Chapitre I : Introduction

I. Qu'est-ce qu'une base de données ?

I. 1. Introduction

Une base de données est un ensemble d'informations structurées stockées sur un support en vue de son traitement automatique. Son rôle principal est d'assurer leur pérennité mais surtout de faciliter leur exploitation. Une base de données modélise une situation réelle à laquelle elle doit être fidèle selon un certain niveau d'abstraction. Son objectif principal est de permettre l'accès à une information précise dons une masse importante d'informations.

Elle est constituée de relations représentées sous forme de tables à deux entrées dont les colonnes sont des attributs de la relation et les lignes des enregistrements ou n_uplets (n étant le nombre d'attributs de la base). Elle est modélisée et administrée via un système de gestion de base de données (SGBD).

I. 2. Instance et schéma d'une base de données

I. 2. 1. Schéma d'une base de données

La structure d'une base de données bien conçue change très peu durant son cycle de vie. Elle est constituée de relations dont chacune est décrite par des attributs, une clé primaire et éventuellement une/des clé(s) étrangère(s). Chaque attribut possède un domaine dans lequel il prendre toutes ses valeurs. Ces informations (nom de relation, attributs, domaines des attributs, clé primaire, clés étrangères) décrivent le schéma d'une relation. L'ensemble des schémas des relations qui composent une base de données donne le schéma de cette base de données.

I. 2. 2. Instance d'une base de données

Le contenu d'une base de données change constamment durant son exploitation. Des mises à jour sont fréquemment faites par les utilisateurs. Chacune d'elles exécutée avec succès sur une table modifie son instance. Il est donc nécessaire de connaitre ce que contient chaque table à chaque instant. L'instance d'une table est constituée de l'ensemble des n_uplets qui s'y trouvent à un instant donné. L'instance d'une base de données est constituée de l'ensemble des instances de ses tables à un instant donné.

II. Qu'est-ce qu'un Système de Gestion de Bases de Données (SGBD) ?

II. 1. Introduction

Un SGBD est une application permettant de gérer des bases de données. Pour ce faire il contient des modules permettant d'implémenter des bases de données, de les sécuriser, et de les

manipuler. L'exploitation d'une base consiste en grande partie à y effectuer des mises à jour (insertion, modification, suppression) et à y chercher des informations respectant un ou plusieurs critères de recherche. Un SGBD est composé essentiellement de trois grandes parties appelées gestionnaires que sont : le gestionnaire de fichiers, le gestionnaire de transactions et le gestionnaire de requêtes.

II. 2. Les composants d'un SGBD

II. 2. 1. Le gestionnaire de fichiers

Le rôle du gestionnaire de fichiers est d'assurer la gestion des données sur les supports de stockage de masse (disque dur, bande magnétique, CD, etc.). Il permet de sauvegarder les données sous forme de tables avec des lignes (n_uplets) et des colonnes (attributs). Il donne la possibilité de lier les différentes tables pour permettre l'implémentation des liens entre elles. Il assure donc la création et la modification de la base et de ses tables en plus de l'enregistrement (sauvegarde) des données, de leurs mises à jour et de leur recherche sur les supports de stockage. Il gère également la restriction à l'accès des données suivant les privilèges des utilisateurs. Il est le seul gestionnaire capable d'accéder aux données enregistrées sur le support de stockage.

II. 2. 2. Le gestionnaire de transactions

II. 2. 2. 1. Qu'est-ce qu'une transaction ?

Une transaction est une suite d'instructions regroupées dans un objet fermé par une instruction de validation et vérifiant les propriétés **ACID** (**A**tomicité, **C**ohérence, **I**solation, **D**urabilité).

- ✓ **Atomicité :** Toutes les instructions contenues dans une transaction doivent être entièrement exécutées ou totalement annulées. C'est la loi du tout ou rien.
- ✓ **Cohérence :** Une base de données doit toujours être cohérente. Ainsi l'exécution d'une transaction doit laisser la base de données dans un état cohérent.
- ✓ **Isolation :** Les modifications apportées à la base par les instructions d'une transaction en cours ne doivent être visibles par les autres transactions qu'après validation de celle-ci. Ainsi, une donnée en cours de modification ne doit pas être visible ni modifiable.
- ✓ **Durabilité :** Lorsqu'une transaction est exécutée et validée, les modifications apportées par ses différentes instructions sur la base doivent y rester jusqu'à ce qu'une nouvelle instruction y apporte d'autres modifications.

