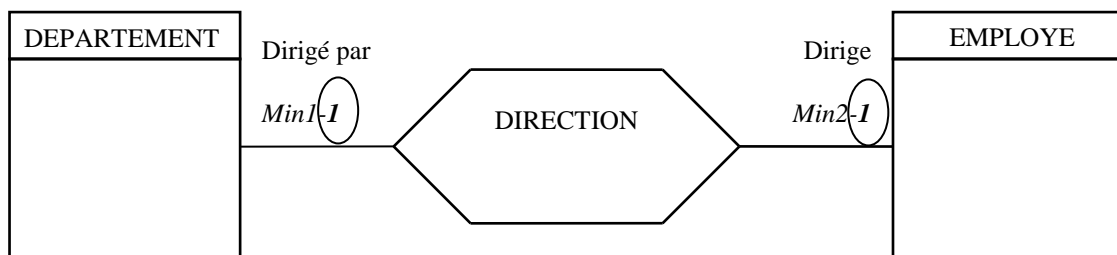
La notion de type de relation entre ensembles n'intervient que dans le calcul des cardinalités **maximum**! Les cardinalités maximum ne peuvent être que de 1 ou N.

A. Relation de type 1 à 1

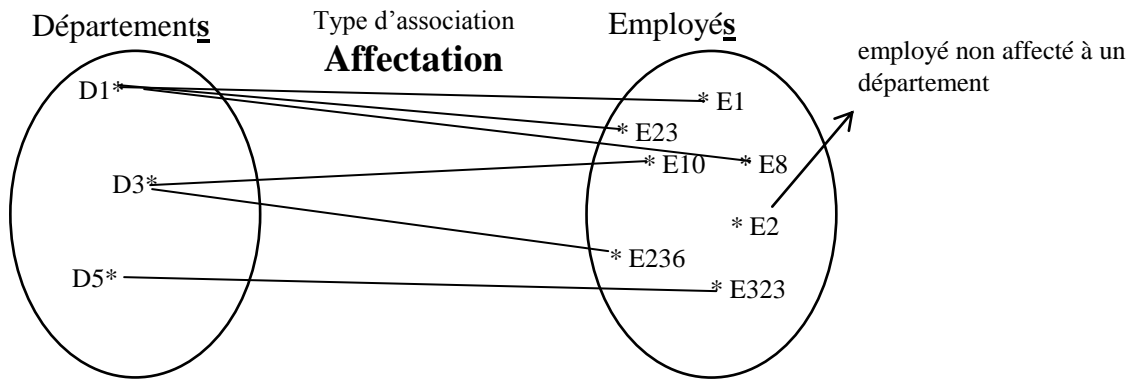


Au maximum:

- un département a *au plus* un employé qui est directeur
- un employé est directeur *d'au plus* un département

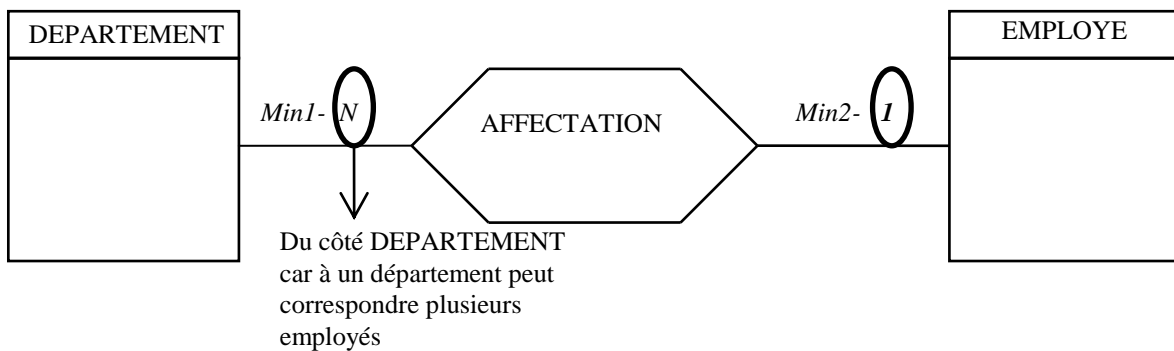


B. Relation de type 1 à N

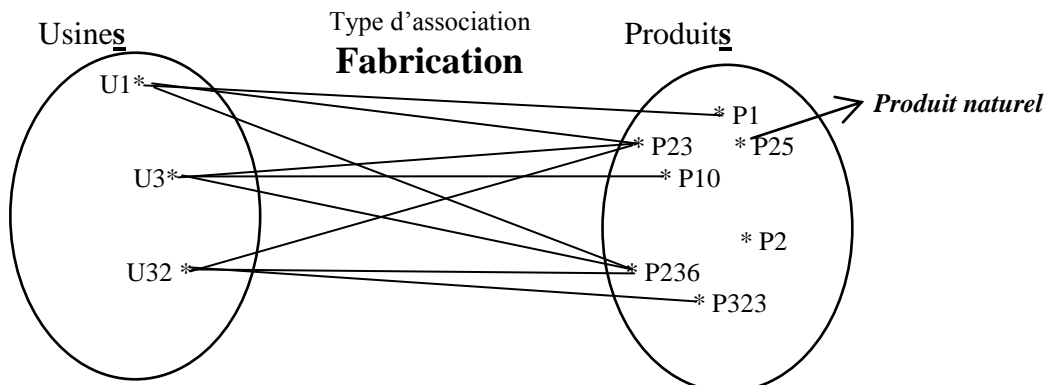


Au maximum:

- un département peut occuper plusieurs employés
- un employé est affecté à au plus un département

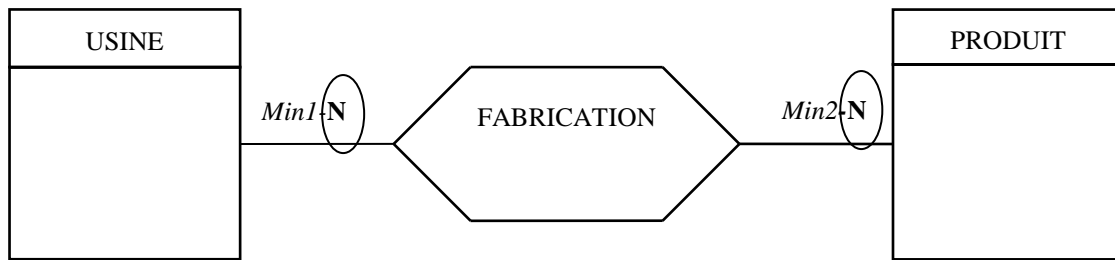


C. Relation de type N à N



Au maximum:

- une usine fabrique plusieurs produits
- un produit peut être fabriqué par plusieurs usines



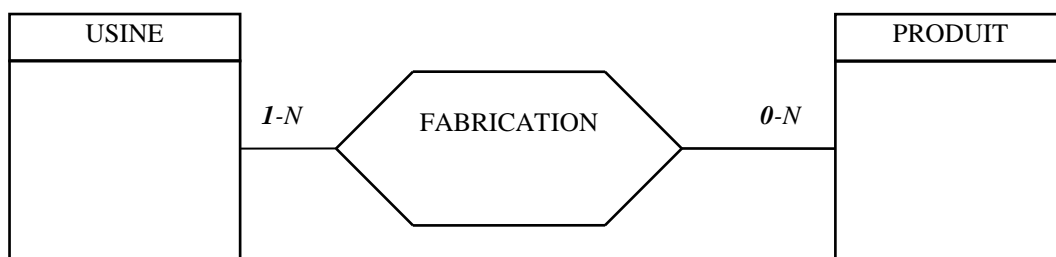
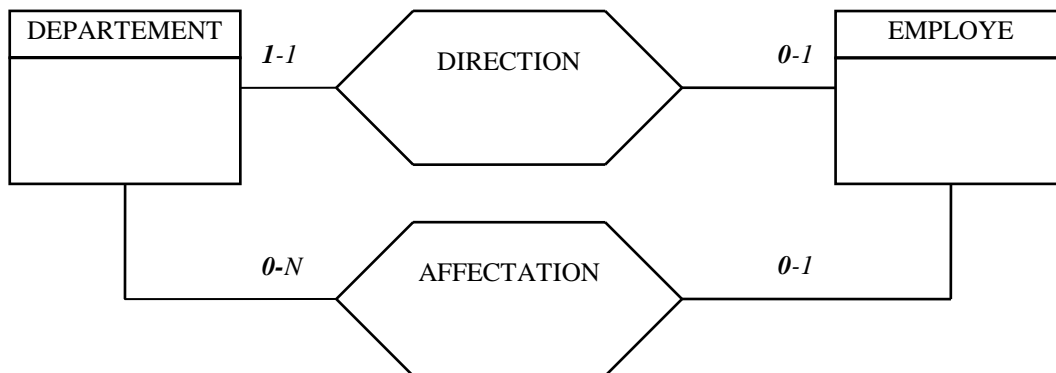
2.3.3. Cardinalités minimales

Les cardinalités minimales expriment le fait qu'un T.A. est facultatif ("peut") ou obligatoire ("doit").

La cardinalité minimale d'un T.A. facultatif est égal à 0.
 La cardinalité minimale d'un T.A. obligatoire est égal à 1.

Il s'agit de contraintes sur le *nombre minimum* d'entités associées.

Exemples:



Interprétation des cardinalités minimales :

T.A. DIRECTION:

- un département est toujours sous la direction d'**un et un seul** employé
- un employé **peut** ne pas être directeur

T.A. AFFECTATION:

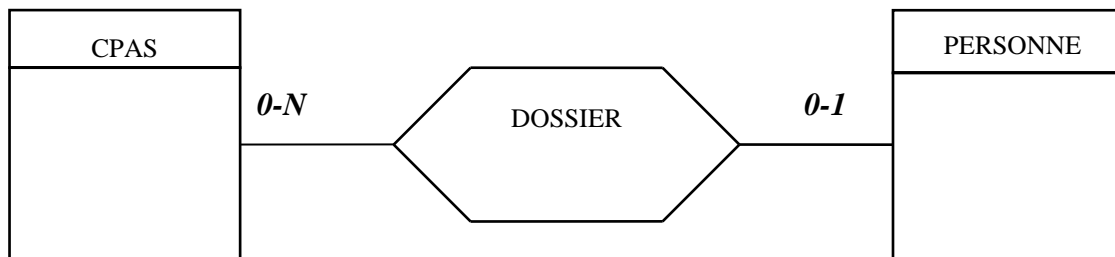
- un département **peut** n'avoir aucun employé qui lui est affecté (ex: le département "machine", si on suppose que l'employé qui le dirige est affecté au département "direction")
- un employé **peut** ne pas être affecté à un département

T.A. FABRICATION:

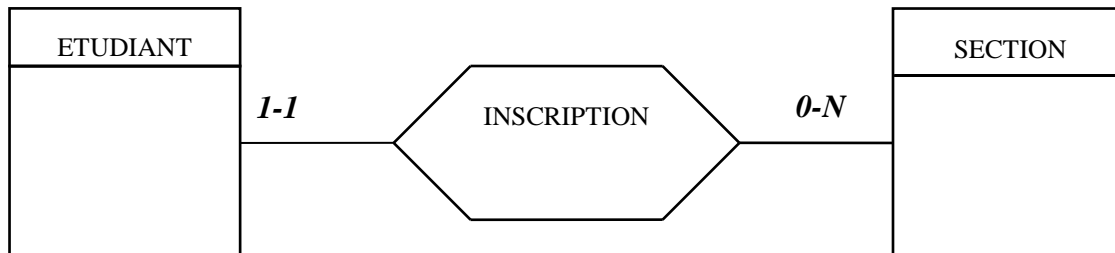
- une usine fabrique toujours **au moins un** produit
- un produit **peut** ne pas être fabriqué par une usine (produit naturel)

2.3.4. Exemples

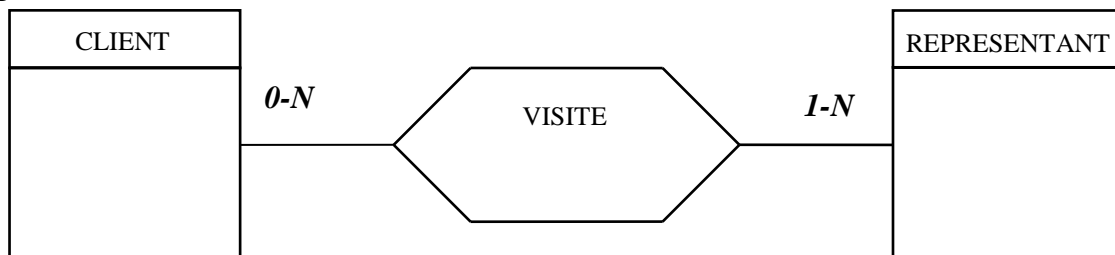
①



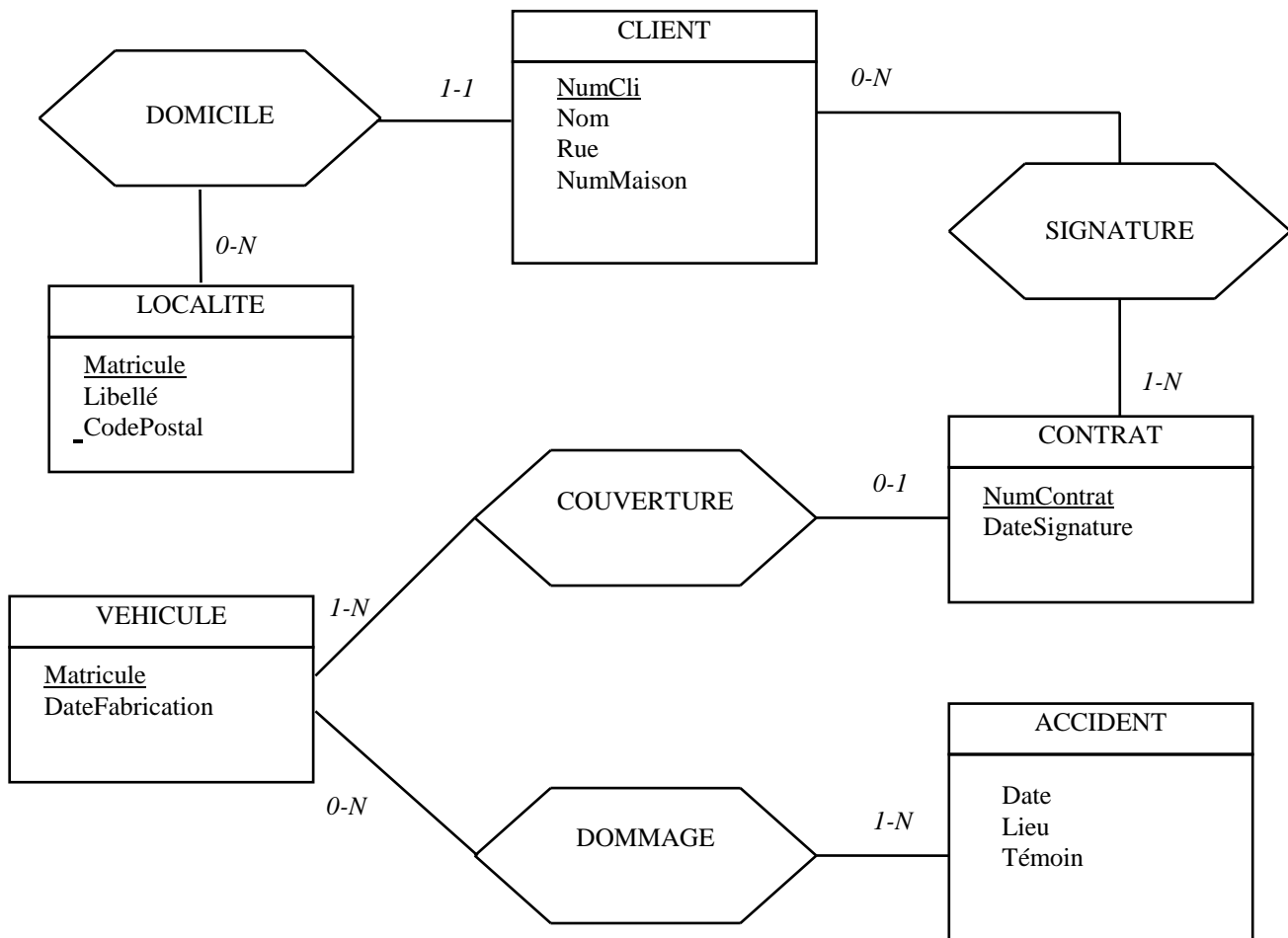
②



③



④ Une compagnie d'assurances demande à un de ses informaticiens de créer le schéma conceptuel de sa base de données. Ne sont stockées dans la base de données que les informations nécessaires à la compagnie d'assurances.



Interprétation:

- Tout client est domicilié dans une et une seule localité (cardinalités **1-1**).
- Une localité reprise dans la base de données n'est pas forcément la localité du domicile d'au moins un client (cardinalités **0-N**). *Notons que le libellé de la localité n'identifie pas à lui seul une localité! De même, le code postal n'identifie pas à lui seul une localité!*
- Un client peut avoir signé plusieurs contrats (cardinalités **0-N**).
- Tout contrat est signé par au moins un client et peut être signé par plusieurs clients (cardinalités **1-N**).
- Un contrat ne couvre pas forcément un véhicule; l'agence ne s'est pas spécialisée exclusivement dans l'assurance automobile. Un même contrat ne peut couvrir qu'un véhicule à la fois. Les cardinalités sont donc **0-1**.

- Si un véhicule est repris dans la base de données de l'agence, c'est qu'il fait l'objet d'au moins un contrat d'assurance. Un même véhicule peut faire l'objet de plusieurs contrats (par exemple, l'assurance vol fait l'objet d'un contrat indépendant de la couverture classique). Les cardinalités sont donc **1-N**.
- Un véhicule peut être impliqué dans plusieurs accidents. Il peut n'avoir été impliqué dans aucun accident (cardinalités **0-N**).
- Un accident répertorié par l'agence implique au moins un véhicule enregistré dans la base de données, et peut en impliquer plusieurs (cardinalités **1-N**).

2.4. Construction d'un schéma conceptuel

Le schéma conceptuel est élaboré par les analystes. Ceux-ci se basent sur les interviews qu'ils ont avec le client demandeur de l'application à réaliser, afin de maîtriser les concepts du domaine d'application et leurs liens. On peut voir une interview comme étant l'énoncé d'une succession de *phrases* ou *propositions* émises par le client à propos de son domaine d'application.

Malheureusement, ces propositions peuvent être **incomplètes**, **redondantes**, voire mutuellement **contradictaires** ou carrément **fausses**.

2.4.1. Décomposition de l'énoncé

- *Quand c'est possible*, l'énoncé doit être décomposé en propositions élémentaires du type:
sujet - verbe - complément.

Exemple: *Tout client a un nom.*
 Une commande est passée par un client.
 Un service traite des données.

Ce type de proposition élémentaire affirme l'existence de **deux concepts** (le sujet et le complément) et **un lien** (le verbe).

Tout client a un nom.
→ **concepts:** *client* et *nom*
→ **lien:** *possession*

- Il existe cependant d'autres types de propositions élémentaires.

Exemple: *Il existe des fournisseurs.*
 On s'intéresse aux accidents.

Ce type de proposition élémentaire affirme l'existence d'**un seul concept sans lien**.

- Il faut parfois reformuler certaines phrases complexes.

Exemple 1: *Un employé est identifié par un numéro de matricule et est caractérisé par un nom et une adresse.*

Il faut éclater cette proposition en trois phrases simples:

- ① *un employé est identifié par un numéro*
- ② *un employé est caractérisé par un nom*
- ③ *un employé est caractérisé par une adresse*

Chacune de ces propositions affirme l'existence de **deux concepts et un lien**.

Exemple 2: *Le coût du produit devra ...*

Il faut scinder cette proposition en deux:

① *tout produit a un coût*

② *le coût devra ...*

- Il faut également expliciter les raccourcis du langage que sont les pronoms possessifs.

Exemple: *Il peut contracter une assurance (il = client).*

Son département ... (= l'employé a un département et le département ...).

- Pour les propositions de type "sujet verbe complément", càd "**A verbe B**", qui affirment l'existence de deux concepts et un lien, on cherche :

- pour un exemplaire de A, **combien a-t-on d'exemplaires** de B au minimum et maximum?

- pour un exemplaire de B, **combien a-t-on d'exemplaires** de A au minimum et maximum?

Exemple: *Une commande est passée par un seul client.*

Un fournisseur peut fournir plusieurs produits.

N.B. Les réponses sont parfois dans les phrases à travers des mots comme:

- tout, certains, chaque
- pouvoir, devoir
- au moins un, un seul, au plus un
- des, les, ...

Propositions générales et particulières

Il est fondamental pour un concepteur de bases de données de faire la distinction entre des propositions générales et des propositions particulières.

Exemple: *Toute voiture a un numéro minéralogique*

 ⊃ proposition **générale** (niveau *type* ou *classe* d'objets)

Ma voiture a le numéro minéralogique ABG910

 ⊃ proposition **particulière** (niveau *occurrence* d'objets)

 il s'agit d'une *instance*, d'une *occurrence*, d'un *exemple*.

Dans un schéma de base de données, il ne faut pas représenter les propositions particulières.

Il faut seulement représenter les propriétés, les caractéristiques et les liens des **classes**.

Mais les propositions particulières sont cependant utiles, car elles peuvent suggérer (par **généralisation à partir d'exemples**) une caractéristique ou un lien de la classe correspondante.

2.4.2. Représentation d'une proposition

On a identifié des *concepts* et des *liens*.

Qu'est-ce que cela devient en termes de *type d'entité* (T.E.) et *type d'association* (T.A.)?

A. Concepts

- Si un **concept semble important** dans le contexte du domaine d'application → **T.E.**

Exemple: *Il y a des clients*

Dans la gestion d'une entreprise de services, la notion de client peut à première vue être un concept suffisamment important → **T.E. Client**

Client

- Si un concept contient **plusieurs caractéristiques** → **T.E.**

Exemple: *Tout employé a un nom*
 Tout employé a une adresse

Le même concept contient deux caractéristiques → **T.E. Employé**

Employé
Nom Adresse

- Si c'est une **simple propriété d'un autre concept** existant → **attribut**

Exemple: *Tout dossier possède un titre*

Si le T.E. *Dossier* existe déjà, le concept *Titre* est un attribut du T.E. *Dossier* → **attribut**

Dossier
... Titre

B. Liens

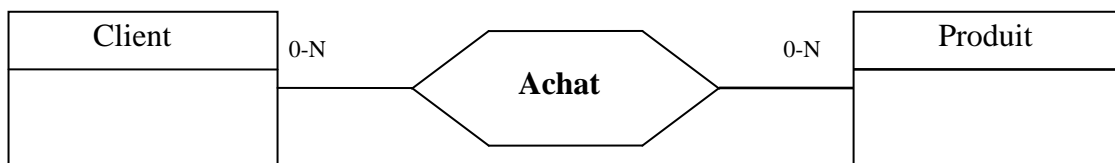
La représentation des liens *dépend du type des concepts reliés*.

