Architecture des SGBD relationnels
Et stockage des données

Introduction

Nous nous intéressons dans ce chapitre a l'architecture des SBBD ainsi qu'à la couche du SGBD qui accède à la BD, c'est-à-dire au système de stockage appelé la mémoire relationnelle.

Ainsi nous aborderons les rappels concernant les bases de données et les SGBD avant de présenter les architectures les plus répandues.

Enfin, la fonction d'un SGBD relative au stockage des données sera présentée. Une base de données est constituée, matériellement, d'un ou plusieurs fichiers volumineux stockés sur un support non volatile. Le support le plus couramment employé est le disque magnétique qui représente un bon compromis en termes de capacité de stockage, de prix et de performance. Il y a deux raisons principales à l'utilisation de fichiers. Tout d'abord il est courant d'avoir affaire à des bases de données dont la taille dépasse de loin celle de la mémoire principale. Ensuite, et c'est la justification principale du recours aux fichiers, même pour des bases de petite taille, une base de données doit survivre à l'arrêt de l'ordinateur qui l'héberge, que cet arrêt soit normal ou dû à un incident matériel.

En première approximation, une relation peut être vue comme un fichier, et chaque tuple correspond à un enregistrement. Les relations sont découpées et stockées par page ou bloc de taille fixe en mémoire secondaire (MS) lesquelles sont amenées en mémoire centrale (MC) à la demande grâce à des mécanismes de mémoire virtuelle.

L'espace de stockage en MS est logiquement divisé en segments. Un segment est un espace virtuel de stockage qui se compose physiquement de plusieurs pages. L'accès à des données stockées sur un périphérique, par contraste avec les applications qui manipulent des données en MC, est une des caractéristiques essentielles d'un SGBD. Elle implique notamment des problèmes potentiels de performances puisque le temps de lecture d'une information sur un disque est considérablement plus élevé qu'un accès en MC.

L'organisation des données sur disque, les structures d'indexation mises en œuvre et les algorithmes de recherche utilisés constituent donc des aspects importants des SGBD.

Un bon SGBD doit minimiser les temps d'accès et l'espace de stockage.

Dans ce chapitre, on abordera l'aspect matériel lié au stockage, nous aborderons ensuite les aspects organisation et structure d'index.

I- Les concepts de bases de données et de SGBD :

1- Le Concept de Bases de données

a) Historique : Au début de l'informatique par les entreprises, une approche au coup par coup (approche classique cartésienne) a été utilisée et qui consistait à définir des systèmes indépendants dotés chacun de ses propres programmes et de sa propre collection de données du monde réel. Cette collection de données était structurée sous forme de fichiers. Chaque système ou application avait son ou ses fichiers que l'on traitait séparément. Les difficultés suivantes sont apparues :

- Etant autonomes, ces systèmes ne représentent pas la façon dont l'entreprise travaille, c'est-à-dire comme un ensemble complexe de systèmes interdépendants (liés) entre eux.
- Les informations obtenues à partir du fichier ne fournissent pas une image complète de la réalité.
- Les fichiers utilisés par différentes applications vont amplifier la redondance des données (grande difficulté pour maintenir l'intégrité des données)

- Grande difficulté pour permettre un accès simultané aux données en temps réel par plusieurs utilisateurs (Insuffisance dans la gestion des données en mémoire secondaire)

Parallèlement aux limitations concernant l'utilisation des fichiers, d'une part l'amélioration des capacités et techniques de stockage part des mémoires plus grandes et plus souples, et d'autre part l'utilisation de techniques d'accès plus performantes, ont fait évolué les exigences des utilisateurs. Ceux-ci ne se contentaient plus de l'automatisation d'applications routinières et autonomes mais exigeaient des :

- Informations complètes
- Informations structurées
- Informations cohérentes
- Informations disponibles rapidement par plusieurs utilisateurs.

La solution était dans l'apparition des Bases de Données (1962-1963) et de leur système de gestion de BD.

b) Définition d'une Base de Données (BD) : Une BD est un ensemble d'informations exhaustif, structuré, non redondant, enregistré sur des supports accessibles par un ordinateur, pour satisfaire simultanément plusieurs utilisateurs de façon sélective.

Exemple1 : Une BD de gestion de l'activité d'une compagnie aérienne concernant les voyageurs, les vols, les avions, le personnel, les réservations, etc. Une telle BD pourrait permettre la gestion des réservations, des disponibilités des avions en fonction des vols à effectuer, des affectations des personnels volants, etc.

Exemple 2 : La BD d'un institut universitaire va contenir des informations sur les étudiants, les enseignants, les modules, …., ainsi que sur les associations qui existent entre ces informations : les modules enseignés par un enseignant, les notes obtenues par un étudiant dans un module etc. Ces informations vont pouvoir être partagées et utilisées par les départements, par le service scolarité, par le service des stages, etc.

2- Système de Gestion de Bases de Données (SGBD)

a) Définitions :

- Un SGBD est un logiciel qui permet d'inter-agir avec une BD. Il permet principalement d'organiser les données sur les supports périphériques et fournit les procédures de recherche et de sélection de ces mêmes données.
- Un SGBD est un logiciel qui prend en charge la structuration, le stockage, la mise à jour et la maintenance d'une base de données. Il est l'unique interface entre les informaticiens et les données (définition des schémas, programmation des applications), ainsi qu'entre les utilisateurs et les données (consultation et mise à jour).
- Un SGBD peut être perçu comme un ensemble de logiciels système permettant aux utilisateurs d'insérer, de modifier et de rechercher efficacement des données spécifiques dans une grande masse d'informations partagées par de multiple utilisateurs. Les informations sont stockées sur mémoire secondaire, en général des disques magnétiques.

b) Historique des SGBD

Les SGBD ont 40 ans d'histoire. Les années 60 ont connu un premier développement des BD sous forme de fichiers reliés par des pointeurs. Les premiers SGBD sont réellement apparus à la fin des années 60.

- Première génération de SGBD basés sur les langages navigationnels et sur les modèles réseau et hiérarchique. Ex : IDS II, IMS et SOCRATE.

- Deuxième génération de SGBD (années 80) et basée sur le modèle relationnel. Ex : Oracle, DB2, Informix. Ces SGBD facilitent l'accès aux données aux utilisateurs, les recherches et mises à jours sont effectuées à l'aide d'un langage non procédural (tel que SQL) et ils supportent en général une architecture répartie, au moins avec des stations clients transmettant leurs requêtes à des serveurs gérant les BD.
- Troisième génération : développée dans les laboratoires depuis le début des années 80. Elle correspond aux extensions objets des systèmes relationnels. Ex : Oracle 8 et 9, DB2 Universal Database. Les systèmes à objets tel que O2 constituent une voie plus novatrice.
- Quatrième génération : prise en compte de l'Internet et le Web, les informations non structurées, les objets multimédia, l'aide à la prise de décision, ….

c) Fonctions d'un SGBD: Parmi les fonctions que doit assurer un SGBD, on peut citer :
   - Description : Le SGBD doit mettre à la disposition de l'utilisateur un outil pour décrire l'ensemble des données qui seront stockées dans la BD.
   - Utilisation : Le SGBD offre à l'utilisateur une interaction avec la BD sous forme d'un dialogue pour rechercher, sélectionner et modifier des données.
   - Intégrité : Pour diminuer le risque d'erreurs sur les données par rapport à la réalité, le SGBD offre à l'utilisateur la possibilité de définir des règles qui permettent de maintenir l'intégrité de la BD. Ces règles sont appelées des contraintes d'intégrité. (Ex : 0<=Note<=20)
   - Concurrence d'accès : Le SGBD doit offrir des mécanismes qui permettent de détecter les cas ou il y aurait conflit d'accès et de les traiter correctement.

d) Objectifs d'un SGBD

Nous présentons les objectifs que les SGBD cherchent à atteindre. Bien sur, peu de SGBD satisfont pleinement tous ces objectifs mais ils y tendent tous plus ou moins.

