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 :

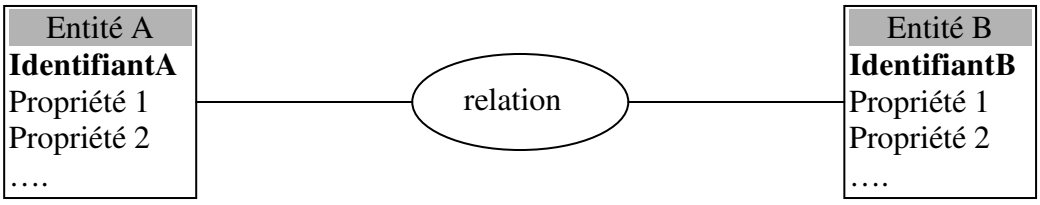
- Clients : **IdClient**, nom, prénom, rue, code postal, pays, tél, email….
- Produits : **CodeProduit**, libellé, prixHT, quantité en stock…
- Commandes : **NumCommande**, date, mode de paiement
- Villes : **CodePostal**, ville

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

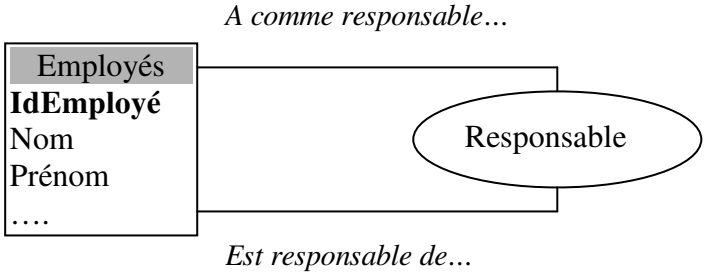
L'étape suivante consiste à énumérer toutes les relations *possibles* entre entités. Si une relation a une chance d'apparaître (et de nous intéresser), alors on doit la considérer dans le MCD. On parle également parfois d'*association.* Dans la description de la situation à informatiser les relations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



Ceci est un exemple, cliquez sur le lien de téléchargement pour obtenir le cours complet.