PARTIE I

Bases de données (SQL)

MySQL

M.AATILA

2021/2022

Langage SQL (MySQL)

Le langage SQL a été e initialement conçu dans les années 1970 par la firme IBM. Il a été ensuite normalisé et est devenu le standard de tous les SGBDR. Ce langage permet de masquer aux programmeurs les algorithmes de recherche des données dans des fichiers physiques eux-mêmes structurés de manière très complexe et différemment selon les SGBDR.

- ✓ La définition des données : création des tables, des contraintes, etc. ;
- ✓ La manipulation des données : sélectionner, insérer, supprimer et modifier ;
- ✓ Le contrôle des données : intégrité, droits d'accès, verrous et cryptage ;

I. Syntaxe du langage SQL :

Comme tout nouveau langage, commençons par apprendre la syntaxe de base.
Tout d'abord on peut mettre autant d'espaces et de sauts de ligne que l'on veut entre les mots du langage. Cependant, on respectera les règles suivantes :

- ✓ Une seule instruction par ligne ;
- ✓ Des lignes pas trop longues (visibles entièrement à l'écran).

1. Les Commentaires :

On peut insérer des commentaires de deux façons :

- ✓ Sur une ligne, à partir de deux tirets -- ;
- ✓ Sur une ligne, à partir de # ;
- ✓ Dans un bloc délimité par /* et par */.

Exemple :

```
/*cette requête sélectionne toutes
les données de la table Exemple*/
select * from Exemple;
-- le * désigne toutes les colonnes
#Chaque instruction se termine par ;
```

2. Noms :

Tous les noms d'objets (table, colonne, variable, etc.) doivent respecter les règles suivantes :

- ✓ Ne pas dépasser 128 caractères parmi : les lettres (non accentuées), les chiffres, les caractères non spéciaux ;
- ✓ Commencer par une lettre ;

Par ailleurs, on n'est pas obligé de respecter la casse (i.e. il n'y a aucune différence entre les majuscules et les minuscules).

3. Opérateurs :

- Les opérateurs arithmétiques disponibles sont : **+**, **-**, *****, **/** et **%** le reste par division entière ;
- Les opérateurs de comparaison logique sont : **<**, **<=**, **=**, **>=**, **>** et **<>** (différent) ;
- Les autres opérateurs logiques sont : **AND**, **OR** et **NOT** ;
- Pour la concaténation des chaînes de caractères on utilise **+**.

Les niveaux de priorité entre ces opérateurs sont usuels, il suffit donc de parenthéser quand on a un doute.

4. Types de données :

Les principaux types disponibles en SQL sont :