- **Indépendance physique**

Réaliser l'indépendance physique c'est réaliser l'indépendance entre structure de stockage (niveau physique) et structure de données dans le monde réel (niveau logique). Au niveau logique, les structures de données sont définies sur la base d'entités ou objets et de liens entre objets. Au niveau physique, les structures de stockage associées sont des enregistrements, des fichiers, des chemins d'accès (mode d'organisation et d'accès, critère de tri, …). Le changement des modalités de stockage de l'information (optimisation, réorganisation, segmentation, etc.) n'implique pas de changements des programmes.

- **Indépendance logique**

Elle consiste à autoriser plusieurs visions différentes de différents groupes d'utilisateurs sur la BD. Ceci permet à chaque groupe de travail de voir les données comme il le souhaite. Cette indépendance logique est possible grâce à la notion de vue (correspondant à un sous schéma de la BD). En résumé, il doit être possible d'ajouter des attributs, d'en supprimer d'autres, d'ajouter ou de supprimer des associations, d'ajouter ou de supprimer des entités, etc. dans des schémas externes mais aussi dans le schéma conceptuel sans modifier la plus grande partie des applications.

- **Manipulation des données par les non-informaticiens**

Les non-informaticiens devront manipuler les données au moyen des langages non procéduraux, c'est à dire en décrivant les données qu'ils souhaitent retrouver (voir mettre à jour) sans décrire la manière de les retrouver (voir de les mettre à jour) qui est propre aux SGBD. Ces langages correspondent aux langages de manipulation des données (LMD) des SGBD. Exemple : Select nom, prénom From Etudiant.

- **Efficacité des accès aux données**

L'objectif est de retrouver rapidement l'information en ayant des méthodes d'accès et d'organisation performantes. Cet objectif est possible grâce :

- au développement d'index sophistiqués.
- L'existence de plusieurs chemins d'accès à une donnée.
- L'existence de techniques d'optimisation de requêtes qui sélectionne le chemin optimal vers une donnée.

- **Administration cohérente (centralisée) des données**

Les fonctions essentielles d'un SGBD sont réparties entre les mains d'un petit groupe de personnes hautement qualifiées appelés administrateurs de la base de données.

L'administrateur doit:

- permettre un contrôle efficace des données
- résoudre les conflits entre divers points de vue pas toujours cohérents
- pouvoir optimiser les accès aux données et l'utilisation des moyens informatiques

L'administration des données est souvent centralisée. L'objectif est de permettre une administration cohérente et efficace des données.

- **Non redondance des données**

Dans les systèmes classiques à fichiers non intégrés, chaque application possède ses données propres. Ceci conduit généralement à de nombreuses duplications de données. Avec une approche Base de Données, la conception intégrée grâce à la notion de schéma de données (une représentation globale des informations pour un ensemble d'application) évite les redondances. L'administration cohérente des données doit veiller à la non duplication physique des données afin d'éviter les mises à jours multiples.

- **Cohérence des données**

Le SGBD maintient la cohérence des données en appliquant les règles d'intégrité et cela à chaque fois qu'il y a mise à jour.

- **Partageabilité des données**

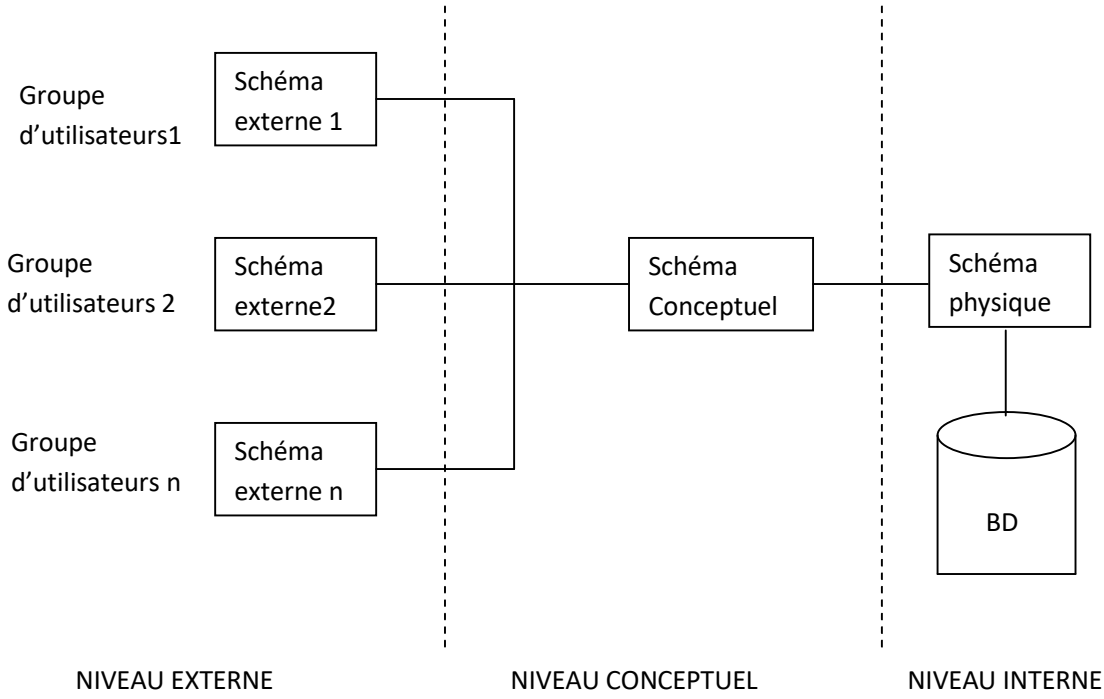
Le SGBD doit permettre à plusieurs applications de partager les données. Pour cela, il doit gérer les conflits d'accès (les détecter et les solutionner). Ceci est possible grâce à la notion de transaction et de définition d'algorithmes de gestion de la concurrence.

- **Sécurité des données**

Les données doivent être protégées contre les accès non autorisés ou mal intentionnés et contre les pannes. La protection contre les accès mal intentionnés se fait grâce à la définition de droits d'accès, de mot de passe, etc. … La protection contre les pannes grâce à la définition des journaux de transaction.

3- Les différents niveaux de représentation d'une BD

Il est couramment admis aujourd'hui qu'il existe trois niveaux de représentation d'une BD ou niveaux d'abstraction définis par le groupe ANSI/X3/SPARC : le niveau interne avec le schéma physique, le niveau conceptuel avec le schéma conceptuel et le niveau externe avec les schémas externes qui correspondent à différents groupes utilisateurs.



