Gestion des bases de données (1ère partie)

Ce polycopié rédigé par F. Horn est basé sur deux polycopiés précédents réalisés par A. Lemay et D. Gonzalez.

I. Objectifs du cours

- savoir construire une base de données sur micro-ordinateur répondant à des besoins individuels, d'un service ou d'une petite structure.
- Comprendre le fonctionnement des « grosses » bases de données d'une entreprise ou d'une administration, pour faciliter l'utilisation et le dialogue avec les informaticiens responsables de ces bases.
- Présentation du programme et difficultés spécifiques des SGBD (réflexion préalable aux manipulations plus importante que pour l'utilisation des autres outils bureautiques).

II. Généralités sur les bases de données

Alors qu'au début de leur histoire les ordinateurs servaient essentiellement à calculer, leur utilisation principale de nos jours est la gestion d'informations. On les retrouve dans tous les secteurs d'activité.

Au départ, les informations étaient stockées sous forme de fichiers créés au fur et à mesure des besoins et au cours du développement de nouvelles applications. La création non maîtrisée de différents fichiers a rapidement posé des problèmes :

- Redondance : les mêmes données finissent par se retrouver dans plusieurs fichiers.
- Manque de cohérence : il est très difficile de répercuter les mises à jour sur l'ensemble des fichiers concernés
- Manque de structuration : l'absence d'une vision globale fait que les données sont trop spécifiques et ne permettent pas leur réutilisation pour de nouveaux traitements.

D'où l'idée de remplacer ces différents fichiers par une seule base de données. Une base de données est définie comme étant « **un ensemble de données organisé en vue de son utilisation par des programmes correspondant à des applications distinctes, et de manière à faciliter l'évolution indépendante des données et des programmes** » (Journal Officiel, 17/01/1982) :

- Par rapport à des fichiers disparates, une base de données unifie la structuration et la mémorisation des informations grâce à un **modèle** (ou schéma) **unique** et **cohérent** des données.
- Ce modèle unique de données ne doit pas être lié à une application spécifique qui en figerait la structure et doit être suffisamment général pour s'adapter à toutes les situa-

tions particulières (d'où la nécessité dans la conception d'une base de données d'une analyse globale et prospective des besoins).

- Dans une base de données, les données sont décrites indépendamment des programmes (ou traitements) qui les utilisent. Il doit être possible de modifier les programmes appliqués sans avoir à redéfinir les données.

Un logiciel permettant d'utiliser ces données est un *système de gestion de base de données* (SGBD). Il permet de décrire, mémoriser, interroger, modifier, traiter, maintenir les données constituant une base. Il permet de définir des règles précises permettant de maintenir la cohérence (l'intégrité, la consistance) des données d'une base en veillant à ce que des données identiques ne soient pas dupliquées. Il permet également d'appliquer des contraintes sur les données et d'assurer des fonctions de confidentialité, de sécurité et de partage des données pour des accès concurrents.

Différents logiciels existent permettant cette opération. Nous allons utiliser ici le logiciel **Access** comme SGBD. Ce logiciel permet une conception aisée de bases de données de "petite" taille avec un nombre restreint d'utilisateurs. Il est à noter que plusieurs autres SGBD plus performants (mais également plus complexes) existent par ailleurs. On peut citer notamment Oracle, SQL Server, Paradox, MySQL, PostgreSQL parmi beaucoup d'autres.

Il existe trois types de modèles de bases de données, les modèles hiérarchiques, les modèles en réseaux et les modèles rationnels. Le modèle hiérarchique est le plus ancien ; dans ce modèle, l'organisation des données repose sur une structure arborescente (on peut faire l'analogie avec la gestion des fichiers sur un ordinateur) : chaque information n'a qu'un seul supérieur hiérarchique et n'est accessible qu'à partir d'un point unique (la racine). Le deuxième modèle est le modèle en réseaux (modèle CODASYL). Chaque information peut être associée à plusieurs autres (plusieurs « supérieurs hiérarchiques ») et servir de point d'entrée (il n'y a plus d'informations privilégiées), les relations entre les données étant stockées dans la base avec les données (on peut faire l'analogie avec les liens hypermédias). Le dernier modèle est le modèle relationnel sur lequel sont basés la plupart des SGBD actuels (dont **Access**) et qui est le seul que nous étudierons. Dans ce modèle, les informations sont stockées dans des tables qui sont reliées entre elles par des relations. L'interrogation de la base de données se fait à l'aide de *requêtes*, ces requêtes étant écrites à l'aide d'un langage commun à la plupart des SGBD : le SQL (Structured Query Language). **Access** a comme avantage par rapport à la plupart de ses concurrents de permettre une écriture en mode graphique des tables, de leurs relations et de la plupart des requêtes. De plus, il intègre un système de création d'applications claires et simples pour chaque base de donnée. Pour concevoir une base de données relationnelle, il existe différentes méthodes la plus utilisée (en France) étant la méthode **Merise**.

III. Méthode Merise

1. Principes généraux

La méthode Merise a été créée en France en 1978 sous l'impulsion du ministère de l'industrie, par un groupement de six sociétés de services et un centre de recherche informatique. Cette méthode utilise le système dit *d'entités-relations*. Il s'agit d'un outil et d'une technique d'analyse permettant de construire des schémas théoriques de raisonnement sur des applications tournant avec des bases de données dites *relationnelles* (comme celles d'**Access**).

A noter que nous ne présenterons ici qu'une partie de la méthode Merise, puisque la méthode Merise générale traite de l'intégralité de la conception de la base de données : elle ne s'intéresse pas uniquement à la partie correspondant au stockage des données, mais également à leur traitement.

La méthode Merise considère quatre phases dans la création d'une base de données :

1. **La phase d'analyse** : cette phase, qui ne sera pas étudié dans ce document, est une phase essentielle qui consiste à
   - étudier l'existant : y a-t-il un système qui gère déjà tout ou partie de l'information, qu'il s'agisse d'un logiciel ou d'un ensemble de documents papiers ? Comment ces informations sont elles stockées ? Quelles sont les informations stockées ? Que manque-t-il ? Qu'est ce qui convient ou ne convient pas aux utilisateurs ?
   - interroger les futurs utilisateurs : qu'attendent-ils du futur SGBD ? Quelles sont les opérations qu'ils désirent automatiser ?
   - recueillir les informations existantes, étudier les divers liens qui peuvent exister entre ces informations, mettre en évidence les règles de gestion employées…
2. **La phase conceptuelle :** elle consiste à représenter l'organisation des données de manière générale. Elle aboutit à la création du *modèle conceptuel des données* (MCD) dans lequel les données sont représentées sous forme d'entités liées entre elles par des relations.
3. **La phase logique ou organisationnelle :** dans cette phase, la base de données est représentée sous une forme *logique* plus proche de sa représentation réelle au sein du SGBD : les informations sont représentées uniquement sous forme de tables au sein d'un *modèle logique des données* (MLD).
4. **La phase physique ou opérationnelle :** elle consiste à construire réellement la base de données au sein du SGBD (ici **Access**). Cette partie ne sera pas décrite dans cette section, mais dans les suivantes.

A retenir : les quatre phases de la méthode Merise :

1. analyse (étude de l'existant et enquête)
2. conceptuel (création du MCD)
3. logique (création du MLD)
4. physique (conception de la base de données dans Access)

2. Modèle Conceptuel de Données

Après la phase d'analyse, nous pouvons commencer à représenter les informations sous forme conceptuelle dans un modèle de données. Un modèle de données est un formalisme permettant de décrire les données intervenant dans un système d'informations et les liens existant entre ces informations de façon claire, simple, complète et non ambiguë. Le *Modèle Conceptuel de Données* (MCD) que nous allons construire contient deux éléments principaux : les *entités* et les *relations*.

Une entité (ou objet) est un élément du problème. La notion d'entité est réfractaire à toute définition formelle. Une entité est une chose (concrète ou abstraite) qui existe et est distinguable des autres entités. Elle est définie par un ensemble de propriétés. Chacune des propriétés est l'un des éléments qui caractérise l'entité. Il faut distinguer une *entité* et une *occurrence d'entité* (ou *instance*). Une entité correspond au type général d'une donnée (ex : le type "employé") alors qu'une occurrence d'une entité est un représentant particulier de cette entité (l'employé "Jean Martin"). Une occurrence d'une entité est un élément particulier correspondant à l'entité et associé à un élément du réel.

