

## CHAPITRE I :

### ■ Définition : Bases de Données.

Une *base de données* est une collection de données **cohérentes** et **structurées**. Les données d'une base de données sont mémorisées de manière *persistante* et gérées par un Système de Gestion de Bases de Données (SGBD). Au début, Les bases de données avaient une portée plus limitée (Personnel d'une entreprise, les stocks, Les clients, les commandes, données bancaires, etc....). De nos jours, le domaine des Bases de Données englobe tout ce qui touche aux données (recherches Web, fouille de données, BDs médicales et scientifiques, Intégration d'information). Les Bases de Données sont derrière presque tout ce qui se fait sur le Web (recherches Google, requêtes Amazon, eBay, etc...).

¶ Une organisation en Bases de Données est caractérisée par les avantages suivants :

- Saisie des données uniforme.
- Des traitements standards.
- Contrôle de la validité des données.
- Partage des données entre plusieurs utilisateurs ou applications.

¶ Propriétés de l'organisation en Bases de Données :

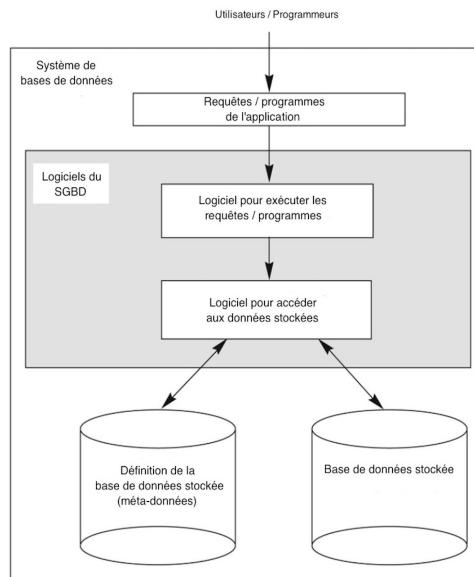
- Usage multiple des données.
- Accès facile, rapide (même quand le volume des données est massif), souple, sécurisé et puissant.
- Disponibilité, exactitude, cohérence et protection des données.
- Indépendance des données et des programmes, ce qui facilite l'évolution de la BD et de ses applications.

Les propriétés et avantages d'une organisation en Bases de Données sont assurées par le SGBD.

### ■ Définition (SGBD)

Un Système de Gestion de Bases de Données (SGBD) est un système informatique qui assure la gestion de l'ensemble (Structuration, stockage, Sécurité, maintenance, mises à jours et consultation) des informations stockées dans une base de données.

Exemples de SGBD : Sybase, Access, Db2, Oracle, SQL Server, MySQL ...



## Architecture d'un S.G.B.D

#### Langages et interfaces d'un SGBD :

- Langages de conception : E/A (Entité/Association), UML ; Utilisés pour la conception haut-niveau d'applications (données et traitements).
- Langages base de données : SQL, XQuery, SPARQL, ... ce sont des langages déclaratifs où l'utilisateur spécifie quoi (et non comment) avec une puissance d'expression limitée (par rapport à un langage de programmation comme C ou Java). Ces langages sont utilisés pour la définition des schémas, interrogation et mises-à-jour ou administration.
- Langages de programmation : PL/SQL, Java, PHP, ... langages impératifs avec une interface SQL utilisés pour la programmation des applications sur les données.

#### Langage des Bases de Données SQL :

SQL se subdivise en 3 sous-langages :

- LDD (Langage de Définition de Données) : création, modification et suppression des objets que peut manipuler une BD (tables, vues et index, etc.).
- LMD (Langage de Manipulation de Données) : ajout, suppression, modification et extraction des données.
- LCD (Langage de Contrôle de Données) : sécurisation et validation des données.