Le processus de développement d'un BD comprend deux grandes phases :

- une phase de conception : le résultat est un schéma conceptuel. Au cours de cette phase, la validation s'effectue à travers les schémas externes.
- Une phase de réalisation : résultats : schéma physique et la BD.

a) Le niveau interne : Le schéma physique a pour but de spécifier comment les données seront stockées sur les organes périphériques (disque, bande, …) de l'ordinateur. Ce schéma contient les spécifications de stockage (fichiers, enregistrements, …), de mode d'accès et des liaison inter-fichiers.

b) Le niveau conceptuel : Le schéma conceptuel est la partie fondamentale dans l'architecture d'un système de BD. Il a pour but de décrire en termes abstraits mais fidèles une certaine réalité d'une organisation et de ses processus de gestion qui nécessitent la mise en œuvre d'une BD. Il correspond à la structure canonique (conforme aux règles) des données relatives à un domaine (c'est à dire leur structure sémantique). Le passage du monde réel au schéma conceptuel correspond à un processus de modélisation. Ce processus consiste à traduire les objets du monde réel en catégories d'objets suivants des modèles bien définis. Un modèle est un ensemble de concepts et de règles pour utiliser ces concepts. Il existe différents modèles :
   - Le modèle Entité-association
   - Le modèle hiérarchique
   - Le modèle Réseau

- Le modèle relationnel
- Le modèle objets
- Etc.

c) Le niveau externe : Ce niveau correspond à la vision de tout ou d'une partie du schéma conceptuel par un groupe d'utilisateurs concerné par une application et chargé de mettre en œuvre des programmes d'utilisation. Il s'agit donc de décrire, à l'aide d'un schéma externe parfois appelée vue, la façon dont seront perçues les données par un programme d'application (ou un groupe d'utilisateurs). Un schéma externe ou vue peut être considéré comme un sous schéma du schéma conceptuel. Les vues ont également pour objectif la validation du schéma conceptuel. Il existe une technique de construction du schéma conceptuel à partir de l'intégration de vues.

4- Mise en œuvre d'un SGBD

Un SGBD comprend des langages spécialisés utilisés pour décrire la BD et pour manipuler et accéder aux données contenues dans la BD.

4.1- Langage de Définition de Données (LDD) :

- ✓ La première tâche consiste à représenter le schéma conceptuel à l'aide d'un langage : le langage de définition de données (LDD). Le LDD est propre à chaque SGBD et dépend bien entendu du type de modèle de données. Le LDD est un langage descriptif qui permet de décrire et de nommer d'une part les objets que l'utilisateur perçoit dans son application et d'autre part les associations qui existent entre ces objets, ainsi que les contraintes d'intégrités.
- ✓ Le LDD est aussi utilisé pour la spécification des schémas externes (les vues)
- ✓ La tâche de spécification du schéma conceptuel à l'aide du LDD est définie par l'administrateur de la BD.

4.2- Le langage de Manipulation de Données (LMD) :

- ✓ Une fois que le schéma conceptuel a été défini grâce au LDD, il faut pouvoir crée, mettre à jour, interroger et effectuer toute manipulation que l'on souhaite sur la BD. Pour cela, on dispose d'un LMD pour écrire les programmes d'application.
- ✓ Le LMD comporte des instructions d'insertion, de suppression, de mise à jour, de recherche et d'interrogation des données.

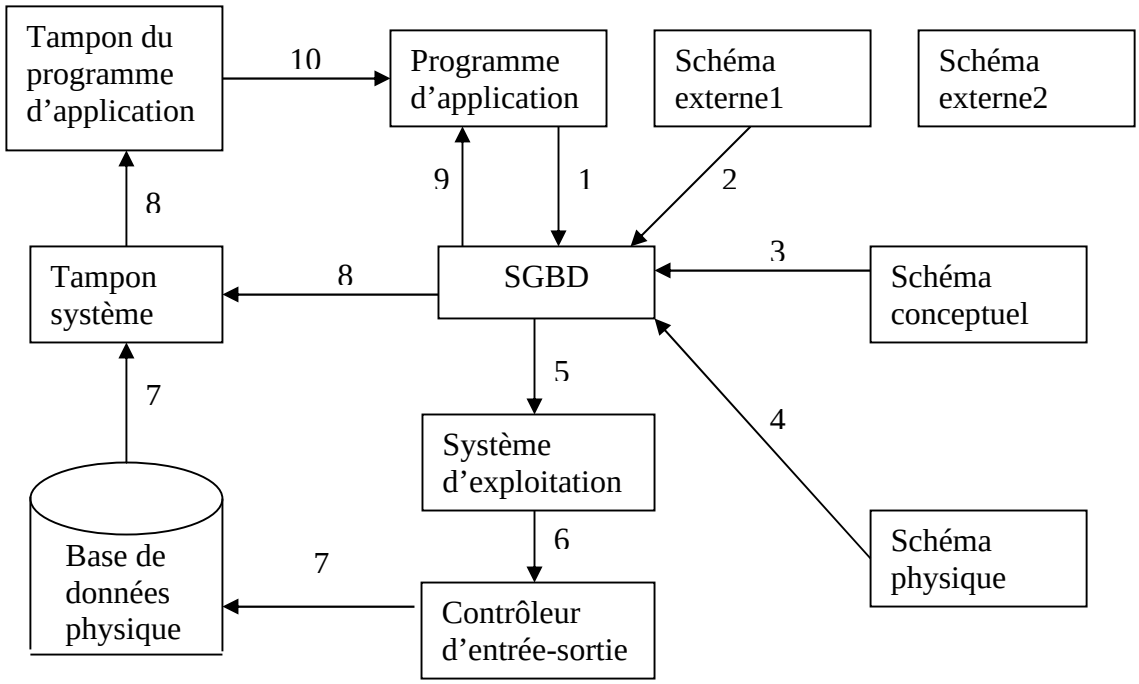
5- Fonctionnement d'un SGBD

Les grands principes du fonctionnement d'un SGBD sont repris dans le schéma suivant qui représente l'architecture d'un SGBD :

Cette architecture met en évidence le flot des ordres et le flot des données.

1- La demande de lecture est envoyée au SGBD
2- La demande est analysée à l'aide du schéma externe (notamment les droits d'accès sont vérifiés)
3- Le SGBD consulte le schéma conceptuel et déduit le type logique des données à extraire
4- Le SGBD consulte le schéma physique et en déduit l'enregistrement physique à lire
5- Le SGBD transmet un ordre de lecture au système d'exploitation

6- Le système d'exploitation lance l'ordre de lecture au contrôleur des unités périphériques qui gère la BD
7- Transmission des données recherchées dans une zone mémoire
8- Le SGBD sélectionne parmi l'ensemble des données reçues dans son tampon seulement celles nécessaires au programme d'application et les transmet dans le tampon du programme
9- Le SGBD informe éventuellement le programme d'application de déroulements anormaux
10- Le programme d'application dispose de la donnée demandée et peut par conséquent passer à l'opération suivante.