Une relation est un lien *possible* qui relie deux entités. Elle correspond à une association perçue dans le réel entre deux entités. Par exemple, si un employé peut être affecté à un entrepôt, il y aura une relation "affectation" entre l'entité entrepôt et l'entité "employé". Cela ne signifie pas nécessairement qu'il y aura affectation pour chacun des employé, juste qu'il est possible qu'un employé soit affecté à un entrepôt. Une relation peut éventuellement être reliée à plus de deux entités et peut avoir certaines propriétés.

Après avoir fait une analyse aussi complète que possible du problème à informatiser, la construction du MCD se fait en quatre étapes :

1. **repérage des entités**,
2. **construction des entités, choix des propriétés**,
3. **construction des relations**,
4. **choix des cardinalités.**

a. Repérage des entités

Une entité est un composant du problème : une personne, une facture, un livre… C'est la représentation d'un objet matériel ou immatériel pourvu d'une existence propre et conforme aux choix de gestion de l'organisation. Dans la description de la situation à informatiser les entités correspondent souvent aux noms. Comme dit plus haut, ce que l'on considère comme entité est un type général (ex : l'entité *personne* représente toutes les personnes) à ne pas confondre avec une occurrence d'entité (Jean Martin étant une personne, on le considère comme une occurrence de l'entité *personne*). Une entité doit avoir une existence indépendamment de tout autre entité.

Exemple : On considère le problème (très simplifié) suivant :

Une société qui vend des produits veut informatiser la gestion des commandes de ses clients. Chaque commande d'un client peut comporter plusieurs produits différents.

Dans cet exercice, les entités sont :

- l'entité "produits" : un produit commercialisé par la société
- l'entité "clients" : une personne qui achète des produits à la société
- l'entité "commandes" : une liste de produits commandés par un client à la société

b. Construction des entités

L'étape suivante correspond à la construction des entités. On commence par donner un nom à chacune des entités. Il faut ensuite rechercher les propriétés (ou attributs) de ces entités. Une propriété est une donnée élémentaire que l'on perçoit sur une entité. Chacune des propriétés d'une entité prend une valeur parmi une variété de valeurs possibles (le domaine de l'attribut). Une propriété peut être obligatoire ou facultative. On devra garder à l'esprit les points suivants :

- toute propriété est élémentaire. Elle n'est pas la composition d'éventuelles propriétés plus petites : plutôt qu'une propriété unique *adresse*, il est préférable d'avoir des propriétés *rue, code postal, ville, pays*….
- une propriété ne doit pas être "instable" ou "calculable" : si une propriété peut être obtenue par calcul à partir d'autres éléments qui vont apparaître dans la base de donnée (notamment d'autres propriétés), on ne doit pas la considérer : il est inutile d'avoir une propriété *montant de la commande* si celui-ci peut être calculé à partir d'autres propriétés.
- toute entité doit posséder une propriété particulière appelée sa **clé** (ou **identifiant**). Une clé doit caractériser de manière unique chaque occurrence de l'entité. L'identifiant d'une entité est une propriété de l'entité telle qu'à chaque valeur de la propriété correspond **une et une seule** occurrence de l'entité. Par exemple, le nom de famille d'une personne ne peut pas être considéré comme une clé d'une entité "personne" puisque deux personnes peuvent avoir le même nom de famille. Le numéro de sécurité sociale est par contre tout à fait acceptable. Il vaut mieux éviter les identifiants trop longs (on préférera un code de quelques chiffres à un intitulé d'une vingtaine de lettres par exemples). Une « bonne » clé ne doit pas comprendre un sous-ensemble qui pourrait lui-même être une clé (notion de *minimalité*).
- si aucune des propriétés "naturelles" ne peut servir de clé, on en rajoute une artificiellement (par exemple "CodeProduit" ou "IdClient").
- Chaque propriété ne doit dépendre que d'une seule entité.

Une entité se représente ensuite graphiquement sous la forme d'une boîte dans laquelle on indique en titre le nom de l'entité suivi de toutes ses propriétés. On indique d'une manière particulière l'identifiant.

Entité
Identifiant propriété 1 propriété 2 ...

Exemple : Dans l'exemple de la gestion des commandes de la société, on peut construire les entités suivantes (les propriétés sont indiquées après le nom de l'entité, l'identifiant est en gras) :

- Clients : **IdClient**, nom, prénom, rue, code postal, ville, pays, tél, email….
- Produits : **CodeProduit**, libellé, prixHT, quantité en stock…
- Commandes : **NumCommande**, date, mode de paiement….

Remarque : il est également possible de transformer la propriété « ville » de l'entité Clients, en une entité Villes dont l'identifiant serait le code postal. On aurait dans ce cas quatre entités :

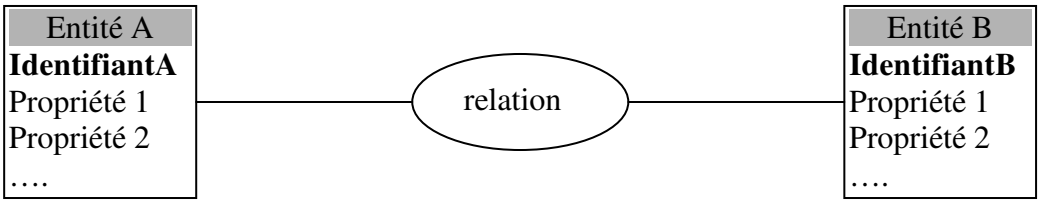
- Clients : **IdClient**, nom, prénom, rue, code postal, pays, tél, email….
- Produits : **CodeProduit**, libellé, prixHT, quantité en stock…
- Commandes : **NumCommande**, date, mode de paiement
- Villes : **CodePostal**, ville

Cette solution est (un peu) plus complexe à construire mais elle présente l'avantage de nécessiter moins de saisie et d'espace mémoire et de faciliter une éventuelle actualisation des données (ville qui change de nom….).

c. Construction des relations

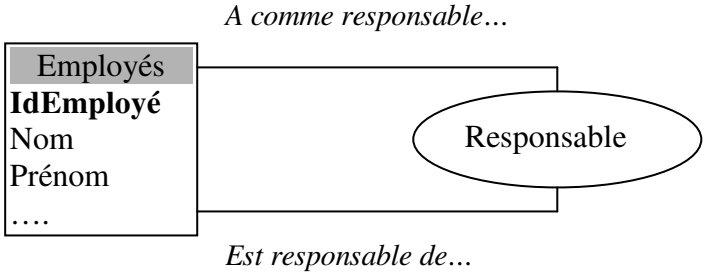
L'étape suivante consiste à énumérer toutes les relations *possibles* entre entités. Si une relation a une chance d'apparaître (et de nous intéresser), alors on doit la considérer dans le MCD. On parle également parfois d'*association.* Dans la description de la situation à informatiser les relations correspondent souvent aux verbes.

Une relation se représente de la manière suivante :



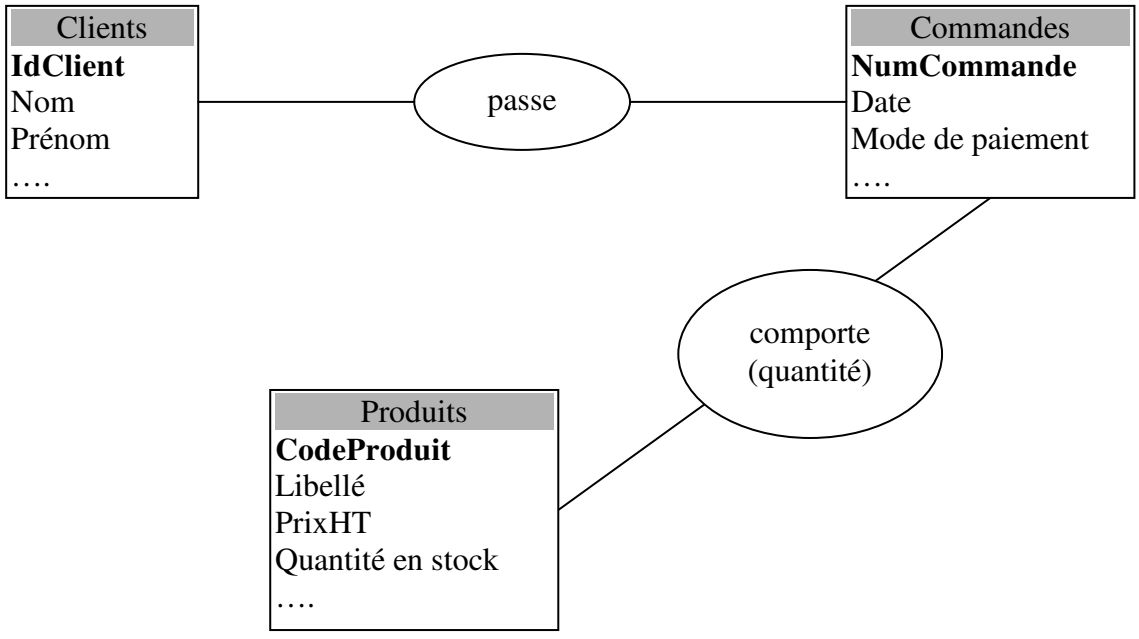
On notera les points suivants :

