Chapitre 1 : Généralités sur les bases de données

1 Problèmes posés par les organisations classiques

L'approche classique de mise en place d'une application informatique dans une entreprise, consistait le plus souvent à l'écriture d'un certain nombre de programmes destinés à l'exploitation d'un ensemble de fichiers qu'il fallait aussi créer. Les principaux problèmes posés par cette démarche sont : la redondance des informations et la dépendance entre les données et les programmes qui les manipulent.

1.1 Redondance des informations

La création anarchique de fichiers ajoutée à l'absence de coordination entre les groupes de développement entraîne une duplication non contrôlée des informations (préexistence de ces informations dans d'autres fichiers propres à d'autres applications). Cette duplication engendre des redondances c'est à dire qu'une même information peut se trouver dans plusieurs fichiers distincts. Par exemple, l'identité d'un employé (nom, prénom, adresse, etc.) peut figurer dans un fichier F1 propre à une application P1 calculant la paie, et dans un fichier F2 propre à une autre application P2 de gestion des remboursements des frais médicaux. Une telle situation complique les opérations de mise à jour car il est nécessaire pour une même information mise à jour dans un fichier de la mettre à jour dans tous les fichiers où elle est supposée exister. (Ex : si on change l'adresse d'un employé dans le fichier F1, il faut faire aussi ce changement dans tous les fichiers contenant cette information). Ceci devient de plus en plus difficile dès que le nombre de fichiers est élevé et souvent on a tendance à retarder la mise à jour dans les autres fichiers ce qui engendre une situation incohérente qui se traduit par le fait que l'interrogation de la même information par deux utilisateurs différents chacun sur son propre fichier fournira deux résultats différents.

1.2 Dépendance entre les données et les programmes

Généralement, la création de fichiers faite lors de la mise en place d'une application sous entend la réalisation d'un certain nombre d'opérations telles que :

- La structuration des enregistrements : définir les champs, leurs types et leur ordre
- Le choix d'une organisation : séquentielle, séquentielle indexée, directe, etc.
- Le choix du support : disque dur, bande magnétique, etc.

Cette création était donc figée car faite en fonction d'un ou de plusieurs programmes. Ceci veut dire que les données contenues dans les fichiers sont directement associées aux programmes qui les exploitent et ceci par le biais d'une description contenue dans ces programmes eux-mêmes.

Exemple :

Si on a à mettre en place une application de gestion de prêts dans laquelle on suppose qu'un emprunteur peut emprunter jusqu'à trois (3) ouvrages, on doit créer un fichier ayant à peu près la structure suivante :

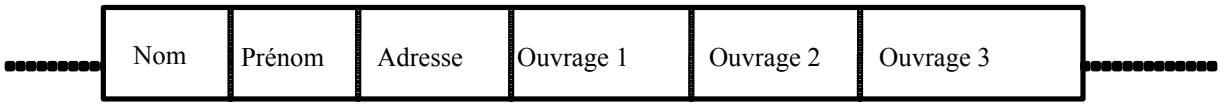

Figure 1.1 : Allure des enregistrements du fichier de l'application ''gestion de prêt''

Les programmes qui utilisent un tel fichier en donnent la description à l'intérieur du programme même. Si on se réfère par exemple au langage COBOL, une telle description aurait l'allure suivante :

```
01      PRET
        02          NOM             PICTURE             A(20)
        02          PRENOM          PICTURE             A(20)
        02          OUVRAGE         OCCURS              3       TIMES
                    03              COTE_OUVRAGE        9(6)
```

Si ce fichier est utilisé par les programmes P1, P2, P3 (voir Figure 2) et si on doit le modifier pour le besoin de P2 (prise en compte de l'âge de l'emprunteur par exemple !), il faudra modifier P1 et P3 en conséquence. Ceci est dû au fait que tous les programmes *voient* le fichier de la même façon bien qu'ils n'aient pas tous besoin des mêmes informations. De même qu'un changement de l'organisation du fichier sur disque pour le besoin d'un programme Pi (passer par exemple d'une organisation séquentielle à une organisation directe afin d'optimiser les temps d'accès de Pi) obligera les autres programmes à prendre en compte ce changement et en quelque sorte à suivre Pi même s'ils n'ont pas vraiment besoin de ce type d'organisation. De même qu'une réorganisation des champs de l'enregistrement (changement de l'ordre des champs à l'intérieur de l'enregistrement) entraînera une modification dans les programmes et une reconstruction du fichier sur le support.

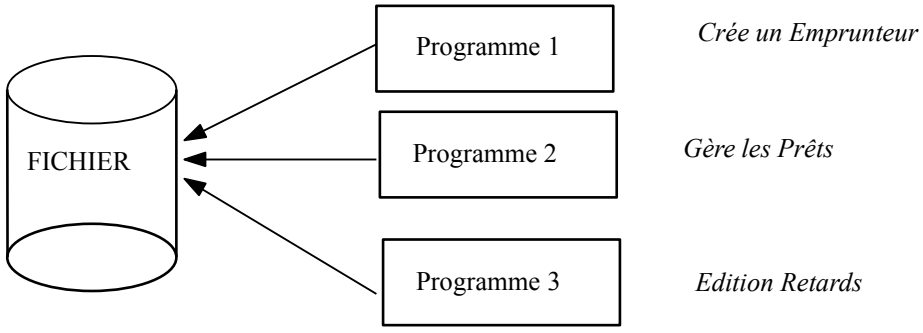

Figure 1.2 : Vue d'un même fichier par 3 programmes différents

On peut dire qu'il manque un moyen de filtrer les informations afin de ne retenir que celles qui sont utiles à un programme donné. Ceci montre qu'il existe une dépendance étroite entre les données et les programmes qui les manipulent qui se traduit par le fait qu'il n'est pas possible de changer la structure et l'organisation des données sans modifier en conséquence les programmes.

2 Bases de données : Notions générales

2.1 Définition

Une base de données est un ensemble d'informations structurées permettant la mise en place d'une série d'applications informatiques destinées à une grande variété d'utilisateurs.

Exemple:

Dans une entreprise, les informations concernant son fonctionnement :

- Employés
- Produits Fabriqués
- Moyens matériels (Machines, Véhicules, Magasins, etc.)

peuvent être rassemblées et mises à la disposition de nombreux utilisateurs (cadres de l'entreprise, gestionnaires, opérateurs, etc.).

2.2 Objectifs d'une base de données

Parmi les principaux objectifs visés par une base de données, on peut citer :

2.2.1 Partage de l'information

Une base de données permet le partage d'un ensemble unique d'informations par plusieurs utilisateurs. Cependant, il faut que cette mise en commun soit faite tout en préservant la vue particulière que chaque utilisateur peut avoir des informations, et en s'assurant que la simultanéité des traitements qui peuvent être effectués ne risque pas de dégrader l'intégrité de la base de données.

2.2.2 Organisation des données indépendamment des programmes

Afin de construire un ensemble d'informations structurées non redondant et qui soit partageable par plusieurs utilisateurs, il est nécessaire de faire abstraction des traitements particuliers de tel ou tel utilisateur (ou programme) pour tenter d'organiser les informations en fonction de leur nature et des liens réels qui existent entre elles. C'est de cette manière qu'on arrivera à garantir le maximum d'indépendance entre données et programmes.

2.3 Rôle d'une Base de données

Contrairement aux approches classiques, la création d'une base de données qui soit partagée par plusieurs utilisateurs est le reflet d'une évolution dans la gestion de l'entreprise. Son rôle est de rendre possible :

La centralisation de l'information : *l'information n'est plus éparpillée dans différents*

fichiers à différents endroits)

L'intégration *(tout ce qui se fait dans un service est visible par d'autres services)*

La diffusion de l'information archivée *(si l'information est disponible à un seul endroit, elle est facile à diffuser)*

Ceci a pour avantages :

d'améliorer la cohérence de l'information *(une seule valeur pour une même information)*
de réduire les redondances *(une même information n'est stockée si possible qu'une seule fois)*
de réduire les efforts de saisie et de mise à jour des informations *(i.e. une information qui doit être stockée une seule fois ne sera saisie qu'une seule fois. De même que sa mise à jour ne se fera qu'une seule fois)*

2.4 La démarche de conception d'une base de données

2.4.1 Principe général

Il est communément reconnu que la conception d'une base de données doit se faire en utilisant une méthode de conception qui définit la démarche à suivre. Plusieurs méthodes de conception existent à cet effet et nous citerons comme exemple la méthode MERISE. Pour certaines méthodes, on dispose même d'un outil logiciel d'aide à la conception appelé aussi un Atelier de Génie Logiciel (AGL) constitué d'un ensemble de logiciels permettant l'automatisation d'un certain nombre de tâches lors des différentes phases du processus de conception (génération automatique de la structure de la B.D., de programmes d'accès et de manipulation, etc.).

Néanmoins, quelle que soit la méthode utilisée, la conception d'une base de données passe par un processus de modélisation permettant de modéliser une certaine partie du monde réel afin de caractériser les *entités* qu'on manipule (étudiants, Comptes Bancaires, Ouvrages, etc.). De plus on essaye de caractériser les *attributs* de ces entités en fonctions des problèmes que doit résoudre l'existence de la B.D. : Gestion de la scolarité, Gestion de Prêt d'ouvrages dans une bibliothèque, etc.

