Introduction aux Systèmes de Gestion de Bases de Données Relationnelles

– Olivier Losson –

L'objectif de ce cours est l'acquisition des connaissances fondamentales relatives aux systèmes de gestion de bases de données relationnelles (SGBDr), de la conception du schéma de données à la construction du modèle relationnel et à l'exploitation des données.

La première partie présente dans ses grandes lignes la notion de SGBDr ainsi que les principes fondamentaux du modèle relationnel. La seconde partie se focalise sur la construction du modèle de données, en détaillant comment le modèle entité-association permet d'obtenir un premier schéma relationnel et comment ce dernier est normalisé pour obtenir une base cohérente. La troisième partie concerne la phase d'exploitation des données, et porte en particulier sur le langage de sélection et de manipulation de données SQL.

- Sommaire -

I. Vers un système de gestion des données	2
I.1. Introduction.....	2
I.2. Objectifs et propriétés d'un SGBD	2
I.3. Composants et organisation de la mise en œuvre d'un SGBD.....	3
I.4. Niveaux et modèles de description des SGBD	4
I.5. Principes fondamentaux du modèle relationnel.....	5
II. Construire un modèle de données.....	6
II.1. Phases de construction.....	6
II.2. Le modèle entité-association	6
II.3. Passage au modèle relationnel.....	8
II.4. Nécessité de la normalisation dans la conception d'une BDr.....	9
II.5. Dépendances fonctionnelles et normalisation des relations.....	10
II.6. Intégrité structurelle.....	13
III. Exploiter les données	15
III.1. Bases de l'algèbre relationnelle	15
III.2. Le langage SQL (Structured Query Language).....	17

I. Vers un système de gestion des données

I.1. Introduction

I.1.1. Quelques jalons historiques

- 1960 Uniquement des systèmes de gestion de fichiers plus ou moins sophistiqués
- 1970 Début des SGBD réseaux et hiérarchiques proches des systèmes de gestion de fichiers
L'ouvrage de Codd paru en 1970, "Un modèle relationnel pour les grandes banques de données partagées", jette les bases théoriques du concept de BDr
- 1980 Les SGBDr font leur apparition sur le marché
- 1990 Les SGBDr dominant le marché et apparaissent les SGBD orientés objets (SGBDOO)

I.1.2. Qu'est-ce qu'une base de données ?

De façon informelle, on peut considérer une **Base de Données** comme un ensemble structuré de données, centralisées ou non, servant pour les besoins d'une ou plusieurs applications, interrogeables et modifiables par un groupe d'utilisateurs en un temps opportun.

Plus formellement, une BD est un ensemble d'informations *exhaustives, non redondantes, structurées et persistantes*, concernant un sujet.

Une condition nécessaire pour mériter le titre de base de données est, pour un ensemble de données non indépendantes, d'être **interrogeable par le contenu** (pouvoir retrouver toutes les données qui satisfont un critère quelconque).

Un **Système de Gestion de Base de Données** peut être défini comme un ensemble de logiciels prenant en charge la structuration, le stockage, la mise à jour et la maintenance des données. Autrement dit, il permet de décrire, modifier, interroger et administrer les données. C'est, en fait, l'interface entre la base de données et les utilisateurs (qui ne sont pas forcément informaticiens).

I.2. Objectifs et propriétés d'un SGBD

I.2.1. Objectifs d'un SGBD

Un SGBD doit résoudre certains problèmes et répondre à des besoins précis :

- **Indépendance physique** : la façon de définir les données doit être indépendante des structures utilisées pour leur stockage
- **Indépendance logique** : un utilisateur doit pouvoir percevoir seulement la partie des données qui l'intéresse (c'est ce que l'on appelle une *vue*) et modifier la structure de celle-ci sans remettre en cause la majorité des applications
- **Manipulation aisée** des données par des non informaticiens, ce qui suppose des langages "naturels"
- **Accès efficaces aux données** et obtention de résultats aux interrogations en un temps "acceptable"
- **Administration centralisée** des données pour faciliter l'évolution de leur structure
- **Non-redondance** : chaque donnée ne doit être présente qu'une seule fois dans la base afin d'éviter les problèmes lors des mises à jour

- **Cohérence (ou intégrité)** : les données ne doivent présenter ni ambiguïté, ni incohérence, pour pouvoir délivrer sans erreur les informations désirées. Cela suppose un mécanisme de vérification lors de l'insertion, de la modification ou de la suppression de données
- **Partage** des données pour un accès multi-utilisateur simultané aux mêmes données. Il faut entre autre assurer un résultat d'interrogation cohérent pour un utilisateur consultant une base pendant qu'un autre la modifie
- **Sécurité** des données : robustesse vis-à-vis des pannes (il faut pouvoir retrouver une base "saine" en cas de plantage au cours de modifications) et protection par des droits contre les accès non autorisés