Remarque : Si une partie des instructions d'une transaction est exécutée et que pour une raison ou une autre le système n'arrive pas à exécuter les autres, les effets de ces opérations doivent être annulés.

II. 2. 2. 2. Rôle du gestionnaire de transactions

Le rôle du gestionnaire de transactions est d'assurer l'exécution correcte des transactions lancées par les utilisateurs. Il assure leur exécution en veillant à ce que les différentes tâches de chacune d'elles suivent le bon ordre, tout en vérifiant le respect des propriétés ACID. Il coordonne aussi l'exécution des différentes transactions lancées simultanément.

II. 2. 3. Le gestionnaire de requêtes

Il sert d'intermédiaire entre les données sauvegardées dans la base et les utilisateurs. Il traduit les requêtes de création, d'interrogation et de mise à jour exprimées par les utilisateurs dans un langage de haut niveau (SQL par exemple) en une séquence d'opérations basiques sur les informations enregistrées dans la base. Ces requêtes sont souvent intégrées dans des programmes applicatifs pouvant accéder à la base.

III. Dépendance fonctionnelle

Il y'a dépendance fonctionnelle entre deux attributs d'une entité si à chaque valeur de l'un correspond une et une seule valeur de l'autre.

Par exemple connaissant le nom du bâtiment et le numéro d'une salle, on connaît sa capacité, si oui ou non elle est climatisée, son type (Salle de TP, TD, etc.), etc.

Bâtiment, Num_Salle → Capacité

Bâtiment, Num_Salle → Climatisé

Bâtiment, Num_Salle → Type.

L'ensemble contenant la/les propriété(s) de gauche (Bâtiment, Num_Salle) est appelé ***source*** et celui contenant la/les propriété(s) de droite (Capacité, Climatisé, Type) est appelé ***but***. Une dépendance fonctionnelle dont la source est composée d'un seul attribut est appelée DF **simple**. Si la source est composée de plusieurs attributs on a une dépendance fonctionnelle **composée**.

III. 1. Dépendance directe

Une dépendance fonctionnelle X → Y entre deux attributs X et Y d'une relation R est dite directe s'il n'existe pas un attribut Z de R tel que les dépendances fonctionnelles X → Z et Z → Y soient correctes.

III. 2. Dépendance élémentaire

Une dépendance fonctionnelle est dite élémentaire si sa source ne contient pas d'attribut superflu. Si X, Y → Z et X → Z alors Y est superflu dans la dépendance X, Y → Z qui n'est donc pas élémentaire.

Remarque :

$$\begin{cases} X \rightarrow Y \\ X \rightarrow Z \\ X \rightarrow W \end{cases} \quad ==> X \rightarrow Y, Z, W$$

IV. Avantages et inconvénients des bases de données relationnelles

Les bases de données relationnelles jouent un rôle très important dans le traitement automatique des données. Elles permettent de structurer et sauvegarder les données d'un système d'information pour faciliter leur exploitation. Pour ce faire, les données sont stockées sous forme de relations liées entre elles. Cette structuration permet de trouver les données satisfaisant une condition dans un ensemble d'information assez volumineux. Une base de données peut être centralisée ou répartie sur plusieurs sites différents. L'administrateur doit veiller à ce qu'il n'y ait ni redondance, ni incohérence, ni perte de mise à jour. Il doit également mettre en place les politiques de sécurité tout en faisant de sorte d'avoir les meilleures performances possibles. Le SGBD doit pouvoir autoriser l'accès simultané aux informations par plusieurs utilisateurs tout en sauvegardant la cohérence et en assurant la sécurité de la base.

IV. 1. Les avantages des bases de données relationnelles

Les bases de données relationnelles ne se sont pas imposées sans raison comme une norme de facto dans le traitement électronique des données. Les points suivants s'appliquent au modèle de bases de données relationnelles.