Le cas le plus général est celui où la B.D. est partagée par plusieurs utilisateurs. Ces utilisateurs n'ont pas tous la même vue des données de la base, et n'ont pas tous à voir la base dans sa totalité car chaque utilisateur n'est concerné que par une partie de celle-ci.

Exemple :

Dans une entreprise l'ensemble des informations sur les départements, les employés, les produits, le matériel, etc. peuvent être rassemblées sous forme d'une B.D. et il est bien rare qu'un utilisateur de cette base ait besoin de toutes ces informations à la fois.

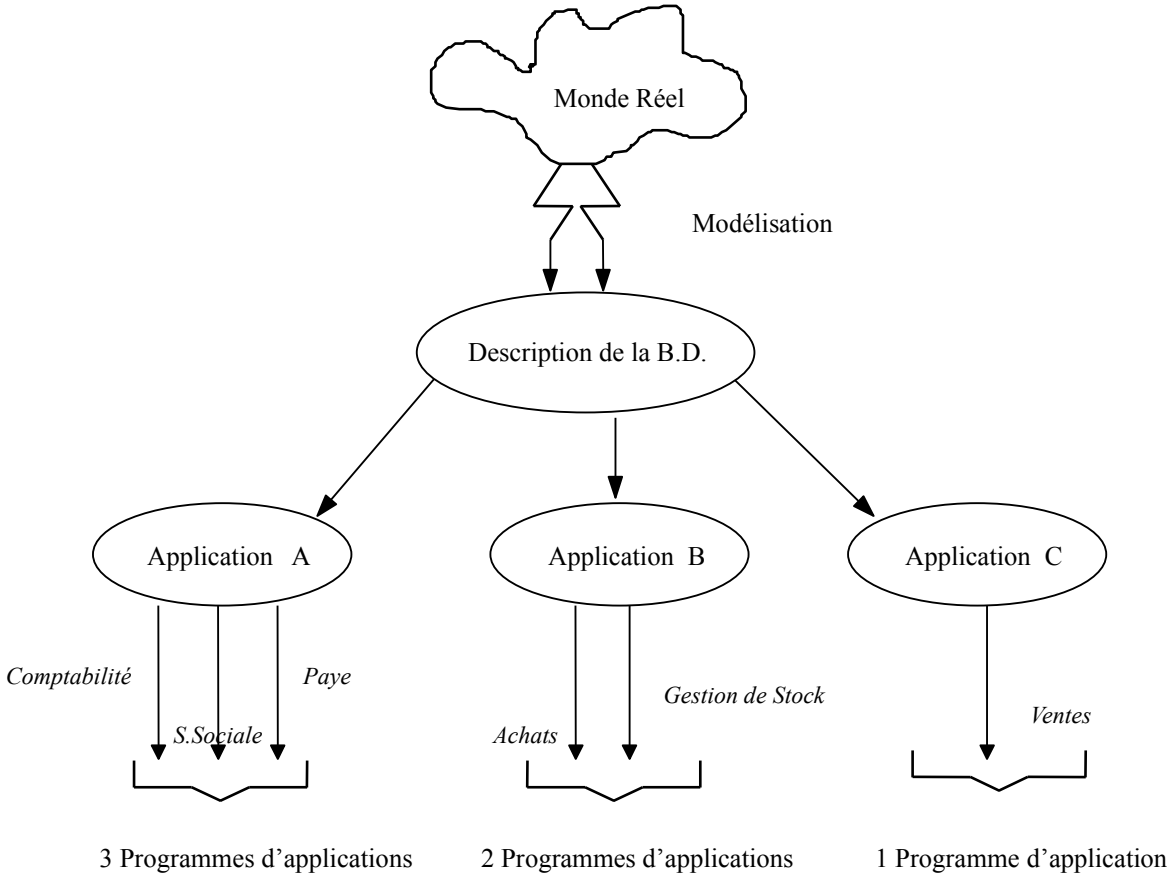

Figure 1.3 : Processus de Modélisation d'une B.D.

2.4.2 Remarque

Il serait plus juste de remplacer le terme *Utilisateur* par celui d'*Application* car on définit en général une B.D. afin de mettre en place une série d'applications ayant chacune ses programmes et ses propres utilisateurs exploitant le même sous-ensemble de données.

Exemple :

Dans une B.D. de Gestion Universitaire, on peut mettre en place une application de gestion de Prêts qui sera composée des programmes P1, P2 et P3 (voire Figure 2) et dont les utilisateurs seront ceux concernés par les traitements réalisés par ces programmes (Secrétaire, Documentaliste, Archiviste, etc.).

2.5 Problèmes posés par la Centralisation de l'information sous forme de B.D.

Dans la pratique, la centralisation de l'information sous forme d'une B.D. unique pose un certain nombre de problèmes liés directement à l'intégrité et à la sécurité de ces informations. Parmi ces problèmes, on peut énumérer les suivants :

2.5.1 Nature et type de l'information

Lors de la mise en place d'une B.D., l'utilisateur décrit les propriétés que doivent vérifier ses données. Ces propriétés peuvent se situer à différents niveaux :

Appartenance d'une donnée à un ensemble de valeurs (Ex : l'âge d'un employé est un entier positif compris entre 0 et 150 0< âge < 150)

Déclaration de propriétés invariantes au cours du temps (Ex : Un enseignant à une heure Donnée ne peut se trouver que dans une seule salle).

Relation d'ordre total à respecter lors du stockage des données (Ex : Les employés doivent être stockés dans la B.D. par ordre croissant de leur numéro ou par ordre alphabétique sur leur nom)

Toutes ces propriétés doivent être préservées tout au long de l'existence de la base de données.

2.5.2 Sûreté physique, sûreté de fonctionnement et point de reprise

Il s'agit de protéger l'information contre un mauvais fonctionnement soit de la machine, soit du système qui géré la base de données.

Dans le premier cas, on peut délimiter les enregistrements qui ont été altérés ou perturbés alors que dans le second cas le problème est beaucoup plus complexe. Une des solutions en usage consiste à prendre à intervalles réguliers des copies de la B.D. et à enregistrer l'ensemble des transactions (opérations) effectuées sur la base. Ceci permettra en cas d'incident de régénérer une copie consistante (i.e. sans défauts) de la base.

2.5.3 Partage de l'information

Lorsque deux programmes P1 et P2 veulent se partager la même donnée A, il peut y avoir perte d'intégrité.

Exemple :

P1 accède à	A	et la transfère dans son buffer propre
P2 accède à	A	et la transfère dans son buffer propre
P1 modifie	A	dans son buffer puis la recopie dans la B.D.
P2 modifie	A	dans son buffer puis la recopie dans la B.D. venant ainsi écraser les modifications faites par P1.

La solution à ce problème serait par exemple celle de l'exclusion mutuelle qui est une technique utilisée dans les systèmes d'exploitation.

2.5.4 Problème des données confidentielles

Il s'agit de protéger les données contre des utilisateurs indiscrets. On dispose en général pour cela de procédures sélectionnant les accès à la base de données. Lorsqu'un utilisateur veut faire un accès, on distingue deux phases :

- *La phase d'identification* : qui a pour but d'identifier l'utilisateur qui veut se connecter à la base de données. Ceci est possible grâce à un mot de passe, une carte spéciale, etc.

- *La phase d'autorisation* : qui après identification de l'utilisateur, permet de déterminer ce que peut faire cet utilisateur sur tel ou telle données (consulter seulement, consulter et mettre à jour, etc.)

3 Systèmes de gestion de bases de données (SGBD)

L'exploitation d'une base de données nécessité la mise en place d'un système informatique appelé communément : *système de gestion de base de données*

3.1 Définition

Un SGBD peut être vu comme un système informatique (un logiciel) spécialisé dans le traitement de gros volumes d'informations et permettant à différents utilisateurs d'interagir avec la base de données.

Un SGBD doit permettre de définir la structure de la base et d'y introduire les données correspondantes. De plus, une fois la base créée, il faudra d'une part la mettre à jour et d'autre part l'exploiter ou l'interroger.

3.2 Fonctions principales d'un SGBD

Un SGBD possède 3 fonctions principales :

1- Fonction Description

2- Fonction Manipulation

3- Fonction Utilisation

3.2.1 Description des données

La modélisation conduit à la définition des entités qui vont constituer les données de la base, de préciser leurs caractéristiques ainsi que les liaisons qui existent entre elles.

Ceci se fait grâce à un langage de description de données ou LDD (Data Definition Langage ou DDL en anglais) offert par le SGBD. Le but essentiel de ces langages est de fournir une indépendance totale des données vis à vis des supports où elles seront stockées (on peut comparer la DATA

DIVISION d'un programme COBOL à la description de la base)

3.2.2 Manipulation des données

Lorsque la structure de la base est décrite, il faut pouvoir stocker les informations correspondantes. Ceci implique nécessairement un certain nombre de mécanismes pour construire des *enregistrements* correctement structurés et qu'il faut ensuite écrire sur un support physique (mémoire secondaire). De plus, il faudra pouvoir accéder à de tels enregistrements pour tous les problèmes de mise à jour et d'interrogation.