- Une relation est en général entre **deux** entités. Il est possible d'avoir des relations entre plus que deux entités. Par exemple, une relation **Vente** entre **Acheteur**, **Vendeur** et **Lieu** pour une base de donnée de transactions immobilières. Il est néanmoins souvent possible (et préférable !) de se restreindre à des relations entre deux entités. Dans le cas ici, la relation **Vente** pourrait être remplacée par une entité **Acte de vente** qui est en relation avec l'acheteur, le vendeur et le lieu.
- Il est tout à fait possible d'avoir plusieurs relations entre deux entités.
- Il est également possible d'avoir une relation dite *réflexive*, c'est-à-dire entre une entité et elle-même. Par exemple, on peut avoir une relation **Responsable** entre une table **employés** et elle-même. Dans ce cas, il convient tout de même de remarquer que chacune des "pattes" de la relation a une signification différente. Ici, l'une des "pattes" signifiera *est responsable de* et l'autre signifiera *a comme responsable*.



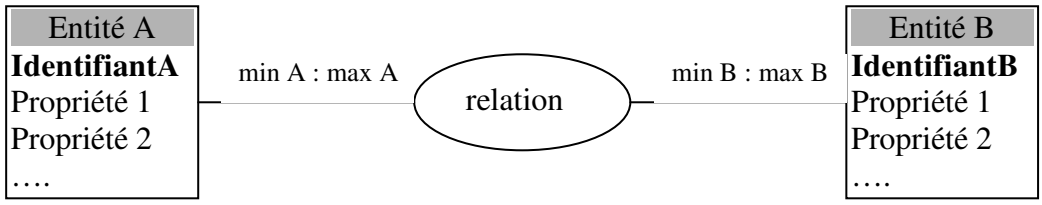
- Une relation peut avoir des propriétés. Par exemple, si une relation **Comporte** lie l'entité **Commandes** et l'entité **Produit**, elle possède certainement la propriété "quantité" (une facture contient un produit *x* en quantité *y*). D'ailleurs, si une propriété dépend de plus d'une entité (comme c'est le cas ici avec la quantité qui dépend à la fois de la facture et du produit), c'est certainement qu'elle dépend d'une relation, et non pas d'une entité.
- Il faut éviter les relations que l'on peut déduire d'autres relations par transitivité. Par exemple, dans une base de données gérant une université, si on dispose d'entités **étudiant**, **formation** et **cours**. On a les relations **fait partie** entre formation et cours (un cours fait partie d'une formation) et **inscription** entre étudiant et formation. Il est inutile d'avoir en plus une relation **inscription** entre étudiant et cours : tout étudiant inscrit à une formation est systématiquement inscrit à tous les cours qui composent la formation.

Exemple : Dans l'exemple de la gestion des commandes de la société, on a les relations suivantes. La relation « commande comporte produits » a une propriété « quantité » qui correspond à la quantité d'un produit qui a été commandé.



d. Choix des cardinalités

Une fois les relations établies, il convient ensuite de caractériser le nombre de fois ou chacune de ces relations peut apparaître réellement. Ceci se fait à l'aide des cardinalités. Dans une relation classique (i.e. entre deux entités), quatre cardinalités sont à déterminer.

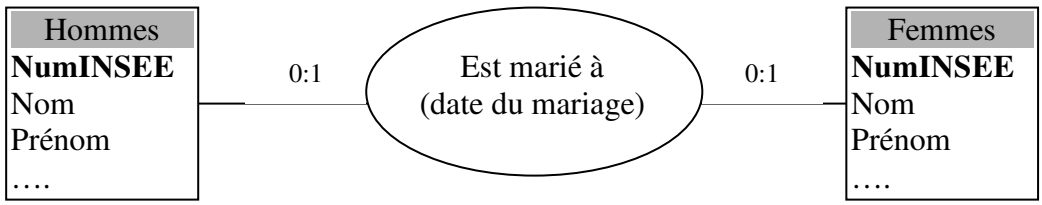


- min_A est le nombre *minimal* de fois où une occurrence de l'entité *A* participe à une relation du type considéré. Il s'agit en général de 0 ou 1.
- max_A est le nombre *maximal* de fois où une occurrence de l'entité *A* participe à la relation. Il s'agit en général de 1 ou *n* (*n* pour *plusieurs* fois, ou un nombre quelconque de fois).
- min_B et max_B fonctionnent de la même manière, mais en considérant l'entité *B*.

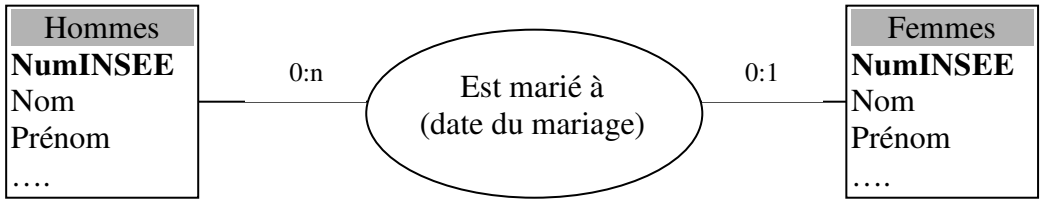
Notons qu'il est souvent difficile de choisir entre une cardinalité de type 0:*n* et une cardinalité de type 1:*n*. Dans le premier cas, la participation à la relation est facultative pour les occurrences de l'entité, alors qu'elle est obligatoire dans le second cas : toute occurrence de l'entité participe à la relation. Il faut toutefois souligner que ce choix a souvent peu d'importance.

Pour illustrer la notion de cardinalité, prenons l'exemple d'une base de données destinée à enregistrer les mariages entre les hommes et les femmes dans des sociétés ayant des régimes matrimoniaux différents. Nous avons deux entités (les hommes et les femmes) et une relation (« est marié à ») avec une propriété « date du mariage ». Les cardinalités minimales seront égales à 0 si l'on prend en compte tous les hommes et toutes les femmes (y compris les célibataires) et à 1 si l'on ne prend en compte que les hommes et les femmes mariés. Les cardinalités maximales seront différentes selon le régime matrimonial en vigueur :

1- dans une société interdisant la polygamie et la polyandrie

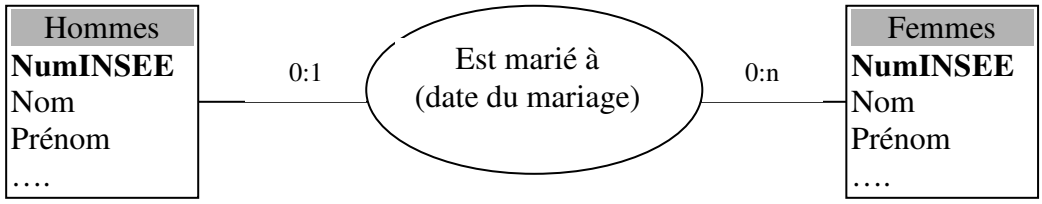


2- dans une société autorisant la polygamie mais interdisant la polyandrie

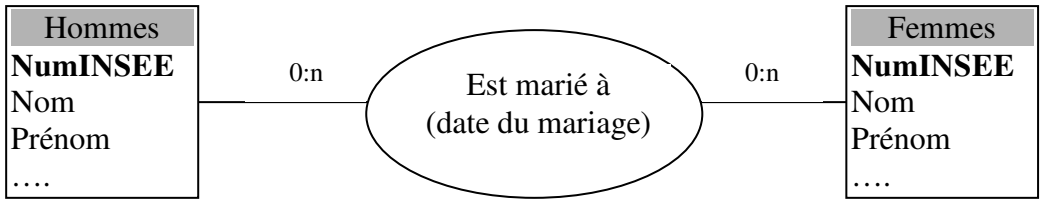


Remarque : si seule la bigamie était autorisée, la cardinalité maximale pour l'entité Hommes serait 2 (et non pas n).