- ✓ **Modèle de données simple :** les bases de données relationnelles sont basées sur un modèle de données relativement facile à mettre en œuvre et à gérer. De nombreuses informations, telles que les données clients, les listes de commandes ou les mouvements de compte que les entreprises souhaitent stocker pendant une longue période peuvent idéalement être mappées avec la structure de relation sous forme de table sur laquelle le modèle de bases de données relationnelles est basé.
- ✓ **Faible redondance des données :** le modèle relationnel des bases de données fournit des règles précisément définies pour éviter la redondance avec les différentes formes

normales. Si les spécifications de normalisation sont appliquées de manière cohérente, les systèmes de bases de données relationnelles permettent un stockage des données pratiquement sans redondance. Cela facilite la gestion et l'entretien des stocks de données, puisque les modifications ne doivent être effectuées qu'à un seul endroit.

- ✓ **Cohérence élevée des données :** les bases de données relationnelles normalisées permettent un stockage cohérent des données et contribuent ainsi à la cohérence des données. Les systèmes de bases de données relationnelles offrent également des fonctions permettant de définir et de vérifier automatiquement les conditions d'intégrité. Les transactions qui mettent en danger la cohérence des données sont exclues.
- ✓ **Traitement des données quantitatives :** le système de bases de données relationnelles est basé sur un traitement des données quantitatives dans lequel chaque entité est décomposée en valeurs atomiques. Cela permet de lier différentes entités via le contenu ainsi que des requêtes de base de données complexes telles que les jointures.
- ✓ **Langage de requête uniforme :** pour écrire des requêtes en bases de données relationnelles, le langage SQL standardisé par un comité de ISO et IEC a été établi. Le but de cette standardisation est de permettre aux applications développées de s'exécuter en grande partie indépendamment du système de gestion de base de données sous-jacent. Cependant, le support SQL varie encore considérablement d'un SGBD à l'autre.

IV. 2. Inconvénients des bases de données relationnelles

Selon le scénario dans lequel les systèmes de bases de données relationnelles sont utilisés, des avantages tels que le modèle de données simple basé sur plusieurs tables liées peuvent également être interprétés comme des inconvénients. En outre, les caractéristiques centrales du modèle des bases de données relationnelles sont difficiles à concilier avec les exigences modernes de programmation des applications (telles que l'orientation objet, le multimédia et les données de grande taille).

- ✓ **Affichage tabulaire des données :** tous les types de données ne peuvent pas être compressés dans un schéma rigide de tables bidimensionnelles liées. Les types de données abstraites et les données semi ou non structurées qui se produisent dans le cadre d'applications multimédia et de grandes solutions de données ne fonctionnent pas avec le modèle des bases de données relationnelles.
- ✓ **Pas de schéma de base de données hiérarchique :** contrairement aux bases de données orientées objet, les bases de données relationnelles n'offrent pas la possibilité d'implémenter des schémas de base de données avec des classes hiérarchiquement

structurées. Des concepts tels que les entités subordonnées qui héritent de propriétés d'entités supérieures ne peuvent pas être implémentés avec elles. Par exemple, on ne pout pas créer de sous-tuples. Tous les tuples d'une base de données relationnelle se trouvent au même niveau hiérarchique.

- ✓ **Segmentation des données :** le principe de base des SGBDR, qui consiste à diviser l'information en tables séparées (normalisation), conduit inévitablement à une segmentation des données. Tout ce qui est connexe n'est pas nécessairement stocké ensemble. Cette conception de base de données entraîne des requêtes complexes sur plusieurs tables au niveau de l'application. Le nombre plus élevé de segments interrogés qui en résulte a généralement un impact négatif sur les performances.
- ✓ **Performances inférieures à celles des bases de données NoSQL :** le modèle de bases de données relationnelles impose des exigences élevées en matière de cohérence des données, ce qui est préjudiciable à la vitesse d'écriture des transactions.

V. Comment concevoir une base de données ?

La conception d'une base de données est une étape très importante et délicate qui demande beaucoup d'attentions et de créativité. Elle commence par la mise en place du cahier des charges s'il n'existe pas. Le cahier des charges contient les informations importantes fournies par le client et qui doivent être prises en compte tout au long de la conception de la base. C'est à partir du cahier des charges, que le schéma de la base est créé.

