Taki Academy
www.takiacademy.com

STI

Classe : 4ème sciences de l'informatique

Résumé :
Chapitre : Le langage SQL

Sousse (Khezama - Sahloul) Nabeul / Sfax / Bardo / Menzah El Aouina / Ezzahra / CUN / Bizerte / Gafsa / Kairouan / Medenine / Kébili / Monastir / Gabes / Djerba

Q Magazines

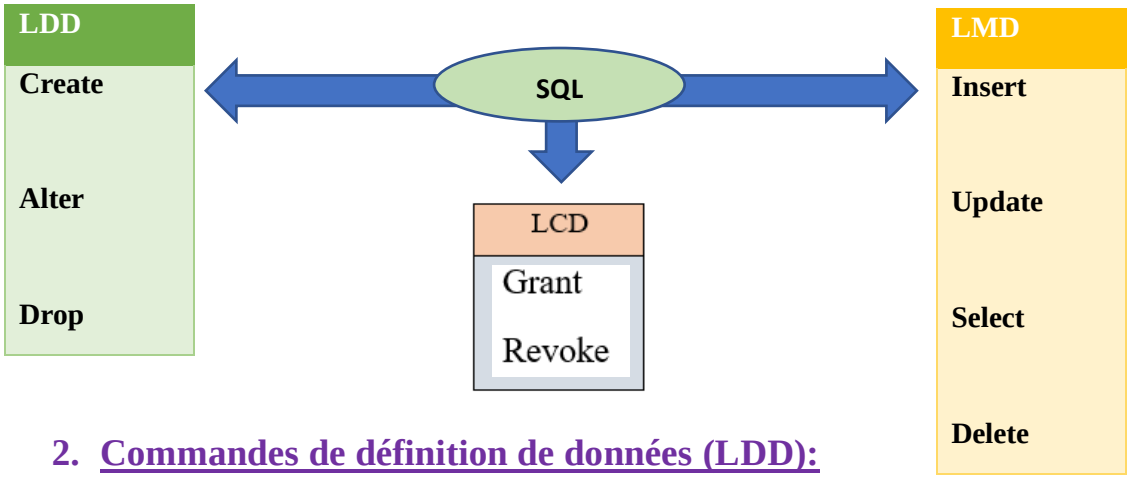
Le langage SQL

1. Modes de création :

Il existe deux modes pour créer les éléments d'une base de données

- **Le mode assisté :** consiste à créer les éléments de la base de données à travers des assistants graphiques.
- **Le mode commande :** consiste à créer les différentes structures de la base de données à l'aide des commandes du langage SQL.

Ce langage est composé de trois familles de commandes



2. Commandes de définition de données (LDD):

Permettent de créer, modifier et supprimer les structures de la base de données :

- La commande du langage SQL permettant de créer une nouvelle base de données est la commande **CREATE DATABASE**. La forme générale de cette commande est la suivante :

```
CREATE DATABASE nom_base ;
```

- La commande du langage SQL permettant de créer une table est la commande **CREATE TABLE**. La forme générale de cette commande est la suivante :

```
CREATE TABLE nom_table (définition colonne ) ;
```

- **La clause « définition colonne »**

Permet de préciser les caractéristiques d'une colonne, elle à la syntaxe suivante :

- Type de données :
  - **INT** (n) : Numérique à n chiffres
  - **DECIMAL** (n, m) : Numérique à n chiffres dont m décimales

Nom_colonne Type [[NOT] NULL] [DEFAULT Valeur] [Contrainte colonne]

  - **VARCHAR** (n) : Chaîne de caractères de longueur maximale n
  - **DATE / TIME** : Date et/ou heure
- L'option **NULL** : La colonne n'est pas obligatoire, l'option NOT NULL veut dire que la colonne est obligatoire ;
- L'option **DEFAULT** : permet d'attribuer une valeur par défaut à cette colonne, elle ne peut pas être indiquée lorsque la colonne est obligatoire ;
- L'option « **Contrainte colonne** » : précise une contrainte d'intégrité relative à la colonne (contrainte de clé primaire, de clé étrangère ou de valeur). La syntaxe est la suivante :

```
[CONSTRAINT nom_contrainte]
   PRIMARY KEY
  |REFERENCES nom_table [(nom_colonne)] [ON DELETE CASCADE ON UPDATE
      CASCADE ]
   CHECK (condition)
```

- **CONSTRAINT** : est optionnel et sert à attribuer un nom à la contrainte ;
- **PRIMARY KEY** : Spécifie que la colonne est utilisée comme clé primaire ;
- **REFERENCES** : Définit une contrainte d'intégrité référentielle ;
- **ON DELETE CASCADE** : Maintient l'intégrité référentielle en supprimant automatiquement les valeurs d'une clé étrangère dépendant des valeurs d'une clé primaire supprimée ;
- **ON UPDATE CASCADE** : Maintient l'intégrité référentielle en modifiant automatiquement les valeurs d'une clé étrangère dépendant des valeurs d'une clé primaire modifiée ;
- **CHECK** : Mot clé associé à une condition pour chaque valeur insérée.