II- Architectures des SGBD:

L'architecture d'un système de base de données est hautement influencée par les propriétés du matériel sur lequel il s'exécute, essentiellement par les aspects de réseau, parallélisme ainsi que la répartition géographique :
• Les réseaux permettent que certaines tâches soient exécutées sur le serveur alors que d'autres soient exécutées sur les systèmes clients. Cette division du travail a donné lieu aux systèmes de base de données Client-Serveur.
• Le traitement parallèle permet aux activités du SGBD d'être accélérées, offrant ainsi une rapidité d'exécution des transactions et un nombre plus important de transactions par seconde. Le besoin à un traitement parallèle des requêtes a mené vers les SGBD parallèles.
• Distribuer les données à travers plusieurs sites permet aux données de résider là où elles sont générées et le plus souvent utilisées. Toutefois cette distribution permet l'accès à ces données par les autres structures de l'organisation.

Les architectures les plus utilisées pour les SGBD sont:

1- Architecture centralisée :

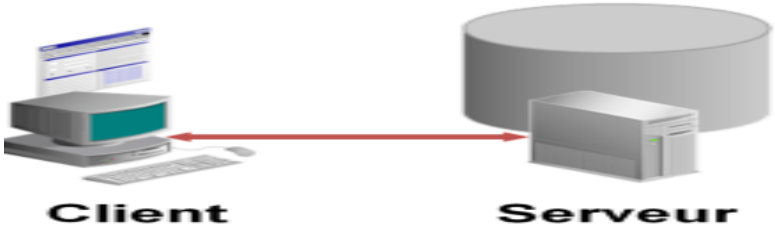
L'architecture centralisée est la plus ancienne. Elle se compose :
· *d'ordinateurs centraux;*
· *de terminaux.*
Tout le travail (les processus) s'exécute sur les systèmes centraux, donc le temps de réponse aux requêtes dépend de la charge du système.
Ce sont des systèmes simples, mais peu flexibles.

2- Architectures Client/Serveur:

L'architecture ***client/serveur découle:***

- ✓ Des améliorations des interfaces graphiques;
- ✓ Les ordinateurs personnels à moindre coûts;
- ✓ Des réseaux locaux;
- ✓ Des bases de données relationnelles;
- ✓ Des langages de traitement de données;
- ✓ Outils de conception assistée par ordinateur.

Pour correspondre au modèle d'architecture **client/serveur,** il faut une communication entre des programmes tournant sur des ordinateurs différents.



2.1 Le client

Le client est un ordinateur qui contient un module informatique intelligent qui est utilisé par un seul usager. Il fournit une interface entre l'usager et l'application informatique.

Il possède son propre système d'opération, celui-ci:

1. Accepte les demandes de l'utilisateur;

2. Ensuite effectuer une requête au serveur d'application;

3. Finalement, affiche le résultat à l'écran.

Un client peut calculer, afficher des données, modifier les données, fournir de l'aide…

Mais il ne peut pas avoir accès directement aux données.

2.2 Le serveur

Le serveur est un ordinateur connecté à un réseau qui fournit des services à d'autres ordinateurs (clients). C'est un module informatique intelligent qui n'est pas accédé directement par l'usager.

Ses fonctions essentielles sont:

1. reçoit des requêtes des ordinateurs clients,
2. exécute les requêtes à l'aide du SGBD,
3. retourne le résultat aux clients.

*Note: L'application client/serveur peut utiliser **plusieurs serveurs** pour fournir différents services aux clients.*

· Ex: Serveur de fichier , d'imprimante, de Fax, de communication et de base de données.

3- Architecture d'un serveur de bases de données Oracle:

Un système de base de données Oracle élémentaire est constitué d'une base Oracle et d'une instance de cette base.

La base de données est constituée de structures physiques et de structures logiques. Celles-ci étant distinctes, il est possible de gérer le stockage physique des données sans affecter l'accès aux structures de stockage logiques.

Une instance se compose de structures mémoire et de processus en arrière-plan (Background). A chaque démarrage d'une instance, une zone de mémoire partagée appelée mémoire SGA (System Global Area) est allouée et les processus en arrière-plan sont lancés. Le terme "processus" désigne les travaux qui s'exécutent dans la mémoire des ordinateurs. Un processus peut être défini comme un "thread de contrôle" ou comme un mécanisme du système d'exploitation capable d'exécuter un ensemble d'étapes. Lorsqu'une instance est démarrée, le logiciel Oracle l'associe à une base de données précise. Ce processus est appelé *montage de la base de données*. La base peut alors être ouverte et mise à la disposition des utilisateurs autorisés.

Chaque instance de base de données est associée à une seule base. S'il existe plusieurs bases sur le même serveur, il existe une instance distincte pour chacune. Une instance de base de données ne peut pas être partagée.

Les processus qui participent au fonctionnement du système de base de données incluent:

• **Processus Serveur**: Ce sont les processus qui reçoivent les commandes SQL des utilisateurs (transactions et interrogations), les exécutent et retournent les résultats. La majorité des SGBD utilisent des processus dédiés pour chaque session utilisateur, alors que certains utilisent un seul processus serveur pour plusieurs sessions.

• **Processus LockManager**: Ce processus gère la fonctionnalité de verrouillage qui inclut l'obtention de verrou, la libération des verrous et la détection de l'interblocage.

• **Processus DatabaseWriter**: Un ou plusieurs processus qui écrivent les blocks modifiés dans la mémoire sur le disque selon un rythme donné.

• **Processus Log Writer**: Ce processus écrit les enregistrements de journalisation du log buffer vers le stockage permanent (sur disque). Les processus serveur rajoute les enregistrements de journalisation vers le log buffer et c'est bien ce processus qui les enregistre sur disque.
•**Processus Checkpoint**: Ce processus effectue des checkpoints périodique.
•**Processus ProcessMonitor**: Ce processus surveille les autres processus, et si l'un d'eux rencontre une situation d'échec il lance les actions de récupération nécessaires.
La mémoire partagée contient les structures suivantes:
• Buffer pool,
• Log buffer,
• Locktable,
• Queryplan cache,…

III- Stockage des données :

Un système informatique offre plusieurs mécanismes de stockage de l'information ou mémoires. Ces mémoires se différencient par leur prix, leur rapidité, le mode d'accès aux données et enfin leur durabilité.

1- Supports :

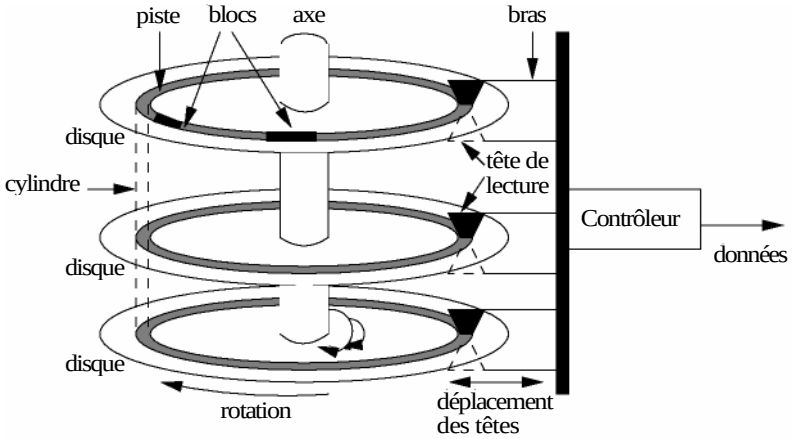
De manière générale, plus une mémoire est rapide, plus elle est chère et plus sa capacité est réduite. Les différentes mémoires *utilisées par un ordinateur* peuvent être classée comme suite (de la plus petite mais la plus efficace à la plus grande mais la plus lente) :

a- La mémoire cache, utilisée par le processeur
b- La mémoire vive ou mémoire principale constituant l'espace de travail de la machine
c- Les disques magnétiques constituent le principal périphérique de type mémoire (grande capacité de stockage)
d- Bandes magnétiques, supports très économiques, caractérisées par leur lenteur d'accès donc déstinées aux sauvegardes.