V. 1. Mise en place du cahier des charges

Le cahier des charges est le document contenant les informations qui seront stockées dans la base, les traitements qui s'y porteront, les utilisateurs qui y accéderont, les privilèges de chacun d'eux, les règles de gestion, etc. Pour cela des interviews sont nécessaire car toutes les informations du cahier des charges doivent venir du client et de ses collaborateurs. Le concepteur de la base se chargera d'exploiter ces informations pour créer le schéma de la base de données. Les principales questions que l'on peut poser lors de l'interview sont :

- Quelles sont les données que l'on doit enregistrer dans la base ?
- Quels sont les traitements qui doivent porter sur les données ?
- Quelles sont les règles de gestion de la structure concernée ?
- Où est ce que les données seront stockées ?
- Qui est ce qui pourra accéder aux données de la base ?

- Pour chaque utilisateur, quels sont les traitements qu'il peut lancer et sur quelle(s) partie(s) de la base ?
- Etc.

V. 2. Analyse des données

Elle consiste à faire une étude approfondie du contenu du cahier des charges pour structurer les données selon leurs caractéristiques et leur importance. Cette structuration se fait de manière descendante. On commence par identifier les objets les plus significatifs qui sont généralement des entités, ensuite leurs attributs, les domaines des attributs, l'identifiant de chaque entité, etc. Les répétitions de données ainsi que les incohérences sont à éviter au maximum. L'identifiant de chaque entité est choisi et les liens entre les différentes entités sont établis. Ces liens permettent d'inter-relier les différentes entités pour faciliter la recherche d'informations basée sur plusieurs entités.

V. 3. Conception du schéma de la base de données

Après une bonne analyse des informations contenues dans le cahier des charges, les entités, leurs attributs, leurs identifiants, les associations qui les lient sont clairement identifiés. Le modèle Entité/Association est donc complété avec le choix des cardinalités des associations et de leurs attributs éventuels.

Le modèle Entité/Association n'est pas directement implémenté dans le SGBD. Il est transformé en modèle relationnel en appliquant les règles de passage du modèle entité-association au modèle relationnel. Elles utilisent principalement les associations entre entités et leurs cardinalités. Le schéma de la base de données est obtenu en ajoutant les domaines des attributs au modèle relationnel.

VI. Architecture d'une base de données

Une base de données est manipulée le plus souvent par les utilisateurs via des interfaces utilisateurs d'applications informatiques. Les utilisateurs accèdent ainsi au serveur de données qui peut être très distant du site où ils se trouvent. Les composants logiciels peuvent être exécutés par une même machine ou par plusieurs machines différentes.

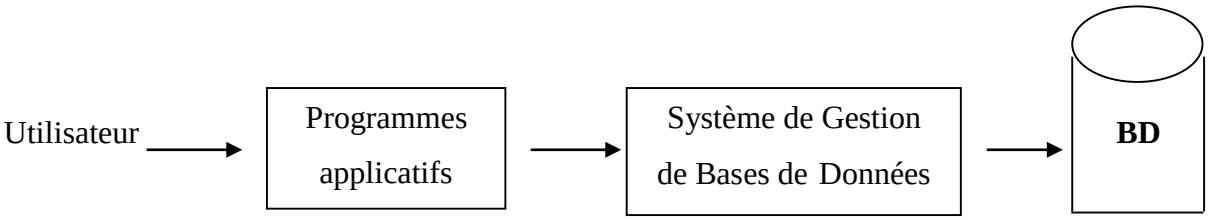

Figure 1 : Environnement d'utilisation d'une base de données

VI. 1. Architecture 2-tiers

On parle d'architecture 2-tiers si une seule machine (un seul serveur) se charge de l'exécution complète des demandes des utilisateurs. Elle sert alors de serveur de données et de serveur de traitement. Les logiciels (programmes applications) qui accèdent à la base sont exécutés par la machine sur laquelle la base de données est implémentée.

VI. 2. Architecture 3-tiers

Dans une architecture 3-tiers, le serveur de données est distinct du serveur de traitement. L'utilisateur envoie sa requête au serveur de traitement, qui à son tour passe par le serveur de données pour obtenir les informations sur lesquelles doivent porter les traitements.