B.1. Entre deux T.E.

S'il s'agit d'un lien entre deux concepts eux-mêmes représentés par **deux T.E.** → le lien est un **T.A.**

Exemple: *Un client achète des produits*

Si *Client* et *Produit* sont représentés chacun par un T.E. → T.A. *Achat*

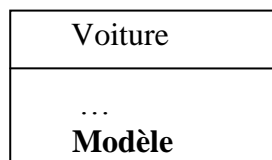


B.2. Entre un T.E. et un attribut

S'il s'agit d'un lien entre deux concepts, l'un étant déjà représenté par **un T.E.**, l'autre étant une simple caractéristique du premier → on crée un **nouvel attribut** qu'on affecte au T.E.

Exemple: *Toute voiture est d'un modèle particulier*

Si le T.E. *Voiture* existe déjà, le concept *Modèle* est un attribut du T.E. *Voiture* → **attribut**

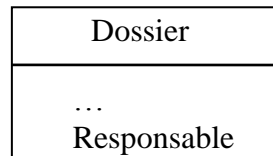


B.3. Entre deux attributs

Comment établir un lien entre un attribut déjà existant sur le schéma et un nouveau concept qui semble n'être qu'une caractéristique de cet attribut?

→ Il faut créer un **nouveau T.E.** auquel on affecte ces deux attributs, ainsi qu'un **nouveau T.A.** entre le T.E. existant et ce nouveau T.E.

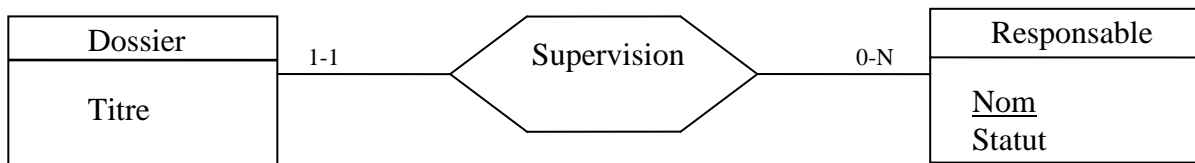
Exemple: 1^{ère} étape: *A tout dossier est associé le nom du responsable*



2^{ème} étape: *Le responsable du dossier a un statut*

"statut" est une caractéristique de *responsable* qui est lui-même un attribut déjà existant.

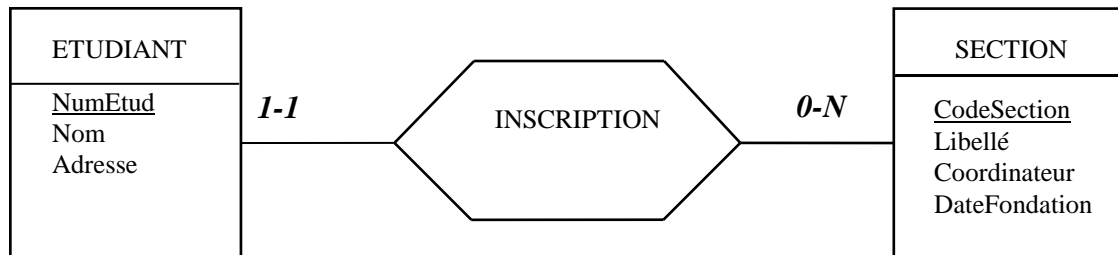
↪ On extrait l'attribut *Responsable* du T.E. *Dossier*. On crée un nouveau T.E. qu'on appelle *Responsable* qui aura deux attributs, un attribut identifiant (*Nom*) et un attribut *Statut*. Un nouveau T.A. *Supervision* est créé pour relier les T.E. *Dossier* et *Responsable*.



2.5. Intérêt de prévoir un type d'association plutôt qu'un attribut

On pourrait se demander quel est l'intérêt pour un analyste de prévoir un T.A. entre deux types d'entités plutôt que de prévoir un attribut dans un des T.E. "mémorisant" ainsi un lien vers l'autre T.E.

Exemple: Soit le schéma suivant:



Pourquoi prévoir ce T.A. INSCRIPTION plutôt qu'un simple attribut *SectionInscrip* dans le T.E. ETUDIANT ? L'attribut *SectionInscrip* pourrait alors prendre comme valeur n'importe quelle chaîne de caractères (y compris un code de section existant).

Le schéma deviendrait alors :



La différence fondamentale entre les deux schémas est que si le S.G.B.D. (Système de Gestion de Base de Données) est prévenu de l'existence du T.A. INSCRIPTION, il **est tenu de vérifier la cohérence de la base de données**. Par conséquent, il s'assurera qu'à chaque création (ou modification) d'un ETUDIANT, celui-ci soit relié à une section **EXISTANTE**. Par contre, si ce T.A. INSCRIPTION n'est pas spécifié au S.G.B.D., celui-ci ne peut maintenir la base de données dans un état cohérent. En effet, rien n'empêchera la création/modification d'un étudiant avec l'affectation à l'attribut *SectionInscrip* d'un code de section inexistante. Une telle base de données est difficilement exploitable : lors de l'interrogation de la base de données, il n'est pas certain de pouvoir retrouver les informations concernant la section de chaque étudiant, puisque certains étudiants risquent d'être enregistrés avec un code de section erroné. Par contre, si le T.A. INSCRIPTION est déclaré, on est certain de pouvoir retrouver pour chaque étudiant les informations concernant sa section.

En conclusion, un S.G.B.D. est un outil formidable pour l'administrateur de la base de données. En effet, bon nombre de *contraintes peuvent être déléguées au S.G.B.D.* C'est à lui désormais d'effectuer toutes sortes de vérifications, et, si une contrainte est violée lors de l'accès à la base de données, de refuser cet accès à l'utilisateur. A l'analyste donc de prévoir toutes ces contraintes **dès la création de la base de données**. La déclaration d'un type d'association entre deux types d'entité est une de ces contraintes.

2.6. Attributs

Un attribut est défini par un **nom**, un **type** et le **domaine des valeurs** admises.

💣 les types et domaines de valeurs n'apparaissent pas dans le schéma entités-associations. Ils seront renseignés dans une documentation annexe. Dans cette documentation, chaque T.E. et chaque T.A. fera l'objet d'une description détaillée.

Il existe *différentes classes* d'attributs.

2.6.1. Identifiant (“key attribute”)

Un attribut est un identifiant dans un T.E. si sa **valeur est distincte pour chaque occurrence** du T.E. (contrainte d'unicité -"uniqueness"). Autrement dit, deux entités ne peuvent pas avoir la même valeur d'identifiant.

L'attribut identifiant est souligné sur le schéma entités-associations.

Exemple: Une personne est identifiée par son numéro de registre national

PERSONNE
<u>N°RegistreNational</u>

2.6.2. Mono-valué ou multivalué (“single-valued or multivalued”)

On parle également d'attribut **simple** ou **répétitif**.

Un attribut **simple** ou **monovalué** se voit affecter une seule valeur dans une même occurrence de T.E.

Exemple: Une personne a un nom

PERSONNE
<u>N°RegistreNational</u> Nom

Un attribut **répétitif** ou **multivalué** se voit affecter plusieurs valeurs dans une même occurrence de T.E. On précise alors le nombre maximum de valeurs que peut prendre l'attribut pour une même occurrence de T.E. (= **cardinalité maximale**). La cardinalité minimale est de 1. *Par convention*, un nom pluriel sera choisi pour les attributs multivalués (ex: prénoms).

Exemple: Une personne peut avoir de 1 à 6 prénom(s)

PERSONNE
<u>N°RegistreNational</u> Nom Prénoms [1..6]

2.6.3. Obligatoire ou facultatif

Un attribut est **obligatoire** si toute occurrence du T.E. **doit** avoir une valeur pour cet attribut.

Exemple: Toute personne a obligatoirement un nom

PERSONNE
<u>N°RegistreNational</u> Nom Prénoms [1..6]

Un attribut est **facultatif** si une occurrence de T.E. **peut** ne pas avoir de valeur pour cet attribut. Un attribut **facultatif** est un attribut dont la valeur peut être *inconnue* (valeur *null*).

Un attribut peut ne pas avoir de valeur pour une entité particulière soit parce que la valeur de cet attribut est inconnue (ex : n° de tel, si la personne possède le téléphone mais ne désire pas communiquer son numéro), soit parce que l'attribut est sans signification pour cette entité-là (ex : nom d'épouse pour une personne de sexe masculin).

Un attribut facultatif est spécifié sur le schéma entité-association par des cardinalités [0..1].

Exemple: Une personne **peut** avoir un numéro de téléphone
Une personne **peut** avoir un nom d'épouse

PERSONNE
<u>N°RegistreNational</u> Nom Prénoms [1..6] N°Tel [0..1] NomEpouse [0..1]

N.B. Un attribut **obligatoire** et **mono-valué** (le plus fréquent) a des cardinalités **[1..1]**. Par convention, on ne note alors aucune cardinalité sur le schéma entités-associations. Tout attribut apparaissant sur le schéma entités-associations *sans cardinalité* a des cardinalités **[1..1]** *implicites*.

Un attribut **facultatif** et **multi-valué** aurait des cardinalités **[0..N]**.

Par exemple, si on admet plusieurs numéros de téléphone (soit 3 au maximum), les cardinalités de l'attribut "NumerosTel" seraient de [0..3].

PERSONNE
<u>N°RegistreNational</u> Nom Prénoms [1..6] NomEpouse [0..1] Numeros<u>Tel</u> [0..3]

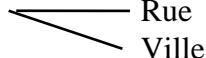
2.6.4. Atomique ou décomposable (“atomic or composite”)

Un attribut est **atomique** ou **élémentaire** si ses *valeurs sont indivisibles*, c’est-à-dire si on ne veut ou on ne peut plus les décomposer en parties plus fines.

Exemple : Nom

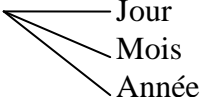
Un attribut est **décomposable** ou **composé** s’il *peut être “découpé” en attributs plus “fins”* ayant une signification à part entière. On prévoit un attribut de type décomposable quand on désire pouvoir accéder à chacune des parties de l’attribut. Par exemple, si l’adresse est stockée uniquement en vue d’imprimer des étiquettes à apposer sur des enveloppes à envoyer par courrier, un attribut adresse de type atomique suffit. Si par contre, des statistiques doivent pouvoir être calculées par localité, l’attribut adresse doit être de type décomposable.

Exemple: Adresse



De même, si la date de naissance est utilisée pour calculer l’âge d’une personne, il faut prévoir un attribut âge de type décomposable.

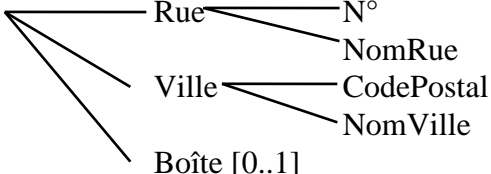
Exemple: DateNaissance



La valeur d’un attribut décomposable est obtenue par la **concaténation** des valeurs des attributs composants.

Plus largement, un attribut décomposable est constitué de **plusieurs** attributs *obligatoires* ou *facultatifs*, *mono-valués* ou *multi-valués*, *atomiques* ou *décomposables*.

Exemple: Adresse



PERSONNE
<u>N°RegistreNational</u> Nom Prénoms [1..6] NomEpouse [0..1] NumerosTel [0..3] Adr Rue N° Boîte [0..1] NomRue Ville CodePostal NomVille

2.6.5. Attribut facultatif versus attribut booléen

Pour rappel, un attribut **facultatif** est un attribut dont la valeur peut être *inconnue* (valeur *null*).

Notation sur le schéma entités-associations d'un attribut ***facultatif*** :

T.E.
AttributFacultatif[0..1]

Un attribut **booléen** est un attribut dont la valeur est soit la valeur *vrai* soit la valeur *faux*.

Un attribut *booléen* qui ne peut prendre que la valeur vraie ou fausse est donc un attribut **obligatoire** (soit *vrai* soit *faux*) ; les cardinalités sont [1..1]. Par conséquent, on ne note *aucune cardinalité* sur le schéma entités-associations pour des attributs booléens obligatoires.

Notation sur le schéma entités-associations d'un attribut ***booléen obligatoire***:

T.E.
AttributBooléenObligatoire

Un attribut **booléen** peut cependant être **facultatif**, si les valeurs possibles sont *vrai*, *faux* et *inconnu* (*null*).

Notation sur le schéma entités-associations d'un attribut ***booléen facultatif***:

T.E.
AttributBooléenFacultatif[0..1]

Illustrons le choix d'attributs facultatifs et booléens sur base d'exemples tirés de la gestion d'un hôtel.

Exemple 1 : Le touriste peut payer par carte visa

Le touriste peut payer par carte visa ou par un autre moyen.

Les valeurs possibles pour l'attribut *CarteVisa* sont: *vrai* ou *faux*.

Il s'agit donc d'un attribut **booléen obligatoire** (cardinalités [1..1] implicites).

Touriste
CarteVisa

Exemple 2 : Le touriste peut avoir réservé son séjour à l'hôtel via une agence de voyage

Soit le touriste a réservé son séjour via une agence de voyage, soit il l'a réservé de sa propre initiative sans passer par une agence de voyage.

Les valeurs possibles pour l'attribut *ParAgence* sont: *vrai* ou *faux*.

Il s'agit donc d'un attribut **booléen obligatoire** (cardinalités [1..1] implicites).

Touriste
CarteVisa ParAgence

Exemple 3 : On ne sait pas toujours si le touriste a réservé via agence de voyage ou non

Il se peut que, pour certains touristes, on ne sache pas s'ils sont passés par une agence de voyage.

Les valeurs possibles pour l'attribut *ParAgence* sont : *vrai*, *faux* ou *null* (valeur inconnue).

Il s'agit donc d'un attribut **booléen facultatif** (cardinalités [0..1]).

Touriste
CarteVisa ParAgence[0..1]

Exemple 4 : *Un touriste peut venir avec son propre véhicule*

Les touristes peuvent arriver à l'hôtel avec leur propre véhicule ou via un autre moyen de transport. Si l'on veut mémoriser si le touriste est arrivé avec son véhicule personnel ou non, il s'agit d'un attribut **booléen obligatoire** (cardinalités [1..1] implicites).

Touriste
CarteVisa ParAgence[0..1] VehiculePersonnel

Exemple 5 : *Si le touriste est venu avec son propre véhicule, le numéro de plaque de la voiture est mémorisé*

En vue de contrôler les véhicules garés dans le parking de l'hôtel, on désire connaître les numéros de plaque des véhicules appartenant aux touristes logeant à l'hôtel. Les touristes qui ont choisi une formule *voyage organisé en car*, par exemple, n'auront aucune valeur pour l'attribut *VehiculePersonnel* (valeur *null*).

Il s'agit donc d'un attribut **facultatif** (cardinalités de l'attribut: [0..1]).

Touriste
CarteVisa ParAgence[0..1] VehiculePersonnel [0..1]

Exemple 6 : *Le touriste peut être accompagné d'un animal domestique*

Soit le touriste est accompagné, soit il n'est pas accompagné d'un animal domestique.

Les valeurs possibles pour l'attribut *AnimalDomestique* sont: *vrai* ou *faux*.

Il s'agit donc d'un attribut **booléen obligatoire** (cardinalités [1..1] implicites).

Touriste
CarteVisa ParAgence[0..1] VehiculePersonnel [0..1] AnimalDomestique

Exemple 7 : *Si le touriste est accompagné d'un animal domestique, on mémorise de quelle espèce il s'agit*

Les touristes qui voyagent sans animaux domestiques auront la valeur *null* pour cet attribut. Il s'agit donc d'un attribut **facultatif** (cardinalités de l'attribut: [0..1]).

Touriste
CarteVisa ParAgence[0..1] VehiculePersonnel [0..1] AnimalDomestique[0..1]

Exemple 8 : *Le touriste peut être accompagné d'animaux domestiques. Si le touriste est accompagné d'animaux domestiques, on mémorise la **liste** des espèces accompagnant le touriste*

Les touristes qui voyagent sans animaux domestiques auront toujours la valeur *null* pour cet attribut.

Il s'agit donc d'un attribut **facultatif** et **répétitif** (cardinalités de l'attribut: [0..N]).

Touriste
CarteVisa ParAgence[0..1] VehiculePersonnel [0..1] AnimalDomestique[0..N]

2.7. Non redondance dans le schéma entités-associations

Le but d'une B.D est de stocker, et donc de représenter, *toutes les données nécessaires aux applications ultérieures*. D'autre part, ne seront représentées que des données qui seront utilisées. Par conséquent, *une B.D doit être complète mais ne contiendra pas de données inutiles ni redondantes*.

Une donnée est **redondante** si elle peut être **calculée ou dérivée** à partir de données déjà stockées dans la B.D.

💣💣 Une telle donnée ne sera **donc pas stockée sous forme d'attribut**, sauf introduction volontaire de redondance pour des raisons de performance. Cette distinction à faire entre des attributs nécessaires et dérivables est essentielle, en particulier dans les bases de données dont seront extraites des statistiques.

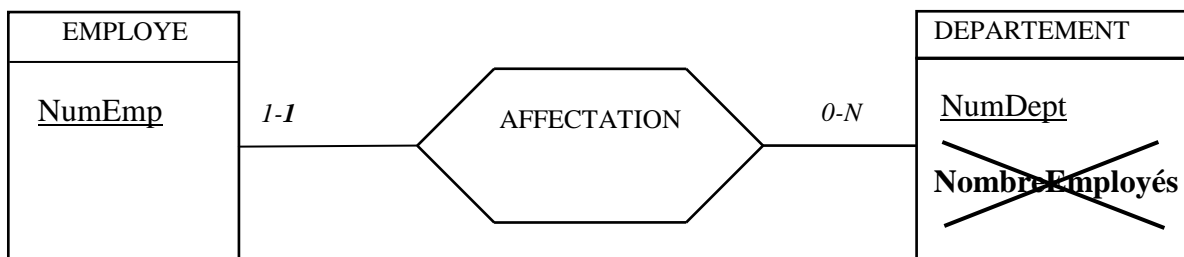
☹ Il s'agit d'une *source fréquente d'erreurs* auprès des étudiants!

Exemple 1 : date de naissance et âge

Si ces deux données sont utilisées dans l'application, il suffit de prévoir l'attribut "date de naissance" seul et en aucun cas un second attribut "âge"; celui-ci peut être *calculé par l'application* à partir de l'attribut "date de naissance".

N.B. Il vaut mieux stocker la date de naissance plutôt que l'âge. En effet, le premier attribut a une valeur fixe, tandis que le second devrait régulièrement être mis à jour.

Exemple 2 : soit le schéma suivant

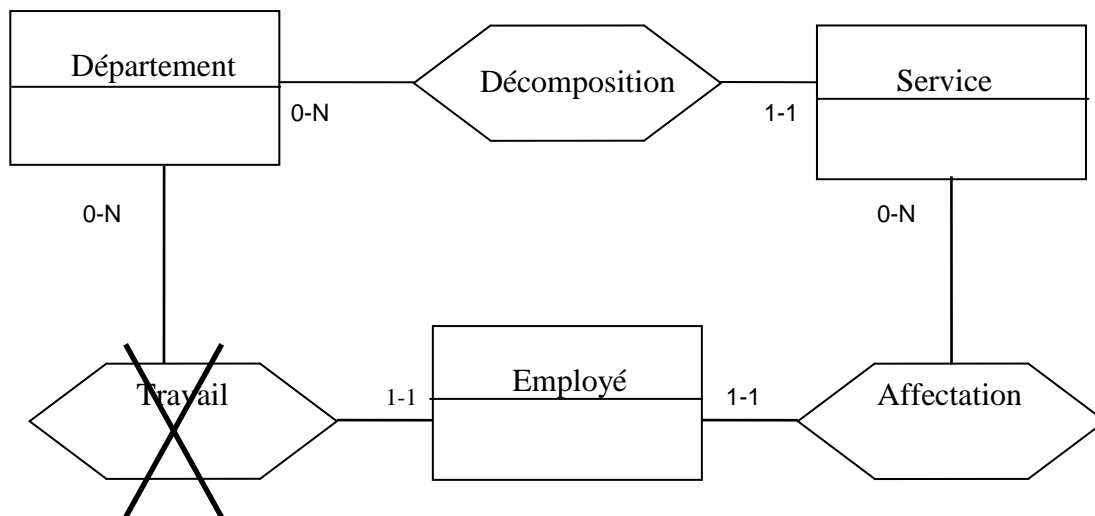


Si le nombre d'employés travaillant pour un département donné doit être connu, il est **inutile** et théoriquement **faux** d'ajouter un attribut *NombreEmployés* au T.E. *Departement*. Cette donnée est **dérivable** à partir du T.A. *AFFECTATION*. En effet, l'existence du T.A. *Affectation* signifie qu'il est possible, à partir d'un employé, de trouver son département, et à partir d'un département de retrouver **tous** les employés qui y travaillent.

Il est possible de **demandeur au S.G.B.D. de compter ces liens** pour un département donné.

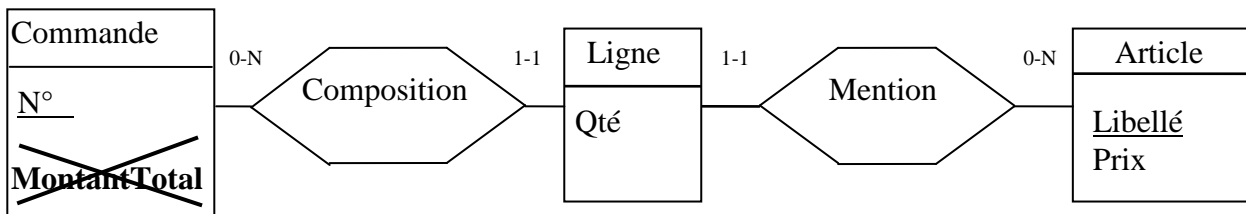
Le principe est de **ne pas surcharger inutilement une B.D**, sauf *recherche de performance*.

Exemple 3 : soit le schéma suivant



Grâce aux cardinalités 1-1, on sait qu'un employé est affecté à un seul service, et qu'un service appartient à un seul département. Par conséquent, un employé ne peut appartenir qu'à un seul département. Il est donc **inutile** et conceptuellement **faux** de rajouter un T.A. *Travail* entre les T.E. *Département* et *Employé*! Il s'agirait de **redondance**.