Pour ce faire, le SGBD fournit un langage de manipulation de données ou LMD (Data Manipulation Language en anglais). Ce langage est en général construit autour de quelques primitives d'accès et de manipulation (Get, Retrieve, Update, etc.).

3.2.3 Utilisation des données

A ce niveau, on désire par exemple interroger la base de données, c'est à dire rechercher parmi l'ensemble des entités stockées celles qui répondent à des critères de choix très divers. C'est une fonction qui définit directement le lien entre l'utilisateur et les données au sein d'une application.

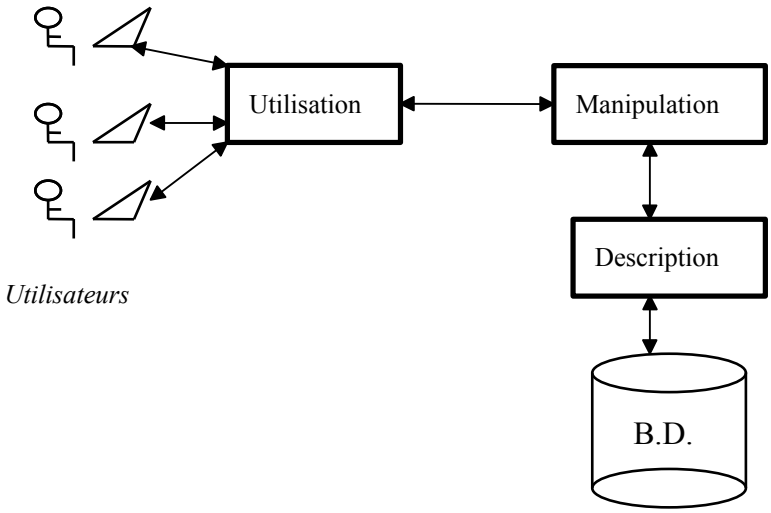

Figure 1.4 : Agencement des fonctions d'un SGBD

A ce niveau, il faut préciser que la mise en place et l'exploitation d'une base de données fait appel à une panoplie d'individus appelés *"Utilisateurs"* ayant chacun un rôle à jouer dans ce processus. Il est donc important de bien définir les différents types d'utilisateurs d'un SGBD et de préciser les rôles de chacun.

3.3 Les différents types d'utilisateurs d'un SGBD

Il s'agit donc de voir les différents rôles que doivent jouer un individu ou un groupe d'individus pour concevoir, créer, mettre en œuvre et exploiter une base de données.

3.3.1 L'administrateur de la base de données

On désigne ainsi la ou les personnes chargées d'établir une description des données constituant la base. Elles sont chargées de décrire les entités de la base de données et indiquer les liaisons existant entre ces entités, ceci au moyen du DDL offert par le SGBD.

Ce travail est l'aboutissement d'une phase d'analyse de l'environnement réel (Entreprise, etc.). Le choix de la structure est primordial pour l'avenir de la base de données car une fois fixée et une fois la base créée, il est très difficile pour ne pas dire impossible dans de nombreux SGBD, de modifier cette structure.

Souvent, on utilise le terme *" Administrateur de l'entreprise"* pour désigner les personnes chargées de la description formelle des données de la base pour souligner l'ouverture vers le monde réel de ce rôle, et on réserve le terme *"Administrateur de la base"* pour désigner les personnes chargées de l'aspect plus technique de la création de la base : choix de l'organisation des fichiers, des structures de mémoires secondaires, des méthodes d'accès aux données, etc.

3.3.2 L'administrateur d'application

Il est chargé de décrire la portion de la base de données concernée par une application particulière. En effet, dans la pratique chaque application n'est concernée que par une portion plus ou moins importante des données de la base. Cette description sera utilisée par les programmes qui vont constituer l'application en question. Ces derniers ne verront donc la base de données que par cette description.

Là aussi, l'administrateur d'application utilise le DDL offert par le SGBD pour décrire la portion de la base de données qui concerne une application donnée (appelée sous schéma ou vue).

3.3.3 Le programmeur d'application

Il est chargé d'élaborer les programmes pour exploiter la base de données en fonction de la description qui a été faite par l'administrateur d'application. Le programmeur d'application utilise le LMD offert par le SGBD ainsi que d'autres sous-programmes conservés généralement dans une librairie (i.e. bibliothèque de sous-programmes).

3.3.4 L'utilisateur

Il s'agit de caractériser ici la personne qui se sert simplement de la base de données et qu'on appelle couramment *l'utilisateur final* (End User en anglais).

Exemple :

Dans une agence de réservation de billets d'avion, la personne qui tape sur son terminal quelques commandes pour effectuer une réservation est une utilisatrice au même titre qu'un chef d'entreprise qui lui aussi demande de temps en temps à une base de données de son entreprise un certains nombre d'informations reflétant l'état de son entreprise (produits non vendus, commandes en attente, etc.).

Ce sont là deux exemples typiques d'un utilisateur final d'une base de données *(End User)*. Leur point commun est qu'ils ne sont pas informaticiens. C'est donc pour cette catégorie là que l'administrateur et le programmeur d'application ont conçu et réalisé des programmes qu'ils n'ont plus qu'à activer au moyen d'un langage de commandes qui devrait être *le plus naturel possible*.

4 Niveaux de description d'une Base de Données

La description d'une base de données peut se faire à différents niveaux, suivant que l'on regarde plus du côté de l'utilisateur que du côté du stockage des données sur les supports physiques. On distingue communément trois (3) niveaux de description d'une B.D. :

1- niveau conceptuel
2- niveau externe
3- niveau interne

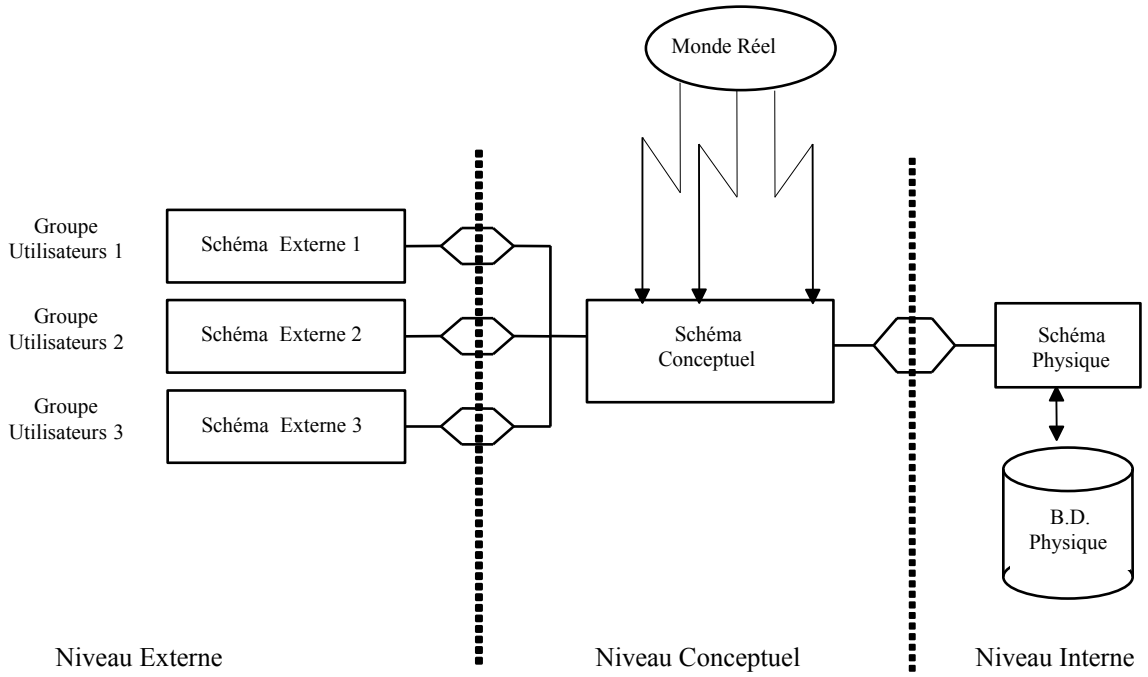

Figure 1.5 : Les niveaux de description d'une base de données

4.1 Niveau conceptuel

Le schéma conceptuel est la partie fondamentale dans l'architecture d'une base de données. Il a pour but de décrire en termes abstraits mais fidèles une certaine réalité d'une organisation et de ses processus de gestion qui ont nécessité la mise en place d'une B.D.

Le passage du monde réel au schéma conceptuel correspond à un processus de modélisation où les objets du monde réel ayant les mêmes caractéristiques sont classés en catégories et désignés par des noms (Etudiants, Véhicules, etc.). Le SGBD fournit un langage de description qui permet de spécifier

le schéma conceptuel.

Dans le processus de modélisation, le concepteur de la B.D. spécifie le schéma conceptuel en utilisant les possibilités offertes par un *modèle de données*.

4.1.1 Définition d'un modèle de données

Un modèle de données est un outil formel destiné à décrire la réalité de manière indépendante de tout traitement informatique.

Un modèle de données doit permettre de regrouper les objets du monde réel auxquels on s'intéresse en classes d'objets de nature identique.

Exemple :

Dans une application de gestion universitaire, on pourra regrouper les étudiants dans une classe d'objets qu'on appellera ETUDIANTS et les modules dans une autre classe d'objets qu'on appellera MODULES. Par la suite on ne fera référence aux objets de ces classes que par l'intermédiaire de ces noms.

