SGBD : Système de Gestion de Base de Données

Contexte d'apparition

- Dans les années 60, les premiers systèmes d'exploitation comportaient déjà des primitives d'entrée/sortie (BIOS - basic input ouput system)
- qui permettaient au programmeur de faire abstraction de la gestion physique des données persistantes (au niveau des pistes, blocs ou secteurs, sur les supports de mémoire externe).

- Apparition des Systèmes de Gestion de Fichiers (SGF) qui prennent désormais en charge les mécanismes d'accès aux données organisées en fichiers:

  - **séquentiel,**
  - **relatif,**
  - **indexé,**
  - **séquentiel indexé...**

  - On peut citer, par exemple, les systèmes de fichiers SAM, ISAM, OSAM, HISAM des mainframes IBM

- Les SGF permettent une **relative** «indépendance» entre les programmes et les données.

- Plusieurs programmes (en général conçus à l'intérieur d'une même application) peuvent travailler sur le ou les mêmes fichiers aux conditions suivantes:

  - qu'ils détiennent la même définition du fichier (la description du fichier figurant explicitement à l'intérieur des programmes);

    - très contraignant puisque toute remise en question des structures des fichiers entraîne des modifications au sein des programmes y accédant;

  - qu'ils utilisent le même SGF (généralement incompatibles entre eux); la notion même de structure de fichiers n'est pas uniforme; ceci rend les applications peu portables et difficilement évolutives;

- Les langages supports restent des langages procéduraux pour lesquels la gestion d'application multi-fichiers devient rapidement pénible :
  - programmation explicite des mécanismes d'accès,
  - vérification de nombreuses contraintes d'intégrité,
  - mise-à-jour éclatées induisant des risques d'incohérence…,
- une proportion importante du code des programmes d'application ne se rapporte pas à l'application traitée mais prend en compte les techniques d'accès du SGF sous-jacent;

- **En plus d'un alourdissement des programmes, la programmation devient fastidieuse.**
- De plus, l'informatisation s'effectue de façon verticale (on privilégie l'aspect traitement).
- Le découpage fonctionnel d'un système d'information se traduisait par :
  - la prise en charge des **fonctionnalités** des différents «**services**»
  - Chaque service exploitant des traitements spécifiques sur des **fichiers propriétaires**.

Evolution du contexte informatique

- **L'évolution du contexte informatique se traduit par:**

- **Évolution des mémoire de masse:** amélioration des temps d'accès, augmentation des capacités, diminution des prix.
- **Pour les systèmes et logiciels de base**, apparition de :
  - **accès multiples:** permettant à plusieurs utilisateurs d'utiliser en même temps la même base de données;
  - **mode conversationnel:** permet de rendre la consultation plus facile, moins informatique;
  - progrès dans la **gestion de la mémoire**.
  - Développement continuel des **base de données en réseaux**.

- **Pour les modes de développement d'applications, passage de l'artisanat à la production rationnalisée:**
  - atelier de développement comprenant des possibilités de gestion des programmes, des versions, des documentations, des chaînes de traitement,
  - langages de 4 Génération comportant des générateurs d'écran (form), d'états (report), de menus,
  - outils de maquettage, de prototypage,
  - mécanismes assurant la cohérence du Système d'Information (référentiel, dictionnaire de données),
  - multiples outils (debug, test...).

- Dans les mentalités, passage d'une informatique centralisée, à une informatique ouverte axée sur le service aux utilisateurs:
  - une diversification des besoins et donc des utilisateurs;
  - une volonté d'autonomie des utilisateurs: ceux-ci veulent être considérés comme des partenaires actifs:
  - langage de haut-niveau, non procéduraux;
  - possibilité de faire accéder les différents acteurs aux sous-modèles qui les concernent;
  - environnement informatique hétérogène: machine-logiciels-hommes;

- des méthodes d'analyse et de conceptions de systèmes informatiques qui privilégient désormais une approche horizontale basée sur le **principe de séparation des données et des traitements**
  - les programmes sont conçus autour d'un réservoir d'information (défini indépendamment);
  - cela implique de pouvoir concevoir une base pouvant intervenir dans toutes les applications (potentielles),
    - accessible à de multiples utilisateurs n'ayant pas tous les mêmes besoins, la même culture, les mêmes responsabilités, la même vision des choses.
- Face à ces limitations, les avancées technologiques vont permettre l'émergence du concept de base de données et donc des SGBD