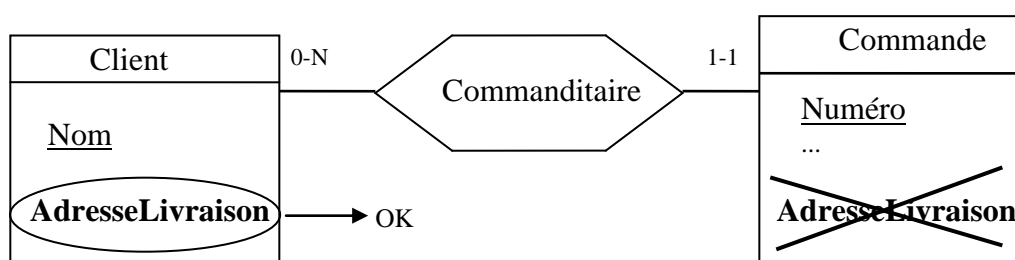
Exemple 4 : soit le schéma suivant



Une ligne (de commande) appartient à une seule commande et fait référence à un seul produit. De plus, le T.E. *Ligne* contient la quantité commandée (*Qté*) et le T.E. *Article* reprend le prix unitaire (*Prix*).

Il est donc redondant de rajouter un attribut *MontantTotal* au T.E. *Commande*. C'est une donnée dérivable, calculable.

Exemple 5 : soit le schéma suivant



Où mémoriser l'adresse de livraison des commandes ?

Faut-il rajouter l'attribut *AdresseLivraison* dans le T.E. *Commande* ou dans le T.E. *Client* ?

Il faut viser le moins de redondance possible. Il faut donc choisir la solution qui offrira le moins de redondance. Si on part de l'hypothèse que toutes les commandes d'un même client sont livrées à la même adresse, l'attribut *AdresseLivraison* doit être attaché au T.E. *Client*; elle ne sera ainsi stockée qu'une seule fois dans la B.D. En effet, si l'on attache l'attribut *AdresseLivraison* au T.E. *Commande*, il y aura répétition de l'adresse pour chaque commande.

2.8. Pertinence des propositions

2.8.1. Contradiction des propositions

Une source de contradiction est *l'évaluation des nombres d'éléments en liaison* dans les proposition du type " A verbe B ".

Exemples: *Un véhicule a un propriétaire*
 Un véhicule peut avoir plusieurs propriétaires.

Deux réactions sont possibles (après discussion avec le client!) :

- ① Rejet de la proposition erronée
- ② Choix de la **proposition la plus générale**:
 "un seul propriétaire" est un *cas particulier* de "plusieurs propriétaires"
 ↳ on retient "plusieurs propriétaires "

2.8.2. « Bruit » dans les propositions

Il est possible que certaines propositions de l'énoncé ne doivent pas être prises en compte. Il s'agit de "**bruit**".

Exemple 1: *Le calcul des factures se fait tous les deux jours.*

Cette information concerne la fréquence d'un **traitement**. Il ne faut donc pas prévoir de donnée supplémentaire dans la B.D.

Mais de telles propositions *peuvent cacher une structure à incorporer* dans la base de données.

Exemple 2: *Une prime annuelle de 1000 euros est accordée après une ancienneté de 10 ans*

A première vue, cette proposition ne concerne pas la base de données, mais le traitement "calcul des salaires". Cependant, pour que le traitement puisse se faire, il faut connaître *l'ancienneté* de l'employé.

Il faut donc que l'ancienneté de chaque employé soit mémorisée dans la B.D.

Remarquons que puisque l'ancienneté évolue, il est préférable de stocker **la date d'entrée en fonction** qui, elle, reste fixe.

Exemple 3: *Le montant en devise de la facture sera calculé à partir du taux de la devise atteint le jour de la commande*

Cela ressemble à du bruit: il s'agit d'une information sur la façon de calculer le montant de la facture. Mais pour que le traitement puisse se faire, il faut mémoriser *la date de la commande* afin de retrouver le cours de la devise ce jour-là, ou carrément *le taux de la devise*.

Exemple 4: *Les véhicules que la compagnie assure ...*

Supposons que la base de données ne concerne que cette compagnie-la. Il est donc inutile de prévoir T.E. *Compagnie*; il n'aurait qu'une seule occurrence (une seule entité).

En conclusion,

La base de données doit comprendre toutes les données nécessaires aux traitements, sans redondance inutile

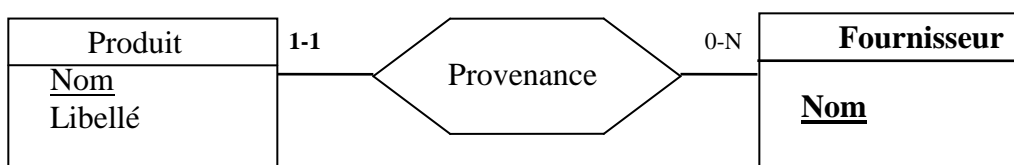
2.8.3. Autres recommandations

Encore quelques conseils découlant du Bons Sens.

① Si un concept n'a qu'une caractéristique, c'àd **si un T.E. ne contient qu'un seul attribut**, il est peut être nécessaire de le remettre en question! Ce concept n'est *peut être qu'un simple attribut* d'un autre T.E.

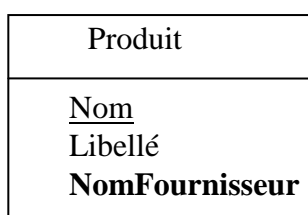
Exemple : Supposons qu'on ne mentionne *que le nom* du fournisseur pour chaque produit.

Avant:



Puisque d'après les cardinalités 1-1, un produit n'a qu'un seul fournisseur, et qu'on ne mentionne que le nom du fournisseur (qui est identifiant), on peut en faire un simple attribut du T.E. produit.

Après:



② Si **une même caractéristique** (d'un concept) est citée à **plusieurs endroits**, il faut peut-être en faire un **T.E.**

Exemple : *Le nom du fournisseur apparaît à plusieurs endroits*

On peut envisager la création d'un T.E. *Fournisseur* qui contiendra (au minimum) l'attribut *Nom* et la création de T.A. pour relier ce T.E. *Fournisseur* aux concepts le mentionnant auparavant.

2.9 . T.A. particulier

2.9.1 . T.A. de degré supérieur à 2

💣 Attention, il y a des propositions complexes qu'il ne faut pas réduire à des propositions élémentaires de type "A verbe B", sous peine de perdre des informations!

Les T.A. peuvent être ternaires, quaternaires, ... Il n'y a pas de limitation de degré pour un T.A.

Quel que soit le degré du T.A., les cardinalités minimum et maximum doivent être précisées pour tout T.E. participant à ce T.A.

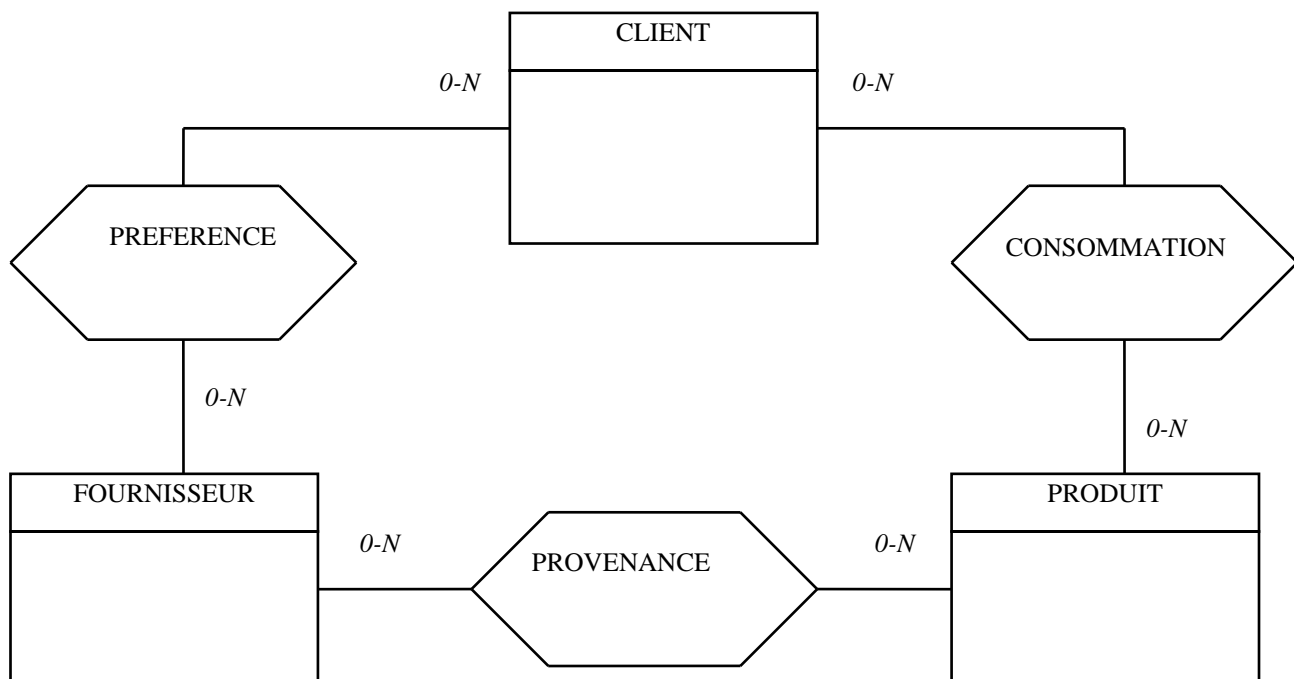
Exemple :

Un **client** achète un **produit** chez un **fournisseur**.

Il faut pouvoir stocker des occurrences du type :

- *Luc achète un PC chez Priminfo*
- *Pol achète une imprimante chez Flag 2000*
- *Anne achète du papier chez Dell*
- *Luc achète du papier chez Flag 2000*
- *Laure achète une imprimante chez Dell*
- *Anne achète un scanner chez Priminfo*
- *Pol achète un PC chez Flag 2000*
- *Louis achète un scanner chez Dell*
- *Laure achète du papier chez Flag 2000*

Le schéma ci-dessous est-il correct ?



Un tel schéma permet de retrouver :

- *les fournisseurs d'un client*
ex : Priminfo et Flag 2000 pour Luc
- *les fournisseurs d'un produit* :
ex : Priminfo et Flag 2000 pour les PC
- *les produits consommés par un client* :
ex : PC et papier pour Luc

Il y a malheureusement **perte d'informations**: il est, par exemple, impossible de retrouver *chez qui Luc a acheté un PC* ou *à quel(s) client(s) Flag 2000 vend du papier*.

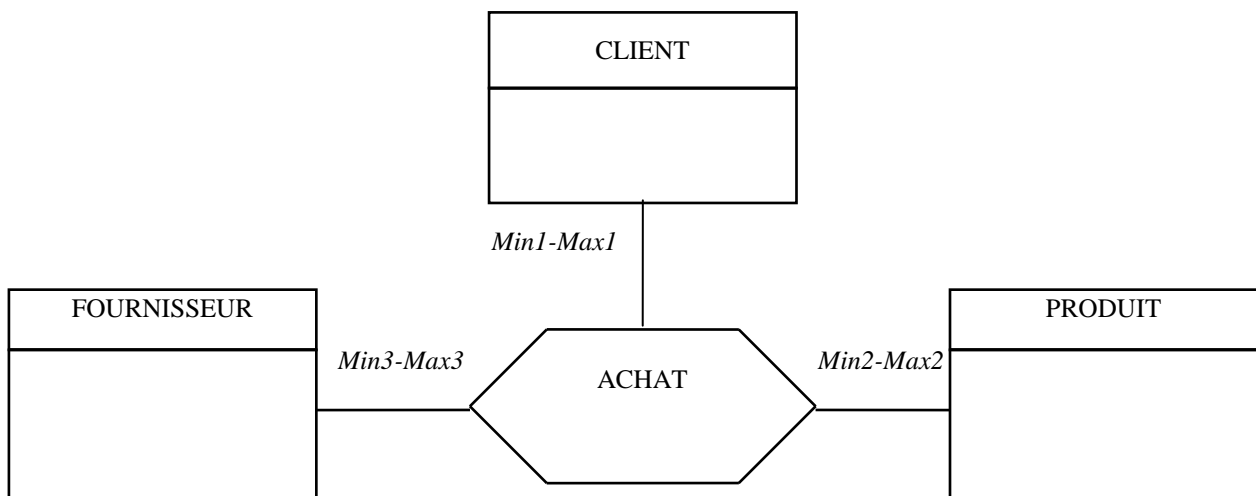
Ce schéma est donc **faux**.

Il y a un concept nouveau qui représente la proposition complexe toute entière; c'est le concept **achat**.

Une telle proposition doit donc être reformulée comme suit:

- ① les clients effectuent des **achats**
- ② tout **achat** concerne un produit
- ③ les **achats** s'effectuent chez les fournisseurs

Un achat est un **lien**, une relation **entre** des **clients**, des **produits** et des **fournisseurs**. Il s'agit d'un T.A. **ternaire** (c'est-à-dire de degré 3).



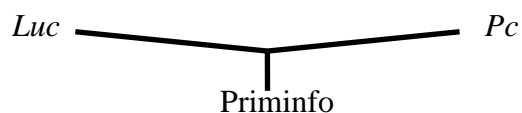
Quelles sont les cardinalités associées à chacun des rôles de ACHAT?

Les rôles sont :

- CLIENT *achète*
- PRODUIT *est acheté*
- FOURNISSEUR *fournit*

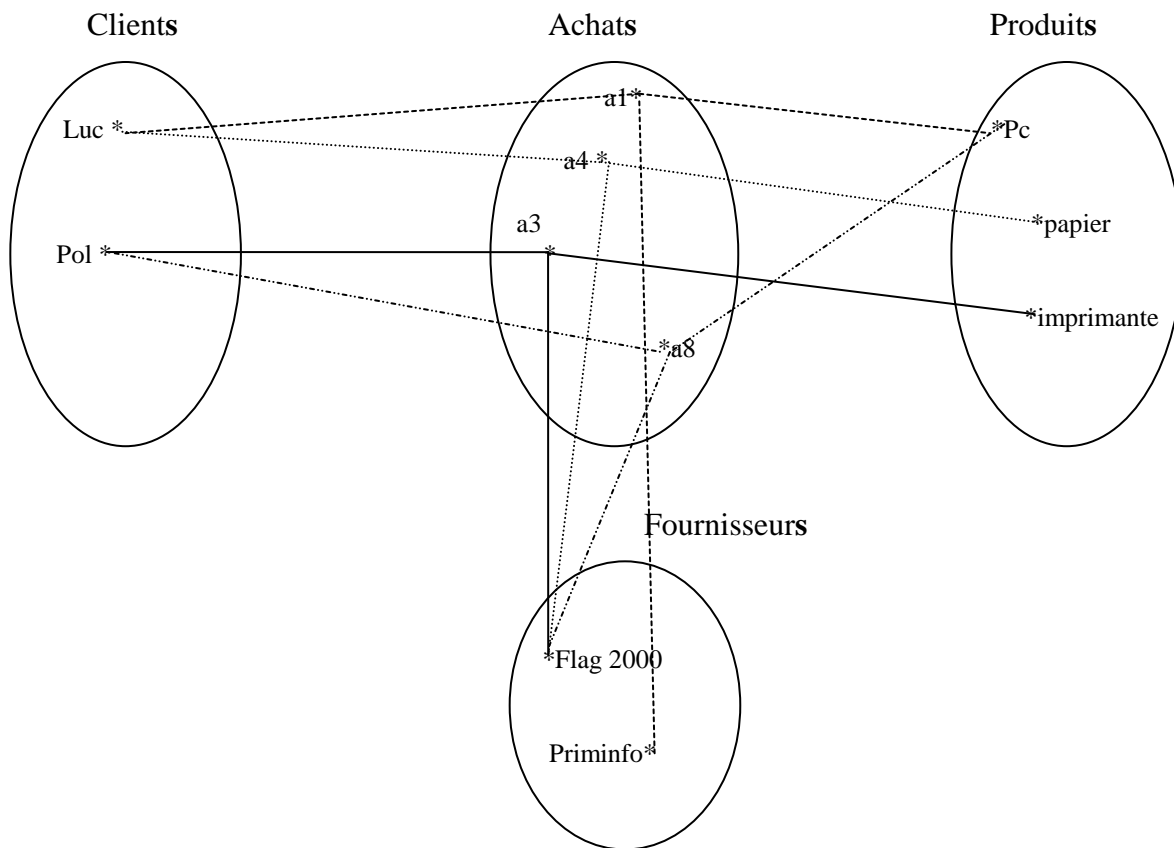
Toute **occurrence** du T.A. ACHAT est un lien entre une occurrence de CLIENT, une occurrence de PRODUIT et une occurrence de FOURNISSEUR. Toute occurrence d'ACHAT est donc une "étoile" à exactement 3 branches.

Exemple d'occurrence de ACHAT:



N.B. Dans la base de données correspondant à ce schéma, une occurrence d'achat mémorise le fournisseur chez lequel un client est allé s'approvisionner au moins une fois pour un produit donné. Par exemple, Luc a acheté un PC chez Priminfo. Si Luc a acheté trois PC chez Priminfo, cela ne fait l'objet que d'une occurrence d'achat dans la base de données.

Au point de vue des *ensembles d'occurrences* :



Calcul des cardinalités :

-Au point de vue client : (Min1-Max1)

A combien d'occurrences de ACHAT participe au minimum et au maximum une occurrence de CLIENT ? Min1=0 et Max1=N.

-Au point de vue produit : (Min2-Max2)

A combien d'occurrences de ACHAT participe au minimum et au maximum une occurrence de PRODUIT ? Min2= 0 et Max2=N.

-Au point de vue fournisseur : (Min3-Max3)

A combien d'occurrences de ACHAT participe au minimum et au maximum une occurrence de FOURNISSEUR ? Min3= 0 et Max3=N.

Les cardinalités se calculent donc en terme du nombre d'occurrences du T.A. auquel une occurrence du T.E. participe.

Un vrai T.A. ternaire ne contient que des rôles dont la **cardinalité maximum est N**.

Par conséquent, si un des rôles d'un T.A. ternaire est de cardinalité maximum égale à 1, c'est qu'il ne s'agit pas d'un vrai T.A. ternaire. Ce "pseudo" T.A. ternaire peut alors être réduit à plusieurs T.A. binaires.

Ce principe peut être généralisé à tout T.A. de degré >2 (T.A. quaternaire, ...).

Exemple :

Dans la gestion des réservations de place sur des vols dans une compagnie aérienne, un couple pilote/opérateur-radio est associé à chaque vol. Les couples pilote/opérateur-radio ne sont pas fixes, ils varient d'un vol à l'autre. Cette façon de voir les choses pourrait laisser sous-entendre qu'il existe un T.A. ternaire entre les T.E. *Vol*, *Pilote* et *Opérateur-radio*. Or, comme un vol n'est associé qu'à un couple pilote/opérateur-radio, la cardinalité maximum associée au rôle joué par le vol est de 1. Il ne s'agit donc pas d'un vrai T.A. ternaire.

Schéma incorrect

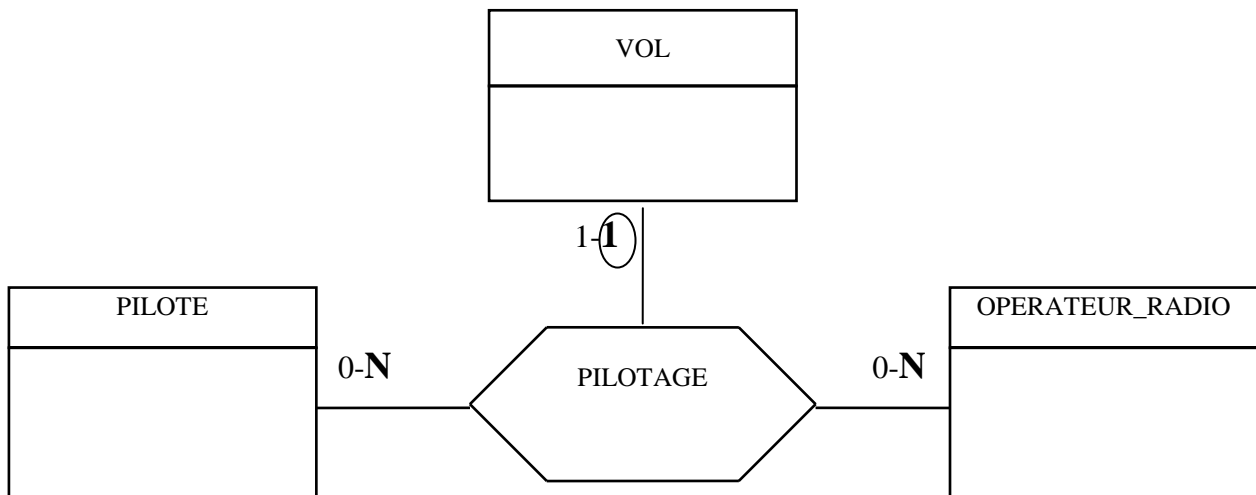
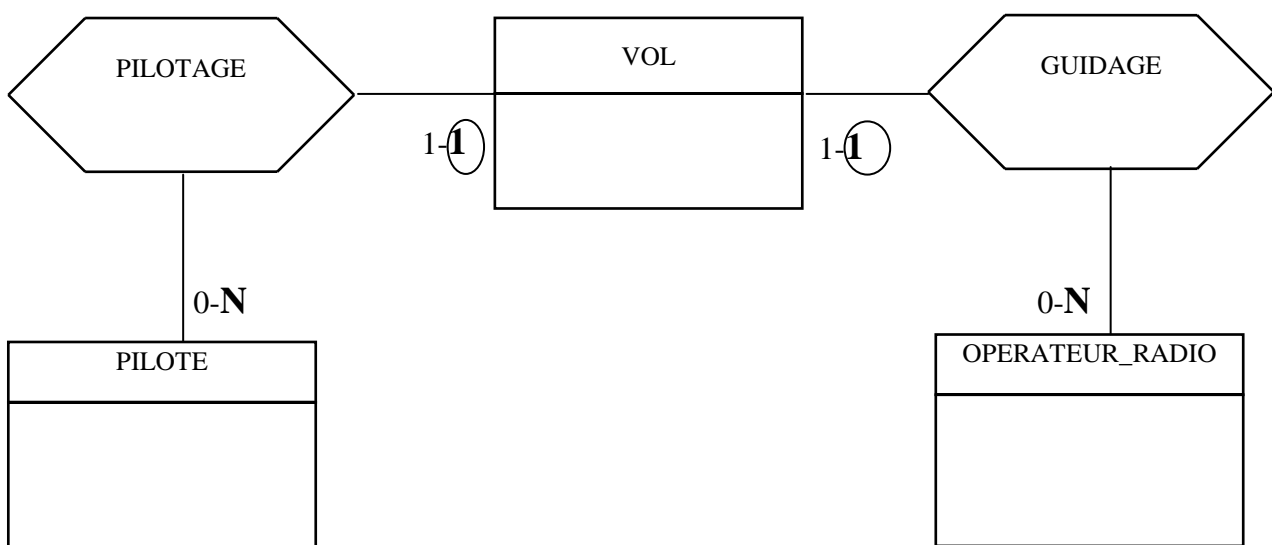


Schéma correct :

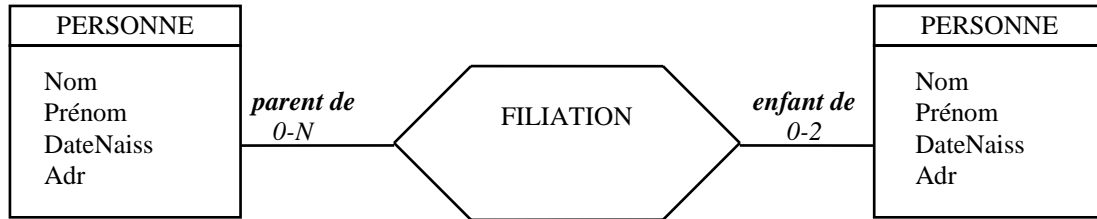


2.9.2 . T.A. récursive ou cyclique

Supposons qu'on veuille représenter des liens de filiation (liens parents-enfants) entre individus, ainsi que les liens de couples (mari et femme).