Un modèle de données doit aussi permettre de décrire les *liaisons* ou *associations* qui peuvent exister entre les classes d'objets.

Exemple :

Dans une application de gestion universitaire, l'inscription est un phénomène qui associe un objet étudiant appartenant à la classe ETUDIANTS à un ou plusieurs objets modules appartenant à la classe MODULES. On pourra dans ce cas créer une association qu'on nommera INSCRIPTION entre la classe ETUDIANTS et la classe MODULES afin de modéliser cette réalité.

4.1.2 Classification des modèles

Un schéma conceptuel est donc le résultat d'un processus de modélisation fait en respectant les possibilités d'un modèle de données. Le modèle de données est une caractéristique de tout SGBD. Il existe trois grandes classes de modèles de données qui se distinguent par la nature des associations qu'ils permettent de modéliser. Ce sont :

Les modèles hiérarchiques

Les modèles réseaux

Les modèles relationnels

4.1.2.1 Le modèle hiérarchique

A l'aide du modèle hiérarchique, le schéma conceptuel peut être vu comme un graphe arborescent dont les nœuds correspondent aux classes d'objets (entités) et les arcs entre deux nœuds aux liaisons ou associations entre les entités. Un tel graphe possède donc un nœud racine (*sur lequel n'arrive aucun arc !*) et les autres nœuds sont des fils, petit-fils, etc., de cette racine. Avec le modèle hiérarchique, le nombre de flèches pouvant arriver sur un nœud est donc égal à un (sauf pour le nœud racine).

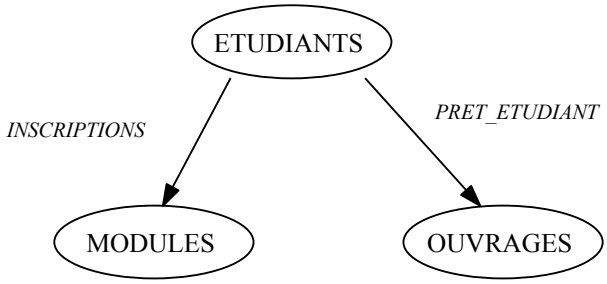

Figure 1.6 : Un exemple de modèle hiérarchique

4.1.2.2 Les modèles réseaux

A l'aide de ce modèle, le schéma conceptuel peut être vu comme un graphe général où les nœuds correspondent aux classes d'objets et les arcs entre deux nœuds aux associations. A la différence du modèle hiérarchique on peut avoir ici plusieurs arcs qui arrivent sur le même nœud. De même que la notion de nœud racine n'existe pas avec le modèle réseau.

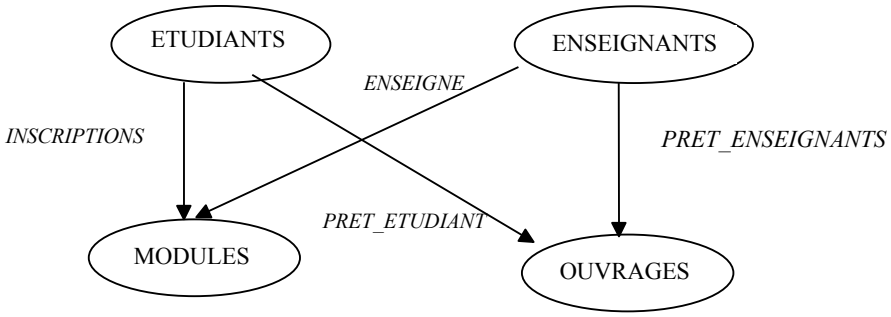

Figure 1.7 : Un exemple de modèle Réseau

4.1.2.3 Le modèle relationnel

Ce modèle est fondé sur la théorie mathématiques des relations. Le schéma conceptuel peut être vu comme un ensemble de tables (ou relations) à n colonnes, n désignant le degré de la relation. Avec le modèle relationnel, une table sert à représenter aussi bien une classes d'objets qu'une association entre des classes d'objets. Ainsi, la distinction entre nœud et arc comme dans les autres modèles n'est pas nécessaire avec le modèle relationnel. Chaque élément d'une table est appelé un n-uplet. Par exemple la table INSCRIPTION décrit l'association entre la classe d'objets ETUDIANTS et la classe MODULES et qui permet de modéliser le fait qu'un étudiant peut s'inscrire à 0, 1 ou plusieurs modules.

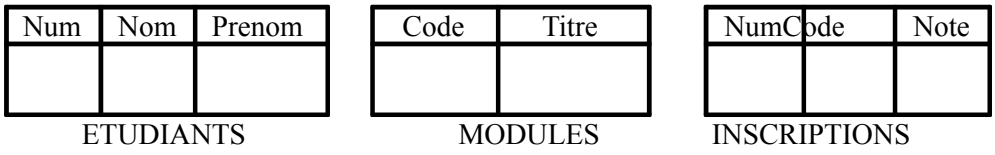

Figure 1.8 : Un exemple de modèle Relationnel

4.2 Niveau externe

Ce niveau correspond à la vision de tout ou partie du schéma conceptuel par un groupe d'utilisateurs concerné par une application. Il s'agit de décrire à l'aide d'un *schéma externe ou Vue* la façon dont seront perçues les données par un programme d'application.

Exemple :

Dans une base de données de gestion universitaire, un groupe d'utilisateurs concerné par les *INSCRIPTIONS* des étudiants n'a pas besoin d'avoir une vision globale de la base et peut se limiter à la partie qui englobe les informations relatives aux *étudiants* et aux *Modules*.

Un schéma externe peut être considéré comme un sous schéma du schéma conceptuel. Grâce à cette notion de schéma externe, chaque groupe d'utilisateurs perçoit les données à sa façon Par exemple, une donnée vue comme donnée numérique par un groupe peut être vue comme une chaîne de caractères par un autre (cas d'une date par exemple). Un groupe peut ne pas voir certaines caractéristiques (attributs) d'une entité (ex : note obtenue dans un module) qui seront par contre visibles par un autre groupe (ou application).

4.3 Niveau interne

Ce niveau a pour but de spécifier comment les données sont stockées sur les supports physiques. Cette spécification est faite par le biais d'un *schéma physique* ou schéma de stockage. Ce schéma permettra par exemple de :

- Décrire la structure des fichiers qui constituent la base de données (nom d'un fichier, organisation, adresse sur le support, etc.)
- Définir les méthodes d'implantation (fichier plat, inversé, etc.)
- Préciser les chemins d'accès aux enregistrements (index, chaînage, calcul d'adresse, etc.)

Cependant, il faut noter que tous ces aspects ne doivent pas affecter les applications sauf sur le plan des performances d'accès à la base afin de garantir l'un des objectifs visés par une base de données à savoir : l'indépendance entre les données et les programmes. Le souci du schéma physique est donc de pouvoir changer l'organisation physique des données sans modifier le schéma conceptuel ni les programmes d'application. Par exemple, pour augmenter les performances d'accès à la base de données, on peut être amené à :

- Changer l'organisation d'un fichier *(Passer par exemple d'une organisation initialement séquentielle à une organisation séquentielle indexée ou directe).*
- Déplacer physiquement le fichier vers une autre adresse sur le support
- Modifier les chemins d'accès aux enregistrements (changer d'index, ajouter d'autres indexes, etc.).

5 Les efforts de standardisation dans le domaine des B.D.

Historiquement, les efforts de développement des SGBD avaient donné naissance à toute une gamme de notions nouvelles qui variaient d'un système à un autre. Plusieurs groupes d'usagers

avaient alors ressenti très tôt le besoin de préciser les principales caractéristiques concernant la terminologie, la syntaxe, la sémantique ainsi que les structures logiques des langages DDL et DML qui devaient être offerts par un SGBD. Ces groupes composés d'usagers d'un même secteur industriel ou de clients d'un même fabricant d'ordinateurs avaient influencé directement ou indirectement les efforts de standardisation et de développement des SGBD. Parmi ces groupes, les plus distingués sont :

- CODASYL (COnference on DAta SYstems Language)
- ANSI (American National Standard Institute)
- GUIDE/SHARE (groupe d'usagers IBM)

5.1 Le groupe CODASYL

Ce groupe a profondément influencé le développement des logiciels SGBD. Il a été formé en 1959 aux USA lors d'une réunion des représentants d'environ quarante (40) organisations comportant des fabricants d'ordinateurs(IBM), des agences gouvernementales, et des usagers importants du matériel informatiques(grandes compagnies).

Ce groupe avait pour mandat de *Concevoir et développer des techniques et des langages pour aider au développement des systèmes informatiques.* Le premier projet du groupe fût de développer les spécifications d'un langage de programmation adapté aux traitements de gestion. Ce fût la naissance du langage *COBOL* (*CO*mmon *B*usiness *O*riented *L*anguage).

A partir de 1966, le groupe se pencha sur les possibilités d'inclure dans COBOL la syntaxe nécessaire pour exploiter les bases de données. Vers 1968, un rapport intitulé *"COBOL Extension To Handle Data Bases"* fût publié par le groupe.