Type	Taille	Description
BIGINT(*M*)	8 octets	Ce type est un entier pouvant aller de -9 223 372 036 854 775 808 à 9 223 372 036 854 775 807.
BIT	1 octet	Ce type est booléen et peut prendre la valeur 0, 1, «*OFF*» ou «*ON*».
BOOL	1 octet	Ce type est booléen et peut prendre la valeur 0, 1, «*OFF*» ou «*ON*».
BLOB		Ce type est un bloc de données dont la taille maximale est de 65 535 octets.
CHAR(*longueur*)	*longueur*+1 octets	Ce type est une chaine de caractères ayant une longueur constante.
DATE	3 octets	Ce type est un date ayant le format «AAAA-MM-JJ».
DATETIME	8 octets	Ce type est un date et une heure ayant le format «AAAA-MM-JJ hh:mm:ss».
DECIMAL(*M*,*Dec*)		Ce type est un nombre ayant longueur texte prédéfinit au niveau entier et de ses décimals.
DOUBLE[(*M*,*Dec*)]	8 octets	Ce type est un nombre à virgule flottante ayant longueur texte prédéfinit au niveau entier et de ses décimals.
DOUBLE PRECISION[(*M*,*Dec*)]	8 octets	Ce type est un nombre à virgule flottante ayant longueur texte prédéfinit au niveau entier et de ses décimals.
ENUM(*chaine1*,*chaine2*,...)	1 à 2 octets	Ce type est une énumération de données non-combiné pouvant aller de 0 à 65 535 possibilités.
INT(*M*)	4 octets	Ce type est un entier pouvant aller de -2 147 483 648 à 2 147 483 647.
INTEGER(*M*)	4 octets	Ce type est un entier pouvant aller de -2 147 483 648 à 2 147 483 647.
LONGBLOB		Ce type est un bloc de données dont la taille maximale est de 4 294 967 295 octets.
LONGTEXT		Ce type est un texte dont la taille maximale est de 4 294 967 295 octets.
MEDIUMBLOB		Ce type est un bloc de données dont la taille maximale est de 16 777 215 octets.
MEDIUMINT(*M*)	3 octets	Ce type est un entier pouvant aller de -8 388 608 à 8 388 607.
MEDIUMTEXT		Ce type est un texte dont la taille maximale est de 16 777 215 octets.
NUMERIC(*M*,*Dec*)		Ce type est un nombre ayant longueur texte prédéfinit au niveau entier et de ses décimals.
REAL[(*M*,*Dec*)]	8 octets	Ce type est un nombre à virgule flottante ayant longueur texte prédéfinit au niveau entier et de ses décimals.
SET(*chaine1*,*chaine2*,...)	1 à 8 octets	Ce type est un ensemble de données combiné pouvant aller de 1 à 64 données.
SMALLINT(*M*)	2 octets	Ce type est un entier pouvant aller de -32 768 à 32 767.
TEXT		Ce type est un texte dont la taille maximale est de 65 535 octets.
TIME	3 octets	Ce type est un date et une heure ayant le format «hh:mm:ss».
TIMESTAMP	4 octets	Ce type est un date et une heure ayant le format «AAAA-MM-JJ hh:mm:ss» allant du 1 janvier 1970 à l'année 2037.
TINYBLOB		Ce type est un bloc de données dont la taille maximale est de 255 octets.
TINYINT(*M*)	1 octet	Ce type est un entier pouvant aller de -128 à 127.
TINYTEXT		Ce type est un texte dont la taille maximale est de 255 octets.
VARCHAR(*longueur*)	*longueur*+1 octets	Ce type est une chaine de caractères ayant une longueur variable.
YEAR	1 octet	Ce type est une année de date pouvant pouvant aller de 1901 à 2155.

Consulter l'aide de MySQL pour plus de détails.

II. Définition d'une base de données relationnelle (LDD)

1. Création d'une base de données : CREATE DATABASE

On crée une base de données par la commande CREATE DATABASE. Dans la base de données seront enregistrés :

- ✓ le catalogue de la base de données,
- ✓ les objets utilisateurs (tables, vues, règles, triggers, procédures, …),
- ✓ les index, contraintes et types de données
- ✓ le journal des transactions

Les tables, vues, règles … sont créées dans un deuxième temps par d'autres commandes.

Syntaxe :

```
CREATE DATABASE   <nom base de données> ;
```

Le nom de la base de données doit être valide c'est-à-dire commençant par une lettre et ne pas comporter de caractères spéciaux sauf '_'.

✓ Exemple : Création d'une base de données Gestion_Etudiant :

```
CREATE DATABASE GESTION_ETUDIANT ;
```

2. Suppression d'une base de données DROP DATABASE

Pour supprimer une base de données, il faut impérativement qu'aucun utilisateur ne soit en train de l'utiliser. Les données de chaque table de la base de données sont supprimées et irrécupérables. Structure de la commande :

```
DROP DATABASE IF EXISTS   <nom base de données> ;
```

✓ Exemple : Suppression de la base de données Gestion_Etudiant :

```
DROP DATABASE IF EXISTS GESTION_ETUDIANT ;
```

3. Création de table CREATE TABLE

Lors de la création d'une table dans une base de données existante, il faut préciser :

- ✓ Pour chaque colonne : son nom et son type de données ;
- ✓ Une clé primaire (qui permet d'identifier chaque ligne de façon unique).

On peut éventuellement préciser pour chaque colonne si null est interdit et/ou une valeur par défaut.

Exemple de création d'une table :

```
CREATE TABLE clients  (
numClient  CHAR(8) PRIMARY KEY , -- clé primaire
nomClient  VARCHAR(64) NOT NULL, -- null interdit
classementClient  INT DEFAULT 0 , -- valeur par défaut
) ;
```

4. Modification d'une table ALTER TABLE

Pour modifier une table existante, on utilise l'instruction ALTER TABLE.