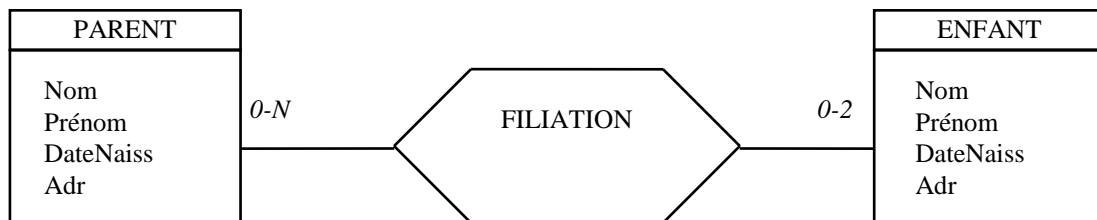
Les schémas ci-dessous sont-ils corrects ?

Première version:



Deux types d'entités qui apparaissent sur un schéma entité-association sont supposés représenter *deux concepts distincts*. Ce schéma est donc **incorrect**.

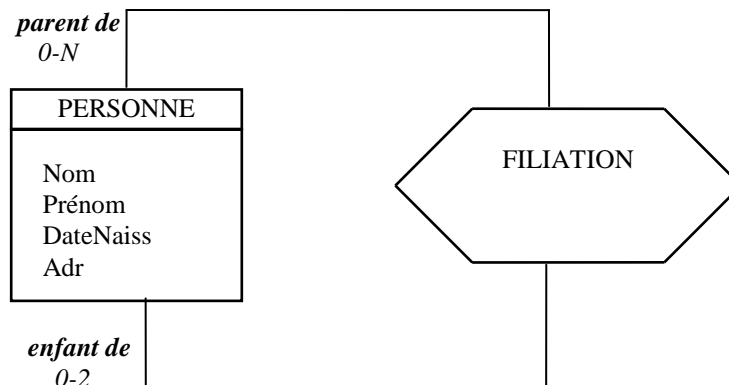
Seconde version:



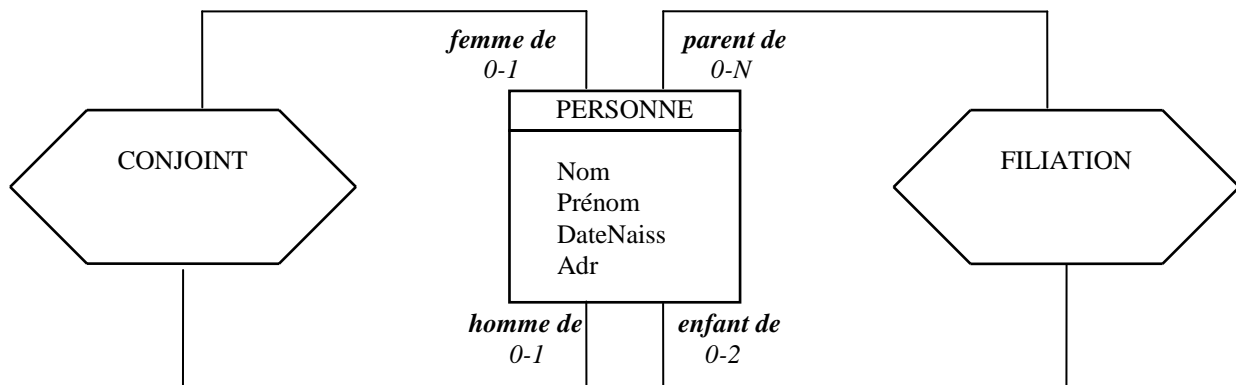
Tout *enfant* peut *lui-même être parent*. Si c'est le cas, les informations concernant la même personne seront *représentées deux fois* dans la B.D. D'où **redondance**. Ce schéma est également à rejeter.

Il est donc nécessaire de pouvoir représenter des T.A. dites **récurrentes** ou **cycliques**.

Le schéma suivant est **correct** et **dépourvu de toute redondance** :



Si les liens entre conjoints doivent aussi être représentés, le schéma devient :

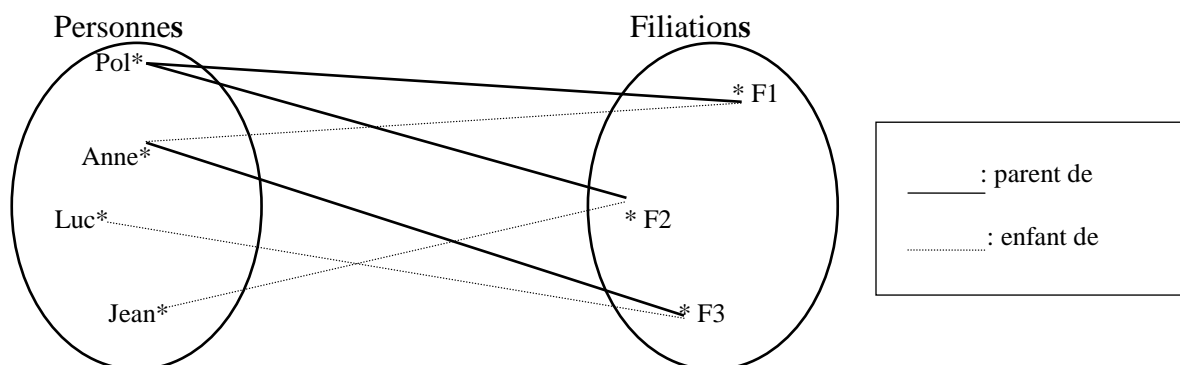


N.B.1: Dans les T.A. récursives, il est **indispensable** de *préciser les rôles* sur le schéma. En effet, le même T.E. joue deux rôles différents; il est donc important de savoir à quel rôle correspondent quelles cardinalités. Un schéma contenant des T.A. récursives sans rôle est *ambigu*.

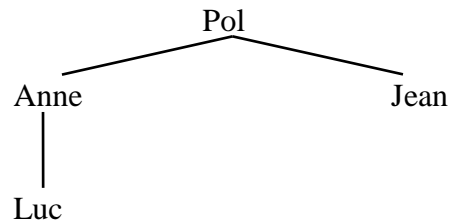
N.B.2: Le T.A. récursive FILIATION ne signifie pas qu'une même occurrence de PERSONNE à un lien de filiation avec elle-même. Il s'agit de liens entre des occurrences distinctes de PERSONNE.

N.B.3. Les cardinalités associées au rôle *enfant de* sont 0-2 et non 2-2, même si dans la réalité, tout individu est l'enfant de deux parents. Il ne faut pas perdre de vue qu'on établit le schéma de la base de données et non de la réalité. Il est impossible d'enregistrer dans une base de données les deux parents de tous les individus repris dans cette base de données: il faudrait y enregistrer l'humanité entière. On admet donc dans la base de données des individus dont on ne connaît pas (càd dont on n'a pas enregistré) les parents.

Au niveau occurrences :



Ces trois occurrences de FILIATION (f1, f2 et f3) permettent de représenter l'arbre généalogique suivant :



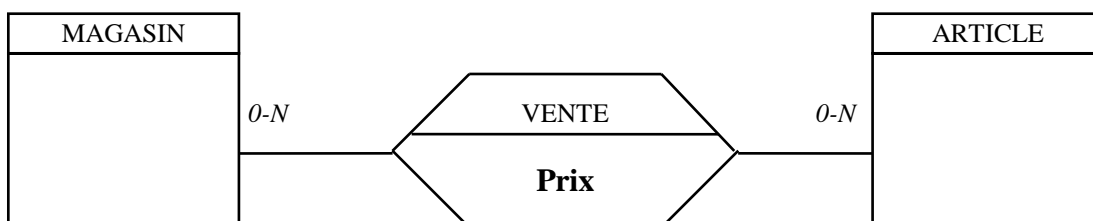
2.9.3 . T.A. avec attributs

Une extension possible au schéma entité association est *l'ajout d'attributs à un T.A. N à N*.

Un attribut peut être attaché à un T.A. N à N, s'il s'agit d'une caractéristique qui dépend du lien entre les T.E. et non pas uniquement à un des T.E. reliés.

Exemple 1 :

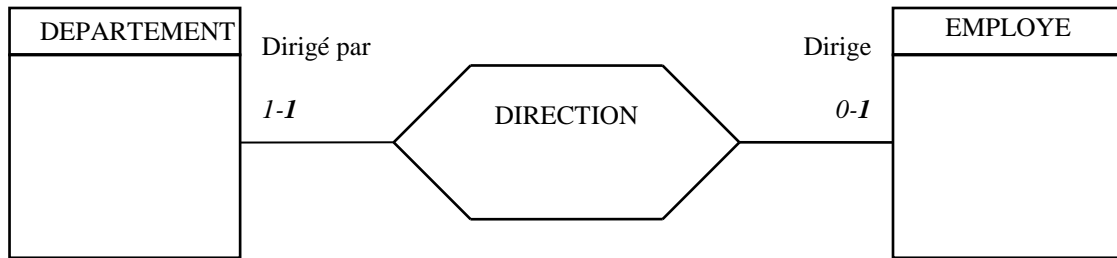
Un magasin vend des articles et un article peut être vendu par plusieurs magasins.
Le prix de vente d'un article dépend du magasin.



L'attribut *Prix* ne peut pas être associé à MAGASIN car un magasin n'a pas de prix fixe pour tous ses articles. L'attribut *Prix* ne peut pas être associé à ARTICLE car un article n'est pas vendu au même prix dans tous les magasins. Le prix varie en fonction de l'article ET du magasin. Le prix est donc bien une caractéristique du T.A. VENTE.

Exemple 2 :

Reprenons le T.A. DIRECTION entre les T.E. DEPARTEMENT et EMPLOYE: un employé ne peut être directeur que d'un département à la fois, et un département doit avoir un et un seul directeur.



On ne considérait que les directeurs *courants*; on ne mémorisait pas l'historique des directeurs des départements.

Que faut-il modifier au schéma ci-dessous pour tenir compte de cet historique?

Les cardinalités maximales (=1) passent à N. En effet, un même département peut avoir vu se succéder plusieurs directeurs. D'autre part, un même employé peut avoir dirigé plusieurs départements à des moments différents dans le temps.

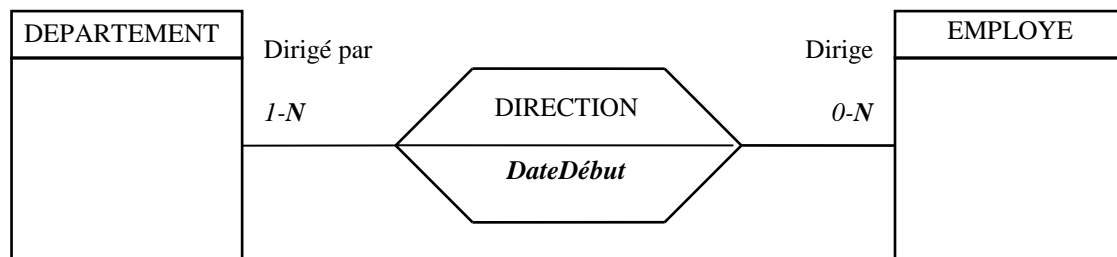
D'autre part, si l'on désire pouvoir retrouver le directeur *actuel* de chaque département ainsi que *l'ordre de succession des différents directeurs* d'un même département, il faut ajouter l'attribut *DateDébut* au T.A. DIRECTION spécifiant la date de début de chaque mandat de directeur.

Pour retrouver le directeur *actuel* d'un département donné, il suffit de retrouver, parmi toutes les occurrences du T.A. DIRECTION concernant ce département, celle qui a la date de début la plus récente.

Si l'on désire retrouver tous les directeurs d'un département donné et les classer par ordre de succession au poste de directeur, il suffit de retrouver toutes les occurrences du T.A.

DIRECTION concernant ce département et de les trier par date de début croissante.

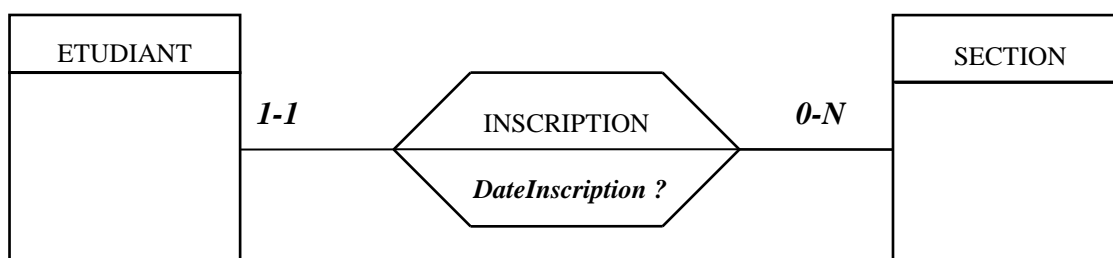
Le schéma devient:



Notons qu'il est inutile d'associer des attributs à des T.A. 1 à N.

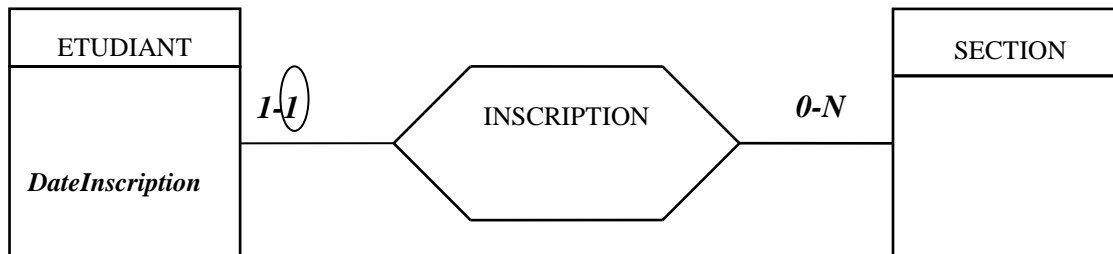
En effet, l'attribut est alors une caractéristique du T.E. qui joue le rôle côté cardinalité maximum = 1.

Exemple 1 :



La date d'inscription pourrait être considérée d'un point de vue logique comme une caractéristique du T.A. Inscription.

Cependant, comme un étudiant n'est relié qu'à un seul lien inscription, la date d'inscription peut être placée dans le T.E. Etudiant; la date d'inscription devenant alors une caractéristique de tout étudiant.

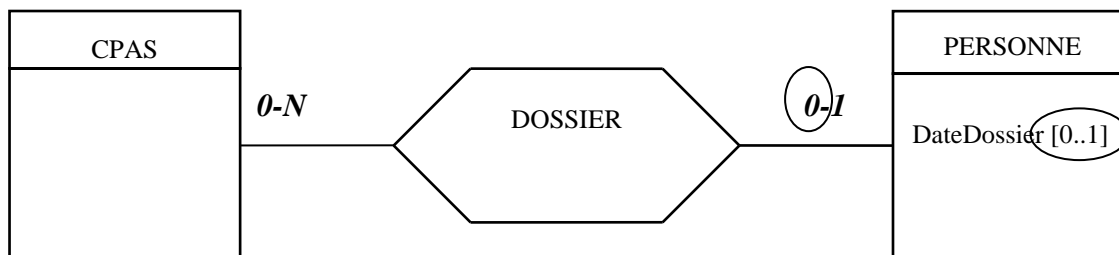


Pas d'attribut attaché à un T.A. 1 à N :

Placer l'attribut dans le T.E. qui joue le rôle côté cardinalité maximum = 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.

Exemple 2 :



2.10. Contraintes additionnelles

Le **formalisme** du schéma entité-association est **limité**; il ne permet pas de représenter toutes les informations. Il ne suffit donc pas à lui seul. *Une documentation annexe complétera le schéma.*

Cette documentation comprendra:

- ① la définition de tous les termes apparaissant sur le schéma
 - T.E.
 - T.A.
 - Attribut
- ② des **contraintes** dites **additionnelles** car complétant le schéma.

Ces contraintes peuvent être exprimées en langage naturel (français, anglais,...) ou en pseudo-langage (cfr exemples plus loin).

Contraintes d'identifiant

Un T.E. peut avoir **un ou plusieurs** identifiant(s). Il peut également n'en avoir **aucun**.

☛ Rappelons que, au cours de son existence, une occurrence d'entité *ne peut voir la valeur de son identifiant changer*. Un identifiant est **fixe**.

2.10.1. Identifiant composé d'attributs

① Le cas le plus simple est celui d'un identifiant unique **composé d'un seul attribut**. Ce dernier est alors souligné.

Exemple :

OUVRAGE
<u>NumOuv</u>

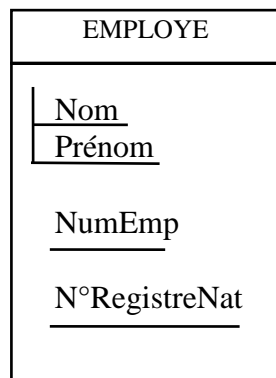
② Un T.E. peut également posséder un identifiant **composé de plusieurs attributs**. Ils sont soulignés et reliés par une ligne verticale.

Exemple:

PROFESSEUR		
<table><tr><td><u>Nom</u></td></tr><tr><td><u>Prénom</u></td></tr></table>	<u>Nom</u>	<u>Prénom</u>
<u>Nom</u>		
<u>Prénom</u>		

③ Un T.E. peut posséder **plusieurs identifiants**. Chaque attribut identifiant ou groupe d'attributs identifiant est souligné, mais indépendamment des autres.

Exemple:



Un employé a 3 identifiants :

- NumEmp
- N°RegistreNat
- Nom +Prénom

2.10.2. Identifiant hybride

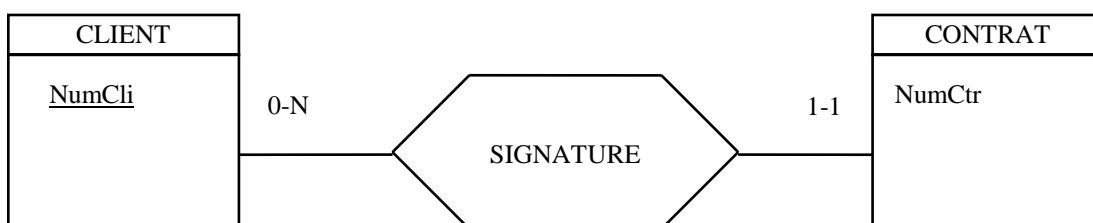
Est appelé identifiant **hybride** un identifiant composé de:
0, 1 ou plusieurs attribut(s)
ET
un ou plusieurs rôle(s) via un ou plusieurs T.A.(s).

En effet, l'identifiant d'un T.E. peut dépendre de l'identifiant d'un autre T.E. qui lui est relié via un T.A.

Exemple 1:

Dans une compagnie d'assurance, un client signe des contrats. Supposons qu'un contrat n'est signé que par un seul client, et qu'un contrat n'a pas un numéro unique au sein de la compagnie. Le T.E. CONTRAT n'a donc pas d'identifiant constitué uniquement d'attributs. Par contre, les contrats signés par un même client possèdent un numéro d'ordre (*NumCtr*).

L'attribut 'NumCtr' est donc identifiant au sein de l'ensemble des contrats appartenant à un même client.



Le formalisme des schémas entité-association ne permet pas de représenter ce type d'identifiant **hybride**. Il faut donc avoir recours à une **contrainte supplémentaire**. L'identifiant de CONTRAT (id(CONTRAT)) est donc constitué de l'identifiant de CLIENT (on se contentera de noter le nom du T.E. relié, soit CLIENT ici) et de l'attribut *NumCtr*.

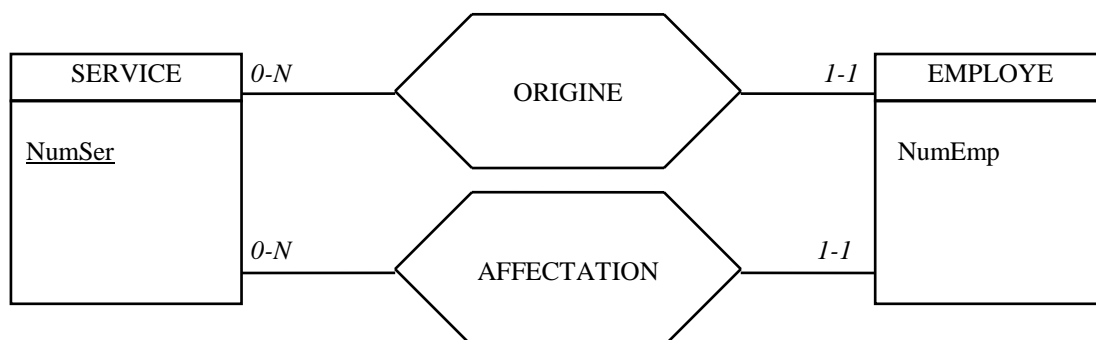
La contrainte est donc, en pseudo-langage:

id(CONTRAT): CLIENT, NumCtr

N.B.: Si il existe plusieurs T.A. entre CLIENT et CONTRAT, il est indispensable de préciser dans la contrainte via quel T.A. le T.E. CLIENT intervient dans l'identifiant de CONTRAT, soit le T.A. SIGNATURE (sinon *risque d'ambiguïté*). La contrainte devient :

id (CONTRAT): SIGNATURE.CLIENT, NumCtr

Exemple 2 :



L'attribut *NumEmp* de EMPLOYE n'est pas identifiant au sein de la société. Par contre, la valeur de *NumEmp* pour un employé particulier est unique au sein d'un service. Cependant, des *réaffectations peuvent avoir lieu*. Au cours de son existence, une occurrence de EMPLOYE *pourrait donc voir la valeur de son identifiant varier*; ce qui est **à exclure!** Une solution est de garder trace du *service d'origine* de l'employé, et de définir l'identifiant d'EMPLOYE comme étant le n° du service d'origine combiné à l'attribut *NumEmp*.