Chacun de ces sous-langages propose ses mots-clés propres. Voici les principales primitives de chaque sous langage :

LDD	LMD	LCD
CREATE	SELECT	GRANT
ALTER	INSERT	REVOKE
DROP	UPDATE	COMMIT
	DELETE	ROLLBACK

Le langage SQL peut être autonome (par ex. SQL seul) ou intégré dans un langage de programmation, à travers une API (Application Programming Interface) comme JDBC (Java DataBase Connectivity).

#### Le Langage de Définition de Données : LDD

Un LDD permet de spécifier le schéma d'une base de données relationnelle, il regroupe l'ensemble des possibilités permettant de créer une base de données, de décrire des données : tables, vues, colonnes, contraintes, index...

Les tables obéissent à une logique de création qui doit respecter des règles strictes appelées : Contraintes d'Intégrité (CI). On associe des contraintes d'intégrité au contenu des tables.

Une contrainte d'intégrité est un automatisme du SGBD qui garantit que les valeurs d'une colonne ou d'un groupe de colonnes satisfont à une condition déclarée.

Les contraintes assurent la cohérence des données (concept d'intégrité). Elles sont décrites lors de la définition des tables de la base (ordre CREATE TABLE).

Les contraintes appliquent des règles sur les données d'une table chaque fois qu'une ligne est insérée, supprimée ou mise à jour.

#### Types de contraintes :

- contrainte de valeurs : toute règle s'appliquant à la valeur d'un attribut (min et max par exemple), (contrainte de domaine) : CHECK
- unicité d'occurrences (unicité de lignes) : UNIQUE
- clé primaire (unicité et indexation pour l'intégrité d'entité) : PRIMARY KEY
- clé étrangère : Un attribut dans une table est lié à la clé primaire d'une autre table (intégrité référentielle) : FOREIGN KEY
- caractère obligatoire (un attribut doit toujours avoir une valeur 'NOT NULL') ou facultatif des valeurs.

#### Les contraintes peuvent s'exprimer :

- Soit au niveau colonne (contraintes locales qui fait partie de la définition de la colonne), appelées : Contraintes de colonnes et fait toujours référence à une seule colonne

Les contraintes de colonnes sont :

- NULL ou NOT NULL (caractère facultatif /obligatoire) ;
- UNIQUE (unicité des valeurs) ;
- PRIMARY KEY (clé primaire élémentaire) ;
- REFERENCES (intégrité de référence) ;
- CHECK (contraintes de valeurs).

Une contrainte au niveau de la colonne est définie par la syntaxe suivante :

**Syntaxe : Déclaration des contrainte de colonne.**

```
[CONSTRAINT nom_contrainte]
NOT] NULL / UNIQUE / PRIMARY KEY / REFERENCES [propriétaire.]nom_table
[ON DELETE CASCADE] / CHECK (condition)
```

- Nom\_Contrainte - bien que non obligatoire, il est toujours conseillé de donner un nom à la contrainte, vous permettant ainsi de l'identifier facilement.
  - Soit au niveau table (contraintes globales définie après la définition des colonnes) : valables pour un ensemble de colonnes d'une table.

Les contraintes de tables sont :

- UNIQUE (composition des colonnes) ;
- PRIMARY KEY (clé primaire composée);
- FOREIGN KEY... REFERENCES... (composition de colonnes);
- CHECK (condition particulière sur les valeurs).

**Syntaxe : Définition des contraintes de table :**

```
[CONSTRAINT nom_contrainte]
UNIQUE / PRIMARY KEY (colonne1 [, colonne2]...) / FOREIGN KEY (colonne1
[, colonne2] ...) REFERENCES [propriétaire.]nom_table [(colonne1 [, colonne2] ...)]
[ON DELETE CASCADE] / CHECK (condition)
```

- La clause [ON DELETE CASCADE] signifie que tout sera supprimé en cascade lors d'une suppression de table.
  - **Les contraintes se définissent :**
    - soit lors de la création des tables, dans l'ordre CREATE TABLE ;
    - soit après la création des tables, par l'ordre ALTER TABLE permettant certaines modifications de la structure des tables.