Définition

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

L'objectif des SGBD

- Les bases de données et les systèmes de gestion de bases de données ont été créés pour répondre à un certain nombre de besoins et pour résoudre un certain nombre de problèmes. Leurs principaux objectifs sont les suivants :

1. **g**arantir **l'indépendance** physique et logique entre les données et les programmes d'application,
2. **a**ssurer la **persistance** des données,
3. **p**ermettre une **administration centralisée** des données,
4. **gérer la mémoire** de façon **optimale** et assurer l'efficacité de l'accès aux données,
5. **g**érer le **partage des données** entre utilisateurs et les accès concurrents,
6. **a**ssurer la **fiabilité**, l'**intégrité** et la **cohérence** des données, éviter les redondances,
7. **a**ssurer la **sécurité** des données,
8. **Résistence aux pannes**,
9. **a**ssurer les interrogations **interactives**, la consultation **déclarative**, et l'accès à de **non-informaticiens**.

1. L'indépendance entre les données et les programmes d'application :

- les données et les programmes étant séparés, **ce ne sont plus les applications qui sont chargées de structurer, d'organiser** et de **vérifier** les données et de les **stocker** dans des fichiers.
- Les données sont définies et structurées par le SGBD qui offre cette structure aux applications.
- L'indépendance est **à la fois logique** (au niveau de la définition des données) **et physique** (au niveau du stockage et de la gestion de la mémoire).

  - **Indépendance physique**
    - La façon dont les données sont définies doit être indépendante des structures de stockages utilisées.
  - **Indépendance logique**
    - Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrés dans une vision globale.

2. Le SGBD assure la persistance des données :

- alors que les résultats d'une application sont liés à l'exécution de l'application, l'existence des données d'une base est assurée indépendamment de leur utilisation.
- Les données d'une base **ont donc une durée de vie supérieure** à la durée de vie du programme qui les crée ou qui les utilise.

3. L'administration centralisée des données :

- une des fonctions essentielles des SGBD est la définition des structures de données, la définition et l'organisation des structures de stockage, les modifications de ces structures, les contrôles de validité et de cohérence.
- Il est essentiel de centraliser ces fonctions : c'est l'administrateur du système qui gère l'ensemble des données, indépendamment de leur utilisation et des applications.
- Les usagers ou les applications sont déchargés de toute tâche d'administration, et peuvent utiliser les données de la base sans se préoccuper de l'administration de ces données tout en étant assurés de leur validité et leur cohérence.
- Un SGBD permet d'optimiser l'administration des données en **centralisant ces tâches** dans l'entreprise.

4. Gérer la la mémoire de façon optimale :

- chaque SGBD optimise le stockage et l'accès aux données, en utilisant des techniques complexes inaccessibles aux programmes d'application eux-mêmes.
- En évitant les redondances, il optimise également le volume des données stockées.
- Avec un SGBD, on est donc sûr d'avoir accès à de volumineux ensembles de données de manière efficace car :
  - un SGBD permet d'obtenir des réponses aux interrogations en un temps "raisonnable".
  - un SGBD utilise un mécanisme permettant de minimiser le nombre d'accès disques.
  - Tout ceci, bien sur, de façon complètement transparente pour l'utilisateur.

5. Gérer le partage des données entre usagers :

- Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment.
- Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations et quand on est dans un contexte mono-utilisateur, cela n'est plus le cas quand il s'agit de modifications dans un contexte muti-utilisateurs.
  - **Il s'agit alors de pouvoir :**
    - permettre à deux (ou plus) utilisateurs de modifier la même donnée "en même temps" ;
    - assurer un résultat d'interrogation cohérent pour un utilisateur consultant une table pendant qu'un autre la modifie.
- c'est le SGBD qui va se charger de gérer les accès concurrents, les mises à jour, le tout en fonction des droits de chacun.
- Chaque usager aura l'illusion d'être seul à utiliser les données de la base, qu'il verra assemblées et structurées comme il le souhaite