1.2.2. Propriétés d'un SGBDr

Des objectifs cités ci-dessus découlent les propriétés fondamentales d'un SGBDr :

- **Base formelle** reposant sur des principes parfaitement définis
- **Organisation structurée** des données dans des tables interconnectées (d'où le qualificatif *relationnelles*), pour pouvoir détecter les dépendances et redondances des informations
- Implémentation d'un **langage relationnel ensembliste** permettant à l'utilisateur de décrire aisément les interrogations et manipulation qu'il souhaite effectuer sur les données
- **Indépendance des données** vis-à-vis des programmes applicatifs (dissociation entre la partie "stockage de données" et la partie "gestion" - ou "manipulation")
- **Gestion des opérations concurrentes** pour permettre un accès multi-utilisateur sans conflit
- **Gestion de l'intégrité des données**, de leur protection contre les pannes et les accès illicites

1.3. Composants et organisation de la mise en œuvre d'un SGBD

1.3.1. Composants des SGBD

Un SGBD va donc posséder un certain nombre de composants logiciels chargés de :

- **la description des données** au moyen d'un *Langage de Définition de Données* (LDD). Le résultat de la compilation est un ensemble de tables, stockées dans un fichier spécial appelé dictionnaire (ou répertoire) des données
- **la manipulation des données** au moyen d'un *Langage de Manipulation de Données* (LMD) prenant en charge leur consultation et leur modification de façon optimisée, ainsi que les aspects de sécurité
- **la sauvegarde et la récupération après pannes**, ainsi que des mécanismes permettant de pouvoir revenir à l'état antérieur de la base tant qu'une modification n'est pas finie (notion de *transaction*)
- **les accès concurrents aux données** en minimisant l'attente des utilisateurs et en garantissant l'obtention de données cohérentes en cas de mises à jours simultanées

Une **architecture Client-Serveur** met en œuvre un ou plusieurs ordinateurs (Clients), qui exécutent un programme applicatif, communiquant avec un ordinateur distant (Serveur) qui traite leurs requêtes. Dans ce modèle, la communication des requêtes et des résultats entre le programme applicatif du client et le SGBDr, qui se trouve sur le serveur, est assurée par une couche logicielle médiatrice (Middleware).

1.3.2. Organisation de la mise en œuvre des BD

Les données constituent aujourd'hui une ressource vitale et stratégique pour l'entreprise. Les besoins concernent autant les données propres que celles de l'extérieur, avec pour objectif de sécuriser, maintenir et mettre à jour ces informations. Quatre catégories de fonctions sont impliquées dans cette *gestion de données* :

- les tâches liées à l'**architecture de données** consistent à analyser, classifier et structurer les données au moyen de modèles confirmés
- l'**administration de données** vise à superviser l'adéquation des définitions et des formats de données avec les directives de standardisation et les normes internationales, à conseiller les développeurs et les utilisateurs, et à s'assurer de la disponibilité des données à l'ensemble des applications. L'administrateur assume en outre des responsabilités importantes dans la maintenance et la gestion des autorisations d'accès
- les professionnels en **technologie de données** ont en charge l'installation, la supervision, la réorganisation, la restauration et la protection des bases. Ils en assurent aussi l'évolution au fur et à mesure des progrès technologiques dans ce domaine.
- l'**exploitation de données** consiste à mettre à disposition des utilisateurs les fonctions de requête et de reporting (générateurs d'états), ainsi qu'à assurer une assistance aux différents services pour qu'ils puissent gérer leur stock propre de données en autonomie (service *infocentre*).

I.4. Niveaux et modèles de description des SGBD

I.4.1. Niveaux de description

Trois niveaux de description (ou d'abstraction) des données sont définis (norme ANSI/SPARC) :

- **Niveau interne** (ou **physique**) : concerne le stockage des données au niveau des unités de stockage, des fichiers; c'est ce que l'on appelle le *schéma interne*
- **Niveau conceptuel** (ou **logique**) : décrit la *structure* des données dans la base, leurs propriétés et leurs relations, indépendamment de toute préoccupation technologique d'implémentation ou d'accès par les utilisateurs; c'est ce que l'on appelle le *schéma conceptuel*
- **Niveau externe** : décrit comment chaque utilisateur perçoit les données; c'est ce que l'on appelle le *schéma externe* ou *vue*.