3- dans une société autorisant la polyandrie mais interdisant la polygamie

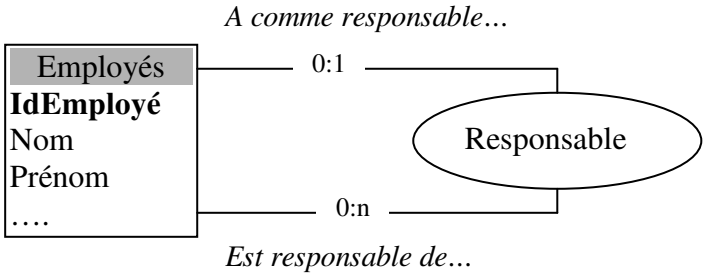


4- dans une société autorisant la polygamie et la polyandrie

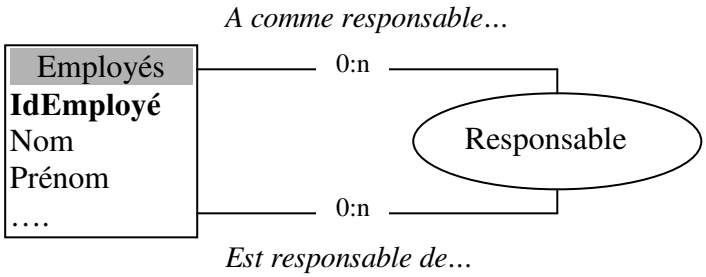


Autre exemple : la relation réflexive **Reponsable** entre une table **employés** et elle-même. Il faut distinguer le cas où un employé ne peut avoir qu'un seul supérieur hiérarchique direct du cas où il peut en avoir plusieurs (par exemple le technicien informatique d'un UFR a deux responsables hiérarchiques directs : le directeur de l'UFR et le directeur du Centre de Ressources Informatique).

Dans le premier cas (un seul supérieur hiérarchique direct), les cardinalités sont :



Dans le deuxième cas (possibilité de plusieurs supérieurs hiérarchiques directs), les cardinalités sont :

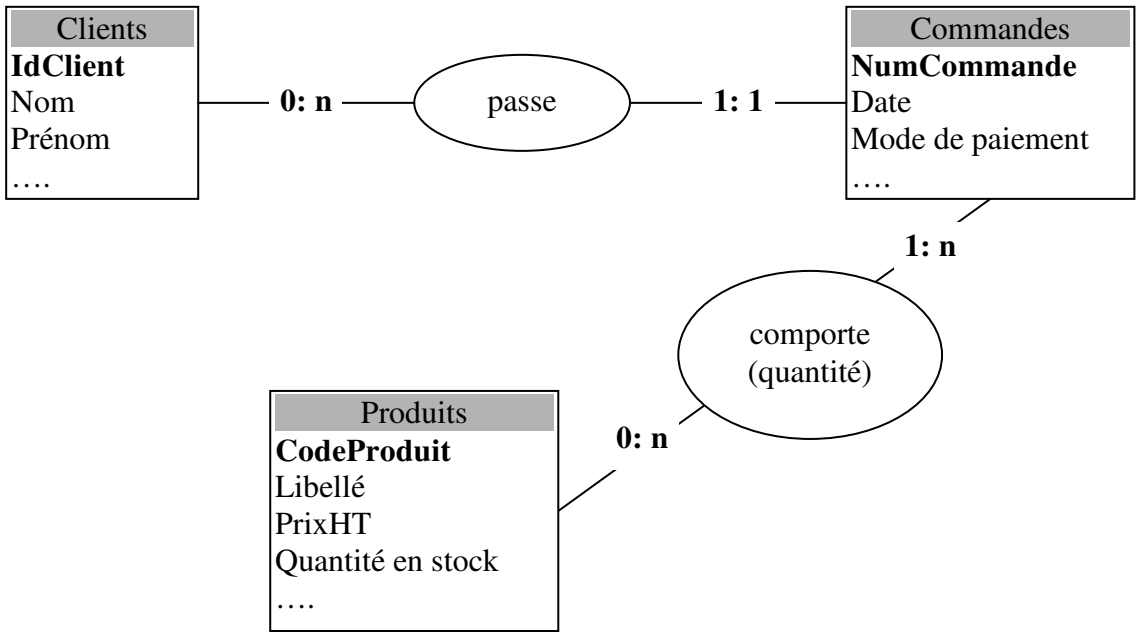


Exemple : Dans l'exemple de la gestion des commandes de la société, les cardinalités sont les suivantes :

Dans la relation « Clients passe Commandes », pour l'entité Clients la cardinalité minimale est 0 (si l'on considère que le fichier clients contient des « clients » qui n'ont encore jamais passé de commandes) et la cardinalité maximale est n (un client peut passer plusieurs commandes !). Dans cette même relation, pour l'entité Commandes la cardinalité minimale est 1 (une commande est obligatoirement passée par un client !) et la cardinalité maximale est 1 (une commande ne peut être passée par deux clients différents !).

Dans la relation « Commandes comporte Produits », pour l'entité Commandes la cardinalité minimale est 1 (une commande comporte au moins un produit) et la cardinalité maximale est n (puisqu'une commande peut comporter plusieurs produits différents). Dans cette même relation, pour l'entité Produits la cardinalité minimale est 0 (si l'on considère que le fichier produits contient des produits qui n'ont encore jamais été commandés !) et la cardinalité maximale est n (un produit peut figurer dans plusieurs commandes différentes).

Le MCD complet est donc :



e. Cas particuliers et pièges

Quelques points particuliers sont à garder à l'esprit lors de la réalisation d'un MCD.

- Un identifiant est obligatoire pour chaque entité.
- Il ne doit pas y avoir de redondance d'informations : une information quelconque ne doit pas être représentée plus d'une fois dans le MCD.
- Évitez autant que possible les relations entre plus de deux entités. Souvent, il est possible de remplacer la relation par une entité.
- Restez dans la mesure du possible avec des cardinalités de valeurs 0, 1 ou *n*. Il est de toute manière souvent possible de se ramener à ce cas dans les rares cas où des cardinalités d'un autre type semblent plus naturelles.

- Dans l'idéal, il faut trouver un bon compromis entre niveau de détail et "taille" de la base de données. Il est toujours possible de multiplier les entités, mais il vaut mieux le faire que si cela a vraiment du sens et un intérêt dans le problème. Par exemple, si on a une entité personne, on peut considérer l'adresse comme une entité séparée (reliée à personne par une relation "habite à") ou comme une propriété de la personne (ce qui est fait usuellement). En règle générale, il est plus économique de définir l'adresse comme une propriété, mais dans un cas où il est fréquent que des personnes habitent au même endroit, la règle de non-redondance incite plutôt à utiliser une nouvelle entité.

A retenir : La méthode générale de construction du MCD :

1. recherche des *entités*,
2. recherche des *propriétés* (dont la *clé* de chaque entité),
3. recherche des *relations* entre entités,
4. recherche des *cardinalités* (0:1, 1:1, 0:*n* ou 1:*n* ?)

3. Modèle Logique de Données

Une fois le MCD construit, l'étape suivante dans la conception de la base de données consiste à concevoir le *modèle logique de données*, ou MLD. Ce MLD montre l'organisation des données sous forme de tables et est très proche de la manière dont les données vont être effectivement organisées dans **Access**. L'étape de transformation du MCD en MLD est assez simple et passe par trois étapes :

1. transformation des entités en tables,
2. transformation des relations du MCD,
3. suppression des tables inutiles.

a. Construction des tables

La première étape consiste à transformer toutes les entités du MCD en tables du MLD. Cette transformation est directe : il suffit de recopier les entités. Il s'agit essentiellement d'un changement de vocabulaire :

- une **entité** devient une **table**,
- une **propriété** devient un **champ**,
- un **identifiant** devient une **clé primaire**,
- une **occurrence** d'une entité devient un **enregistrement** de la table.

A noter toutefois qu'il est essentiel qu'il n'y ait pas deux tables qui aient le même nom.

Exemple : la première partie de la construction du MLD de la société est directe. Il suffit de recopier les entités.

Clients
IdClient
Nom
Prénom
….

Commandes
NumCommande
Date
Mode de paiement
….

Produits
CodeProduit
Libellé
PrixHT
Quantité en stock
….

b. Transformation des relations en liens

L'étape suivant consiste à transformer les relations du MCD en liens du MLD. Deux grands cas peuvent se présenter.