La contrainte supplémentaire est donc :

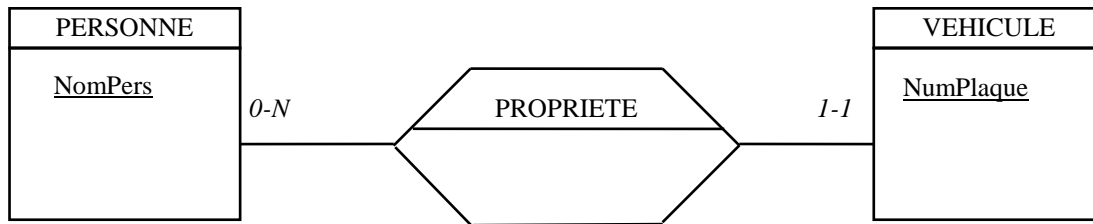
id (EMPLOYE) : ORIGINE.SERVICE, NumEmp

2.10.3. Identifiant de type d'association

- *Identifiant de types d'association implicite*

① 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*.

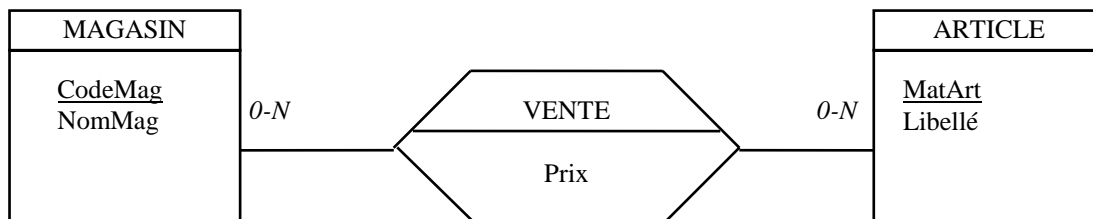
Exemple:



Toute occurrence du T.A. PROPRIETE est un lien entre exactement **une occurrence de PERSONNE** et **une occurrence de VEHICULE**. Comme un véhicule ne peut avoir qu'un propriétaire, toute occurrence du T.A. PROPRIETE est identifiée par le véhicule relié. Autrement dit, toute occurrence du T.A. PROPRIETE est identifiée par l'attribut NumPlaque du véhicule relié.

② **Par défaut**, un type d'association **N à N** est identifié *par la combinaison des identifiants des types d'entité reliés*.

Exemple 1:



Toute occurrence du T.A. VENTE est un lien entre exactement **une occurrence de MAGASIN** et **une occurrence d'ARTICLE**.

Chaque occurrence du T.A. VENTE est donc identifiée par la combinaison de l'identifiant du magasin relié d'une part et de l'identifiant de l'article relié d'autre part.

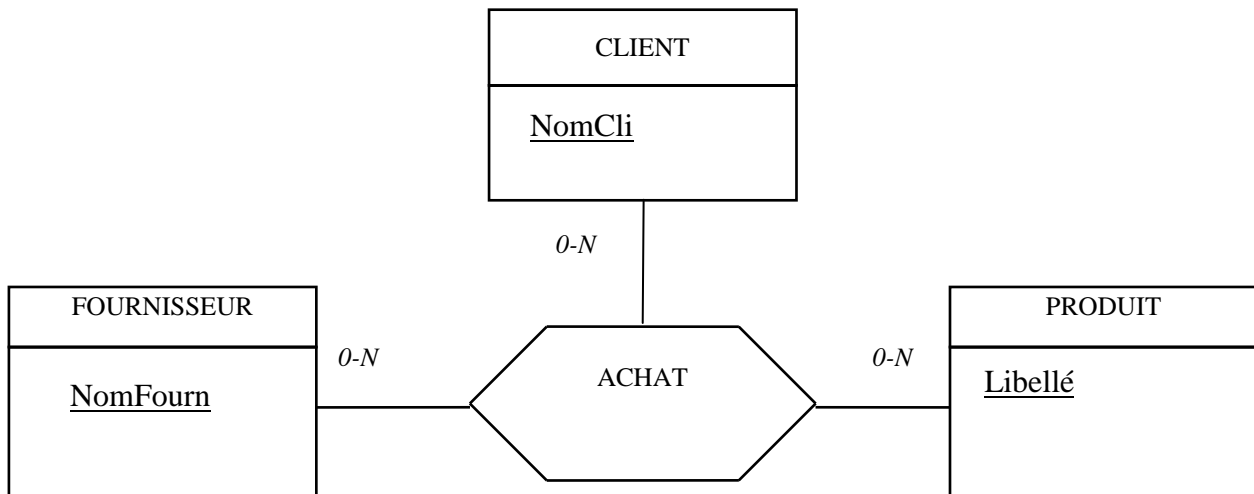
Soient l'occurrence de MAGASIN «Le cellier du vigneron» dont l'identifiant est « CellVig » et l'occurrence d'ARTICLE « Château Lemaître 1997 » » (bouteille de vin rouge de 75cl) dont l'identifiant est « ChLem97 ». L'occurrence du T.A. VENTE qui relie le magasin «Le cellier du vigneron» et l'article « Château Lemaître 1997 » est identifiée par la combinaison CellVig + ChLem97. En effet, il n'y aura pas dans la base de données deux occurrences différentes du T.A. VENTE qui relieront les occurrences CellVig et ChLem97. Notons que **l'attribut prix n'intervient pas dans l'identifiant**.

Le schéma ci-dessus contient **implicitement** la contrainte suivante:

id(VENTE) : MAGASIN, ARTICLE

N.B. Puisque cette contrainte est implicite au schéma, il n'est donc pas nécessaire de l'ajouter explicitement dans les contraintes additionnelles.

Exemple 2 :



Toute occurrence du T.A. ACHAT est un lien entre exactement **une occurrence de CLIENT**, **une occurrence de PRODUIT** et **une occurrence de FOURNISSEUR**.

Chaque occurrence du T.A. ACHAT est donc identifiée par la combinaison de l'identifiant du client, de l'identifiant du produit et de l'identifiant du fournisseur.

N.B. Pour rappel, dans la base de données correspondant à ce schéma, une occurrence d'achat mémorise le fournisseur chez lequel un client est allé s'approvisionner (au moins une fois) pour un produit donné. Par exemple, Luc est allé s'approvisionner en Pc chez Priminfo. Si Luc a acheté trois Pcs chez Priminfo au cours du mois, cela ne fait l'objet que d'**une** occurrence d'achat dans la base de données.

L'occurrence d'achat qui correspond au fait que Luc est allé un jour acheter des Pcs chez Priminfo est identifiée au sein de la base de données par la combinaison:
Luc + Pc + Priminfo.

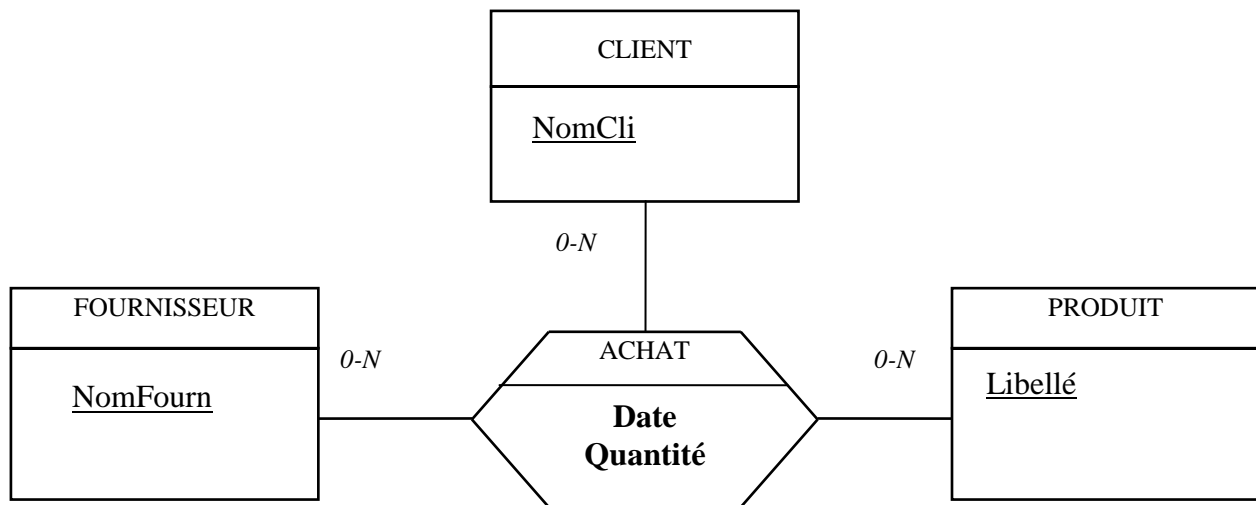
Le schéma ci-dessus contient donc implicitement la contrainte suivante:

id(ACHAT) : CLIENT, PRODUIT, FOURNISSEUR

- **Identifiant de types d'association explicite**

Certains types d'association peuvent cependant ne pas être identifiés par la simple combinaison des identifiants des types d'entité reliés.

Reprenons l'exemple 2 ci-dessus. Supposons que ce qui nous intéresse ce n'est pas d'enregistrer *chez qui quel client* s'approvisionne et *quel produit* il y achète, mais bien les achats réellement effectués par les clients. Dans ce cas, si Luc a acheté **trois fois des Pc** chez Priminfo à des dates différentes, on mémorisera **trois occurrences** d'achats différentes dans la base de données, en précisant à chaque fois la date de l'achat ainsi que la quantité de produit achetée. Il faut alors ajouter deux attributs au T.A. ACHAT, à savoir les attributs *Date* et *Quantité*.



Chaque occurrence du T.A. ACHAT est ici identifiée par la combinaison de l'identifiant du client, de l'identifiant du produit, de l'identifiant du fournisseur et de la date à laquelle l'achat a été effectué.

Ainsi par exemple, l'achat par Luc de 3 Pcs chez Priminfo le 22/2/2005 est identifié au sein de la base de données par la combinaison:

Luc + Pc + Priminfo + 22/2/2005

L'achat par Luc de 2 Pcs chez Priminfo le 28/2/2006 est identifié au sein de la base de données par la combinaison:

Luc + Pc + priminfo + 28/2/2006

Il faut noter que la quantité n'intervient pas dans l'établissement de l'identifiant du T.A. ACHAT. Il faut toujours choisir l'identifiant minimum.

Il est nécessaire de rajouter une contrainte additionnelle au schéma de cette base de données, pour spécifier l'identifiant du T.A. ACHAT, puisqu'il ne s'agit pas de l'identifiant par défaut:

id(ACHAT) : CLIENT, PRODUIT, FOURNISSEUR, Date

N.B. Il ne faut pas souligner l'attribut *Date* sur le schéma; il n'est en effet pas à lui seul identifiant du T.A.

***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 (cfr 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.*

2.11. Transformation de schéma

Il existe des transformations d'un schéma A en un schéma A' qui préservent la sémantique:

$$\text{schéma A} \Leftrightarrow \text{schéma A'}$$

Les deux schémas sont sémantiquement équivalents.

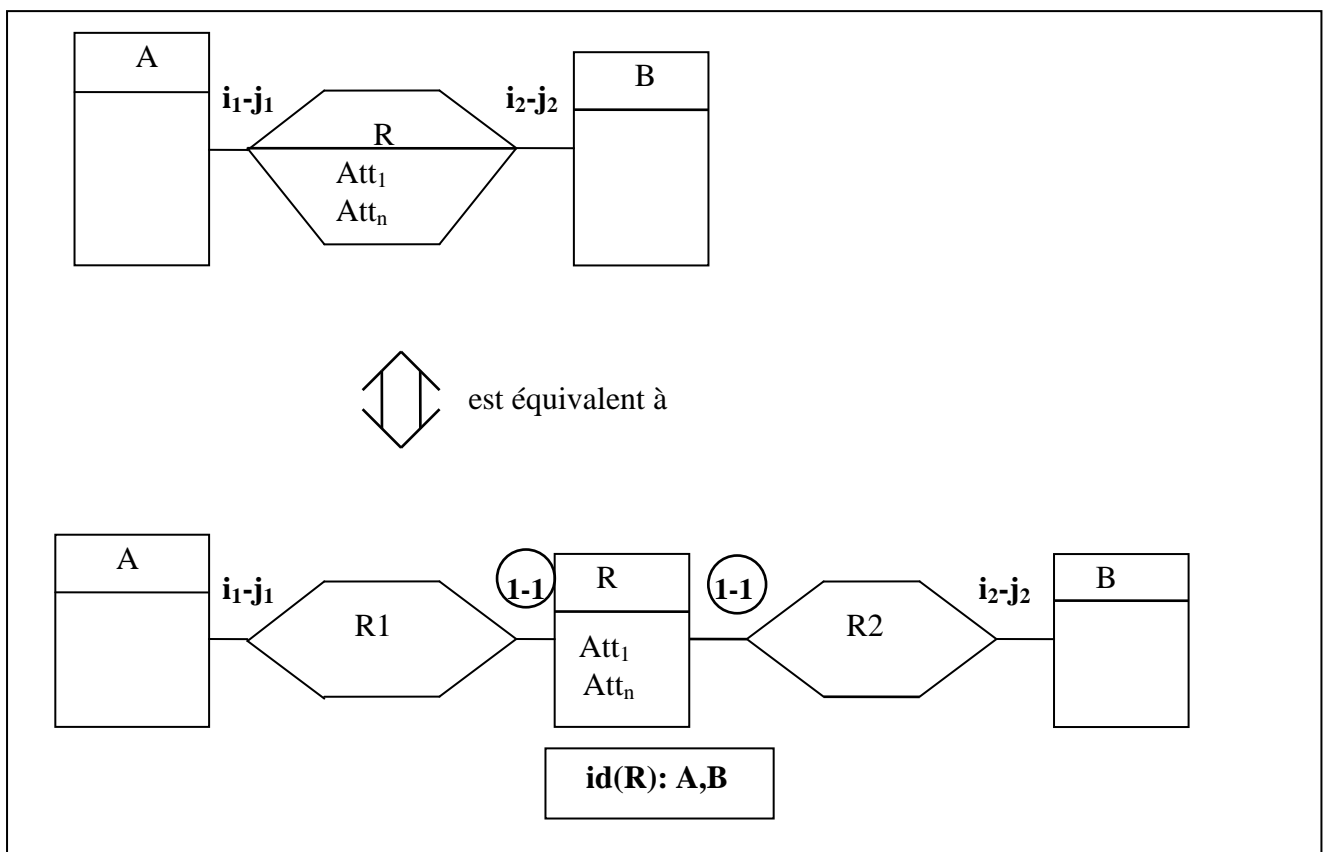
2.11.1. Transformation d'un type d'association en un type d'entité

On peut transformer tout T.A. en une T.E. (et deux nouveaux T.A.).

Les deux représentations sont équivalentes.

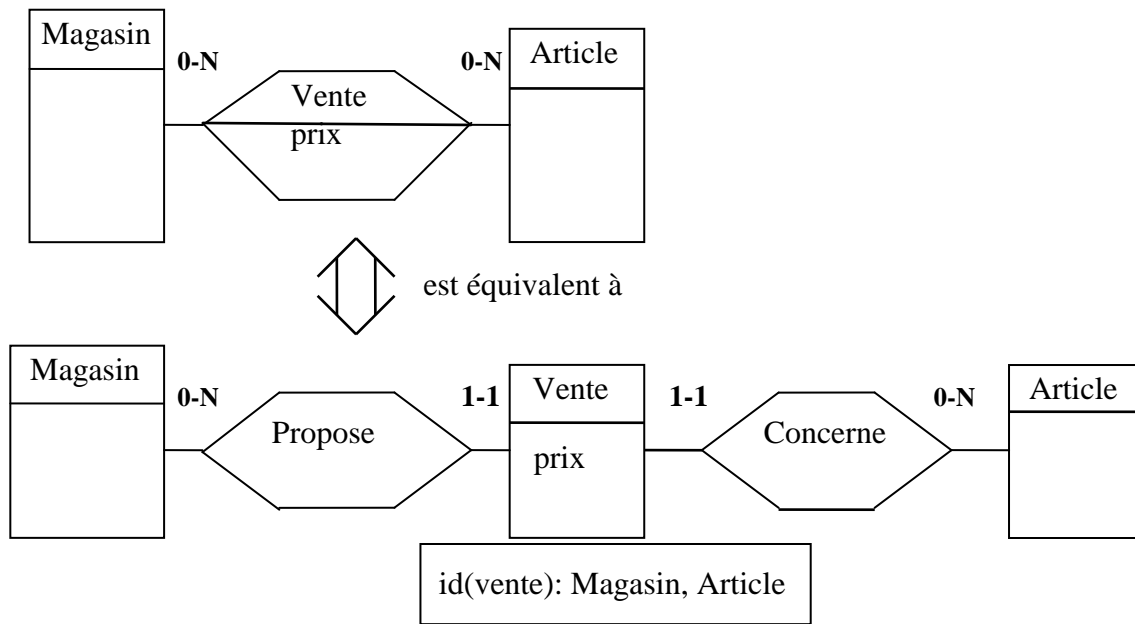
De plus, cette transformation est **réversible**.

Forme générale



L'identifiant du nouveau type d'entité R est la combinaison des identifiants des T.E. reliés, à savoir, A et B.

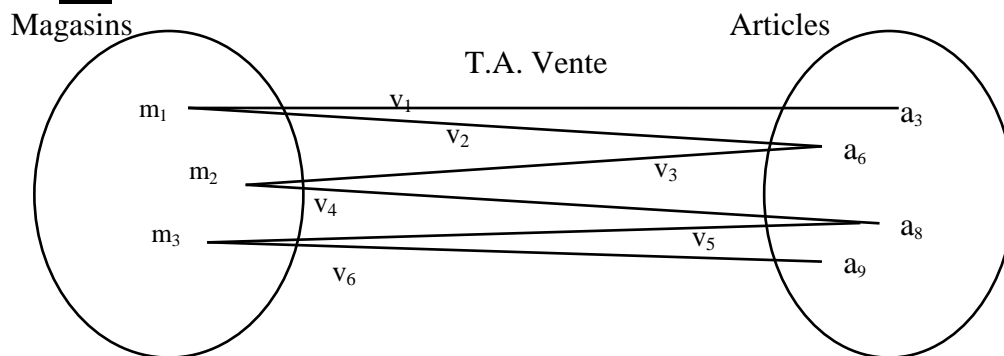
Exemple 1: T.A. binaire



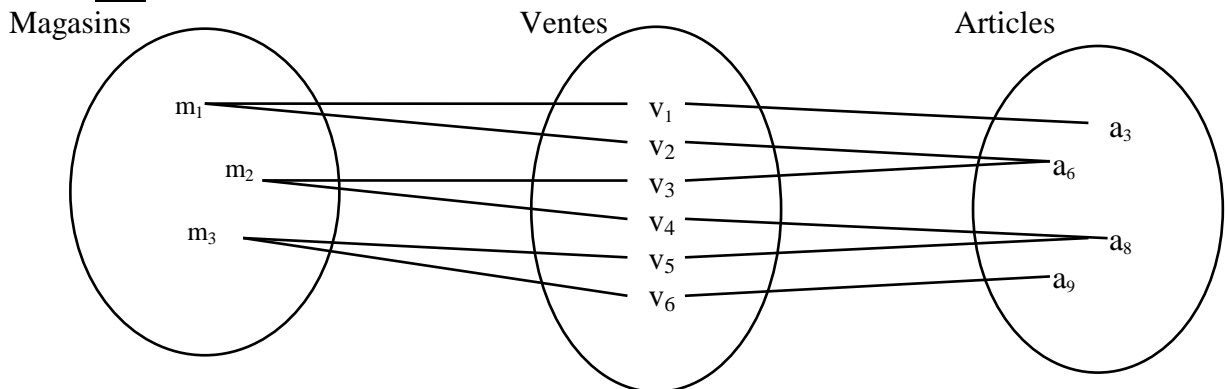
L'identifiant du nouveau type d'entité *Vente* est la combinaison des identifiants des T.E. reliés, à savoir, *Magasin* et *Article*.