La mémoire vive et les disques sont les principaux niveaux à considérer dans les applications de bases de données. Une BD est stockées généralement sur disque (pour des raisons de taille et de persistance) mais les données sont transférées en mémoire vive pour être traitées.
Le SGBD doit minimiser les couts de transferts mémoire principale- mémoire secondaire de manière à améliorer les performances du système.

2- Fonctionnement d'un disque

Le support de stockage le plus utilisé est le disque magnétique. Un disque est une surface circulaire magnétisée capable d'enregistrer des informations numériques. La surface magnétisée peut être située sur une seule face (simple face) ou des deux côtés (double face).



Les disques sont divisés en secteurs, un secteur constituant la plus petite surface d'adressage (l'ordinateur sait lire et écrire des zones débutant sur un secteur et couvrant un nombre entier de secteurs). La taille d'un secteur est le plus souvent de 512 octets.

Un disque est entrainé dans un mouvement de rotation régulier par un axe. Une tête de lecture (2 si le disque est double face) vient se positionner sur une des pistes du disque et y lit ou écrit des données. Une piste est divisée en plusieurs blocs (bloc= ensemble de secteurs, n secteurs). Le système d'exploitation peut choisir, au moment de l'initialisation du disque, de fixer une unité d'E/S supérieur à la taille du secteur, et multiple de cette dernière. On obtient des blocs dont la taille est typiquement 512 octets (un secteur), 1024 octets (2 secteurs), 4096 octets (8 secteurs), … Toute lecture et toute écriture sur disque s'effectue par bloc.

L'accès aux données : un disque est une mémoire à accès direct, contrairement à la bande magnétique. L'accès direct est fondé sur une adresse donnée à chaque bloc au moment de l'initialisation du disque par le système d'exploitation. Cette adresse est généralement composée de : n° du disque dans la pile ou n° de surface (si double face), n° de la piste et n° du bloc dans la piste. La lecture d'un bloc, étant donné son adresse, se décompose en 3 étapes :

- Positionnement de la tête de lecture sur la piste contenant le bloc
- Rotation du disque pour attendre que le bloc passe sous la tête de lecture (la tête n'est pas entrainée dans le mouvement de rotation, elle se déplace sur un plan fixe).
- Transfert du bloc

Le SGBD utilise en mémoire une zone appelée buffer : ensemble de blocs en mémoire principale qui sont des copies des blocs sur le disque (l'idée est de maintenir en mémoire principale une copie aussi large que possible du disque).

IV-Fichiers et méthodes d'organisation :

Une base de données n'est rien d'autre qu'un ensemble de fichiers au niveau physique (stockage). La majorité des SGBD ont leur propre module de gestion de fichiers.
Un fichier est un ensemble d'enregistrements structurés. Un enregistrement correspond à un tuple du niveau logique. Le stockage des enregistrements dans un fichier doit tenir compte du découpage en blocs de ce fichier. Un fichier est organisé en blocs (ou pages), chaque bloc contient un certain nombre d'enregistrements.
Pour un SGBD, un fichier est une liste de blocs, regroupés sur certaines pistes (blocs consécutifs) ou repartis aléatoirement sur l'ensemble du disque et chainés entre eux. Le terme d'organisation pour un fichier désigne la structure utilisée pour stocker les enregistrements du fichier. Une bonne organisation a pour but de limiter les ressources en espace et en temps consacrés à la gestion des fichiers. Une bonne organisation doit réaliser un bon compromis pour les quatre principales opérations : insertion, recherche, mise à jour et suppression.

Les organisations les plus utilisées par les SGBD sont :

1- Organisation séquentielle :

Les enregistrements sont stockés dans l'ordre d'insertion et à la première place disponible. Le seul accès est le balayage séquentiel des enregistrements. L'insertion d'un nouvel enregistrement s'effectue dans la dernière page. Si cette dernière est saturée, alors une nouvelle page est allouée et l'enregistrement est inséré. La suppression est assurée logiquement grâce à un indicateur de suppression.

2- Organisation et méthode d'accès par hachage:

L'idée de base du hachage est d'organiser un ensemble d'éléments d'après une clé. Cette organisation est basée sur l'utilisation d'une fonction de calcul qui, appliquée à la clé, détermine un numéro de paquet dans lequel est placé l'enregistrement (un paquet contient un ensemble de pages). La recherche de l'enregistrement se fait ensuite séquentiellement dans le paquet.

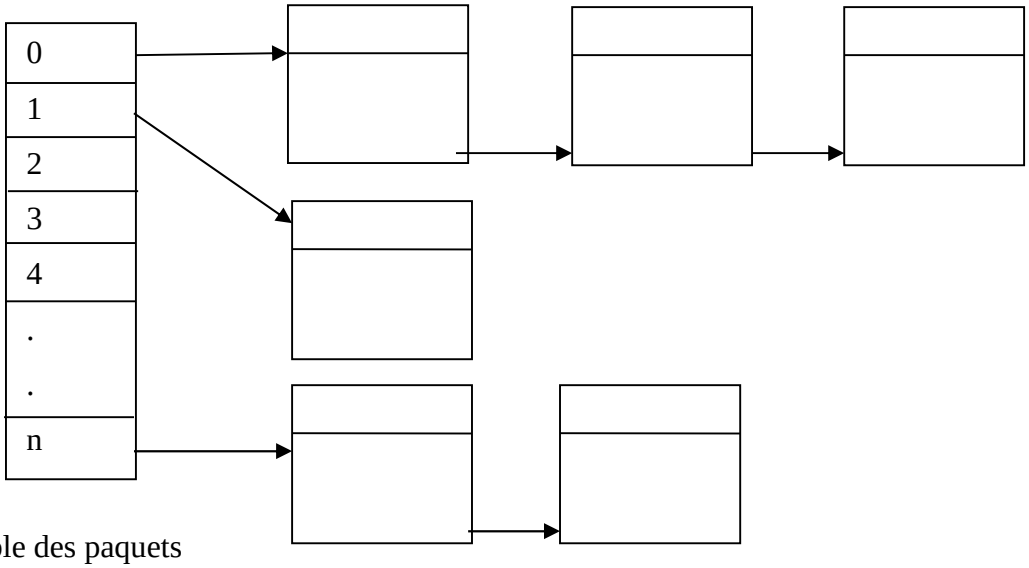


Table des paquets

Algorithme de recherche :
- entrée : valeur de clé c
- calcul h (c) : numéro du paquet
- consultation de la table des paquets pour récupérer l'adresse de la première page du paquet
- rechercher dans cette page l'enregistrement ayant pour clé c
- si la clé existe, alors fin
- sinon passer à la page suivante, répéter l'opération jusqu'à la fin du paquet

Algorithme d'insertion :
- Rechercher si le nouvel enregistrement n'existe pas
- Si non, si le bloc n'est pas saturé alors insérer le nouvel enregistrement, sinon allouer une nouvelle page, insérer le nouvel enregistrement et chainer la nouvelle page aux autres.