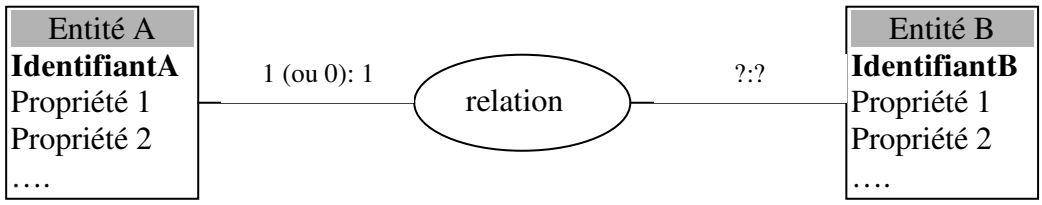
1. le cas où l'une des branches de la relation a une cardinalité maximale de 1 (1:1 ou 0:1)

2. le cas où les deux branches de la relation ont une cardinalité maximale de n (1:n ou 0:n)

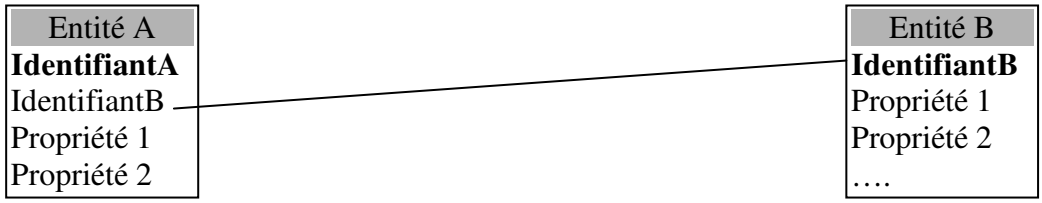
Premier cas : l'une des branches de la relation a une cardinalité maximale de 1 (1:1 ou 0:1)

Dans le cas d'une relation ou l'une des branches a une cardinalité de 1:1 ou 0:1, la transformation de la relation se fait de la manière suivante :

- On ramène dans la table correspondant à l'entité "du côté du 1:1" (ou du 0:1) la clé primaire de l'autre table ainsi que toutes les éventuelles propriétés de la relations.
- On lie la clé primaire ainsi importée avec la clé primaire de la deuxième table.
- Si la relation contenait des propriétés, celle-ci se retrouve également importées du côté du 1:1.

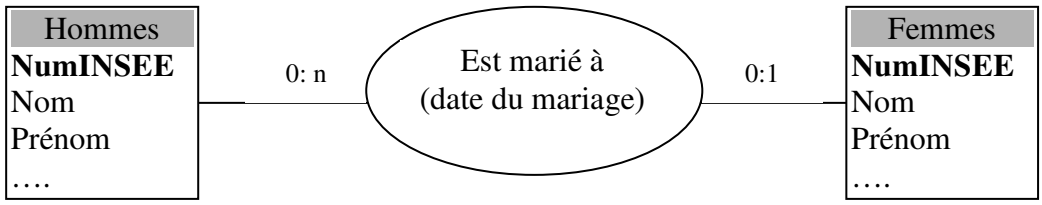


Se transforme en :

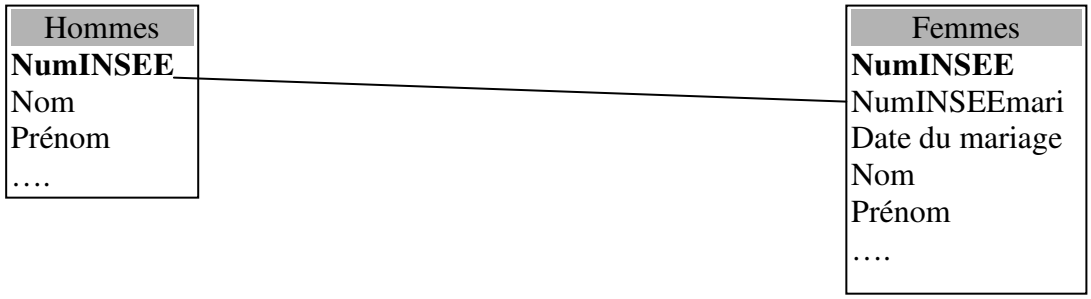


A noter que la clé importée (ici **IdentifiantB** qui se retrouve dans **table A**) ne devient pas une clé de la table : c'est une propriété comme une autre. Notons aussi que le lien se fait *entre champs* (on relie IdentifiantA à IdentifiantB) et non pas, comme dans le MCD, entre les tables.

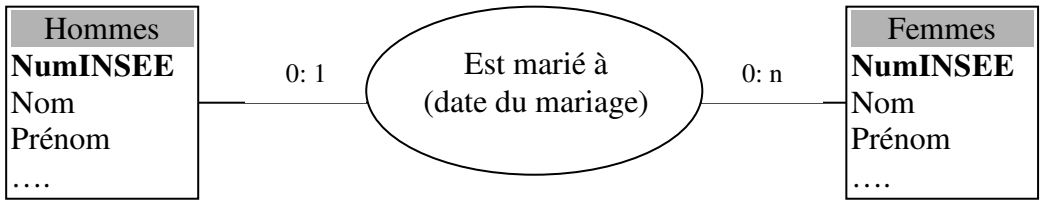
Dans l'exemple de la société autorisant la polygamie mais interdisant la polyandrie, le MCD



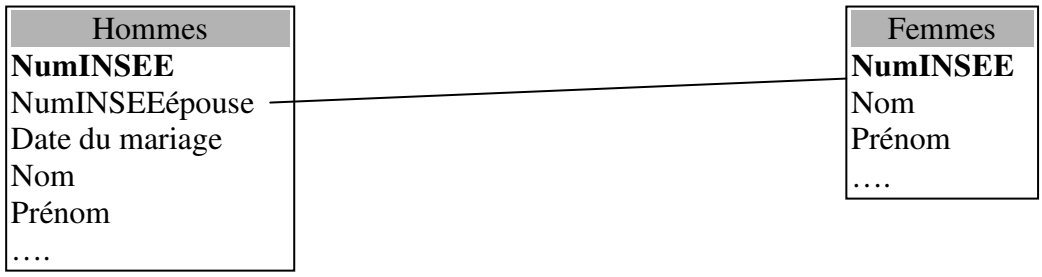
se transforme en :



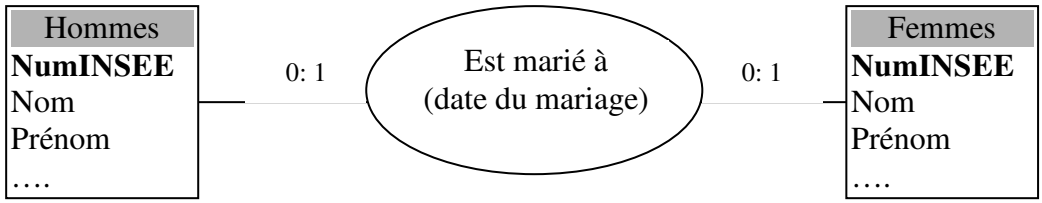
De même, pour la société autorisant la polyandrie mais interdisant la polygamie



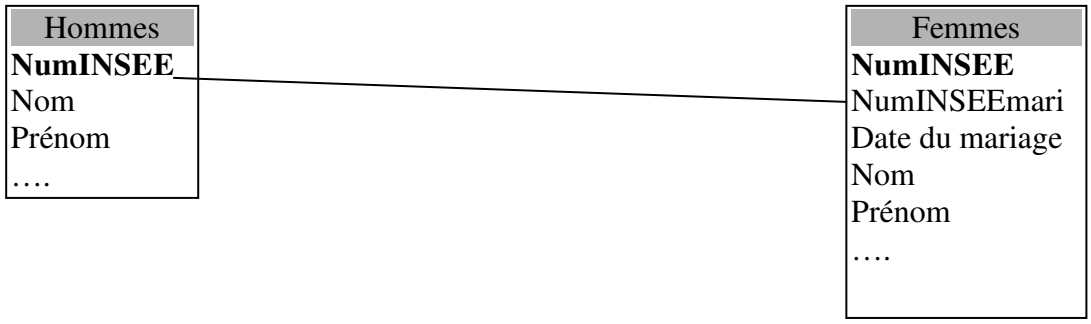
se transforme en :



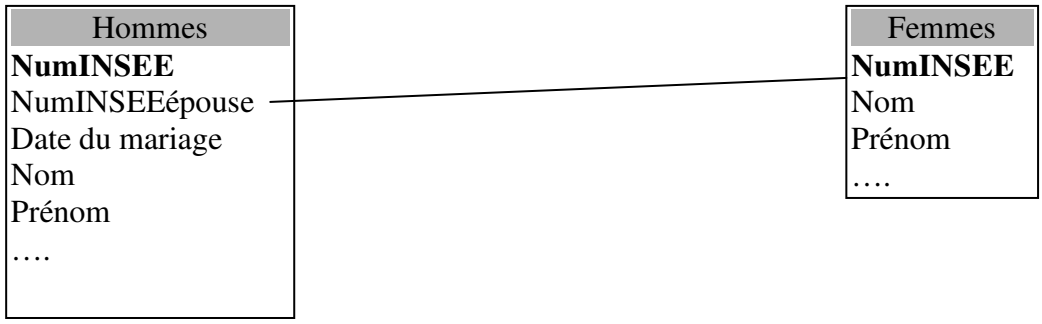
Remarque : si les cardinalités sont 0 : 1 pour les deux entités participant à la relation, on peut soit importer la clé primaire de la deuxième table dans la première, soit importer la clé primaire de la première table dans la deuxième. Par exemple, dans une société interdisant la polygamie et la polyandrie, le MCD :



peut se transformer en :