I.4.2. Modèles de description

Le résultat de la conception d'une base de données est une description des données (appelée *schéma*) en termes de propriétés d'ensembles d'objets et d'organisation logique des données. Pour obtenir une telle représentation à partir d'un problème réel, on utilise un outil appelé **modèle de description**, basé sur un ensemble de concepts et de règles formant le langage de description. Un SGBD peut être caractérisé par le modèle de description qu'il supporte. Une fois la base de données ainsi spécifiée, il est possible de manipuler les données en réalisant des opérations de sélection, d'insertion, de modification et de suppression, et ce au moyen d'un langage spécifique de manipulation de données, mais aussi par des langages de programmation "classiques". Plusieurs types de modèles sont utilisés :

- **modèles réseau** et **hiérarchique** : une base de données est vue comme un ensemble de fichiers reliés par des pointeurs (ces modèles privilégiant l'optimisation des entrées-sorties, ils sont nommés *modèles d'accès*) et les données y sont organisées en réseau et en arbre (respectivement). Un nœud quelconque peut avoir plusieurs pères dans le modèle réseau, un seul dans le modèle hiérarchique
- **modèle relationnel** : les données et les relations qui les unissent sont organisés sous forme de lignes regroupées au sein de tables. Chaque table est composée d'une ou plusieurs colonnes, dont certaines servent à caractériser chaque ligne de manière unique (notion de *clé*, voir section I.5 ci-dessous)
- **modèle objet** : nés dans le prolongement de la programmation orientée objets (POO), les SGBDOO permettent de représenter et de manipuler des objets complexes. Leur avantage est la réduction des coûts de développement et de maintenance, ainsi qu'une performance accrue, puisque la base

"connaît" la structure objet du programme (relations d'héritage, pointeurs entre objets, etc.). On distingue :

- les systèmes d'encapsulation d'objets, surcouches logicielles de bases relationnelles intégrant le code d'interfaçage pour le stockage d'objets. Ces systèmes permettent de bénéficier des gains de productivité des bases objets, mais non des performances
- les bases objets proprement dites sont totalement fondées sur le modèle objet. Elles ne nécessitent plus de langage spécifique de manipulation de données (comme SQL), car la base utilise directement un langage objet (comme C++, Smalltalk ou Eiffel), en implémentant une extension propre aux mécanismes des BD (verrouillage, transactions, intégrité référentielle, ...).

1.5. Principes fondamentaux du modèle relationnel

1.5.1. Table (ou Relation) et Attribut

Une table (ou relation) est un tableau à deux dimensions dans lequel chaque colonne (appelée *attribut*) porte un nom différent et où les données figurent en ligne. Aussi bien pour les colonnes que pour les lignes, une table peut en contenir un nombre quelconque et leur ordre est indifférent. On trouve encore les vocables d'*enregistrement*, *n-uplet* ou *tuple* pour désigner une ligne et de *champ* pour une colonne ; un enregistrement est donc un ensemble de valeurs, chacune renseignant un champ.

ETUDIANTS		
N°Etu	Nom	Promo
034582	Durand	DeugSM
147230	Caspar	DeugSM
128745	Coupet	LEEA
04371	Durand	

Figure 1 : Exemple de relation

1.5.2. Clé

Dans la table de l'exemple ci-dessus, le numéro d'étudiant est un attribut constitué d'un code artificiel. Cet attribut particulier, appelé *clé*, est indispensable pour pouvoir identifier chaque étudiant de manière unique, car aucun autre attribut ni aucune combinaison d'attributs n'auraient pu garantir cela. En effet, l'attribut *nom* peut prendre plusieurs fois la même valeur et n'est donc pas un bon candidat pour être une clé de la table ; de même, si l'on avait disposé du prénom des étudiants, la composition d'attributs $\langle \text{nom}, \text{prénom} \rangle$ n'aurait pas été davantage satisfaisante comme clé, car il est possible que 2 étudiants portent à la fois le même nom et le même prénom.

Une *clé d'identification* ou *clé* d'une table est un attribut ou une combinaison d'attributs qui satisfait :

- une contrainte d'*unicité* : chaque valeur clé désigne de manière unique une ligne de la table ; autrement dit, deux lignes ne peuvent posséder des valeurs identiques pour le(s) attribut(s) de la clé
- une contrainte de *minimalité* : si une clé est composée d'un ensemble d'attributs, cette combinaison doit être minimale (aucun attribut ne peut en être retiré sans violer la règle d'unicité).

Si'il y a plusieurs clés envisageables pour une même table, on choisit parmi celles-ci une *clé primaire*.