Chaque contrainte définie est affectée d'un nom propre (nom de contrainte) stockée dans le dictionnaire des données de la base, implicitement par le SGBD (clé primaire) ou explicitement par la clause CONSTRAINT.

 **Commande de création d'une nouvelle table en PL/SQL :**

```
CREATE TABLE [propriétaire.] nom_table
(col1 TYPE(taille) [DEFAULT expression] [NOT NULL],
 col2 TYPE(taille) [DEFAULT expression] [NOT NULL],
 col3 TYPE(taille) [DEFAULT expression] [NOT NULL] , ...
 [PRIMARY KEY (col n, col M...) ]
[FOREIGN KEY (col N, col M...) REFERENCES nom_table2 (col I)]);
```

- **Types de données SQL :**

- Type caractère fixe
  - **CHAR(longueur)** : permet de stocker une chaîne de caractères de longueur fixe maximum de 2000 caractères.
- Type caractère variable
  - **VARCHAR2(longueur)** : permet de stocker une chaîne de caractères de longueur variable longueur maximale de 4000 caractères. Une chaîne plus courte que la longueur spécifiée n'occupera que l'emplacement correspondant à sa taille réelle.
- Type numérique :
  - **NUMBER[(précision [, échelle])]** ; utilisé pour les nombres entiers, nombres décimaux et nombres en virgule flottante.
    - **précision**: nombre entier de chiffres significatifs de 1 à 38 (38 par défaut).
    - **échelle** : nombre de chiffres à droite du séparateur décimal de -84 à +127.

**Exemple** : NUMBER (6,2) : 8345,34

Variantes :

- **INTEGER** : sur 4 octets ;
  - **SIMALLINT** : sur 2 octets ;
  - **DEC[IMAL](longueur,décimales)** ;
  - **FLOAT**.
- Type date
    - **DATE** : permet de stocker une date et/ou une heure. Le format est DD-MON-YY.
  - Type chaîne longue :
    - **LONG** : permet de stocker une chaîne de caractères d'une longueur maximale de 2Go. Une seule colonne de ce type est autorisée par table.
  - Type binaire :
    - **RAW(n)** : permet de stocker des données de type binaire de longueur fixe (2000 octets maximum). On doit insérer les données hexadécimales sous forme de chaîne de caractères.
  - Autres types
    - **BLOB** : Binary Large Object, jusqu'à 4Go ;
    - **BFILE** : type fichier jusqu'à 4Go;
    - **CLOB** ;
    - **NCLOB** ;
    - **LONG RAW** ;
    - **ROWID** ;
    - **NCHAR(n)**;
    - **NVARCHAR2(n)**

- **Creation de Table:** Examples :

- 1- CREATE TABLE Emp\_S (salaire number(8,2) DEFAULT 9500,  
date\_Ambauche DATE DEFAULT '01-JAN-2011',  
date\_Naiss DATE DEFAULT SYSDATE);
- 2- CREATE TABLE Produit(product\_id number(3) CONSTRAINT prod\_id\_pk PRIMARY KEY,  
Nom\_prod varchar2(25) CONSTRAINT Nom\_prod\_nn NOT NULL );

**Remarque :** La contrainte NOT NULL ne peut être définie qu'au niveau colonne.

```
3- CREATE TABLE produits (id_prod number(3) CONSTRAINT prod_id_pk PRIMARY KEY,
    nom_prod varchar2(25)      CONSTRAINT prod_nom_nn NOT NULL,
    designation_prod varchar2(25) CONSTRAINT products_description_uq UNIQUE,
    prix_prod number(8,2) CONSTRAINT products_productprice_ck CHECK (prix_prod > 50))

4- CREATE TABLE prods(id_prod number(3),
    nom_prod varchar2(25) CONSTRAINT nom_prods_nn NOT NULL,
    designation_prod varchar2(25),
    prix_prod number(8,2),
    remise_prod number(8,2) CONSTRAINT remise_prods_nn NOT NULL,
    id_categorie number(3),
    CONSTRAINT id_prods_pk          PRIMARY KEY (id_prod),
    CONSTRAINT designation_nom_prods_uq UNIQUE(nom_prod, designation_prod),
    CONSTRAINT prix_prod_ck        CHECK (prix_prod > remise_prod),
    CONSTRAINT id_categorie_emp_fk FOREIGN KEY (id_categorie)
        REFERENCES Categories (id_categorie))
```