A partir de 1969, la structure organisationnel de CODASYL fût modifiée et trois comités furent crées :

- PLC (Programming Language Committee) dont le mandat était de développer des spécifications pour les langages de programmation.
- SC (Systems Committee) dont le mandat était de développer des langages et des systèmes de très haut niveau pour aider à la conception des systèmes informatiques.
- PC (Planning Committee) dont le mandat était d'informer les usagers sur les objectifs poursuivis et les résultats obtenus. Il fût dissout vers 1973.

Le comité SC publia deux documents dans lesquels sont rapportés les résultats de ses travaux dans le domaine des bases de données. Les intitulés de ces documents sont :

- *"A Survey of Generalized Database Management Systems " (1969)*
- *"A Feature Analysis of Generalized Database Management Systems " (1971)*

Le comité PLC créa un sous-groupe appelé DBTG (*D*ata *B*ase *T*ask *G*roup) ayant pour mission spécifique d'étudier de très près les questions relatives aux bases de données. Le DBTG publia deux rapports importants sur les résultats de ses travaux dans la revue ACM (Association of Computer Machinery). Ce sont :

- *"Data Base Task Group, October 1969 Report, ACM"*. Dans ce rapport, le groupe avait proposé un DDL et un DML formulés comme une extension au langage

COBOL.

- *"Data Base Task Group, April 1971 Report, ACM"* . Suite aux critiques et suggestions reçues sur le premier rapport, le groupe avait publié ce deuxième rapport afin d'exposer toutes les modifications qui ont été apportés au langage.

5.1.1 Propositions de CODASYL

Les comités de CODASYL (surtout le DBTG) avaient fait les propositions suivantes :

a. La définition de la structure logique de l'ensemble des données (base de données) est faite au moyen d'un schéma (schéma conceptuel) et les vues des applications par autant de sous-schémas que d'applications (schéma externe).

b. Un DDL est spécifié pour la description du schéma et des sous-schémas.

c. La structure logique des données est définie en termes de :

- *Donnée élémentaire ou atome (field)* : c'est la plus petite unité d'information possédant un nom
- *Agrégat* : suite consécutive de données élémentaires ayant les mêmes caractéristiques ou une collection de données apparaissant plusieurs fois consécutivement
- *Enregistrement* : collection d'agrégats et d'atomes rangés consécutivement constituant l'unité d'échange entre la base de données et les applications.
- *SETS :* Association entre un enregistrement père (propriétaire ou Owner) et un enregistrement fils (Membre ou Member).
- *AREA :* Un espace physique sur le support qui possède un nom. La base de données sera constituée d'une ou de plusieurs AREA (un ensemble de pages), chacune pouvant recevoir un ensemble d'enregistrements.

Ces propositions ont été beaucoup influencé par le langage COBOL. L'exemple suivant permet de préciser les notions utilisées :

01 EMPLOYE						
02	NOM		PIC	X(20)		*Atome*
02	PRENOM		PIC	X(20)		*Atome*
02	ENFANTS		OCCURS		5 TIMES	
	03	PRENOM	PIC	X(20)		*Agrégat*
	03	AGE	PIC	99		*répétitif*
02	ADRESSE					
	03	NUMERO	PIC	9(4)		*Agrégat*
	03	RUE	PIC	X(30)		*vecteur*
	03	VILLE	PIC	A(20)		

5.1.2 Architecture du SGBD CODASYL

Le DBTG avait proposé une architecture d'un SGBD dont le principe de fonctionnement est le suivant :

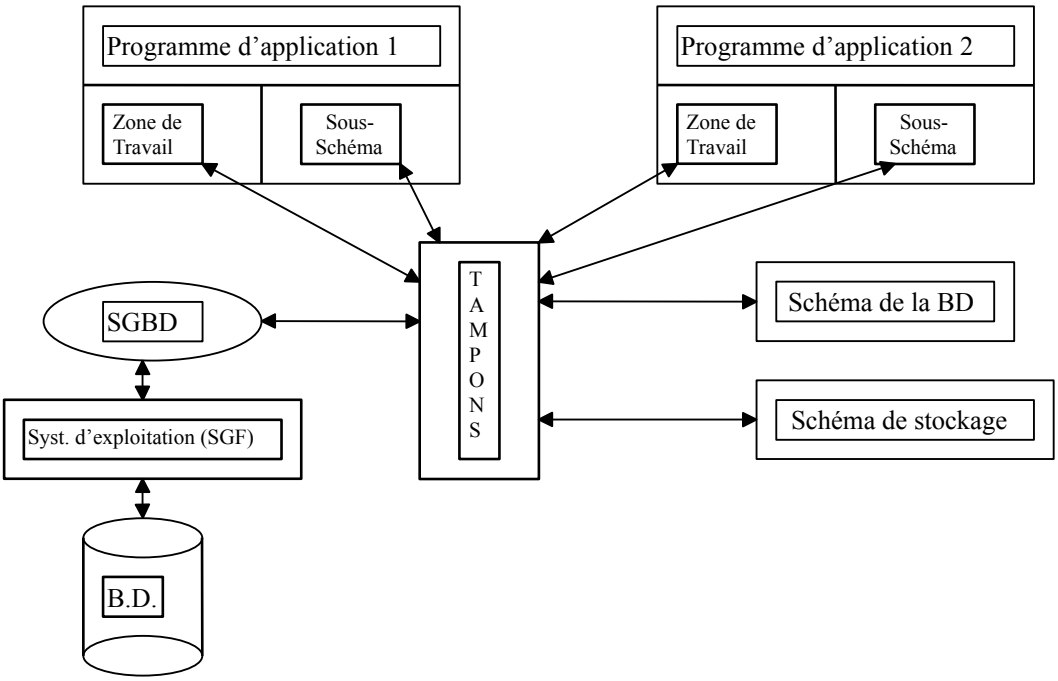

Figure 1.9 : Architecture du SGBD CODASYL

5.2 Le groupe ANSI

Organisme officiel chargé de l'élaboration des normes industrielles aux Etats-Unis. Il créa en 1973 un comité ANSI/X3/SPARC chargé d'étudier les SGBD et les possibilités de normalisation. Ce comité proposa un certain nombre de nouvelles notions qu'il définit ainsi :

1- Administrateur d'entreprise : a une vue d'ensemble des opérations de l'entreprise et de la sémantique des données. Il a la responsabilité globale du contenu et de l'usage de la base de données.

2- Administrateur de la base : a la responsabilité de l'organisation, de la représentation et de l'intégrité de la base de données physique.

3- Administrateur d'application : a la responsabilité de l'exploitation d'une ou de plusieurs applications connexes.

Le comité propose aussi les notions de schéma conceptuel, de schéma externe et de schéma interne. Il propose également un ensemble d'interfaces de type logiciel qui permettent de modifier, consulter, copier et extraire des renseignements du schéma conceptuel et des schémas externes.

5.3 Le groupe GUIDE/SHARE

Composé d'un nombre important d'usagers du matériel IBM. GUIDE représente les usagers de type commercial (entreprises), et SHARE représente les usagers de type scientifique (universités, centres de recherches, etc.).

Le groupe publia en 1971 un rapport s'intitulant : *" The Joint GUIDE/SHARE database Requirment group report"* et dans lequel il avait exposé ses idées sur les approches de développement des SGBD. Il avait aussi initié plusieurs projets de recherches dans le domaine des bases de données. Parmi les thèmes les plus importants qui avaient été traités dans ces projets, on peut citer : l'administration de la base de données et les méthodologies de conception d'une base de données.

6 Mise en œuvre d'un SGBD

La conception, la réalisation et l'utilisation d'un SGBD nécessitent les efforts de nombreuses personnes. Cependant, d'un point de vue externe, les choses peuvent être décrites simplement.

Exemple :

> *un étudiant qui désire s'inscrire se présente à la scolarité. L'employé chargé d'assurer les inscriptions est en général non informaticien et a à sa disposition un terminal lui permettant de communiquer avec la base de données. Il a aussi à sa disposition un ensemble de commandes qu'il doit taper pour afficher les modules en retard, les modules auxquels l'étudiant peut s'inscrire, etc.*

Ainsi, le dialogue entre l'employé et le SGBD paraît simple. Cependant, cette simplicité apparente n'a été rendu possible que grâce au concours de beaucoup d'éléments du SGBD qu'il faudra préciser.

6.1 Le langage de définition de données (DDL)

La première tâche à réaliser est de construire le schéma conceptuel de la B.D. qu'on veut mettre en place. Il est spécifié grâce à un LDD. Ce LDD est propre à chaque SGBD et dépend du type de modèle de données supporté par le SGBD.

Un LDD est un langage descriptif permettant de décrire et de nommer d'une part les classes d'objets que l'utilisateur perçoit dans son application, et d'autre part les associations qui existent entre ces classes d'objets.

Il permet également la description des contraintes d'intégrité, les droits d'accès aux données et les chemins d'accès (indexes, clés, hachage, etc.). Il est utilisé lors de la définition du schéma conceptuel, et aussi pour la définition des différents schémas externes et lorsqu'on veut effectuer certaines modifications du schéma conceptuel.

Dans l'hypothèse ou le schéma externe ferait référence à un modèle de données autre que celui utilisé dans le schéma conceptuel, il serait effectivement nécessaire d'avoir un LDD adapté à ce schéma externe. Dans les SGBD actuellement commercialisés, cette possibilité n'existe pas bien qu'elle puisse probablement devenir une réalité future.