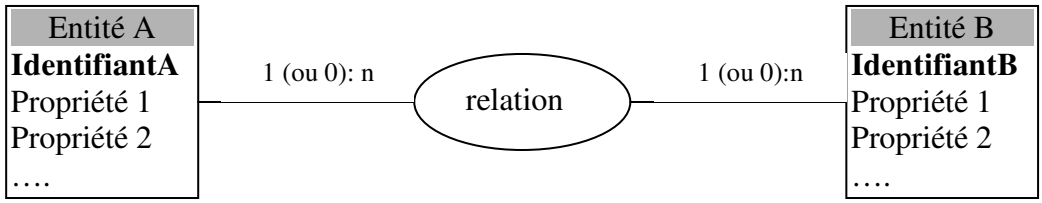
ou se transformer en :



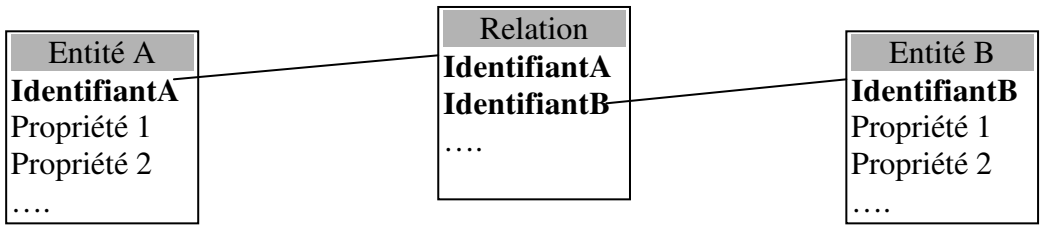
Deuxième cas : les deux branches de la relation ont une cardinalité maximale de n (1:n ou 0:n)

Dans ce cas, la relation du MCD se transforme en une table du MLD :

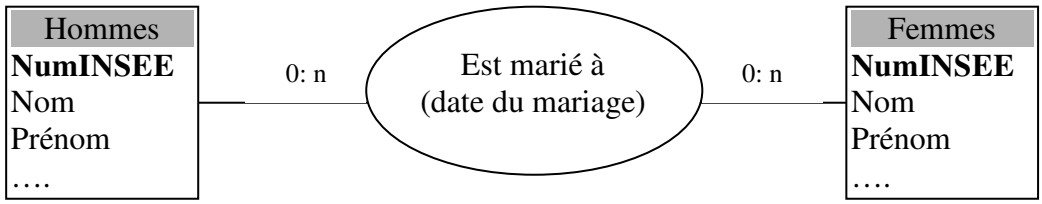
- On crée une nouvelle table correspondant à la relation. Cette table contient toutes les éventuelles propriétés de la relation.
- On intègre à cette table les clés primaires des entités impliquées dans la relation.
- On relie les clés primaires des tables avec les clés importées dans la nouvelle table.
- On choisit enfin la ou les clés primaires de la nouvelle table. L'idée générale est que chaque occurrence de cette entité doit pouvoir être identifiée de manière unique par ses clés primaires. Cela revient en général à choisir comme clé primaire l'ensemble des clés importés des autres tables.



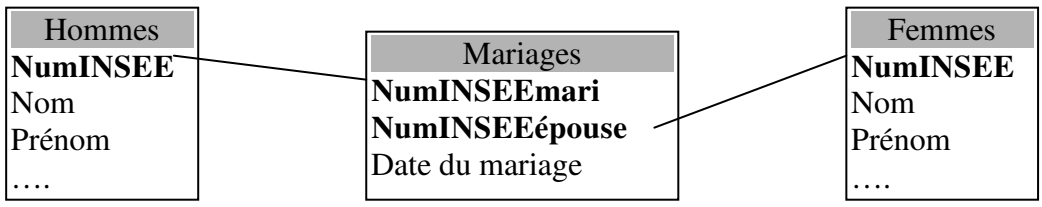
se transforme en :



Dans l'exemple de la société autorisant la polygamie et la polyandrie, le MCD :



se transforme en



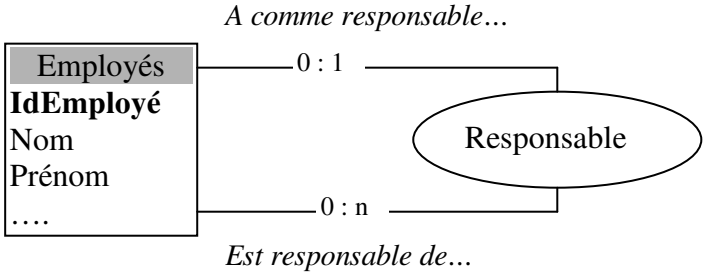
Cas particuliers

Quelques cas particuliers peuvent apparaître.

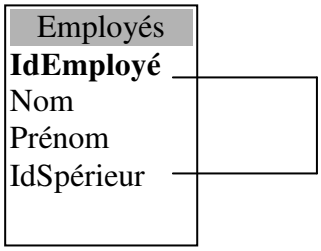
- Si la relation est de type 1:1 - 1:1, on fusionne les deux entités en une. Ce type de relation rare est souvent dû à un problème dans la conception du MCD.
- Si la relation est de type 0:1 - 1:1, on traite la relation comme une relation de type 1:1 - ?:? (en ramenant la clé primaire du côté du 1:1)
- les relations ternaires (entre trois entités, ou plus), se traitent comme d'habitude. Si l'une des branches a une cardinalité de type 1:1, on ramène les clés primaires des autres entités et les propriété de la relation dans l'entité "du côté du 1:1". Si ce n'est pas le cas, la relation se transforme en table.
- Les relations réflexives (entre une entité et elle-même) se traitent comme les autres relations.

Exemple : la réflexion réflexive **Reponsable** entre une table **employés** et elle-même.

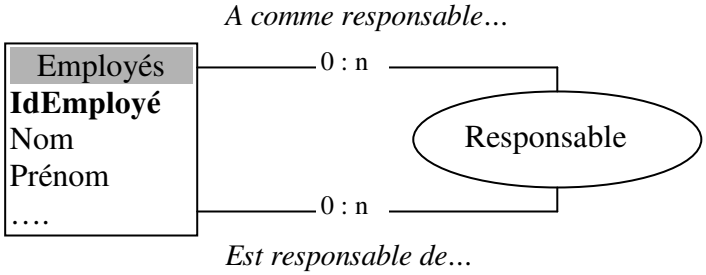
Dans le cas d'un seul supérieur hiérarchique direct :



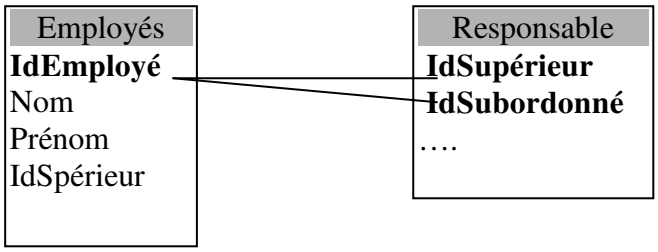
se transforme en :



Dans le cas où il peut exister plusieurs supérieurs hiérarchiques directs :