Algorithme de suppression
- Rechercher l'enregistrement à supprimer
- Soit libérer la place qu'occupait cet enregistrement en mettant à jour le chainage éventuellement
- Soit mettre un indicateur de suppression dans l'en tête de l'enregistrement à supprimer

3- Organisation et méthode d'accès indexée :

Le principe de base est d'associer à la clé d'un enregistrement son adresse relative dans le fichier à l'aide d'une table des matières (index) du fichier.

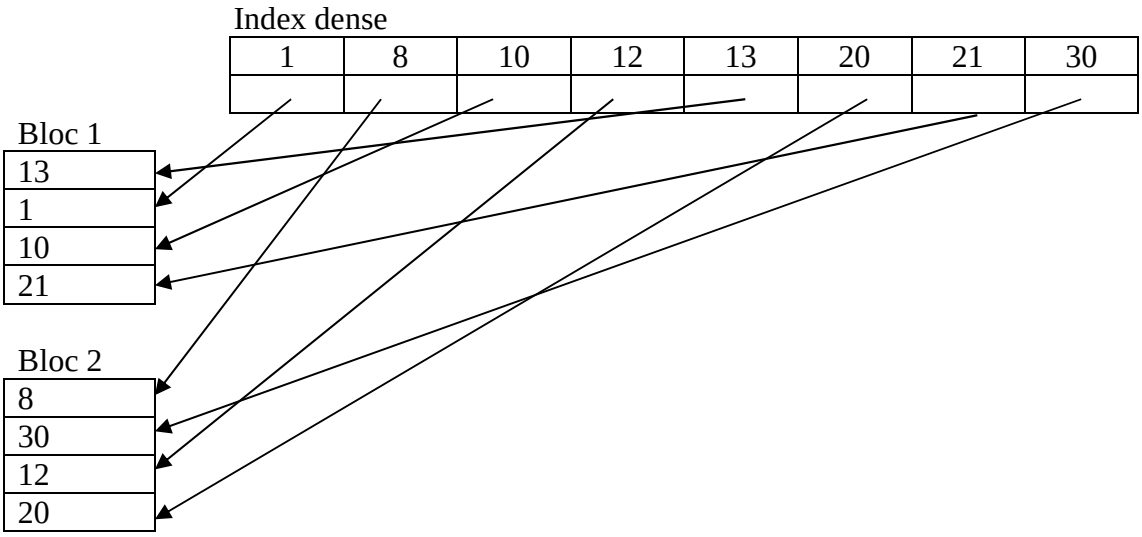
Les principales méthodes d'accès indexées se distinguent par l'organisation de l'index.

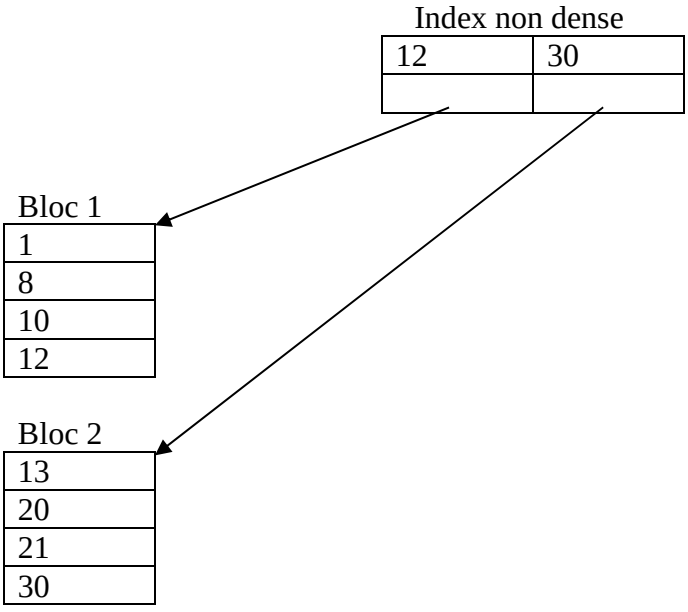
Les étapes successives exécutées pour l'accès à un article dans un fichier indexé sont les suivantes :
- Accès à l'index qui est chargé en MC
- Recherche de la clé de l'enregistrement afin d'obtenir l'adresse

- Accès à l'article (page) et transfert de la page en MC
- transfert de l'article dans la zone du programme utilisateur

Plusieurs variantes d'organisation de l'index :
- Index dense : il contient toutes les clés du fichier
- Index non dense (éparse) : si l'index dense est trop large, on utilise un index eparse ou l'on représente la plus grande clé d'un bloc de données dans l'index et l'adresse du bloc. Dans ce cas, le fichier est trié et divisé en bloc, à chaque bloc lui est associé une entrée dans l'index.

(c,p) : (plus grande clé du bloc, adresse relative du bloc)





Algorithme de recherche :
- accès à l'index
- recherche dans l'index de la clé d'enregistrement désiré,
- récupération dans l'index de l'adresse relative de l'enregistrement (si index dense) ou de l'adresse relative du bloc (si index non dense)
- conversion de l'adresse relative en adresse réelle
- accès à l'enregistrement ou au bloc
- transfert de l'enregistrement dans la zone du programme utilisateur

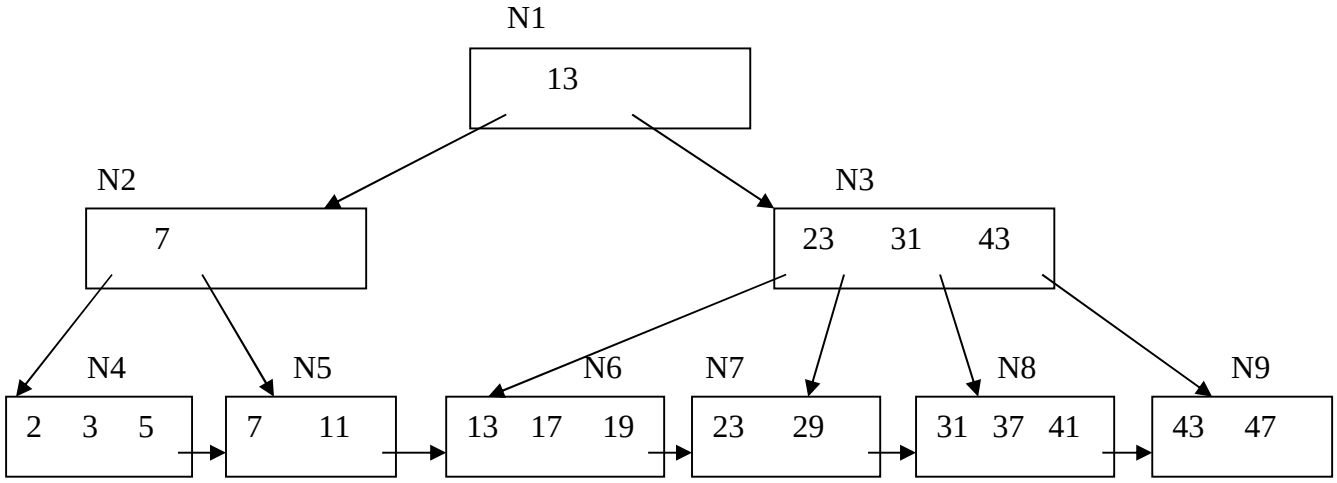
Un index étant lui-même un fichier, on peut définir un autre niveau d'index et ainsi de suite : nous obtenons alors un index hiérarchisé à plusieurs niveaux. Un cas particulier de ces index hiérarchisés est l'index en B arbre.