6.2 Le langage de manipulation de données (DML)

Une fois les différents schémas définis, il faut pouvoir créer, mettre à jour, interroger et effectuer toute manipulation que l'on souhaite sur la B.D. On dispose pour cela d'un LMD pour écrire les programmes d'applications. Un LMD a toutes les caractéristiques d'un langage de programmation : schémas conditionnels, schémas itératifs, affectation, etc. Toutefois, il comporte des instructions spécifiques qui lui permettent de référencer les données que l'on souhaite manipuler, et d'exprimer le type de manipulation à effectuer : insertion, mise à jour, suppression, recherche et interrogation.

Dans de nombreux SGBD, ces instructions sont invoquées à partir d'un langage de programmation classique(COBOL, FORTRAN, PL1, etc.) au moyen d'un ordre de type CALL faisant appel à une routine du SGBD qui réalise la fonction supportée par cette instruction. On dit alors que ces langages de programmation sont des langages hôtes.

Par contre, dans certains SGBD, le LMD est entièrement indépendant des langages de programmation et possède sa propre syntaxe. On parle dans ce cas de langage de manipulation de données autonomes. De même que certains SGBD permettent aussi bien l'utilisation du LMD sous forme de langage autonome qu'à partir d'un langage hôte (ex : SQL).

6.3 Exécution d'un programme d'application par le SGBD

Un programme d'application est écrit à partir des connaissances qu'on a sur la base de données, c'est à dire au travers d'un schéma externe puisque l'application ne voit la B.D. qu'à travers un schéma externe. Le SGBD devra interpréter les instructions exprimées en terme du schéma externe, pour les convertir en termes du schéma conceptuel, puis en ordres sur la base de données physique.

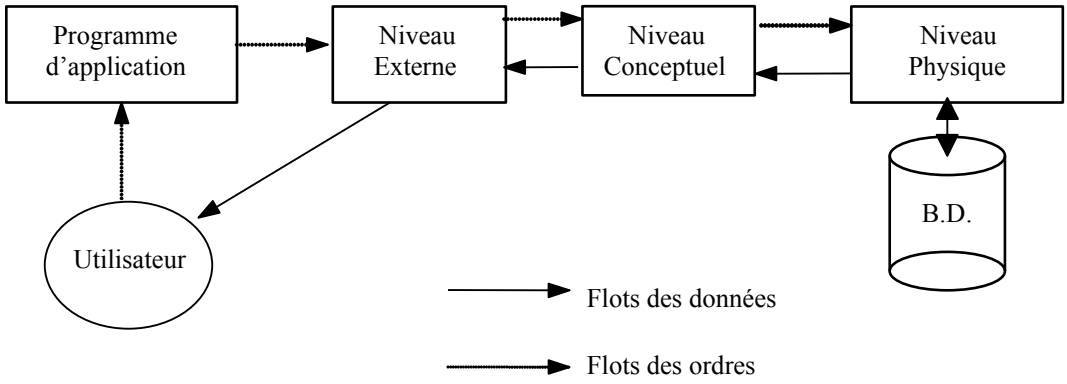

Figure 1.10 : Cheminement des données et des requêtes dans un SGBD

L'architecture d'un SGBD a pour but d'assurer la circulation du flot des ordres et du flot des données. Cette architecture est très diverse selon les différents SGBD (CODASYL ou ANSI). Néanmoins, l'essentiel demeure identique et permet de préciser approximativement le principe de fonctionnement d'un SGBD (Figure 11).

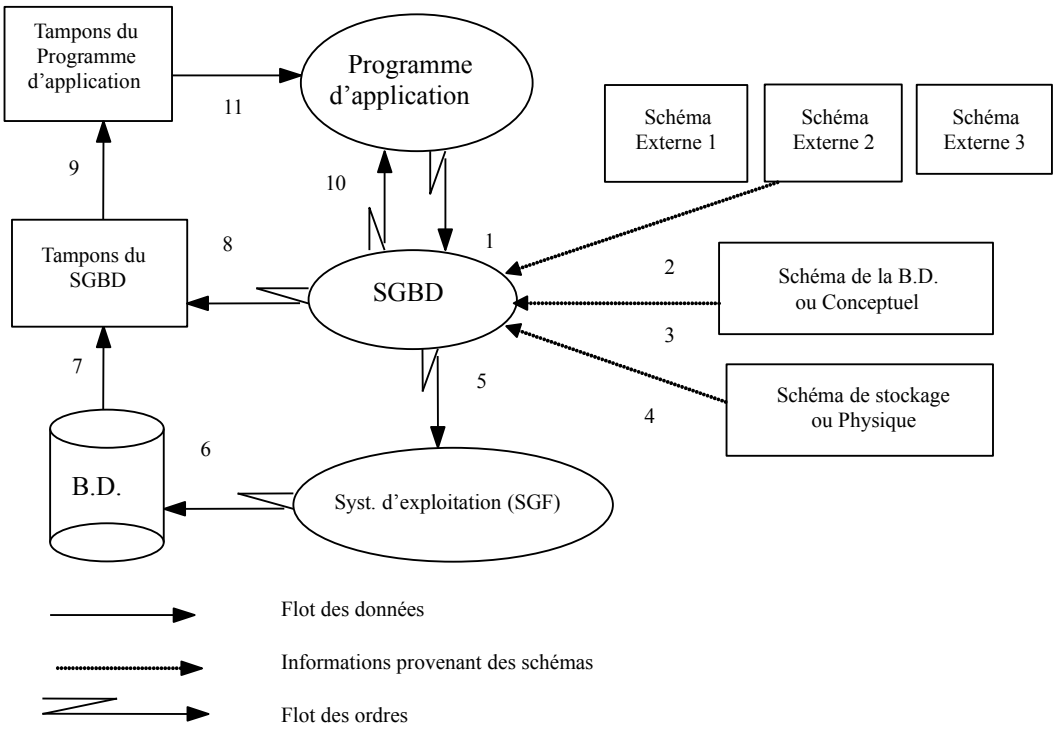

Figure 1.11 : Exécution d'un programme par un SGBD

Cette architecture fait apparaître les éléments principaux suivants :

- Le noyau du SGBD : logiciel qui assure le bon fonctionnement de la base de données
- Le système d'exploitation : logiciel qui assure le fonctionnement de la machine et sur lequel le SGBD a été construit
- Les différents schémas : externes, conceptuel, physique qui sont stockés sous formes de catalogues internes obtenus à partir des descriptions fournies au système grâce au LDD.

6.3.1 étapes de fonctionnement du SGBD (cas d'une lecture de données)

1. La demande de lecture est envoyée par le programme d'application au SGBD (ce qui correspond à la flèche N° 1 sur la figure précédente)
2. La demande est analysée par le SGBD en utilisant le schéma externe propre à cette application. Ceci lui permet de vérifier d'une part que l'application a le droit d'accéder aux données et d'autre part pour extraire et transmettre les caractéristiques de la données à partir de ce schéma.
3. Le SGBD consulte le schéma conceptuel pour déduire le type logique de données à extraire (Table, enregistrement, segment, etc.)
4. Le SGBD consulte le schéma physique pour déduire l'enregistrement physique à lire (Page, Secteur, etc.)
5. Le SGBD transmet à cet effet un ordre de lecture au système d'exploitation (accès à des

pages, blocs, secteurs, etc.)

6. Le système d'exploitation reçoit l'ordre de lecture et l'analyse en consultant certains paramètres du schéma physique puis lance un ordre de lecture au canal ou contrôleur de l'unité périphérique (lire des enregistrements physiques en ignorant le contenu).

7. Les données recherchées sont transmises dans une zone de mémoire qui constitue le tampon du SGBD (tables, enregistrements ou articles, etc.)

8. Le SGBD sélectionne parmi l'ensemble des données reçues dans son tampon seulement celles qui sont nécessaires au programme d'application. Il effectue toute transformation nécessitée par la correspondance schéma externe $\longleftrightarrow$ schéma conceptuel (filtrage des données reçues)

9. Le SGBD transmet ces données dans le tampon du programme d'application.

10. Le SGBD informe éventuellement le programme d'application des déroulements anormaux qui auraient pu se produire lors de l'opération afin que celui-ci réagisse.

11. Le programme d'application dispose de la donnée demandée et peut entreprendre l'opération suivante.

Le principe est le même au cas ou l'opération est une écriture ou une mise à jour. Ce principe concerne uniquement un seul programme d'application. Dans la pratique, il y aura plusieurs programmes d'application qui se déroulent en parallèle. Il appartient au SGBD de les gérer et plus particulièrement de détecter le cas ou différents programmes souhaitent accéder à la même donnée.

7 Approches de développement d'un SGBD relationnel

Il existe deux approches de développement d'un SGBD relationnel :

1. Une approche descendante qui a donné naissance à la première génération de SGBD relationnels

2. Une approche ascendante qui a donné naissance à la deuxième génération de SGBD relationnels

7.1 Approche descendante

Cette approche a été adoptée très tôt pour le développement de nombreux SGBD commercialisés avec succès (DBASE II, III, etc.).

Elle a consisté à concevoir le SGBD comme une interface relationnelle construite autour d'un système de gestion de fichiers existant (SGF). C'est donc au SGF que revient la tâche de gérer les supports physiques où est stockée la B.D.