Au niveau occurrences:

① via un **T.A.** *Vente*

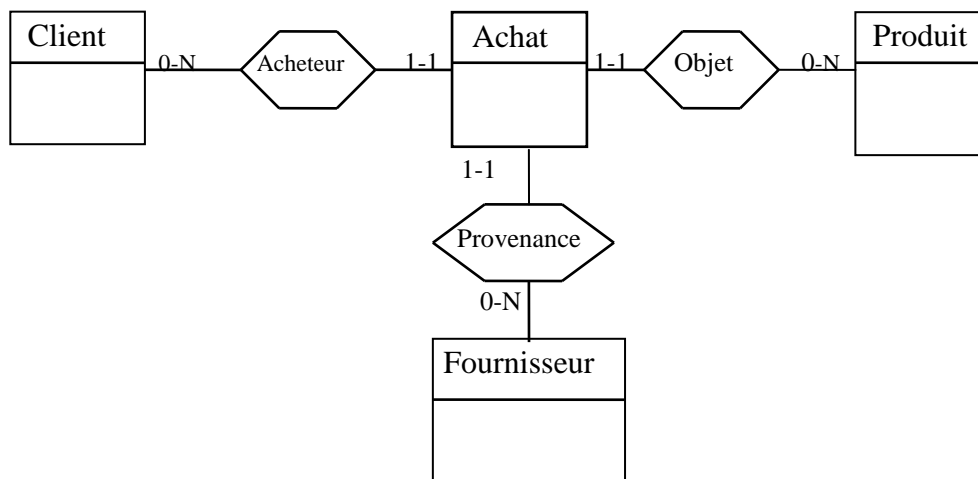
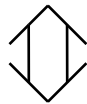
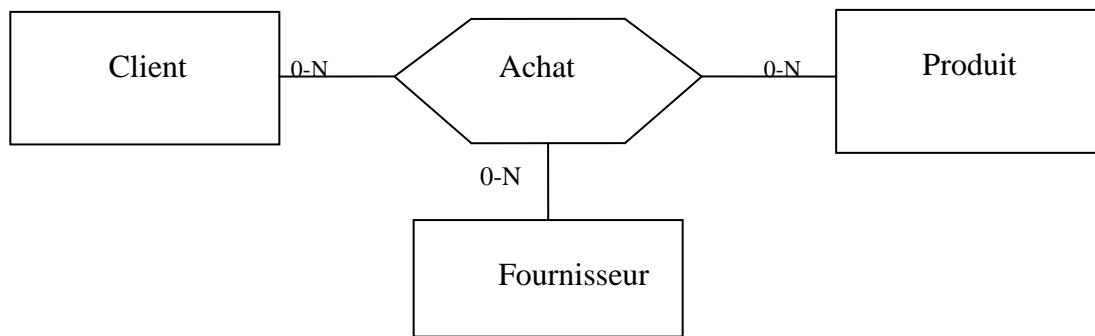


② via un **T.E.** *Vente*



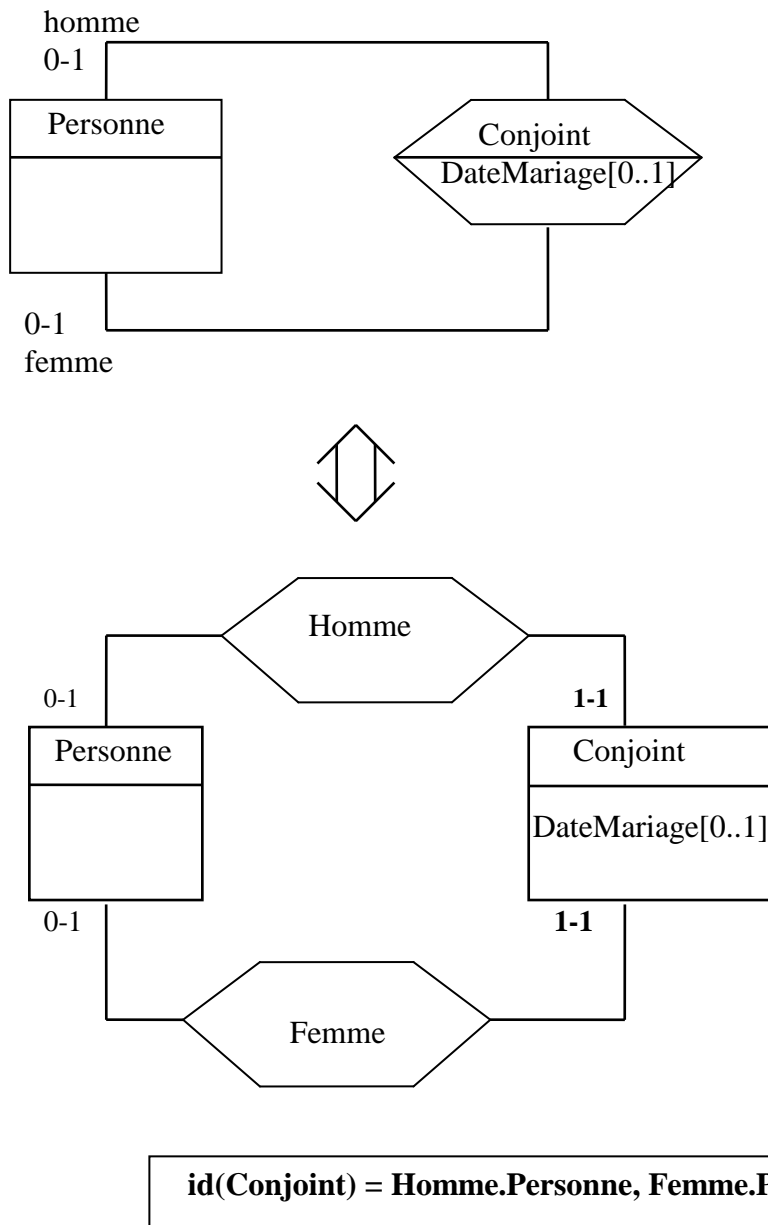
Si on choisit de représenter *Vente* sous forme d'un T.E., cela revient à représenter au niveau occurrences l'ensemble des ventes.

Exemple 2: T.A. ternaire



id(Achat) = Client, Produit, Fournisseur

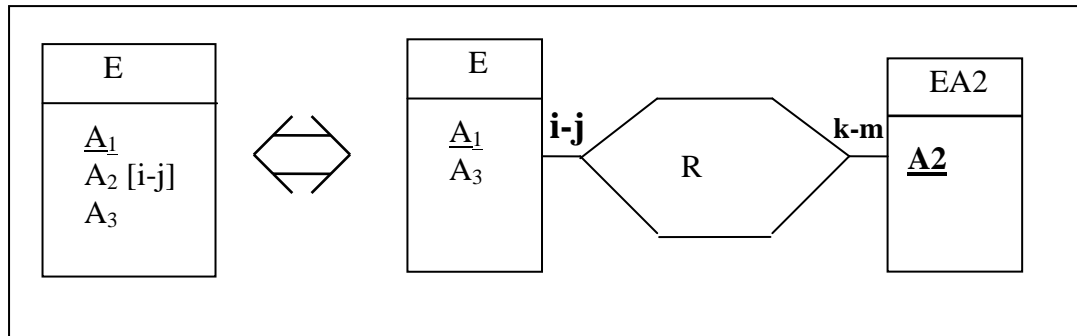
Exemple 3: T.A. récursive



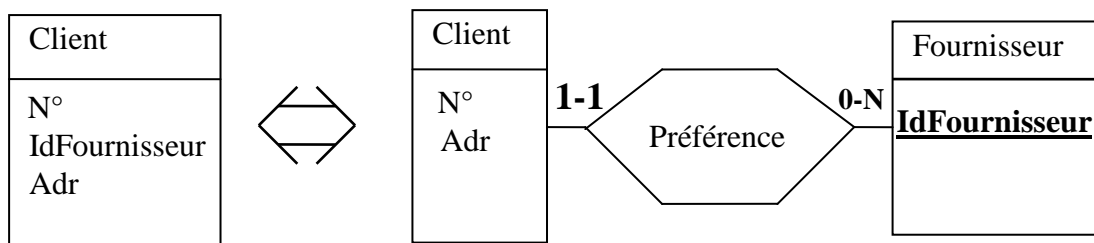
Une occurrence de *Conjoint* est un couple entre exactement un homme et une femme.

2.11.2. Transformation d'attribut en type d'entité

Forme générale

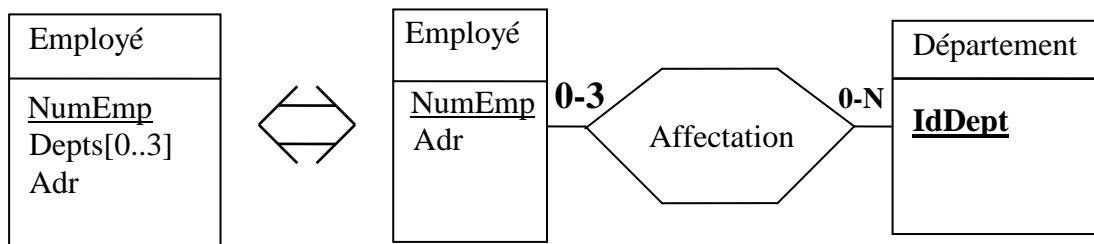


Exemple 1 :



Car l'attribut *IdFournisseur* est obligatoire et monovalué.

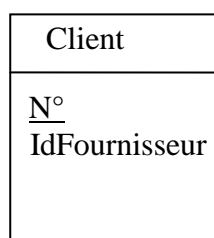
Exemple 2 :



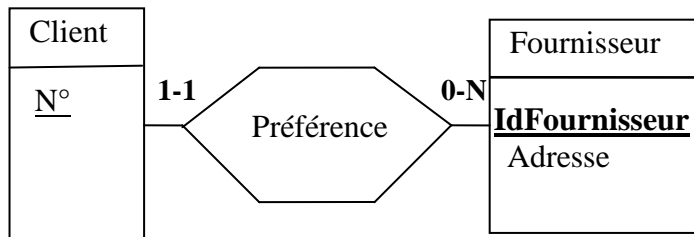
Utilité

① Quand on se rend compte qu'un attribut représente un concept à part entière, on en fait un T.E.. C'est par exemple le cas, quand on doit lui rajouter une caractéristique.

Exemple: soit :



On veut ajouter la notion d'adresse du fournisseur.



② Quand le même attribut est *cité* à plusieurs endroits, il peut s'agir d'un concept à part entière à isoler, càd en faire un nouveau T.E.

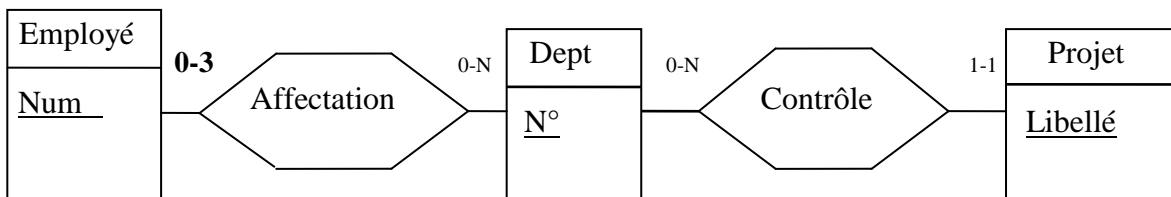
Exemple : L'attribut *N°Dept* apparaît dans

- le T.E. *Employé*, car un employé peut être affecté à un ou plusieurs (max 3) département(s)
- le T.E. *Projet*, car un projet est contrôlé par un département.

Avant:



Après:



Partie 2

Analyse logique

Chapitre 3. Bases de données relationnelles

Le modèle relationnel a été introduit par Codd dès 1970. Ce modèle est basé sur une structure unique et uniforme: la **RELATION**. Le modèle relationnel a une base théorique solide.

La plupart des bases de données actuelles sont de type relationnel.

Le langage d'utilisation des bases de données relationnelles le plus connu est **SQL**, pour **S**tructured **Q**uery **L**anguage. SQL a été développé par IBM.

Il s'agit d'un langage d'accès normalisé aux bases de données relationnelles. Ce n'est pas un langage de programmation à proprement parlé (SQL est NON PROCEDURAL), mais un langage de requêtes basé sur la théorie des ensembles. Pour interroger la base de données, on n'indique plus au SGBD **COMMENT** retrouver l'information, on se contente de spécifier ce qu'on veut obtenir (le **QUOI**). On parle encore de langage **déclaratif**.

Un standard SQL a été défini conjointement par ANSI (American National Standards Institute) et ISO (International Standards Organization).

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.

La relation est une **table de valeurs**.

3.1.1. Table

Une table représente **un concept** qui peut être:

	un objet ,	une personne ,	un fait ,	une situation , ...
	↓	↓	↓	↓
<i>Exemples:</i>	<i>voiture</i>	<i>propriétaire</i>	<i>emprunt</i>	<i>accident</i>
	<i>ouvrage</i>	<i>client</i>	<i>construction</i>	
	<i>article</i>	<i>emprunteur</i>		
	<i>magasin</i>			

La notion de table correspond à la notion de **TYPE D'ENREGISTREMENTS** des bases de données réseaux.

Une table contient des lignes et des colonnes.

3.1.2. Ligne

Une ligne représente **une occurrence** du type d'entité.

La notion de **LIGNE** correspond donc à la notion d'**ENREGISTREMENT** des bases de données réseaux.

Ainsi, une ligne de la table **CLIENT** reprend-elle toutes les informations d'un client particulier, par exemple, le client Dupond:

Dupond	Georges	32, rue de Fer, 5000, Namur	081/21.22.23	24/12/63	marié
--------	---------	-----------------------------	--------------	----------	-------

Les données d'une même ligne *sont en **relation** les unes avec les autres*; elles forment un tout. D'où, le nom de relation pour une table.

☛ Il ne faut toutefois pas confondre **relation** (ou table) avec la notion **de type d'association** du schéma conceptuel. Nous verrons plus loin comment représenter en relationnel les types d'associations.

3.1.3. Colonne

Une colonne représente **un attribut** du type d'entité.

Les attributs permettent d'exprimer le rôle joué par chacune des valeurs d'une même ligne.

La notion de **COLONNE** correspond à la notion de **CHAMP** dans les bases de données réseaux.

Pour chaque attribut/colonne, on définit un **nom**, un **type**, une **longueur**, et éventuellement un **domaine** des valeurs permises.

☛ Par opposition aux bases de données réseaux où les attributs *multivalués* et *décomposables* pouvaient être gérés par le SGBD, il n'y a **qu'une seule classe d'attribut** disponible dans les bases de données relationnelles.

Tous les attributs sont atomiques et monovalués .

Il n'est donc pas possible par exemple de prévoir un seul attribut multivalué prénoms listant les différents prénoms d'une personne, ou un attribut décomposable adresse permettant de

distinguer la rue, le numéro, la boîte postale, la ville et le code postal (cfr section 3.4.1. pour la représentation des attributs décomposables et multivalués).

☛ Contrairement aux bases de données de type réseaux, il est ***possible de rajouter, modifier ou supprimer des colonnes après création de la table.***

3.1.4. Valeur

La valeur est donnée par l'intersection d'une ligne et d'une colonne. Il s'agit de la valeur d'un attribut pour une occurrence particulière de la table.

Plaque	Marque	Modèle	Date-fabrication	Kilométrage
XL935	Renault	Espace	12/08/97	55.897

La notion de **VALEUR** correspond à la **VALEUR D'UN CHAMP** dans les bases de données réseaux.

Il existe une valeur particulière, à savoir, la valeur **NULL**. Une ligne particulière peut prendre la valeur NULL pour un attribut particulier. La valeur null peut avoir trois significations:

① La valeur de l'attribut est inconnue pour certaines occurrences.

Exemple: le numéro de compte d'un client particulier est inconnu (différent de la valeur 0 si l'attribut numéro de compte est défini de type numérique)

② L'attribut ne s'applique pas à certaines occurrences.

Exemple: nom d'épouse pour les hommes

③ Certaines occurrences ne possèdent pas de valeur pour l'attribut.

Exemple: numéro de téléphone pour les personnes sans téléphone

3.2. Notion d'ordre

3.2.1. Ordre des lignes

Une table est un ensemble de lignes. La notion d'ensemble est à prendre au sens mathématique. Les éléments d'un ensemble ne sont pas ordonnés. Les lignes d'une table ne sont donc pas ordonnées. Même si au *niveau physique*, les lignes d'une table sont stockées dans un fichier, et par conséquent dans un certain ordre (physique), il n'y a **pas de notion d'ordre au niveau logique**.

Il n'est donc pas possible d'accéder à la première ligne, à la 5ème ligne ou à la dernière ligne d'une table.

De même, l'ordre des lignes sélectionnées par une requête est non significatif: demander les voitures de marque Toyota fabriquées entre 1995 et 2000 produira une liste. La même requête exécutée ultérieurement pourrait produire la liste des mêmes voitures mais affichées dans un ordre différent. Ce sera le cas par exemple, si la structure de la table a été modifiée (ajout de nouvelles contraintes, ...).

Il est cependant possible de spécifier dans une requête l'ordre dans lequel on veut voir s'afficher les lignes qui satisfont la requête (cfr la clause **ORDER BY** en SQL).

Notons que le nombre de lignes est souvent beaucoup plus élevé que le nombre de colonnes. Les **lignes** sont *dynamiques*, dans le sens où de nouvelles lignes peuvent être insérées, et des lignes existantes modifiées ou supprimées. Les **colonnes**, quant à elles, sont plus *statiques*: le nombre de colonnes est relativement fixe. Bien que cela soit possible en relationnel, on ne rajoute de nouvelles colonnes ou on ne modifie la définition de colonnes existantes que rarement.

3.2.2. Ordre des colonnes

Il en va pour les colonnes comme pour les lignes: aucun ordre logique n'existe entre les colonnes.

L'ordre d'affichage des colonnes en résultat à une *requête ne dépend pas de l'ordre de création des colonnes dans la table*, mais de l'ordre des colonnes tel qu'il est spécifié dans la requête elle-même.

Par exemple, les requêtes

```
select Nom, Prenom, DateNaissance, Statut, Telephone from Personne (1)
et   select Prenom, Nom, Telephone, Statut, DateNaissance from Personne (2)
ne produiront pas le même affichage.
```

Affichage de la requête (1):

Nom	Prenom	DateNaissance	Statut	Telephone
Dupond	Albert	12/12/68	marié	081/21.33.12
Patrice	Pol	01/08/58	célibataire	02/732.25.26
Perssin	Claude	20/02/56	marié	071/81.12.40
...

Affichage de la requête (2):

Prenom	Nom	Telephone	Statut	DateNaissance
Albert	Dupond	081/21.33.12	marié	12/12/68
Pol	Patrice	02/732.25.26	célibataire	01/08/58
Claude	Perssin	071/81.12.40	marié	20/02/56
...

3.3. Identifiant ou clé primaire

La **clé primaire** d'une table est utilisée pour *identifier* de façon univoque chaque ligne de la table. A chaque valeur de la clé primaire correspond *une et une* seule ligne, et à une ligne de la table correspond *une et une* seule valeur de clé primaire.

Une clé primaire (**PRIMARY KEY** en SQL) est composée de **un ou plusieurs attributs**, càd de une ou plusieurs colonne(s).

Le langage SQL n'impose pas que toutes les tables aient un identifiant. Les tables pour lesquelles aucune clé primaire n'est déclarée peuvent contenir alors plusieurs lignes tout à fait identiques.

Rappelons qu'une table n'étant pas ordonnée, il est impossible de demander l'accès à la première ligne, à la 5ème ligne, etc. *On utilisera donc la clé primaire pour sélectionner une ligne spécifique.* Si le numéro d'emprunteur est la clé primaire (identifiant) de la table emprunteur, on pourra, par exemple, accéder à la ligne concernant l'emprunteur ayant pour identifiant le numéro 310.

Un bon identifiant est un **identifiant invariant**. La combinaison des colonnes constituant la clé primaire aura donc une **valeur fixe pendant toute la durée de vie d'une même ligne**.

Par convention, le nom de la ou des colonnes constituant la clé primaire sera **souligné** dans la table.

Exemple:

<u>Nom</u>	<u>Prenom</u>	DateNaissance	Statut	Telephone
Dupond	Albert	12/12/68	marié	081/21.33.12
Patrice	Pol	01/08/58	célibataire	02/732.25.26
Dupond	Claude	20/02/56	marié	071/81.12.40
...

☛ Puisque le rôle d'une clé primaire est d'identifier toute ligne de la table, **aucune clé primaire ne peut prendre la valeur NULL**. En effet, si plusieurs lignes de la table avaient la valeur null pour clé primaire, aucune de ces lignes ne pourrait plus être identifiée de façon univoque par la clé primaire.

Si une même table possède **plusieurs identifiants**, un seul sera choisi pour jouer le rôle de clé primaire. En effet, en langage SQL, une seule combinaison de colonnes peut être déclarée PRIMARY KEY. D'autres identifiants peuvent cependant être spécifiés; ils seront déclarés **clés candidates ou alternatives** (clause UNIQUE en SQL).

3.4. Traduction d'un schéma conceptuel en relationnel

3.4.1. Représentation des types d'entité et des attributs

Un **type d'entité** est représenté en base de données relationnelle par une **table**.

Un **attribut** est représenté par une **colonne**. Une colonne peut être **facultative**. Les attributs facultatifs sont donc représentés par des colonnes facultatives.

☛ Il n'y a **qu'une seule classe d'attribut** disponible dans les bases de données relationnelles.

Tous les attributs sont **atomiques** et **monovalués**.

Représentation des attributs décomposables

Un attribut décomposable doit donc être “éclaté” en plusieurs attributs. L’attribut décomposable *adresse* devra par exemple être éclaté en 5 attributs.

Exemple :

Personne
<u>N°RegistreNational</u> Nom NomEpouse [0..1] Adresse Rue N° NomRue Boîte [0..1] Ville CodePostal NomVille

Ce type d’entité *Personne* sera représenté par la table suivante:

Personne							
<u>NumRN</u>	Nom	NomEp [0..1]	Num	NomRue	Boite [0..1]	CodePostal	NomVille

Les colonnes *NomJF* et *Boîte* seront déclarées facultatives à la création de la table.

Représentation des attributs multivalués

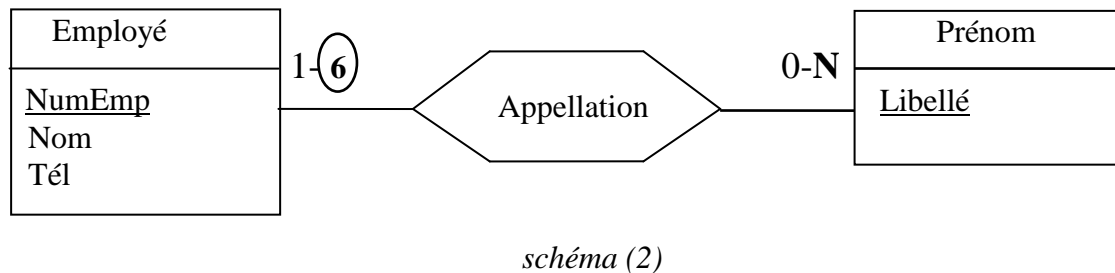
Les attributs multivalués ne peuvent pas être représentés tels quels en relationnel. Une transformation du schéma conceptuel est nécessaire. L’attribut multivalué est extrait du type d’entité et fait l’objet d’un second type d’entité relié au premier par un type d’association.

Exemple :

Employe
<u>NumEmp</u> Nom Prenoms [1..6] Tel

schéma (1)

L'attribut *Prénoms*[1..6] est extrait du type d'entité *Employé* et fait l'objet d'un second type d'entité *Prénom*. Ce dernier est relié au type d'entité *Employé* par un nouveau type d'association. Le type d'entité *Employé* (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 un ou plusieurs prénom(s), et un prénom peut être porté par un ou plusieurs employé(s).

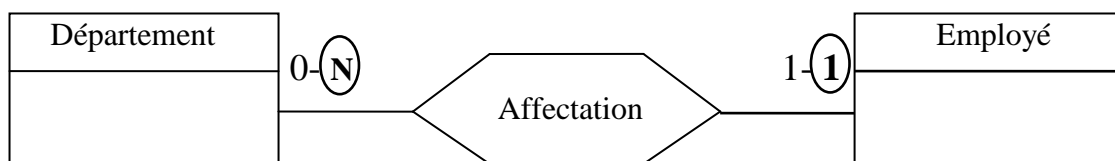


3.4.2. Représentation des types d'association

A. Représentation des types d'association 1 à N

La notion de **clé étrangère** permet de représenter une relation de type 1 à N entre deux tables.