V- L'arbre B :

Afin de mieux caractériser la notion d'index hiérarchisé et de la rendre indépendante des particularités d'implantations, on a été amené à introduire une structure d'arbre, avec un nombre variable de niveaux.
Les B-arbres sont des cas particuliers d'index hiérarchisés à plusieurs niveaux. C'est la structure d'index la plus utilisée dans les SGBD commercialisés.
Nous allons présenter dans ce qui suit une variante de l'arbre B+
Un arbre-B+ est un arbre équilibré ou balancé dans lequel les chemins qui conduisent de la racine à une feuille quelconque sont tous de même longueur.
Caractéristiques d'un arbre-B+ :

- Un paramètre n est associé à chaque index en arbre-B+ : chaque nœud du arbre-B+ a au plus n clés et (n+1) pointeurs (l'arbre est d'ordre n)
- Les clés contenues dans un nœud feuille du arbre-B+ sont des copies du fichier de données. Ces clés sont distribuées sur les feuilles de manière ordonnée (de gauche à droite)
- La racine contient au moins deux pointeurs utilisés. Tous les pointeurs pointent vers des nœuds de niveau inférieur.
- Sur une feuille, le dernier pointeur pointe vers la feuille suivante sur la droite (frère droit).
- Chaque nœud, excepté la racine et les feuilles, a au moins $\left\lceil \frac{n+1}{2} \right\rceil$ fils (plus petit entier non inférieur à $\frac{n+1}{2}$ ou encore l'arrondi supérieure de $\frac{n+1}{2}$)
- Dans une feuille, à l'exception du pointeur vers la feuille suivante, au moins $\left\lfloor \frac{n+1}{2} \right\rfloor$ pointeurs sont utilisés (l'arrondi inférieure de $\frac{n+1}{2}$).

Exemple : Arbre B+ d'ordre 3



3.1- L'insertion dans un arbre-B+

a) Trouver la feuille P où devrait se placer la nouvelle clé
b) S'il y a de place, insérer la clé (en respectant l'ordre) dans la feuille et ajouter le pointeur vers l'enregistrement
c) S'il n'y a pas de place, allouer une nouvelle page et distribuer les clés entre les deux feuilles de manière équilibrée ($\left\lceil \frac{n+1}{2} \right\rceil$ clés dans P, et le reste dans la nouvelle feuille P')

   La plus petite clé k de P' (en supposant que les clés sont triées dans l'ordre croissant de gauche à droite) doit être insérer dans le père de P (soit Q le nœud père de P)
d) S'il y a de la place dans Q, insérer k et ajouter un pointeur vers P'

   S'il n'a pas de place dans Q, éclater Q en deux (soit Q' le nouveau nœud alloué). Laisser dans Q $\left\lceil \frac{n}{2} \right\rceil$ clés et dans Q' $\left\lfloor \frac{n}{2} \right\rfloor$ clés, la clé médiane remonte vers le père de Q

   Répéter l'étape d) au niveau du père de Q.

3.2- La suppression dans un arbre-B+

a) Chercher la feuille P contenant la clé et supprimer la clé. Il faut cependant vérifier que la clé supprimée ne se retrouve pas comme clé dans le B-arbre (vérifier tous les niveaux de l'index).dans ce cas, il faut remonter dans la hiérarchie afin de remplacer cette clé.
b) Si, après suppression, la contrainte d'occupation n'est pas respectée, alors :

b.1) Si P a un frère adjacent (de même père) qui possède plus du minimum de clés alors une clé peut être ramenée à P et répercuter sur le nœud père de P

b.2) Si aucun frère ne peut donner une clé à P, fusionner P avec le frère P' en supprimant l'un deux. Mettre à jour les clés du parent Q de P et éliminer un pointeur et une clé dans Q.

Si Q ne respecte pas la contrainte d'occupation alors répéter l'étape b), sinon fin de l'algorithme.

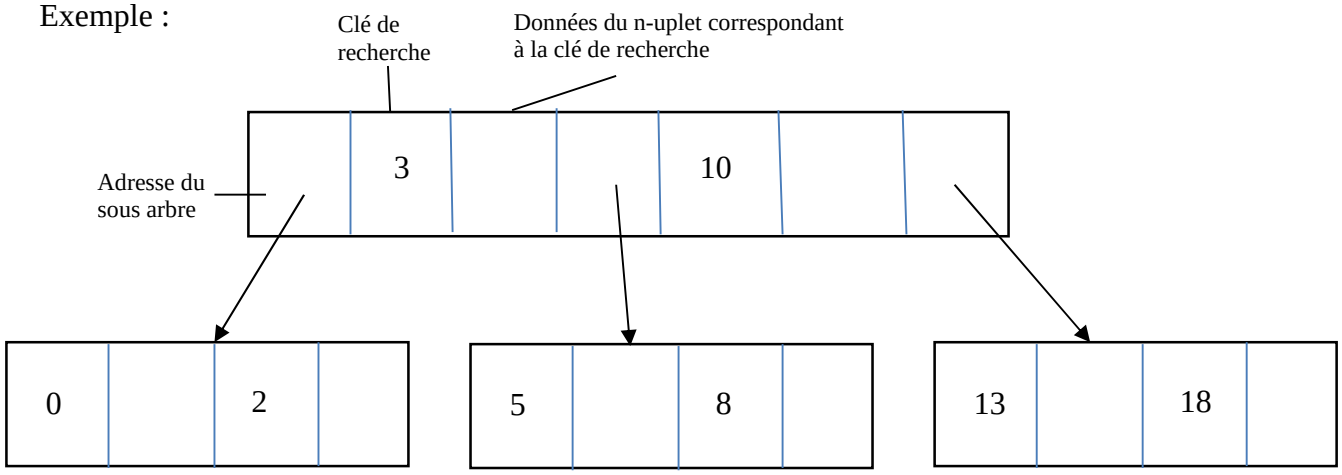
Remarques :

- Pour l'étape b.1) voir en premier le frère gauche, ensuite le frère droit (si le frère gauche ne peut pas donner une clé)
- Pour l'étape b.2) fusionner avec le frère gauche, sinon avec le frère droit s'il n'y a pas de frère gauche

Différences en arbre-B et arbre-B+

Un B-arbre consiste à créer une structure d'index et à stocker les enregistrements dans les nœuds de l'index. Cette structure est dite plaçante, et il ne peut donc y avoir qu'un seul arbre B pour une table.