La correspondance entre les notion de relation (propre au modèle relationnel) et de fichier (propre au SGF) peut être de trois types :

1- Une relation $\longleftrightarrow$ Un fichier : c'est une implantation dite par *" fichier plat"*
2- Un attribut $\longleftrightarrow$ Un fichier : c'est une implantation dite par *" fichier*

transposé"

3- Mixte : c'est une implantation dite par *" fichier virtuel"*

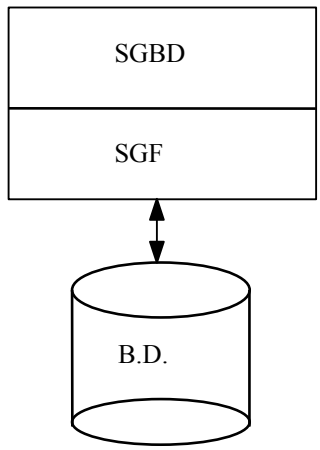

Figure 1.12 : Première approche de conception d'un SGBD/R

7.1.1 Inconvénients

Le SGBD est limité à la fonction de Manipulation des données sans se préoccuper de l'implantation physique. Toute relation est décrite à travers le SGF : l'indépendance physique est artificielle (difficile de changer l'organisation physique pour les soucis de performances d'accès).

C'est une méthode de développement fermée : le développeur est prisonnier des structures physiques des objets du SGF (Si le nombre d'attributs par enregistrement est limité, le nombre d'attributs par relation le sera aussi. Si les enregistrements ont une taille maximale fixe, les tuples de la relation le seront aussi). Le SGBD obtenu n'est donc qu'un SGF aux fonctionnalités étendues ou ce qui revient au même un SGBD aux fonctionnalités réduites.

L'accès à plusieurs relations en même temps est limité du fait qu'il nécessitera l'ouverture d'autant de fichiers que de relations. Ceci est du au fait qu'un SGF ne peut pas permettre pratiquement l'ouverture d'autant de fichiers qu'on le désire.

7.1.2 Avantages

Cette méthode offre l'avantage de pouvoir intégrer dans la B.D. des fichiers utilisés (ou créés) par d'autres applications du fait qu'ils ont été crées par le même SGF. Donc la compatibilité entre les formats rend possible la création d'une B.D. à partir de fichiers générés par d'autres applications (Paie, tableur, facturation, etc.).

Elle rend aussi possible la manipulation de fichiers créés par un SGF avec des opérateurs relationnels en utilisant des moyens de programmation classiques (langages évolués, clipper, etc.). Le niveau interne du SGBD se réduit à une interface très simple avec le SGF puisque le SGBD ne gère pas l'espace physique (un requête se traduira par des ordres de type READ, WRITE, OPEN, CLOSE qui sont caractéristiques de tout SGF).

7.2 Approche Ascendante

C'est l'approche adoptée pour le développement de la plupart des SGBD commercialisés à l'heure actuelle. La différence avec la première est qu'ici le SGBD possède son propre système de gestion de l'organisation physique des données. Ce système est généralement un système de gestion de pages (SGP où 1 page est l'unité de transfert entre la mémoire centrale et le disque ~ 512 Octets).

Avec cette approche, toute la B.D. et son dictionnaire (différents schémas) sont stockés dans un seul fichier système qui est géré par le SGP comme un ensemble de pages (vue comme un ensemble de tables ou relations). Le SGBD est quasiment indépendant du SGF même s'il est possible de les faire communiquer pour tout problème de transfert de données avec d'autres logiciels.

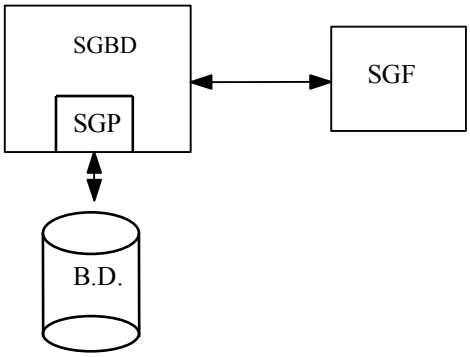

Figure 1.13 : Deuxième approche de conception d'un SGBD/R

7.2.1 Inconvénients

Cette méthode est caractérisée par sa complexité de développement car elle nécessite l'écriture de toutes les fonctions de gestion des supports physiques (i.e. le SGP). A titre indicatif, IBM a mis plus d'une année avec une équipe de 200 personnes (ingénieurs et programmeurs) pour la réalisation du seul niveau interne de son SGBD SYSTEM/R.

De même, des fonctions de conversions SGBD $\longleftrightarrow$ SGF sont nécessaires (Extracteurs) pour permettre le transfert de données dans les deux sens.

7.2.2 Avantages

Avec cette méthode, il n'y a pas de limitation sur le nombre de relations qui peuvent être manipulées en même temps (jusqu'à plusieurs centaines !).

L'accès à une relation ne se fait plus au travers de primitives OPEN et CLOSE de fichiers qui sont propres à un SGF. Le SGBD supporte entièrement les deux fonctions essentielles de description des objets de la base (LDD) et de manipulation (LMD).

Le SGBD ne gère qu'un seul fichier contenant la B.D. et son dictionnaire. Le SGBD est facilement portable vers d'autres environnement car il est indépendant du SGF (il suffit d'adapter le SGP au nouvel environnement). Il est aussi ouvert car il maîtrise bien la structure des objets qu'il utilise d'où la possibilité de l'étendre à d'autres types d'objets (images, sons, documents, etc.).

8 Architecture de quelques SGBD relationnels

Les SGBD relationnels sont à l'heure actuelle les plus diffusés sur le marché. Ils permettent d'organiser les données sous formes de tables. La description de la base de données est faite grâce à un schéma conceptuel ou relationnel permettant de décrire toutes les tables (relations) implantées sur disque. Ce schéma conceptuel servira par la suite à définir un ensemble de schémas externes ou vues décrivant chacun les tables ou relations utilisées par une application. Au schéma relationnel est associé un schéma interne décrivant les chemins d'accès aux tables et les méthodes d'accès (indexes, etc.).

8.1 Le système SQL/DS (Structured Query Langage/Data System)

Ce SGBD est issu d'un projet de recherche lancé par IBM et appelé SYSTEM/R. Il a été initialement commercialisé sur le matériel IBM. Comme on le voit sur la figure, le SGBD est divisé en trois composants :

1- DSC *(Data System Control)*
2- RDS *(Relationnal Data System)*
3- DBSS *(Data Base Storage System)*

DSC *(Data System Control) :* Ce composant assure la communication des usagers avec le système. Il contrôle l'initialisation du système et la terminaison. Il permet aussi la communication des programmes avec le SGBD via des ordres d'appel de type CALL.

RDS *(Relationnal Data System) :* Les requêtes émises par les usagers sont prises en compte par DSC qui les analyse et les transforme de façon à respecter les VUES. Ensuite DSC donne la main à RDS qui les transforme en des appels (les compile) à des modules d'accès aux tables. Il est chargé de déterminer les meilleurs chemins d'accès possibles.

DBSS *(Data Base Storage System) :* Il effectue les accès disques demandés par les modules de RDS. Il assure aussi la gestion de l'espace disque (Allocation, libération,...) et résout les problèmes d'accès concurrents ainsi que la reprise après panne.

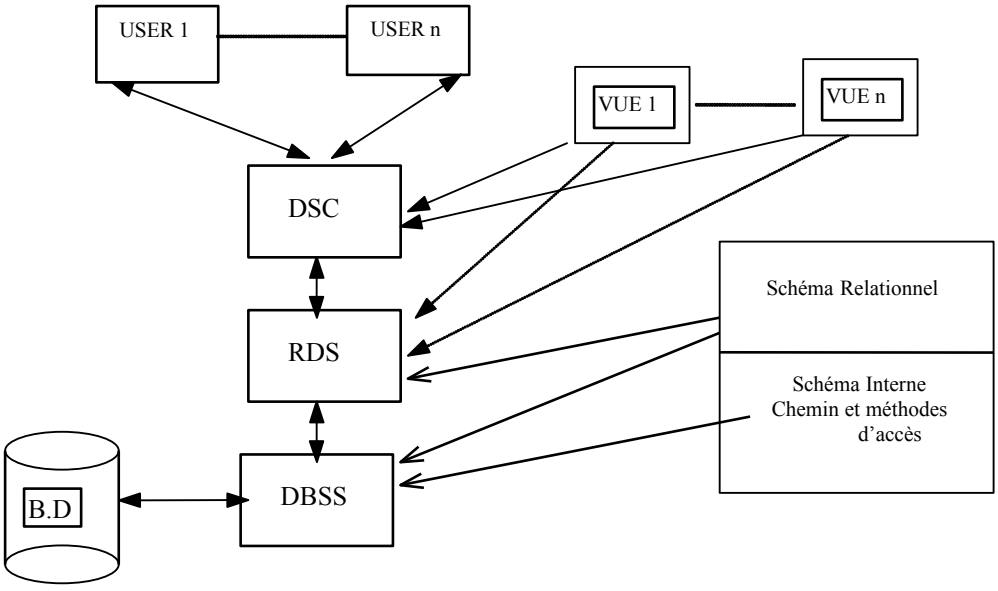

Figure 1.14 : Architecture Fonctionnelle du SGBD SQL/DS (System/R)

8.2 Le système INGRES (IN teractive Graphics & REtrieval System)

Ce SGBD a été réalisé à l'université de BERKLEY par StoneBraker(en 1976). Il a été construit autour du système d'exploitation UNIX et commercialisé initialement sur le matériel de la société Digital Equipment Corporation (PDP et VAX).