6. Assurer la fiabilité, l'intégrité, la cohérence :

- la description du schéma des données **contient un certain nombre de règles** qui permettent de vérifier la fiabilité, l'intégrité, la cohérence de tous les objets présents dans la base

  par exemple, l'âge d'une personne doit être compris entre 0 et 120; l'âge d'un enfant ne peut être supérieur à celui de ses parents, etc..
- Le SGBD contient des procédures spéciales permettant d'effectuer l'ensemble de ces tests.
- Par exemple, les contraintes d'intégrité sont des règles qui précisent la cohérence sémantique des données entre elles.
  - Les données sont soumises à un certain nombre de contraintes d'intégrité qui définissent un état cohérent de la base.
  - Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression des données.
- Eviter la redondance des données

– Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.

7. Sécurité des données :

- tous les utilisateurs d'une même base n'ont pas nécessairement les mêmes droits d'accès aux données.
- Il est utile de pouvoir **définir de tels droits** pour assurer la sécurité de certaines données confidentielles et réservées à certains groupes d'utilisateurs.

8. Résistance aux pannes

- Que se passe-t-il :
  - si une panne survient au milieu d'une modification,
  - si certains fichiers contenant les données deviennent illisibles?
- Les pannes, bien qu'étant assez rares, se produisent quand même de temps en temps.
- ▶ Il faut pouvoir, lorsque l'une d'elles arrive, récupérer **une base dans un état "sain".**
- – Ainsi, après une panne intervenant au milieu d'une modification deux solutions sont possibles :
  - – soit récupérer les données dans l'état dans lequel elles étaient avant la modification,
  - – soit terminer l'opération interrompue.

9. Assurer les interrogations interactives et l'accès aux non-informaticiens :

- les données de la base sont bien gérées et bien organisées.
- Il ne reste plus qu'à les consulter sans avoir à se préoccuper du stockage ou de l'organisation interne.
- Le SGBD offre un certain nombre de fonctions pour accéder aux données de façon déclarative,

  aussi bien pour les programmes d'applications que pour l'usager,

  sans avoir à effectuer **le moindre développement** informatique particulier.

Malheureusement, ces objectifs ne sont pas toujours atteints !!! .

Composants des SGBD

- Un système de gestion de bases de données va donc posséder un certain nombre de composants logiciels et ce, quelque soit le modèle de données qu'il supporte.
- On trouve donc des composants chargés de :

❑ **La description des données**

- Cette partie sera constituée des outils (en gros des langages) permettant de décrire la vision des données de chaque utilisateur et l'intégration dans une vision globale. On y trouve aussi les outils permettant de décrire le stockage physique des données.

❑ **La récupération des données**

- Cette partie prend en charge l'interrogation et la modification des données et ce, de façon optimisée. Elle est composée de langages de manipulation de données spécifiques et d'extensions de langages "classiques". Elle gère aussi les problèmes de sécurité.

❑ **La sauvegarde et la récupération après pannes**

- Cette partie comporte des outils permettant de sauvegarder et de restaurer de façon explicite une base de données.
- Elle comporte aussi des mécanismes permettant, tant qu'une modification n'est pas finie, de pouvoir revenir à l'état de la base avant le début de cette modification.

❑ **Les accès concurrents aux données**

- C'est la partie chargée du contrôle de la concurrence des accès aux données.
- Elle doit être telle que chaque utilisateur attende le moins possible ses données tout en étant certain d'obtenir des données cohérentes en cas de mises à jour simultanées de la base.

Fichiers et structures de données nécessaires au fonctionnement d'un SGBD

- des **fichiers contenant les données** entrées par les utilisateurs,
- un fichier créé par le SGBD, le **dictionnaire des données**, qui contiennent des informations relatives aux données stockées par les utilisateurs (méta-données),
- des fichiers créés par le SGBD, **les index**, qui permettent d'accélérer les recherches d'informations dans les BD,
- des **données statistiques** :
  - **relatives aux données elles-mêmes**
    - » taille des fichiers,
    - » nombre de valeurs différentes par attribut, …..
  - **mais aussi aux opérations effectuées sur ces données**
    - » (nombres d'insertions, suppressions et modifications ;
    - » fréquences de recherche sur les attributs ….