Exemples :

```
ALTER TABLE clients ADD COLUMN adrClient VARCHAR(255) ; -- pour ajouter la colonne adresse
ALTER TABLE clients DROP COLUMN adrClient ;          -- pour retirer la colonne adresse
ALTER TABLE clients ALTER COLUMN numClient INT ;   -- pour reconvertir le type de données
```

Pour supprimer une table existante, il suffit de taper :

```
DROP TABLE IF EXISTS <nom_table> ;
```

Exemple : Pour la suppression de la table Client : **DROP TABLE IF EXISTS Client ;**

III. Contraintes

1. Contrainte PRIMARY KEY (clé primaire)

Elle permet d'indiquer dans une table une colonne (zone) ou un ensemble de colonnes qui doit contenir des valeurs identifiant de façon unique et certaine une ligne de la table.

Exemple : Dans une table Client le numClient est un identifiant permettant de repérer un et un seul salarié. Ce sera donc la "PRIMARY KEY" de la table Client. Une table ne peut posséder qu'une seule clé primaire donc une seule contrainte PRIMARY KEY par table.

Exemple 1 : Déclaration de la contrainte à la fin de la création de la table.

```
CREATE TABLE clients  (
numClient  CHAR(8) ,
nomClient  VARCHAR(64) NOT NULL, -- null interdit
classementClient  INT DEFAULT 0 , -- valeur par défaut
CONSTRAINT PK_Client PRIMARY KEY(numClient ), -- Contrainte Clé primaire
) ;
```

Exemple 2 : Déclaration de la contrainte lors de la définition du champ.

```
CREATE TABLE clients  (
numClient  CHAR(8) PRIMARY KEY , -- clé primaire
nomClient  VARCHAR(64) NOT NULL, -- null interdit
classementClient  INT DEFAULT 0 , -- valeur par défaut
) ;
```

2. Contrainte Unique

Désigne dans une table une colonne (zone) ou un ensemble de colonnes (de zones) qui doit contenir des valeurs ne se retrouvant pas dans une autre ligne de la même table. Plusieurs contraintes UNIQUE sont possibles sur une même table. Exemple :

```
CREATE TABLE clients  (
numClient  CHAR(8) PRIMARY KEY , -- clé primaire
nomClient  VARCHAR(64) NOT NULL, -- null interdit
classementClient  INT ,
CONSTRAINT cst_clsCl UNIQUE (classementClient ),
) ;
```

3. Contrainte FOREIGN KEY (clé étrangère)

Désigne une clé étrangère qui se trouve dans une autre table en clé primaire ; le nom de l'autre table est indiqué derrière références. Si les noms de colonne diffèrent d'une table à l'autre, il faut indiquer le nom de la colonne de la table suivi de références suivi entre parenthèses du nom de la colonne dans l'autre table (là où elle est primaire). Plusieurs contraintes FOREIGN KEY par table sont possibles.

Exemple : table commande

```
CREATE TABLE Commande  (
numCommande  CHAR(8) PRIMARY KEY , -- clé primaire
DateCommande  DATE,
numClient  CHAR(8),
villeCommande CHAR(25),
CONSTRAINT FK_Commande FOREIGN KEY (numClient) REFERENCES Client(numClient),
) ;
```

Remarque : Un nom de contrainte doit être unique à l'intérieur d'une base de données.

4. Contrainte CHECK

Cette contrainte permet d'introduire des contrôles sur les zones de la table :

- ✓ Contrôle de validité par rapport à des constantes ou des listes de constantes ;
- ✓ Contrôle de cohérence entre deux colonnes de la table ;

Dans l'exemple qui suit :

- ✓ Contrôle que DateCommande est comprise entre le '01/01/1980' et le '01/01/2016' ;
- ✓ Contrôle que villeCommande est l'une des villes suivantes : Marrakech ou Rabat.

```
CREATE TABLE Commande  (
numCommande  CHAR(8) PRIMARY KEY , -- clé primaire
DateCommande  DATE,
numClient  CHAR(8),
villeCommande CHAR(25),
CONSTRAINT FK_Commande FOREIGN KEY (numClient) REFERENCES Client(numClient),
CONTRAINT cst_date CHECK( DateCommande BETWEEN '1980/01/01' AND '2016/01/01'),
CONTRAINT cst_ville CHECK( villeCommande IN ('Marrakech', 'Rabat') ),
) ;
```