❖ **Modification de la structure d'une table**

La forme générale de cette commande est la suivante :

```
ALTER TABLE nom_table
        [ADD COLUMN définition_colonne]
        [ADD CONSTRAINT définition_contrainte]
        [MODIFY définition_colonne]
        [DROP COLUMN nom_colonne]
        [CHANGE COLUMN ancien_nom   nouveau_nom]
        [DROP CONSTRAINT nom_contrainte]
        [ENABLE|DISABLE nom_contrainte]
```

- L'option **ADD COLUMN** : permet de rajouter des nouvelles colonnes à la table
- L'option **ADD CONSTRAINT** : permet de rajouter une nouvelle contrainte à la table
- L'option **MODIFY** : modifie certaines caractéristiques d'une colonne existante
- L'option **DROP COLUMN** : permet de supprimer une colonne de la table
- L'option **CHANGE COLUMN** : permet de changer le nom d'une colonne existante
- L'option **DROP CONSTRAINT** : permet de supprimer une contrainte d'intégrité de la table
- L'option **DISABLE** nom_contrainte : permet de désactiver une contrainte d'intégrité
- L'option **ENABLE** nom_contrainte : permet de réactiver une contrainte d'intégrité

❖ **Suppression d'une table en mode commande**

La commande est la suivante :

```
DROP TABLE nom_table ;
```

3. Commandes de manipulation de données (LMD) :

La manipulation de données consiste à :

- Insérer de nouvelles lignes dans une table ;
- Modifier des lignes existantes dans une table ;
- Supprimer des lignes d'une table ;
- Consulter, rechercher des lignes existantes dans une ou plusieurs tables.

1. Insertion de lignes

La forme générale de la commande SQL permettant d'insérer une ligne est la suivante :

- L'ordre des valeurs fournies par le paramètre « liste_valeurs » doit être le même que celui des

```
INSERT INTO nom_table [liste_nom_colonnes] VALUES (liste_valeurs)
```

colonnes données pour liste_nom_colonnes.

- Les conditions suivantes doivent être respectées :
  - Les types de données pour « liste_valeurs » doivent être compatibles avec ceux des colonnes de la table ;
  - Unicité des lignes (contrainte de clé primaire) ;
  - Caractère obligatoire associé à une colonne (clause NOT NULL) ;
  - Contrainte d'intégrité référentielle ;
  - Vérification des conditions de validité (clause CHECK).

2. Modification de lignes

La forme générale de cette commande est la suivante :

```
UPDATE nom_table
SET nom_colonne1=Expression1 [, nom_colonne2=Eexpression2, …]
[WHERE condition] ;
```

3. Suppression de lignes

La forme générale de cette commande est la suivante :

```
DELETE FROM nom_table
[WHERE condition]
```

4. Recherche de données : Requêtes

Une recherche peut consister à effectuer :

- Une **projection** sur certaines colonnes d'une table

```
SELECT [DISTINCT] */ liste_nom_colonne
FROM nom_table ;
```

- Une **sélection** sur certaines lignes d'une table

```
SELECT [DISTINCT] */ liste_nom_colonne
FROM nom_table
WHERE condition ;
```

- Une **jointure** sur deux tables ;

```
SELECT [DISTINCT] */ liste_nom_colonne
FROM nom_table1 AS [alias1], nom_table2 AS [alias2], …
WHERE condition de jointure ;
```

- L'opération relationnelle de jointure réalise une liaison entre deux tables en se basant sur l'égalité des valeurs entre l'une des colonnes de chaque table ;
- Un alias permet de donner un nom synonyme abrégé à la table ;
- Le paramètre condition sert en particulier à préciser la condition de jointure qui doit porter sur les colonnes en commun des deux tables joints (clé primaire – clé étrangère).

- Recherche de données avec **Tri**

La forme générale de la commande de recherche est la suivante :

```
SELECT [DISTINCT] */ liste_nom_colonne
FROM nom_table1 [alias1], nom_table2 [alias2] …
[WHERE condition]
ORDER BY nom_colonne1 [ASC/DESC] [, nom_colonne2 [ASC/DESC] …];
```

- Les colonnes à trier doivent figurer dans la liste des noms de colonnes à afficher
- Par défaut c'est l'ordre croissant qui est pris en considération
- Le tri peut être associé à n'importe quelle opération de projection, sélection et jointure.

- Utilisation des fonctions de calculs dans les opérations de recherche **(fonctions agrégat)**

SQL offre certaines fonctions standards de calculs appelées « Fonctions agrégat ». Ces fonctions ne peuvent être utilisées qu'avec la commande SELECT et en dehors de la clause WHERE.

- La fonction **COUNT** : sert à compter le nombre de lignes du résultat obtenu ;
- La fonction **SUM** : faire la somme des valeurs numériques d'une colonne ;
- La fonction **MIN** : détermine la valeur minimale d'une colonne ;
- La fonction **MAX** : détermine la valeur maximale d'une colonne ;
- La fonction **AVG** (average) détermine la moyenne des valeurs numériques d'une colonne.

- La commande **GROUP BY**

La commande GROUP BY est utilisée en SQL pour grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultats. Cette commande doit être utilisée après la commande WHERE et avant la commande HAVING. La forme générale de la commande est la suivante :

```
SELECT [DISTINCT] */ liste_nom_colonne
FROM nom_table1 [alias1], nom_table2 [alias2] …
[WHERE Condition]
GROUP BY nom_colonne1 , nom_colonne2, …;
```

- La condition **HAVING**

La condition HAVING est presque similaire à WHERE, à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM, COUNT, AVG, MIN ou MAX. La forme générale de la commande est la suivante :

```
SELECT [DISTINCT] */ liste_nom_colonne
FROM nom_table1 [alias1], nom_table2 [alias2] …
[WHERE Condition]
GROUP BY nom_colonne1 , nom_colonne2, …
HAVING fonction (colonne) opérateur Valeur ;
```

Taki Academy
www.takiacademy.com

Sousse (Khezama - Sahloul) Nabeul / Sfax / Bardo / Menzah El Aouina / Ezzahra / CUN / Bizerte / Gafsa / Kairouan / Medenine / Kébili / Monastir / Gabes / Djerba