**Remarques:** 1- Une contraintes de table est :

- Crée après avoir défini les différentes colonnes.
- Peut faire référence à plus d'une colonne (une contrainte qui comprend deux colonnes ensemble).
- Permet de créer plusieurs contraintes sur une même colonne.
- Il n'est pas possible de créer une contrainte **NOT NULL** à l'aide de cette méthode.

2- Dans SQL, la contrainte de clé étrangère **FOREIGN KEY** désigne une colonne (ou un ensemble de colonnes) comme clé étrangère et établit une relation entre une clé primaire (ou unique) dans une table (qui existe déjà dans la Base de Données) ou (dans la même table).

Dans le dernier exemple la table **Catégories** à laquelle on fait référence doit exister déjà dans la Base de Données.

3- Une table peut éventuellement être créée à partir d'une autre table :

```
CREATE TABLE Prod_Remis
AS SELECT nom_prod, designation_prod , prix_prod, remise_prod
FROM prods
WHERE remise_prod > 0 ;
```

La nouvelle table aura donc quatre colonnes (nom\_prod, designation\_prod , prix\_prod, remise\_prod) et sera remplie avec les valeurs sélectionnées.

#### ➤ Commandes connexes

- Pour lister les colonnes d'une table :

```
DESC[RIBE] [propriétaire.]nom_table ;
```

- Pour lister les contraintes d'une table :

```
SELECT TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE,
SEARCH_CONDITION FROM USER_CONSTRAINTS
WHERE TABLE_NAME='nom_table';
```

- Pour supprimer des tables :

```
DROP TABLE [propriétaire.]nom_table [CASCADE CONSTRAINTS];
```

➤ La commande **ALTER TABLE** : Modification du schéma d'une table.

Il est possible de modifier la structure d'une table :

- ajout de colonnes (dénormalisation) ou/et de contraintes ;

- modification de certaines contraintes ;
- changement du caractère obligatoire/facultatif (NULL) ;
- rectification de la largeur d'une colonne (VARCHAR2).

- **Syntaxe de la commande Alter Table :**
- Ajout d'une nouvelle colonne.

```
ALTER TABLE [propriétaire.]nom_table
ADD [(col1 TYPE(taille) [DEFAULT expression] [NOT NULL])]
[contrainte...];
```

Exemple :

```
ALTER TABLE Produits
ADD (couleur varchar2(15));
```

- Suppression d'une colonne clé.

```
ALTER TABLE [propriétaire.]nom_table
[DROP PRIMARY KEY / UNIQUE (colonne1 [, colonne2]...
/CONSTRAINT nom_contrainte
[CASCADE] ]
[ENABLE nom_contrainte / DISABLE nom_contrainte] ;
```

Attention : il est impossible de supprimer une clé primaire ou unique qui est utilisée dans une contrainte d'intégrité référentielle. Il faut supprimer à la fois la clé référencée et la clé étrangère.

On peut éventuellement modifier des colonnes existantes (si elles ne sont pas valutées, ou bien pour augmenter la taille d'une chaîne de caractères).

- Modification d'une colonne

```
ALTER TABLE [propriétaire.]nom_table
MODIFY (colonne nouveau_type (nouvelle_taille) , ...);
```

```
ALTER TABLE [propriétaire.]nom_table
MODIFY (colonne NULL / NOT NULL , ...);
```

Exemple :

```
ALTER TABLE produits
MODIFY (Nom_prod varchar2(35));
```