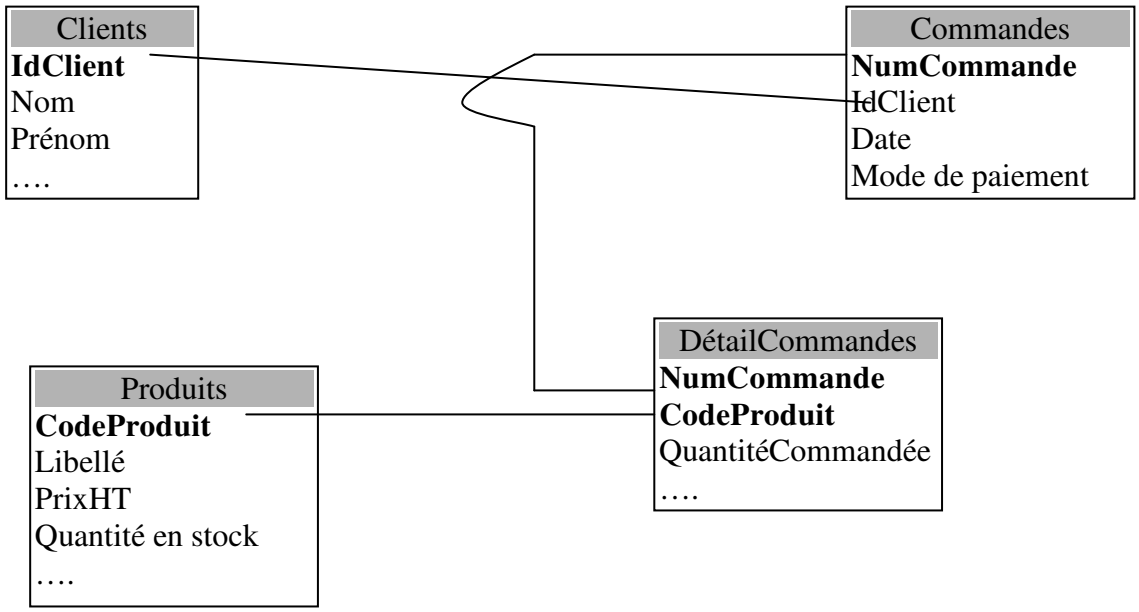
se transforme en :



c. Suppression des tables inutiles

La dernière étape consiste simplement à supprimer les tables inutiles. En général (mais pas toujours), une table qui ne contient qu'un seul champ (sa clé) est inutile : elle ne nous apporte aucune information. L'exemple le plus classique est une entité de type "date".

Exemple : Le MLD correspondant à l'exemple de la société qui veut informatiser la gestion de ses commandes :



A retenir : La méthode de transformation MCD – MLD :

1. les *entités* sont transformées en *tables* (sans modification)
2. les *relations* sont transformées en fonction de leurs cardinalités :

- *une relation de type ?:1 - ?:n entre une entité A et une entité B se traduit par une importation de la clé primaire de l'entité B dans la table de A, et on ajoute un lien entre les deux clés,*
- *une relation de type ?:n - ?:n) se transforme en table dans laquelle on retrouve les clés primaires de A et B.*

3. les tables inutiles sont supprimées : il s'agit essentiellement des tables à un seul champ (leur clé).

IV. Exercices d'application

Construire le Modèle conceptuel des données (MCD) puis le Modèle Logique de données (MLD) des exemples suivants (très simplifiés).

Rappel

La construction du MCD nécessite :

- *le repérage des entités,*
- *la construction des entités et le choix des propriétés, (dont la clé de chaque entité),*
- *la construction des relations,*
- *le choix des cardinalités. (0:1, 1:1, 0:n ou 1:n ?)*

Pour construire le MLD à partir du MCD :

- *les entités sont transformées en tables (sans modification)*
- *les relations sont transformées en fonction de leurs cardinalités :*
  - *une relation de type ?:1 - ?:n entre une entité A et une entité B se traduit par une importation de la clé primaire de l'entité B dans la table de A, et on ajoute un lien entre les deux clés,*
  - *une relation de type ?:n - ?:n) se transforme en table dans laquelle on retrouve les clés primaires de A et B.*

1- Une société qui vend des produits veut informatiser la gestion des commandes de ses clients. Chaque commande d'un client peut comporter plusieurs produits différents.

2- Une entreprise veut informatiser la gestion de ses stocks. Les produits qu'elle commercialise se trouvent dans différents entrepôts.

3- Une compagnie d'assurance automobile a des clients qui signent des contrats. Un contrat couvre un véhicule qui appartient à un client ; celui-ci peut avoir plusieurs véhicules. Il peut arriver que ces véhicules aient des accidents.

4- Un collège souhaite informatiser la gestion des notes de ses élèves. Le collège comprend des classes qui ont des élèves. Les professeurs enseignent différentes matières. Les élèves ont des notes à différentes épreuves de chaque matière.

5- Un comité d'entreprise veut informatiser le catalogue des CD qu'il possède. Pour chaque CD on désire connaître son nom, son genre (variété, classique, jazz…), l'année de sa création, son prix HT et la quantité en stock. Un CD comprend un certain nombre de titres. Chaque titre a un intitulé, est l'œuvre d'un ou de plusieurs auteurs et est interprété par un ou plusieurs interprètes. Pour les auteurs et les interprètes, on se limitera à la connaissance de leur nom et de leur prénom. Prenez garde au fait qu'un même CD peut comporter des titres d'auteurs et d'interprètes différents et que bien évidemment un auteur (ou un interprète) peut avoir composé (ou interprété) des titres sur différents CD. Il faut également tenir compte du fait qu'un auteur peut être également interprète, et qu'un même titre peut figurer sur des CD différents. Voici un exemple d'extraits du contenu d'un CD :

Les enfoirés à l'opéra comique. (1995) prix HT : 90 F. genre : Variété Quantité en stock : 45		
Titres	*Auteurs*	*Interprètes*
Dix ans ça suffit	Patrick Bruel, B. Garcin	Patrick Bruel, Vanessa Pardis
Chansons des jumelles	J. Demy, M. Legrand	Liane Foly, Mimie Mathy
L'italien	J.L. Dabadie, J. Datin	Patrick Bruel, Serge Reggiani
…		

6- La société "Mon beau gîte" veut informatiser la gestion de la location de son gîte à des personnes. Celui-ci a une capacité de 60 places. Les séjours durent une semaine (du samedi au samedi), identifiée par un numéro.

7- Une municipalité veut informatiser la gestion de sa salle des fêtes. Une réservation est effectuée pour une journée (évidemment il ne doit pas être possible d'effectuer deux réservations différentes à la même date !) par un utilisateur. Un utilisateur peut effectuer plusieurs réservations à des dates différentes. Pour les utilisateurs, il est important de savoir s'ils résident (ou non) dans la commune. En effet, la salle est facturée 300 € pour une journée si l'utilisateur réside dans la commune et 400 € sinon. Lors de la réservation, l'utilisateur précisera s'il souhaite également louer la sono qui est facturée 50 €. Enfin l'utilisateur a la possibilité de louer également pour la journée une certaine quantité de certains articles dont la liste figure ci-dessous.

Numéro	Libellé	Quantité disponible	Prix de location unitaire
1	Table	20	5,00 €
2	Chaises	150	1,00 €
3	Assiettes	300	0,20 €
4	Verres	500	0,10 €

Par ailleurs, il faut préciser si la location a déjà été payée ou non.

8- Une banque désire informatiser les opérations de retrait et de dépôt d'espèces de ses clients sur leur compte. Un client peut avoir plusieurs comptes à la banque et un compte peut être commun à plusieurs clients (compte joint).

9- Une librairie veut informatiser sa gestion. Elle a des clients dans différentes villes qui lui passent des commandes. Une commande peut comporter plusieurs livres différents. Un livre est écrit par un ou plusieurs auteurs et est édité par un éditeur.
Cet exercice servira de base à la suite du cours.

V. Exercices supplémentaires

1. Exercice 1

Le bureau de gestion de la présidence de la Communauté Européenne a la responsabilité d'organiser les réunions de plusieurs groupes de travail.

Le directeur du bureau doit pouvoir fixer les dates ainsi que le sujet des réunions et désigner le (ou les) groupe(s) de travail qui y participe(nt). Il doit également pouvoir imprimer un calendrier de toutes les réunions qui ont lieu durant une période donnée.

La secrétaire du bureau doit pouvoir tenir à jour un fichier de personnes (avec leurs noms et leurs coordonnées) ainsi que la composition des groupes de travail. En particulier, elle doit connaître les coordonnées du responsable de chaque groupe de travail.

1. Identifiez les entités de ce problème.
2. Construisez le MCD qui servira à l'informatisation de ces tâches
3. Réalisez le MLD correspondant.

2. Exercice 2

Le camp de vacances « Club Mer du nord » souhaite s'informatiser. Ce club possède une centaine d'emplacements (tente, caravane ou bungalow). Il propose plusieurs activités sportives, certaines étant encadrées par un animateur d'autres non. Certaines animations nécessitent une location de matériel (planche à voile, voilier…).

La direction du centre désire stocker des informations sur tous les clients qui passent au camp : leurs coordonnées, la durée, la date, le type de leur séjour et leur divers locations. Dans le cas (fréquent) ou plusieurs personnes d'une même famille s'inscrivent ensemble, on désire connaître le nom, le prénom, la date de naissance et les coordonnées de chaque membre de la famille.