INGRES se présente sous forme de 4 composants appelés *Process* (Processus)

- *Process 1 :* Assure la gestion des terminaux. Il permet à l'usager de formuler, imprimer, modifier et lancer l'exécution des commandes (requêtes) offertes par le SGBD.
- *Process 2 :* Effectue l'analyse syntaxique des requêtes, la transformation de ces requêtes afin de respecter les vues. Il assure aussi la protection et le contrôle de cohérence des données ainsi que le contrôle des accès concurrents à la base.
- *Process 3 :* Assure la décomposition des requêtes impliquant plusieurs tables en une suite de requêtes simples portant sur une table unique. En cas de requête impliquant une table unique, il effectue lui-même l'accès aux données.
- *Process 4 :* Assure la création de nouvelles tables, indexes ; de même que les suppressions. En gros, il s'occupe de tout ce qui concerne la gestion de l'espace disque. Il prend également en charge les reprises après panne en retardant les mises à jour. Une mise à jour n'est pas directement répercutée sur la base de données mais jusqu'à la fin du programme.

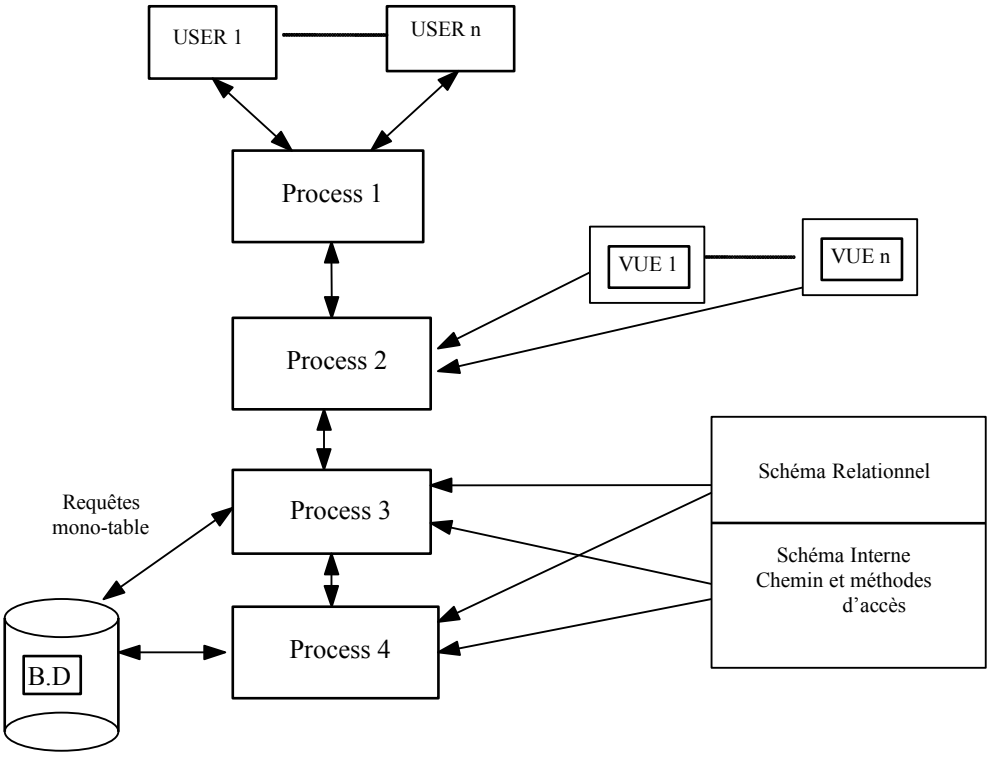

Figure 1.15 : Architecture fonctionnelle du SGBD INGRES

8.3 Le SGBD ORACLE

C'est le SGBD relationnel le plus diffusé actuellement. Il existe aussi bien en version PC et compatibles qu'en version mainframe. Il se présente sous forme d'un ensemble de modules dont un constituant le noyau du SGBD. Depuis sa première diffusion sur le marché (vers 1984), ce SGBD n'a cessé d'être amélioré et plusieurs versions se sont succédées. Si on se réfère par exemple à la version 5 du logiciel, la structure fonctionnelle du SGBD se présente comme suit :

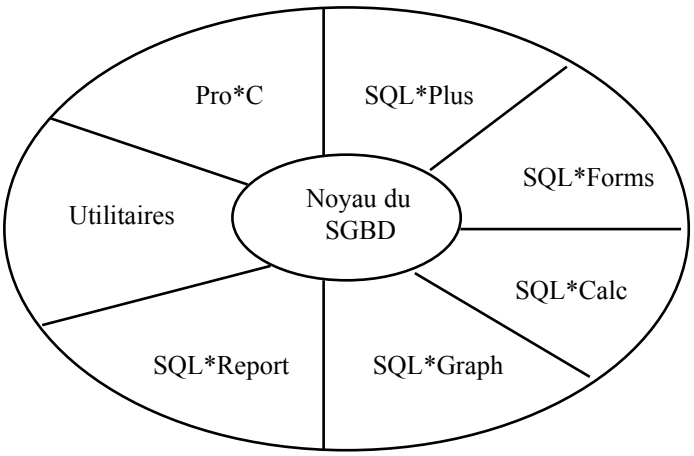

Figure 1.16 : Architecture fonctionnelle du SGBD ORACLE (V5)

On distingue :

- Le Noyau du SGBD : est la couche de base. Il permet la communication avec la B.D. et assure les fonctions suivantes :

  - Contrôle les accès à la base de données
  - Gère les problèmes de confidentialité
  - Assure les reprises après panne
  - Contrôle la cohérence des données

- SQL*Plus : Permet à l'utilisateur d'envoyer des requêtes au SGBD (interrogation, création, mise à jour, etc.)

- SQL*Forms : C'est un générateur d'application interactif (grilles d'écrans, masques de saisie, etc.)

- SQL*Calc : Permet d'associer des requêtes SQL à des cellules d'un tableur intégré à Oracle. le résultat de la requête constituera le contenu de la cellule.

- SQL*Graph : Permet de présenter sous forme graphique des données issues de la base (histogramme, courbe, camembert, etc.)

- SQL*Report : Permet de mettre sous forme de rapports des données issues de la base (Formatage, mise en page, Titre, etc.)

- Pro*C : Précompilateur de programme écrit en C et contenant des requêtes SQL (interrogation, création, mise à jour, etc.). Il existe d'autres précompilateurs disponibles avec le même SGBD mais propre à d'autres langages de programmation (Fortran, Cobol, etc.). Le principe de précompilation est d'analyser lors d'une première passe le programme afin de remplacer les requêtes SQL par des appels à des routines de bas niveau se trouvant dans des bibliothèques du SGBD. Le résultat de la précompilation sera donc un programme source C qui devra passer par les étapes classiques de compilation et d'édition de liens pour enfin donner un programme exécutable.

- Utilitaires : Il en existe une variété, chacun réalisant une fonction précise (Exportation de données vers une autre B.D., Importation de données depuis une autre B.D., allocation d'espace disque supplémentaire pour la B.D., etc.)

L'avantage de ce SGBD est qu'il offre une compatibilité totale entre les versions PC et mainframe (une B.D. créée sur PC peut facilement être transférée sur un gros ordinateur et vice versa).

9 Conclusion

Les SGBD relationnels sont à l'heure actuelle les systèmes les plus diffusés sur le marché. Leur succès est dû au modèle de données sur lequel ils sont basés à savoir : le modèle relationnel. Ce modèle est basé sur des fondements mathématiques (théorie des ensembles et des relations) et offre une approche méthodologique pour la conception d'une B.D. De plus, il faut noter aussi que le succès de ce modèle a été grandement favorisé par l'adoption par l'ISO du langage SQL (Structured Query Language) comme une norme pour les langages de description et de manipulation d'une base de données relationnelle.

Cependant, bien que la supériorité des SGBD relationnels est maintenant reconnue, ceci ne veut pas dire que les SGBD de type réseau ou hiérarchique ne sont plus en usage. Ces SGBD ont aussi évolués avec le temps et certains distributeurs continuent à promouvoir leur produit en allant jusqu'à intégrer dans leur SGBD les concepts du relationnel en arguant que les meilleurs SGBD seront ceux qui intégreront le relationnel et le navigationnel (réseau ou hiérarchique).

Pour mieux comprendre les apports du modèle relationnel dans le domaine des bases de données, nous jugeons utile d'étudier d'abord dans le prochain chapitre les modèles qui l'ont précédé (réseau et hiérarchique) et qui avaient, remarquons-le, fait la vogue pendant longtemps. Ceci nous permettra de familiariser le lecteur avec les concepts de base inhérents à ces modèles et montrer par la même occasion leurs inconvénients majeurs afin de mieux comprendre par la suite pourquoi le modèle relationnel a effectivement pu les surpasser.