5. Valeur par défaut : DEFAULT

Cette notion peut être entrée par contrainte ou par mot clé derrière la définition de la colonne. Cette contrainte permet d'indiquer le contenu d'une zone quand elle n'est pas indiquée lors de l'insertion d'une ligne dans la table.

```
CREATE TABLE Commande  (
numCommande  CHAR(8) PRIMARY KEY , -- clé primaire
DateCommande  DATE,
numClient  CHAR(8),
villeCommande CHAR(25) DEFAULT 'Marrakech', -- Marrakech comme valeur par défaut
CONSTRAINT FK_Commande FOREIGN KEY (numClient) REFERENCES Client(numClient),
CONTRAINT cst_date CHECK( DateCommande BETWEEN '1980/01/01' AND '2016/01/01'),
CONTRAINT cst_ville CHECK( villeCommande IN ('Marrakech', 'Rabat') ),
) ;
```

6. Ajout/Suppression d'une contrainte

Il est parfois nécessaire d'ajouter des contraintes non prévues lors de la création de la table ou de supprimer des contraintes portées à tort sur la table. Pour ce faire, nous allons utiliser la commande déjà vue ALTER TABLE.

```
ALTER TABLE Commande DROP CONSTRAINT  cst_ville ; -- suppression de la contrainte cst_ville
ALTER  TABLE  Commande  ADD  CONSTRAINT   cst_numCmde UNIQUE(numCommande ) ; / *Ajout de la contrainte cst_numCmde */
```

Vous pouvez réaliser plusieurs opérations ADD ou DROP avec un seul ALTER TABLE.

IV. Langage de Manipulation des Données (LMD)

1. Insertion des données dans une table :

En SQL on ne peut insérer des lignes que dans une table à la fois. On peut ajouter des données complètes (on précise alors toutes les colonnes). Les données sont saisies dans une table par la commande INSERT INTO.

i. Pour insérer des valeurs dans toutes les colonnes :

```
INSERT INTO <nom_table>  VALUES (val1, val2, val3,  ....., valn)
```

Exemple : insertion d'un client dans la table client

```
INSERT INTO Client  VALUES ('CL001', 'AHMED', 01) ;
```

ii. Pour insérer des valeurs dans quelques colonnes :

```
INSERT INTO <nom_table> (col1, col2, col3,....., coln) VALUES (val1, val2, val3,  ....., valn)
```

Exemple : insertion d'un client dans la table client

```
INSERT INTO Client (numClient, nomClient) VALUES ('CL002', 'SAID') ;
```

Il faut noter que :

- ✓ Toutes les valeurs ne sont pas nécessaires ;
- ✓ L'indication des colonnes peut se faire dans n'importe quel ordre, il faut simplement respecter la correspondance avec VALUES ;

- ✓ Pour les colonnes définies NOT NULL la valeur est obligatoire. A moins qu'une contrainte DEFAULT soit définie auquel cas c'est la valeur par défaut qui sera prise.

2. Insertion des données dans une table :

La commande UPDATE permet de modifier les valeurs d'un ou plusieurs champs, dans une ou plusieurs lignes existantes d'une table.

```
UPDATE <nom_table> SET column1=value1, column2=value2,
….columnN=valueN WHERE condition;
```

Ou Bien :

```
UPDATE <nom_table> SET (col1, col2,...) = (SELECT ...)
WHERE condition;
```

Exemple : Modifier le nom du client numéro 'CL002' par 'KAMAL'

```
UPDATE Client SET NomClient = 'KAMAL' WHERE numClient = 'CL002';
```

Remarque : La clause **WHERE** est facultative ; si elle n'est pas précisée, LES NOMS DE TOUS LES CILENTS DE LA TABLE PRENDRONS LA VALEUR 'KAMAL' !

3. Suppression des données dans une table :

La commande DELETE permet de supprimer des lignes d'une table. La clause WHERE indique quelles lignes doivent être supprimées.

```
DELETE FROM <nom_table> WHERE condition;
```

Exemple : Supprimer le client numéro 'CL002'

```
DELETE FROM Client WHERE numClient = 'CL002';
```

Remarque : La clause **WHERE** est facultative ; si elle n'est pas précisée, TOUTES LES LIGNES DE LA TABLE SONT SUPPRIMEES !