Fonctions principales d'un SGBD

- Un SGBD doit permettre :
  - de définir la structure de la base
  - d'y introduire les données correspondantes.
  - Et une fois la base créée, il faudra d'une part la mettre à jour et d'autre part l'exploiter ou l'interroger.
- Un SGBD possède **donc 3** fonctions principales :

1- Fonction Description
2- Fonction Manipulation
3- Fonction Utilisation

1 Fonction : 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.
- Pour ce faire, le SGBD fournit un langage de description de données ou **LDD** (Data Definition Langage ou DDL en anglais)
  - 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…

- Le **LDD** est propre à chaque SGBD et dépend du type de modèle de données supporté par le SGBD.
- C'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.**

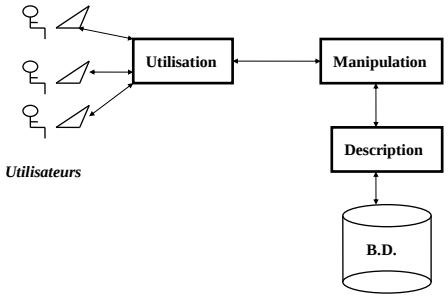
2- Fonction : Manipulation des données

- **Lorsque la structure de la base est décrite, il faut pouvoir stocker les informations correspondantes.**
  - Ceci nécessitera des 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).

- 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'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.**
- 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).

3- Fonction : Utilisation des Données

- A ce niveau, il s'agira d'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.



- Le terme "Utilisateurs" fait référence à une panoplie d'individus ayant chacun un rôle à jouer dans le processus de mise en place et d'exploitation d'une BD .
- Parmi les différents types d'utilisateurs d'un SGBD qu'il est important de distinguer on peut citer :

▶L'administrateur de la base de données

- 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**.
- 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.

▶L'administrateur d'application

- Il est chargé de décrire la portion de la base de données concernée par une application particulière.
- 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 programmes ne verront la base de données que par cette description.

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**).

▶Le programmeur d'application

- 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).

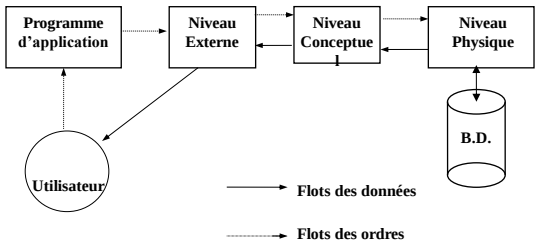
▶L'utilisateur final (End User)

- Représente la personne qui se sert simplement de la base de données
  - Dans une agence de réservation de billets d'avion, la personne qui tape sur son terminal quelques commandes pour effectuer une réservation est un utilisateur final
  - De même qu'un chef d'entreprise qui 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.).
- **Le point commun des utilisateurs de ce type 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.

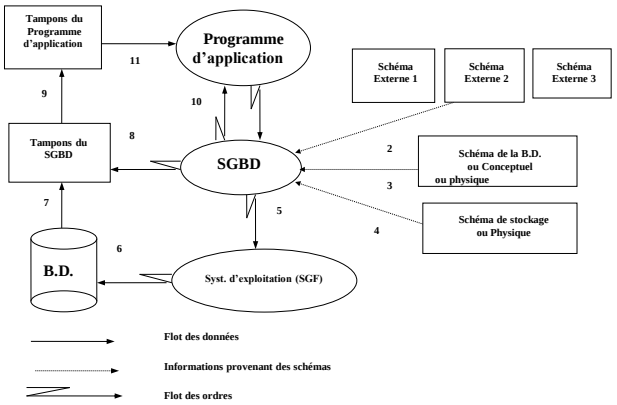
Exécution d'un programme d'application par le SGBD

- 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 (vue)
- Lapplication 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.

- 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



❖ **Le principe de fonctionnement d'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.

Etapes de fonctionnement du SGBD : cas d'1 écriture

- **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.)

Etapes de fonctionnement du SGBD : cas d'1 écriture

- **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.)

Etapes de fonctionnement du SGBD : cas d'1 écriture

- **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 ←→ 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.**

Historique de l'évolution des SGBD

- Historiquement, plusieurs générations de systèmes se sont succédées. Les premières applications informatiques nécessitant les fonctions d'un SGBD furent les applications de gestion : gestion de stocks, comptabilité, paye,etc.
-
- Les deux premières générations de SGBD (réseaux et relationnels) ont donc été conçues à la mesure de ces applications.