Exemple :



Arbre-B :

- Les informations dans un B-arbre sont conservées dans les nœuds de l'arbre (pas de distinction entre fichier d'index et fichier de données)
- L'index est plaçant : la place des données dans les blocs dépend de l'index
- En parcourant l'index, on accède directement aux données
- Pour l'index, une valeur de clé n'apparait qu'une seule fois dans l'arbre
- On ne peut avoir qu'un seul B-arbre pour un fichier de données
- L'arbre B n'est pas intéressant avec des n-uplets de grande taille (on ne eut pas conserver beaucoup de valeurs dans les blocs d'index et donc on perd l'intérêt de l'index)

L'arbre-B+ :

- Les informations sont dans des blocs séparés des blocs d'index
- L'index est non plaçant : la place des données dans les blocs de données ne dépend pas de l'index
- En parcourant l'index on récupère l'adresse des données puis on accède aux données
- Pour l'index, toutes les valeurs des clés apparaissent dans les feuilles
- On peut créer un arbre B+ quelque soit le rangement physique des données
- L'arbre B+ prend peu de place dans les blocs d'index (on conserve la valeur de la clé et une adresse)

VI-Les index bitmap

Un index bitmap considére toutes les valeurs possible pour un attribut, et pour chacune de ces valeurs, on stocke un tableau de bits (dit bitmap) avec autant de bits qu'il y a de lignes dans la table.

Soit A l'attribut indexé, et *v* la valeur définissant le bitmap. Chaque bit associé à une ligne l a la signification suivant :

- Si le bit est à 1, l'attribut A a pour valeur *v* dans la ligne l
- Sinon le bit est à 0.

Quand on recherche les lignes avec la valeur *v*, il suffit donc de prendre le bitmap correspondant à cette valeur, de rechercher tous les bits à 1 et d'accéder aux enregistrements correspondants.

Remarque : un index bitmap est très efficace quand le nombre de valeurs possibles pour un attribut est faible.

Exemple :
Employé(NSS, Nom, Prénom, Situation_Fam)
On a 10 employés, les employés de rang 1, 4,5,6 et 8 sont mariés, les employés de rang 2 et 3 sont célibataires, les employés de rang 7 et 9 sont divorcés et enfin l'employé 10 est veuf.
L'index bitmap pour l'attribut Situation_Fam est le suivant :

	1	2	3	4	5	6	7	8	9	10
Marié	1	0	0	1	1	1	0	1	0	0
Célibataire	0	1	1	0	0	0	0	0	0	0
Divorcé	0	0	0	0	0	0	1	0	1	0
Veuf	0	0	0	0	0	0	0	0	0	1

Bien entendu, il ne peut y avoir qu'un seul 1 au maximum dans une colonne puisqu'un attribut ne peut prendre qu'une seule valeur.
Un index Bitmap est de très petite taille comapé à un arbre-B construit sur le même attribut. Il est donc très utile dans les applications de type 'entrepot de données' gérant de gros volumes de données, et calassant les données par des attributs catégoriels défini sur de petits

domaines de valeurs. Certaine requêtes peuvent alors être éxécutées très efficacement, parfois sans meme recourir à la table. Exemple de requête: combien y a-t-il d'employés dont la situation familiale est 'Marié' ou 'Veuf'?

```
SELECT count(*)
FROM employé
WHERE situation_fam in ('Marié', 'Veuf');
```

Pour répondre à cette requête il suffit de compter le nombre de 1 dans les bitmap associés à ces deux valeurs.

VII- Stockage et indexation dans Oracle

1- Stockage des données :

Un système Oracle stocke les données dans un ou plusieurs fichiers. Ces fichiers sont divisés en blocs dont la taille paramétrable peut varier de 2K et plus (8K par défaut). Au sein d'un fichier des blocs consécutifs peuvent être regroupés pour former des extensions (extent). En général, une extension est affectée à un seul type de données (par exemple les enregistrements d'une table. Enfin un ensemble d'extensions permettant de stocker un des objets physique de la base (une table, un index) constitue un segment. Il y a quatre types de segments :

- Les segments de données contiennent les enregistrements des tables, avec un segment de ce type par table
- Les segments d'index contiennent les enregistrements des index, il y a un segment par index
- Les segments temporaires sont utilisés pour stocker des données pendant l'exécution des requêtes
- Les segments rollback contiennent les informations permettant d'effectuer une reprise sur panne ou l'annulation d'une transaction.

Il est possible de paramétrer, pour un ou plusieurs fichiers, le mode de stockage des données. Parmi les paramètres nous pouvons citer la taille des extensions, le nombre maximal d'extensions formant un segment, le pourcentage d'espace libre laissé dans les blocs, etc. Ces paramètres et les fichiers auxquels ils s'appliquent, portent le nom de tablespace. Un tablespace correspond à un espace physique constitué d'un ou plusieurs fichiers. Une base de données Oracle est organisée sous la forme d'un ensemble de tablespaces, dont un crée au moment de l'initialisation de la base et nommé SYSTEM. Ce tablespace contient le dictionnaire de données, les procédures stockées, les triggers, etc.

Le bloc est la plus petite unité de stockage géré par Oracle. La taille d'un bloc est un multiple de la taille d'un bloc du système d'exploitation. La structure d'un bloc Oracle est constituée de cinq parties :

- L'entête (header) contient l'adresse du bloc et son type (donnée, index, etc)
- Le répertoire des tables donne la liste des tables pour lesquelles des informations sont stockées dans le bloc.
- Le répertoire des enregistrements contient les adresses des enregistrements du bloc.
- Un espace libre est laissé pour faciliter l'insertion de nouveaux enregistrement ou l'agrandissement du bloc
- L'espace des données contient les enregistrements.

Les trois premières parties (non dédiées aux données) occupent environ 100 octets.
Chaque enregistrement dans Oracle est identifié par un ROWID, comprenant trois parties :

1. Le numéro du bloc au sein du fichier
2. Le numéro de l'enregistrement au sein du bloc
3. Enfin l'identifiant du fichier.

2- Indexation

Oracle propose plusieurs techniques d'indexation : arbre B, arbre B+, table de hachage, Bitmap. Par défaut, la structure d'index est l'arbre B+, stocké dans un segment d'index séparément de la table à indexer.
L'index organisé en arbre B+ est crée soit :

- Automatiquement sur les attributs sur lesquels on définit une contrainte de clé primaire ou d'unicité
- Manuellement en utilisant la commande Create index, exemples :

CREATE UNIQUE INDEX IndNomEtudiant On Etudiant (Nom, Prénom)
(la clause unique est optionnelle, quand elle est spécifiée cela veut dire qu'à toute valeur de la clé d'index correspond un seul enregistrement)
CREATE INDEX IndWilaya On Etudiant (wilaya)

Au moment de la création d'un index, Oracle commence par trier la table dans un segment temporaire, puis construit l'arbre B+ de bas en haut . Au niveau des feuilles de l'arbre, on trouve pour chaque valeur de la clé le ou les ROWID des enregistrements associés à cette valeur.

Il est aussi possible de demander explicitement qu'une table soit organisée physiquement

- en arbre B (toujours construit sur la clé primaire), en utilisant la syntaxe suivante : CREATE TABLE ETUDIANT
(CodeET Integer(6), ……, PRIMARY KEY (CodeEt), ORGANIZATION INDEX)

- ou par une table de hachage, en créant d'abord un cluster puis on indique ce cluster au moment de la création de la table :
  CREATE CLUSTER HachEtudiant (Code Integer)
  SIZE 100
  HASHKEYS 500 ;
  La clé de hachage est Code, Size spécifie la taille de chaque entrée de la table de hachage et HASHKEYS spécifie le nombre d'entrées dans la table de hachage.

  CREATE TABLE Etudiant (CodeEt Integer(6), …..)
  CLUSTER HachEtudiant (CodeEt)