Soit le T.A. *Affectation* (de type 1 à N) entre les T.E. *Employé* et *Département*.



Les T.E. *Employé* et *Département* sont tous deux représentés par des tables en relationnel. Le T.A. *Affectation* est représenté en relationnel par une **clé étrangère**.

Une clé étrangère est une **colonne supplémentaire** que l'on ajoute à la table qui provient du T.E. qui joue le rôle dont la cardinalité maximale est de **1**. Cette nouvelle colonne fait référence à la table que l'on veut relier: elle jouera le rôle de **charnière** entre les deux tables. Chaque valeur de la colonne charnière doit référencer à coup sûr **une et une seule ligne** de la table qui a la cardinalité maximale de N.

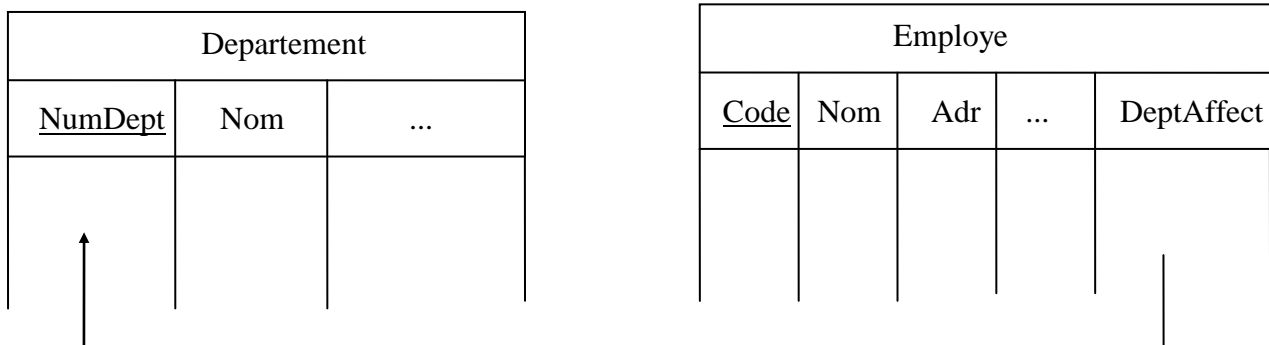
Par conséquent, *la colonne qui joue le rôle de clé étrangère doit référencer l'identifiant de la table que l'on veut relier.*

Dans l'exemple de la relation 1 à N *Affectation*, un employé ne peut être affecté qu'à un seul département. Un département, quant à lui, peut se voir affecter plusieurs employés.

Dans le T.A. *Affectation*, le type d'entité *Employé* a donc une cardinalité maximale de 1 et le type d'entité *Département* une cardinalité maximale de N.

Tout employé doit donc référencer l'identifiant du département auquel il est affecté!

↳ On **ajoute une colonne charnière (clé étrangère) dans la table *Employé***, qui reprendra pour chaque employé *l'identifiant* du département dans lequel il travaille.



La nouvelle colonne intitulée *DeptAffect* doit être déclarée **de même type que l'identifiant** de la table *Departement*, à savoir *NumDept*.

La clé étrangère est symbolisée sur le schéma logique (schéma des tables), par une **flèche** qui part de la colonne charnière vers l'identifiant de la table référencée.

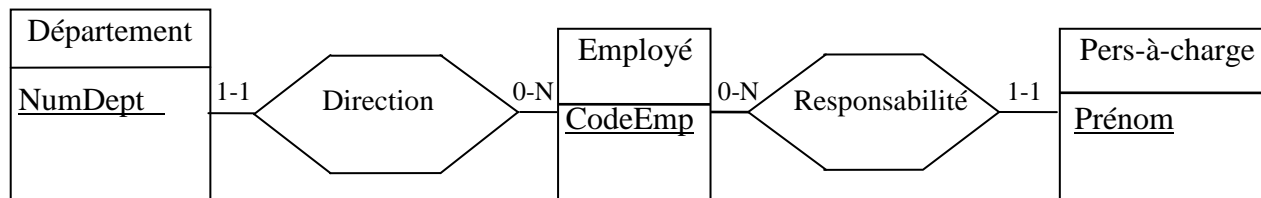
Le nom à donner à la colonne charnière est laissé à l'appréciation du concepteur de la base de données et ne doit obéir à aucune règle. La colonne (clé étrangère) intitulée dans l'exemple *DeptAffect*, aurait pu tout aussi bien être intitulée *Affectation*, *Département*, *NumDept* ou *LieuTravail*.

Notons que le principe des clés étrangères est un principe qui découle du bon sens. En effet, il est impossible de prévoir une colonne additionnelle dans la table *Département* en vue d'implémenter la relation 1 à N *Affectation*. A un département, il faudrait associer plusieurs occurrences d'employé. Plusieurs colonnes devraient par conséquent être rajoutées à la table *Département*, une colonne par employé travaillant dans le département! Ceci est ingérable: *combien de colonnes prévoir?*

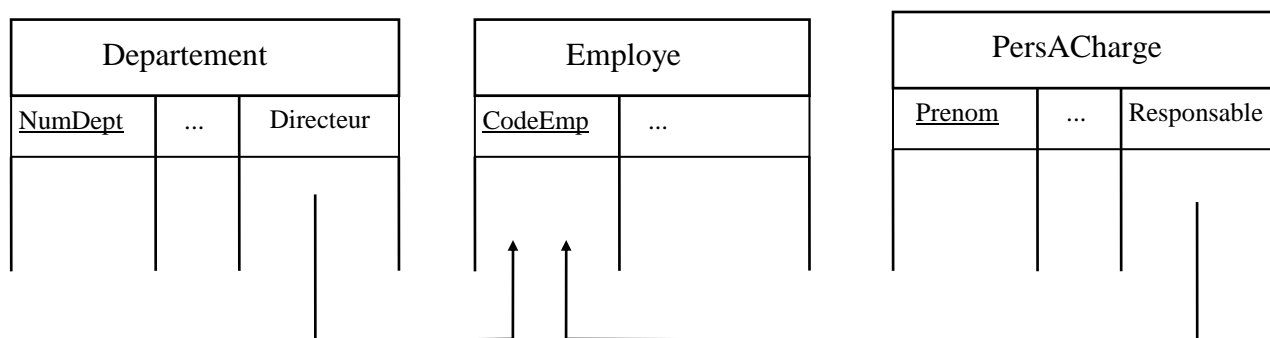
Exemples

Exemple 1:

Soit le schéma entités-associations suivant:

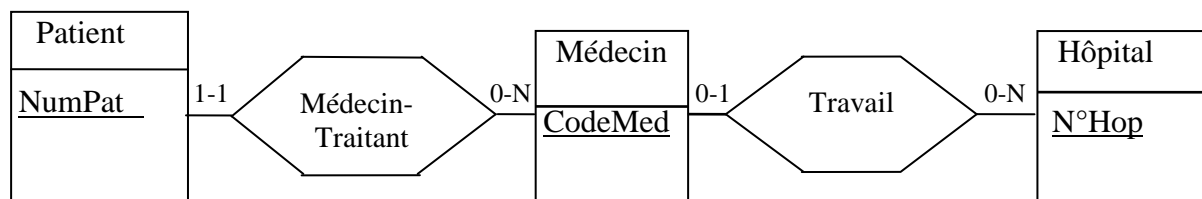


La traduction logique en base de données relationnelle du schéma conceptuel ci-dessus est la suivante:

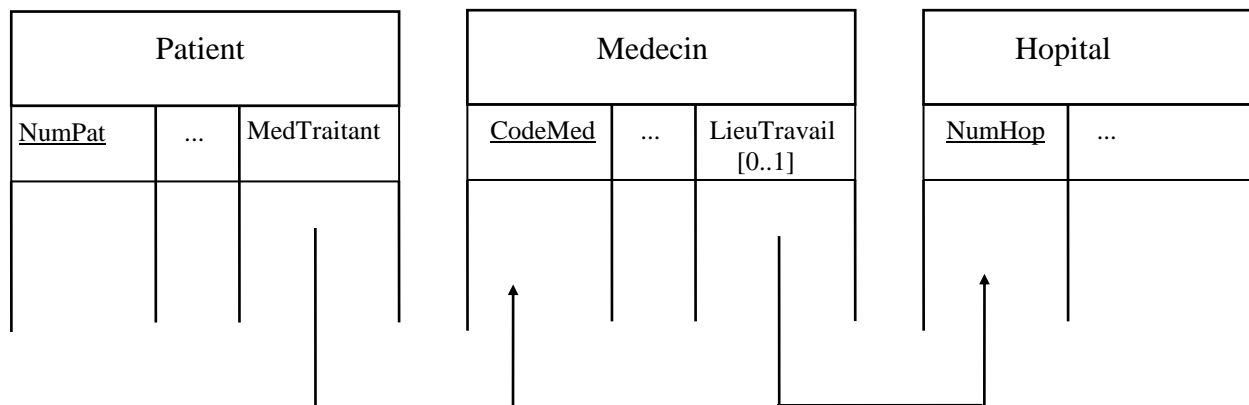


Exemple 2:

Soit le schéma entités-associations suivant:

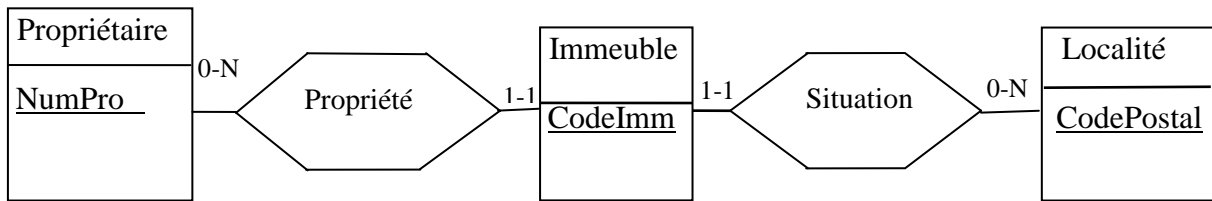


La traduction logique en base de données relationnelle du schéma conceptuel ci-dessus est la suivante:

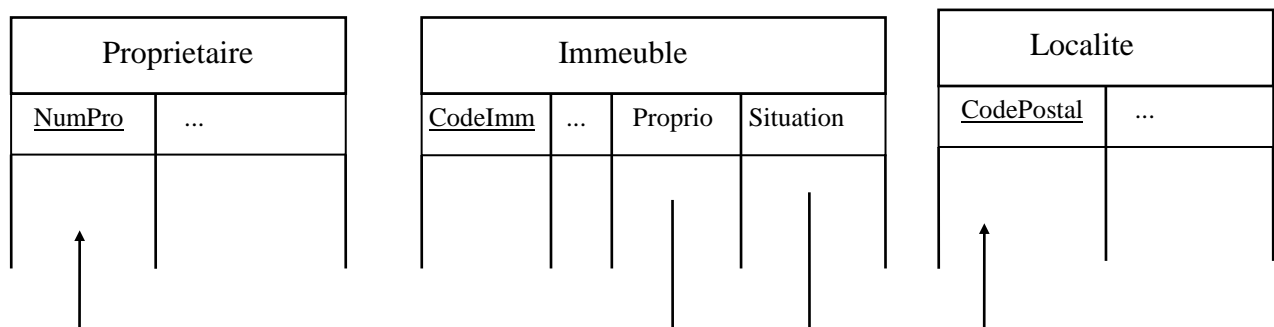


Exemple 3:

Soit le schéma entités-associations suivant:



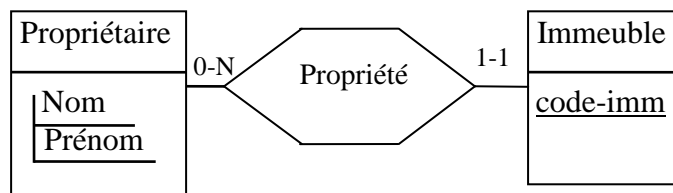
La traduction logique en base de données relationnelle du schéma conceptuel ci-dessus est la suivante:



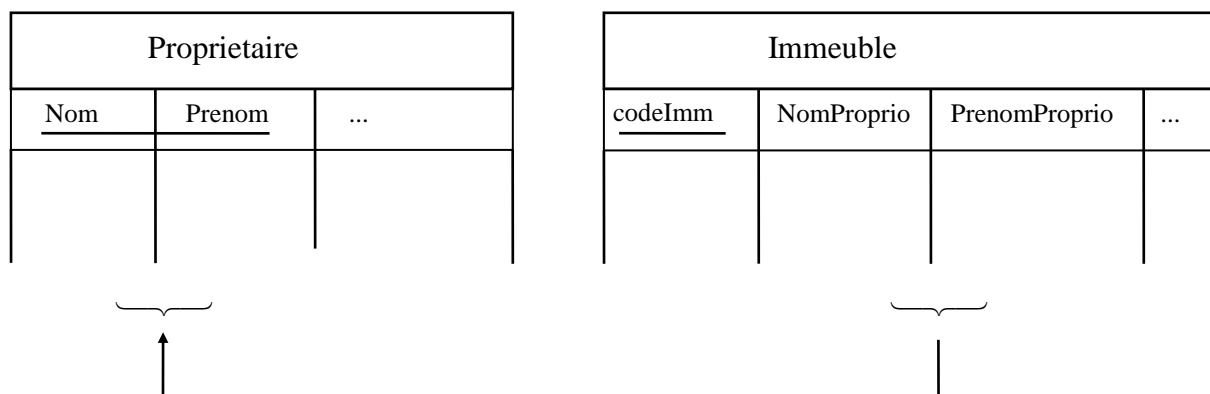
Notons que si l'identifiant de la table reliée est composé de plus d'une colonne, la clé étrangère sera composée d'autant de colonnes.

Exemple 4:

Soit le schéma entité-association suivant:



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



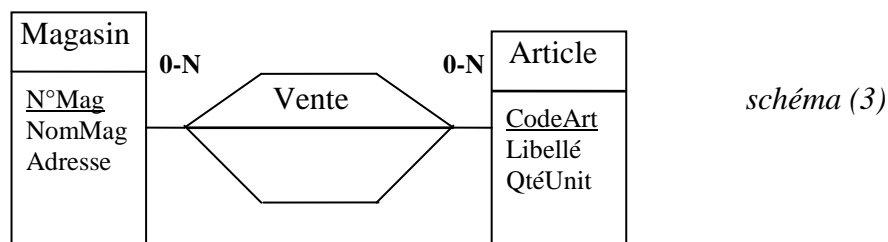
B. Représentation des types d'association N à N

Seuls les types d'association 1 à N sont directement traduisibles en relationnel : une relation 1 à N est représenté par une clé étrangère. Il n'en va pas de même avec les types d'association N à N. Il faut passer par une étape intermédiaire et effectuer une transformation du schéma entité-association : il faut **éclater tout type d'association N à N en deux types d'association 1 à N** qui eux sont représentés en relationnel par des clés étrangères.

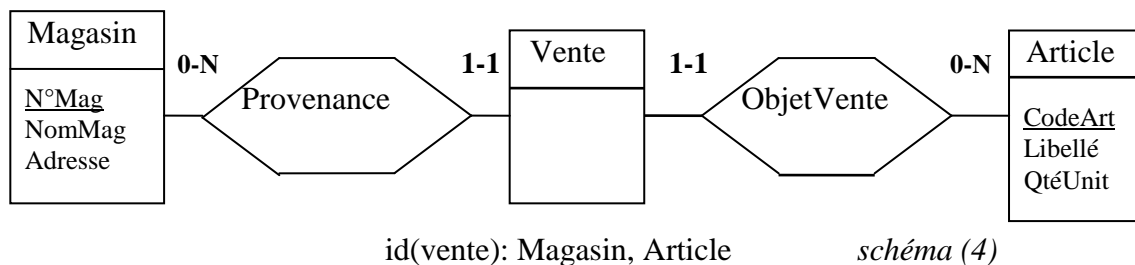
Un type d'association N à N est donc représenté par une **table additionnelle**. La traduction d'un type d'association N à N entre deux types d'entités aboutit donc à la création de trois tables, une table pour chaque type d'entité et une table pour le type d'association N à N.

Exemple 1

Soit le T.A. *Vente* entre les T.E. *Magasin* et *Article*

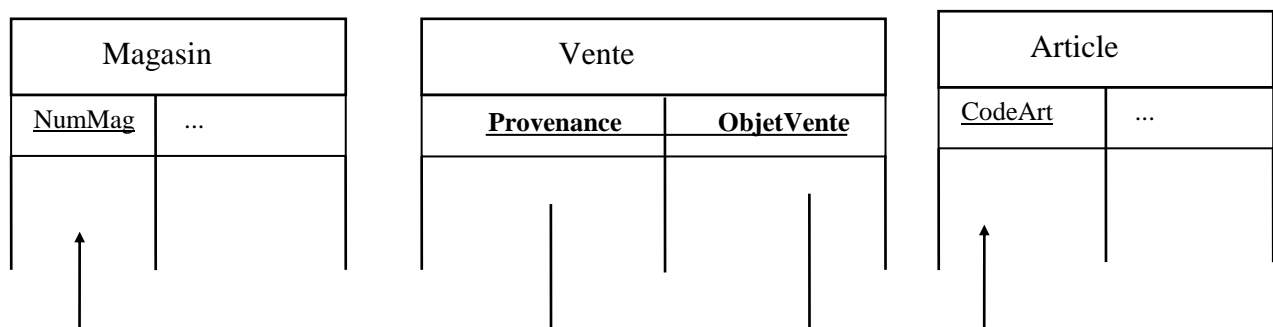


Le T.A. *Vente* peut être considéré comme un concept et non plus comme un lien. Une transformation de ce schéma conceptuel peut être proposée: les schémas conceptuels (3) et (4) sont **sémantiquement équivalents**.



Le T.A. N à N a été éclaté en deux T.A. 1 à N. Les trois T.E. *Magasin*, *Article* et *Vente* seront représentés respectivement par les trois tables *Magasin*, *Article* et *Vente*. Les deux T.A. *Propose* et *Concerne* seront représentés par deux clés étrangères au sein de la table *Vente*.

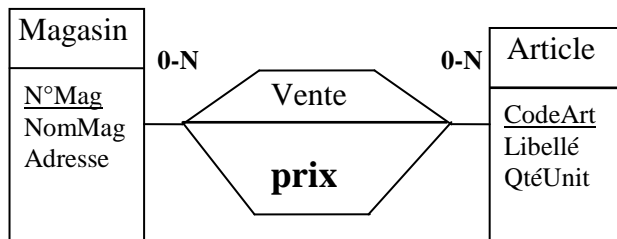
La table *Vente* ne sera en fait constituée que de deux colonnes, les deux clés étrangères.



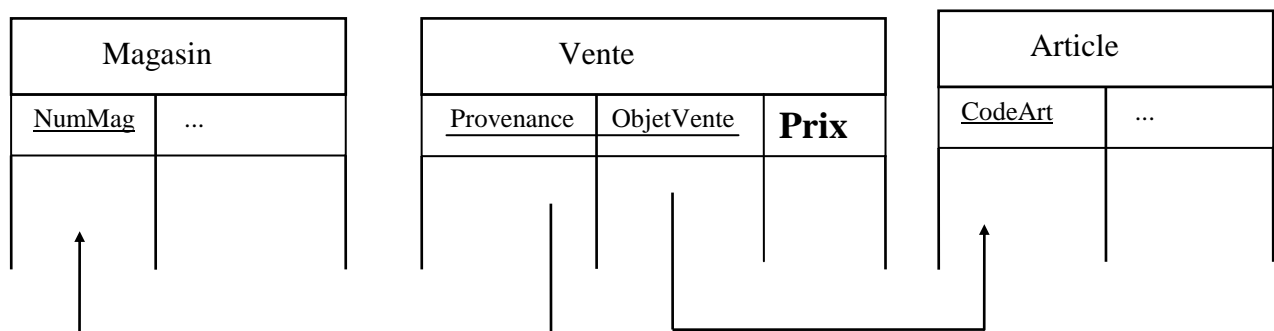
L'identifiant de la table Vente est composé des deux colonnes clés étrangères.

Si le T.E. *Vente* contient des **attributs** propres sur le schéma conceptuel, ceux-ci constitueront autant de colonnes supplémentaires dans la table *Vente*.

Exemple 2

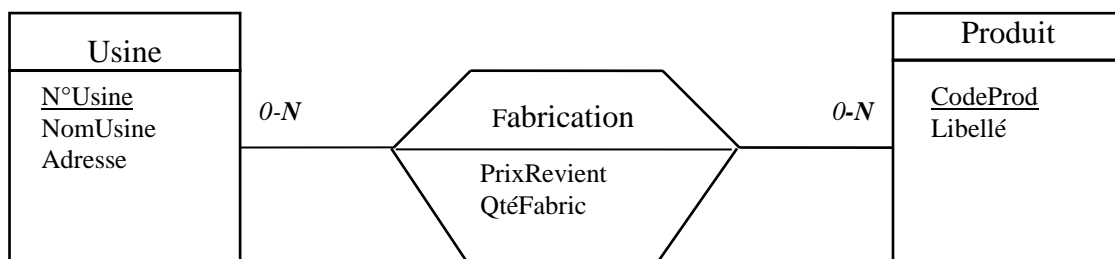


La traduction logique en base de données relationnelle du schéma conceptuel ci-dessus est la suivante

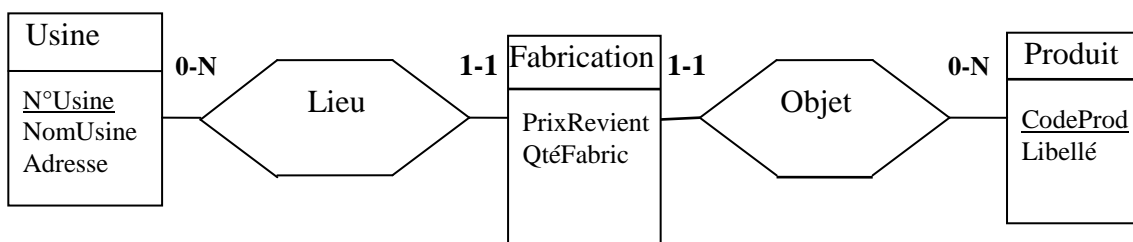


Exemple 3

Soit le T.A. *Fabrication* entre les T.E. *Usine* et *Produit*

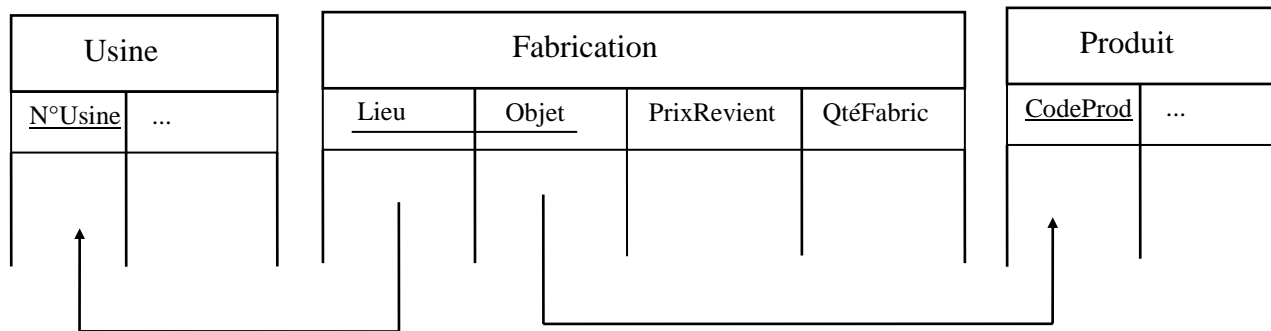


Ce schéma conceptuel doit d'abord être transformé.



id(Fabrication): Usine, Produit

Le schéma conceptuel ainsi transformé est ensuite traduit en tables.