❖ **Les années 70 :**

- Les systèmes de première génération, "réseaux" ou "hiérarchiques«
- Les travaux du DBTG (CODASYL), puis la norme ANSI (1981) ont normalisé le modèle de données et les systèmes qui le supportent.

- Dans ces systèmes, les données sont modélisées par des structures de type graphe (modèle réseau) ou arbre (modèle hiérarchique).
  - Les systèmes **IDS II**, **IDMS** et **Adabas** font partie des systèmes CODASYL diffusés à ce jour.
  - Avec **IMS** et son modèle dit hiérarchique, **IBM** est le seul acteur important à ne pas avoir respecté cette norme.

✍ **La "navigation"** constitue le principal inconvénient de ces systèmes :

- on manipule les données en suivant des pointeurs physiques vers l'information recherchée.
- Les programmes sont donc **dépendants de l'organisation physique** des fichiers sur le disque et les applications doivent donc être profondément modifiées à chaque réorganisation physique.

➢ Les années 80 : les systèmes relationnels

- Le modèle relationnel représente toutes les informations sous forme de tables et offre quelques opérations simples pour manipuler ces tables.
- La simplicité du modèle en tant qu'interface externe a permis la définition de langages de requêtes simples, faciles d'utilisation et puissants, ainsi que de méthodes de conception d'applications basées sur des méthodologies systématiques.
- La technologie des SGBD relationnels est parvenue à un degré de standardisation et de maturité tel que les systèmes sont difficilement différentiables tant en fonctionnalités qu'en performances.
- Le langage d'interrogation SQL a fait l'objet d'une norme et constitue un standard de fait auquel aucun vendeur ne peut maintenant échapper.

❖ Les années 90 : les systèmes orientés-objet

- A partir des années 80 apparaissent de nouvelles applications (CAO, PAO, productique, multimédia, génie logiciel, etc.) **nécessitant l'utilisation de bases de données.**
- Ces nouvelles applications ont mis en évidence les insuffisances majeures des systèmes relationnels :
  - l'inadéquation du modèle relationnel à représenter directement des données complexes, comme
    - ✓ des dossiers médicaux structurés, ou multimédia,
    - ✓ des images radiographiques ou des textes annotés,
  - les performances insuffisantes dans la manipulation de données complexes,
  - la pauvreté graphique et ergonomique des outils d'interface homme-machine.

- Les SGBD orientés-objet sont la réponse directe aux problèmes des nouvelles applications. Ils répondent au problème de pauvreté du modèle en proposant un plus riche et extensible.
- Ils offrent de bonnes performances pour manipuler les données complexes.
- Ils offrent l'intégration complète des concepts de la programmation par objet et des bases de données.
- Les vertus de la programmation par objet sont largement reconnues :
  - puissance des concepts,
  - programmation modulaire,
  - réutilisation du code (notamment par le biais de boites à outils)
  - et maintenance aisée du code.
- La productivité du programmeur en est considérablement améliorée.

- La première apparition du concept date de 84 avec la proposition de David Maier et George Copeland de **construire un SGBD à partir de Smalltalk** .
- A partir de ce prototype, Servio Logic construit un produit commercial, **Gemstone**, mis sur le marché en 88.
- Parallèlement, Ontologic, réalise un produit, **Vbase** , mis sur le marché à peu près au même moment et retiré de la vente un an plus tard pour de nombreuses raisons.
- Dans le laboratoire de Hewlett Packard, un projet de prototypage d'un **SGBD de type fonctionnel**, IRIS, démarré en 83 évolue progressivement vers un produit de type **SGBD orienté-objet**.

- Parallèlement, en Europe, la communauté scientifique s'intéresse aux SGBD permettant de stocker des objets à structure complexe et de nombreux prototypes voient le jour.
- Deux grands projets de recherche en SGBD orientés-objet débutent: le projet ORION à Austin, Texas en 85, et le projet Altaïr, à Rocquencourt, France en 86,
- Des start-up : Object Design, Versant et Objectivity sont créées aux USA en 89, ainsi que **O2 Technology** en France en 90.
- Ces compagnies **mettent des produits sur le marché en fin 90 et début 91**.