Par ailleurs, le système devra également gérer les animateurs : leur paie, les activités qu'ils peuvent encadrer et celles qu'elles encadrent effectivement à une date donnée. Les animateurs étant gérés selon une hiérarchie stricte, on désire également savoir quel est le responsable éventuel de chaque animateur.

Le système devra également gérer les emplacements du camp à savoir leur type et le client qui l'occupe pour tout séjour.

1. Repérez les différentes entités du problème
2. Réalisez le MCD correspondant. Vous veillerez à ne pas stocker d'information redondante.
3. Réalisez le MLD à partir du MCD.

3. Exercice 3

Le Conseil Supérieur de l'Audiovisuel désire constituer une base de données permettant de référencer toutes les émissions de radio nationales diffusées sur les ondes FM. La base doit contient la référence de toutes les émissions diffusées régulièrement et toujours à la même heure, mais pas plus d'une fois par jour (par exemple, les flashs d'actualité diffusés toutes les heures correspondent chacun à une émission différente). Pour simplifier, on considère que toutes les émissions sont quotidiennes. On veut par contre recenser les heures de passage et la durée des émissions. La base doit également enregistrer les coordonnées des directeurs des

radios, des animateurs intervenant dans les émissions, ainsi que des personnels techniques qui y travaillent. Il doit ainsi être possible pour chaque émission de savoir qui l'anime, quels sont les techniciens qui participent à son élaboration, sur quelle radio est-elle diffusée. Pour chaque radio, on désire connaître le nom du directeur.

1. Repérez les différentes entités du problème
2. Réalisez le MCD correspondant. Vous veillerez à ne pas stocker d'information redondante.
3. Réalisez le MLD à partir du MCD.

4. Exercice 4

Il vous est demandé de modéliser le système d'information nécessaire à la gestion des animaux du zoo de la ville de…, en vue d'une informatisation future. Dans ce but, le chef de projet, M. Adémar SUPIAL a interviewé la directrice du zoo, Mme Nathalie GATOR ; en voici un extrait :

M. SUPIAL :

"Quels domaines souhaitez-vous informatiser ou réorganiser ?"

Mme GATOR :

"En priorité les tâches qui nous donnent le plus de travail administratif, c'est-à-dire :

- la gestion de l'alimentation des animaux,
- Le suivi journalier des aliments distribués dans les différentes zones, avec si possible un cumul mensuel,
- la gestion de la répartition des animaux dans les zones de parcage (pouvoir enregistrer les arrivées d'animaux, savoir où se trouve chaque animal,...)…

Je souhaiterais en plus, que devant chaque zone, les visiteurs disposent d'un écran ou d'un affichage électronique connecté à l'ordinateur central sur lequel seraient affichées les caractéristiques des animaux qui s'y trouvent, cela serait-il possible ?"

M. SUPIAL :

"Bien sûr, je trouve même que c'est une excellente idée, mais expliquez-moi d'abord certains points ; comment fonctionnez-vous actuellement ?"

Mme GATOR :

"Tous nos animaux sont répertoriés sur fiches cartonnées que nous mettons régulièrement à jour ; nous y précisons divers renseignements comme l'espèce, la zone où ils se trouvent, et cœtera ; nous inscrivons dans un grand livre les nombreuses distributions d'aliments que nous effectuons dans chaque zone en fonction des besoins spécifiques des espèces ; ces besoins par espèce sont consignés dans un autre livre...

Nous passons d'ailleurs beaucoup de temps à calculer les quantités de nourriture à distribuer, vu que dans certaines zones se trouvent plusieurs dizaines d'animaux. Le stock de nourriture est tenu à jour -plus ou moins bien- par un magasinier."

M.SUPIAL :

"Comment identifiez-vous les animaux ?"

Mme GATOR :

"Chaque animal est identifié par un numéro unique à 6 chiffres qui lui est attribué dès sa naissance ou son arrivée dans le zoo. Ce numéro est reproduit sur l'animal par tatouage, baguage ou autres procédés. On connaît la date de naissance, la date d'arrivée ainsi que le sexe de chaque ani-

mal. Un commentaire éventuel est affecté à chaque animal."

M. SUPIAL :

"Pouvez-vous me préciser ce qu'est une zone de parcage ?"

Mme GATOR :

"Chaque animal est parqué dans une "zone" repérée par un code et à chaque zone correspond une désignation : enclos, cage, aquarium, etc.. Les animaux sont parfois seul dans une zone, mais sont le plus souvent à plusieurs -parfois même, des animaux d'espèces différentes cohabitent dans une même zone-."

M. SUPIAL :

"Rappelez-moi ce que vous entendez par nom d'espèce ?"

Mme GATOR :

"A chaque espèce correspond un nom vulgaire, c'est à dire employé couramment, ainsi qu'un nom scientifique. Pour chaque espèce on connaît sa répartition géographique : code géographique et libellé géographique, par ex. AE = Afrique équatoriale, MA = Madagascar.

Nous notons également le nombre -estimé- d'individus vivants dans le monde et il est important pour nous de savoir si l'espèce est menacée de disparition ou non.

Pour chaque espèce, nous avons répertorié son code famille ainsi que la désignation famille. Bien entendu des animaux d'espèces différentes peuvent appartenir à la même famille."

M. SUPIAL :

"Comment déterminez-vous les besoins en aliments ?"

Mme GATOR :

"Les animaux ont des besoins moyens en nourriture journalière, qui sont fonction de l'espèce et du sexe. Ils sont exprimés en nature de l'aliment et en Kg/jour."

M. SUPIAL :

"Pouvez-vous me donner un exemple ?"

Mme GATOR :

"Bien sûr ; prenez l'éléphant d'Afrique mâle par exemple : il lui faut 80 kg de foin + 10 kg d'avoine + 5 kg de carottes par jour."

M. SUPIAL :

"Que se passe-t-il quand vous êtes en rupture de stock de foin par exemple ?"

Mme GATOR :

"Heureusement c'est rare, mais ça arrive... Dans ce cas nous consultons une liste dans laquelle se trouvent les aliments de substitution : par exemple pour le foin, si je prends la liste, je trouve comme aliment de substitution la luzerne ; notre éléphant recevra donc exceptionnellement 80 kg de luzerne en cas de rupture sur le foin !"

M. SUPIAL :

"Comment identifiez-vous les aliments ?"

Mme GATOR :

"Pour des commodités d'étiquetage, nous avons créé un code aliment en plus de la désignation aliment ; pour chaque aliment notre magasinier est censé tenir à jour la quantité disponible en stock."

M. SUPIAL :

"Comment s'effectue la distribution des aliments ?"

Mme GATOR :

"Dix personnes sont responsables de la distribution de la nourriture et de l'entretien des zones. Ces personnes sont identifiées par leur nom et prénom. Chacun des ces employés est responsable d'une ou

plusieurs zones déterminées. Il n'y a qu'un responsable par zone...

La secrétaire transmet chaque jour à chaque responsable de zone une liste où figure la quantité totale d'aliments à distribuer dans une zone ; celui-ci nous rapporte cette liste en fin de journée en ayant pris soin d'y noter les aliments et les quantités REELLEMENT distribuées, en fonction des disponibilités en stock. Puis la secrétaire inscrit ces informations dans le grand livre destiné à l'historique des distributions par zone. Il serait intéressant que les responsables de zone saisissent eux-même les aliments distribués."

M. SUPIAL :

"Si j'ai bien compris, on ne s'intéresse qu'aux aliments à distribuer par zone, la secrétaire doit donc recalculer les besoins à chaque transfert d'un animal dans une autre zone ?"

Mme GATOR :

"Effectivement, et cela arrive très souvent ; si l'ordinateur pouvait permettre d'automatiser le calcul des distributions d'aliments et la mise à jour du stock, et si en plus nous avions un affichage automatique d'informations pour les visiteurs ça serait formidable !"

M. SUPIAL :

"Donnez-moi un exemple de ce que vous souhaitez afficher devant chaque zone."

Mme GATOR :

"Par exemple devant la zone E3 qui est l'enclos à éléphants, nous aurions :

- BABAR : Eléphant d'Afrique - Mâle - né dans le zoo le 12/03/85
- LEONTINE : Eléphant d'Afrique - Femelle - rachetée au cirque Barnum
- Espèces représentées :
  Eléphant d'Afrique (Loxodonta Africana)
  Famille : Eléphantidés
  Répartition géographique : Afrique équatoriale, Afrique du sud
  Nombre estimé d'individus : 200 000
  Espèce menacée de disparition

Question :

Construisez le MCD puis le MLD nécessaire à la base de donnée qui gérera le zoo.

Sommaire