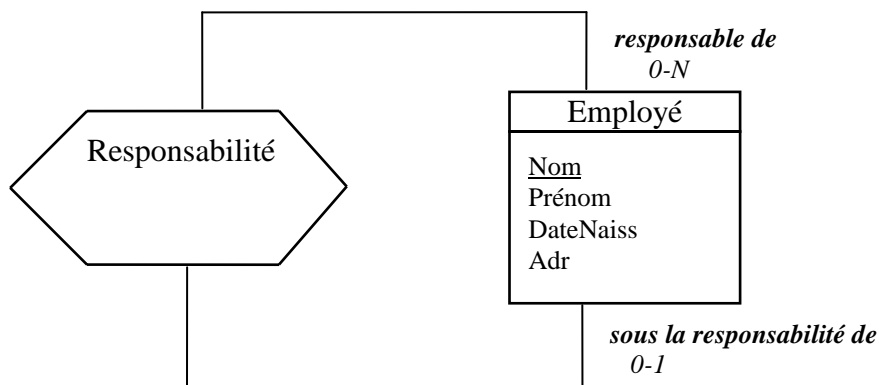


C. Représentation des types d'association récursifs

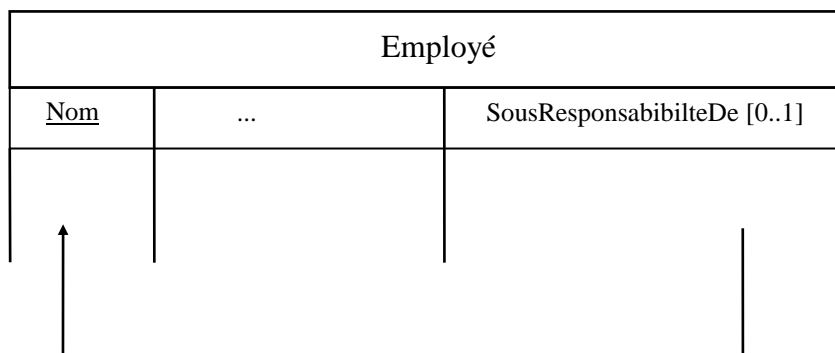
Une association **récursive** est un lien entre **deux occurrences du même type d'entité**. Les deux rôles d'un T.A. récursif sont donc définis sur le même T.E.

Exemple 1

La hiérarchie dans une entreprise peut être représentée via un T.A. récursif responsabilité.

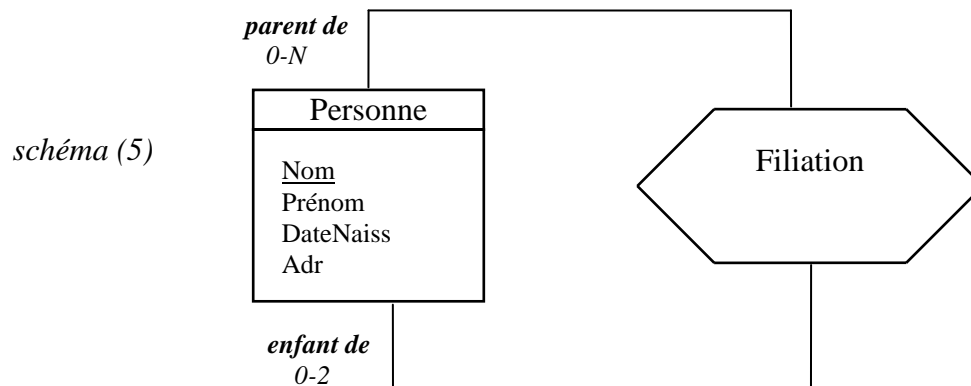


Puisque le T.A. *Responsabilité* traduit une relation 1 à N, il suffit de rajouter une clé étrangère dans le T.E. *Employé*, représentant le rôle ayant la cardinalité maximum = 1, à savoir, le rôle *sous la responsabilité de*.

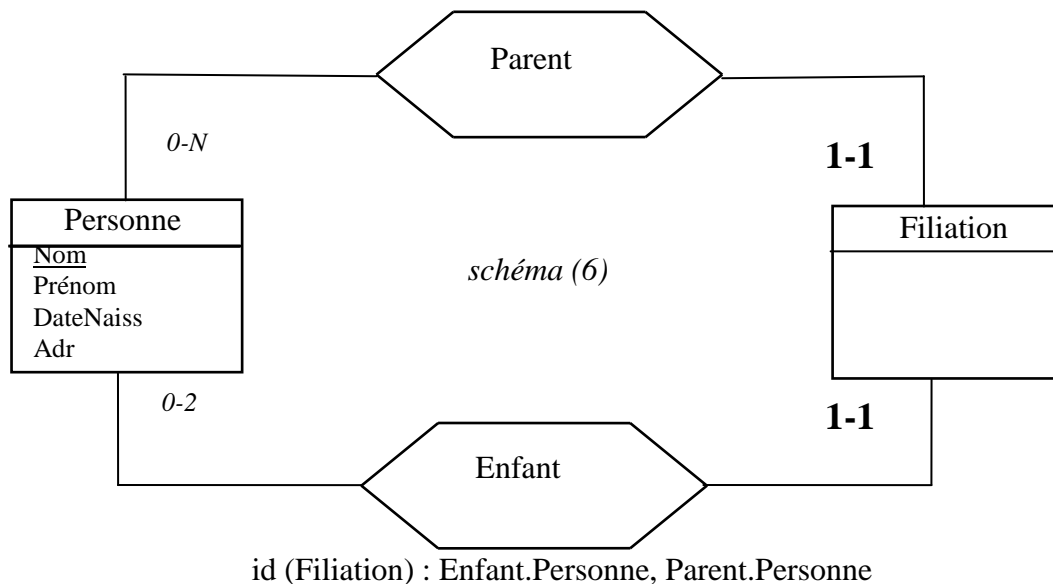


Exemple 2

Un autre exemple de T.A. récursif est la relation de *filiation*. Une occurrence de filiation relie deux occurrences distinctes du T.E. *Personne*: une occurrence qui joue le rôle de l'enfant, et une occurrence qui joue le rôle du parent.



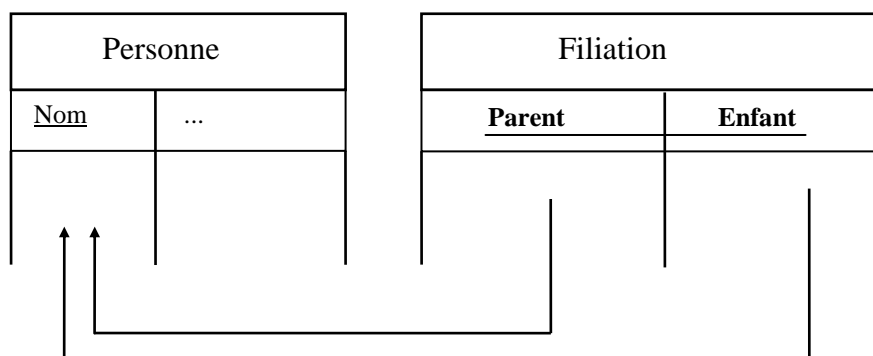
Ce schéma doit d'abord être transformé. La notion de filiation peut être vue comme concept plutôt que comme un lien.



Les schémas (5) et (6) sont sémantiquement équivalents.

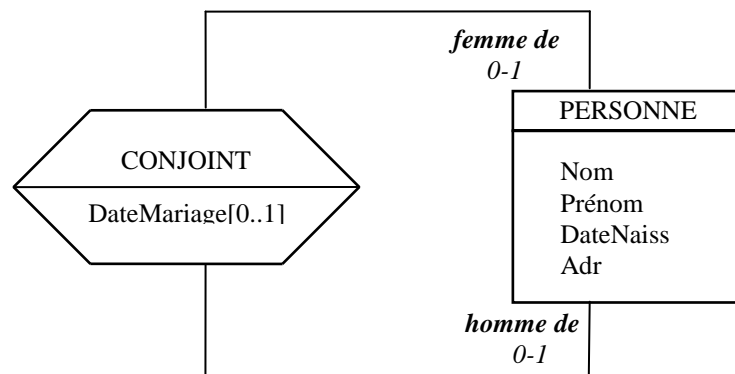
Le T.E. *Personne* sera représenté en relationnel par une table, la table *Personne*.

Le T.E. *Filiation* ainsi obtenu sera représenté par une nouvelle table ne contenant **que deux colonnes**, à savoir, les **deux clés étrangères** vers la table *Personne*.

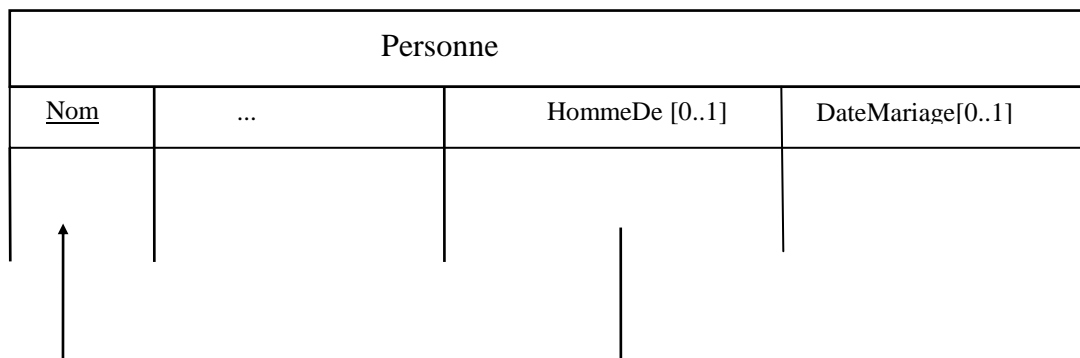


Exemple 3

Les liens entre conjoints peuvent être représentés via un T.A. récursif.



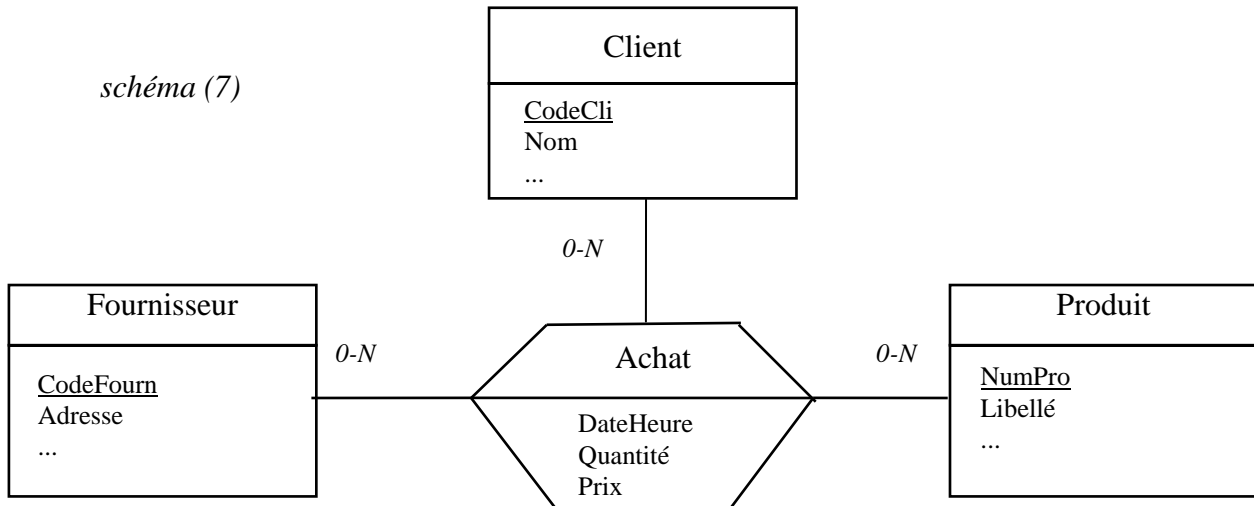
Le T.A. *Conjoint* traduit une relation 1 à 1. Il suffit de rajouter une clé étrangère dans le T.E. *Personne*, représentant un des deux rôles au choix.



Attention, Prévoir deux clés étrangères, une pour le rôle "femme de" et une pour le rôle "homme de" induirait une redondance.

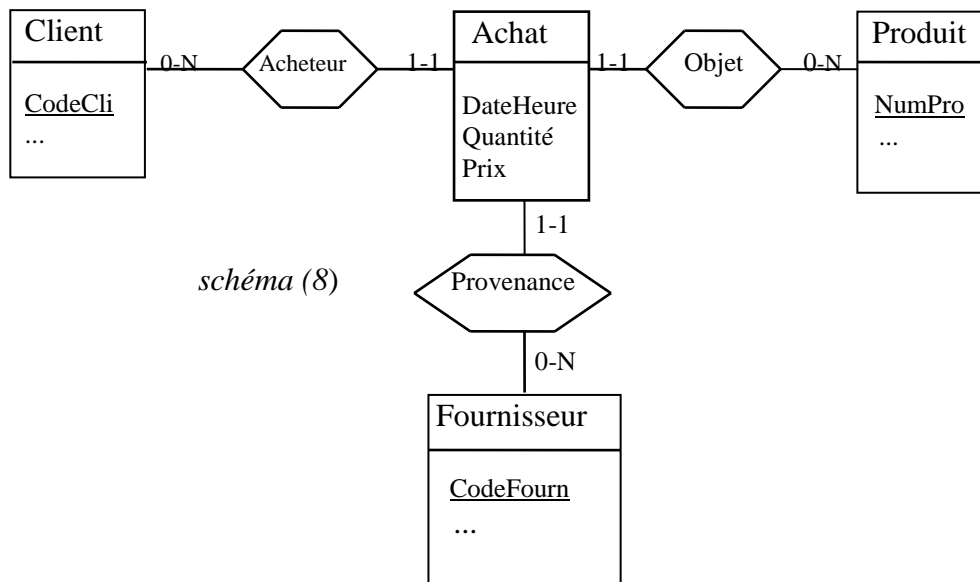
D. Représentation des types d'association de degré supérieur à 2

Soit la notion d'achat qui est un **lien**, une relation **entre des clients, des produits et des fournisseurs**. Il s'agit d'un T.A. *ternaire*, càd de *degré 3*.



id(Achat)= Client, Produit, Fournisseur, DateHeure

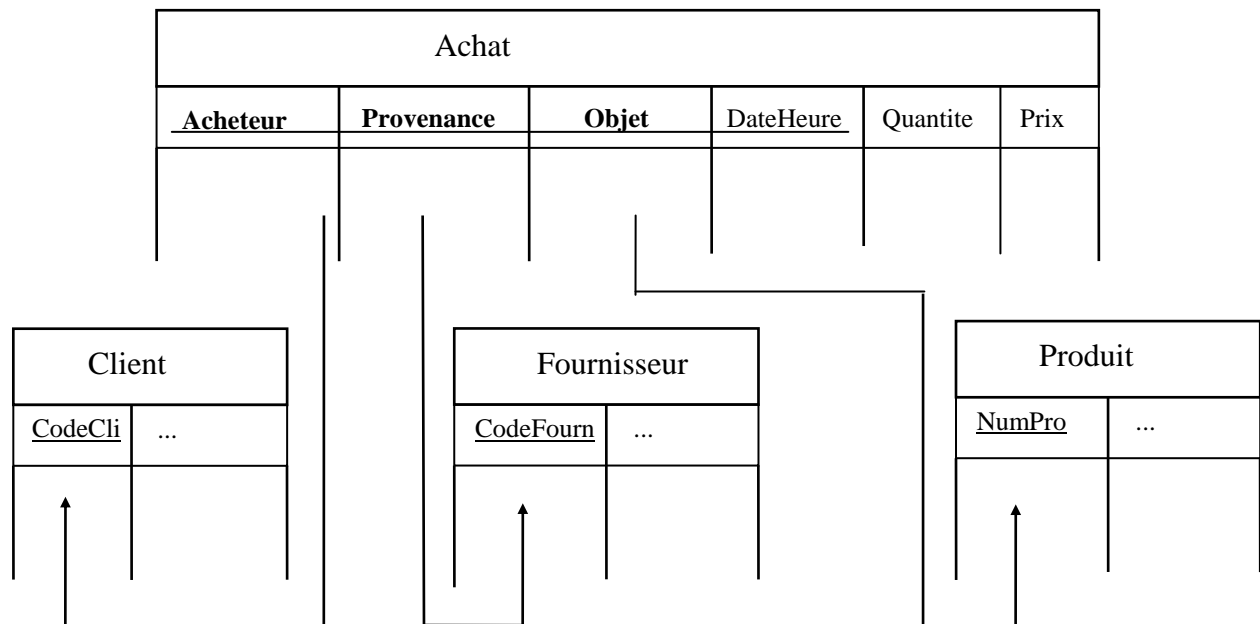
Une transformation de schéma est requise: le T.A. ternaire *Achat* est considéré comme un concept. Il fait donc l'objet d'un type d'entité. Trois nouveaux types d'association sont nécessaires: les T.A. *Effectue*, *Concerne* et *Fournit*.



id(Achat)= Client, Produit, Fournisseur, DateHeure

Les schémas (7) et (8) sont sémantiquement équivalents.

Les 4 T.E. *Client*, *Produit*, *Fournisseur* et *Achat* sont représentés en relationnel respectivement par les tables *Client*, *Produit*, *Fournisseur* et *Achat*. La table *Achat* contiendra, entre autres colonnes, trois clés étrangères vers les trois tables *Client*, *Produit* et *Fournisseur*.



3.5. Conclusion

La plupart des bases de données actuelles sont de type relationnel.

La maîtrise des bases de données relationnelles est donc vitale pour un futur informaticien.

C'est pourquoi, l'apprentissage du langage SQL est prévu au second quadrimestre de la deuxième année. Au cours de ce laboratoire seront étudiés le langage relationnel de définition de données (RDDL) et le langage relationnel de manipulation de données (RDML).

Table des matières

Chapitre 1. Qu'est-ce qu'une base de données?

1.1. Historique	2
1.1.1 Fichiers séquentiels	2
1.1.2. Evolution des supports	2
1.1.3. Evolution des systèmes d'exploitation	3
1.1.4. Evolution des méthodes d'accès aux données et de leur organisation	3
1.1.5. Systèmes documentaires	4
1.1.6. Systèmes de gestion de bases de données	4
1.2. Inconvénients des systèmes de fichiers	4
1.3. Liens entre les données	5
1.4. Caractéristiques d'une base de données	9
1.5. Difficultés d'identifier les concepts et les liens	11
1.6. Systèmes de gestion de bases de données (SGBD)	14
1.6.1. Programmeur	14
1.6.2. Utilisateur	15
1.6.3. Administrateur	15
1.6.4. S.G.B.D.	16

Chapitre 2. Le modèle entités-associations 18

2.1. Introduction	18
2.2. Entités - Types d'entité (T.E.)	20
2.3. Associations - Types d'association (T.A.)	22
2.3.1. Qu'est-ce qu'un type d'association ?	22
2.3.2. Cardinalités maximales	27
A. Relation de type 1 à 1	27
B. Relation de type 1 à N	28
C. Relation de type N à N	28
2.3.3. Cardinalités minimales	29
2.3.4. Exemples	30

2.4. Construction d'un schéma conceptuel	33
2.4.1. Décomposition de l'énoncé	33
2.4.2. Représentation d'une proposition	35
A. Concepts	35
B. Liens	36
<i>B.1. Entre deux T.E.</i>	36
<i>B.2. Entre un T.E. et un attribut</i>	36
<i>B.3. Entre deux attributs</i>	37
2.5. Intérêt de prévoir un type d'association plutôt qu'un attribut	38
2.6. Attributs	39
2.6.1. Identifiant	39
2.6.2. Mono-valué ou multivalué	39
2.6.3. Obligatoire ou facultatif	40
2.6.4. Atomique ou décomposable	41
2.6.5. Attribut facultatif versus attribut booléen	42
2.7. Non redondance dans le schéma entités-associations	46
2.8. Pertinence des propositions	49
2.8.1. Contradiction des propositions	49
2.8.2. « Bruit » dans les propositions	49
2.8.3. Autres recommandations	50
2.9. T.A. particulier	52
2.9.1 . T.A. de degré supérieur à 2	52
2.9.2 . T.A. récursive ou cyclique	56
2.9.3 . T.A. avec attributs	58
2.10. Contraintes additionnelles	61
2.10.1. Identifiant composé d'attributs	61
2.10.2. Identifiant hybride	62
2.10.3. Identifiant de type d'association	64
2.11. Transformation de schéma	68
2.11.1. Transformation d'un type d'association en un type d'entité	68
2.11.2. Transformation d'attribut en type d'entité	72

Chapitre 3. Bases de données relationnelles	75
3.1. Structures de base: les tables	75
3.1.1. Table	75
3.1.2. Ligne	76
3.1.3. Colonne	76
3.1.4. Valeur	77
3.2. Notion d'ordre	78
3.2.1. Ordre des lignes	78
3.2.2. Ordre des colonnes	78
3.3. Identifiant ou clé primaire	79
3.4. Traduction d'un schéma conceptuel en relationnel	80
3.4.1. Représentation des types d'entité et des attributs	80
<i>Représentation des attributs décomposables</i>	81
<i>Représentation des attributs multivalués</i>	81
3.4.2. Représentation des types d'association	82
<i>A. Représentation des types d'association 1 à N</i>	82
<i>B. Représentation des types d'association N à N</i>	86
<i>C. Représentation des types d'association récursifs</i>	88
<i>D. Représentation des types d'association de degré supérieur à 2</i>	91
3.5. Conclusion	92