SQL : Langage de Définition de Données

Plan

Création et suppression d'une base de données

Création

```
CREATE DATABASE nom_base_de_données;
```

Création et suppression d'une base de données

Création

```
CREATE DATABASE nom_base_de_données;
```

Suppression :

```
DROP DATABASE nom_base_de_données [IF EXISTS];
```

`IF EXISTS` : pour éviter le message d'erreur si la BD n'existe pas.

Création et suppression d'une base de données

Création

```
CREATE DATABASE nom_base_de_données;
```

Suppression :

```
DROP DATABASE nom_base_de_données [IF EXISTS];
```

`IF EXISTS` : pour éviter le message d'erreur si la BD n'existe pas.

Pour utiliser la base de données

```
USE nom_base_de_données;
```

Création et suppression d'une base de données

Création

```
CREATE DATABASE nom_base_de_données;
```

Suppression :

```
DROP DATABASE nom_base_de_données [IF EXISTS];
```

`IF EXISTS` : pour éviter le message d'erreur si la BD n'existe pas.

Pour utiliser la base de données

```
USE nom_base_de_données;
```

Pour afficher toutes les bases de données

```
SHOW DATABASES;
```

Création d'une table

Plusieurs moteurs de table

- `MyISAM` : rapide mais n'utilise pas les clés-étrangères [moteur par défaut de MySQL pour les versions antérieurs à 5.5]
- `InnoDB` : plus lent, mais il utilise les clés-étrangères [moteur par défaut depuis la version 5.5.]
- `Memory` : utilise la RAM pour le stockage de données
- `Merge`
- ...

Création d'une table

Création :

```
CREATE TABLE nom_table (
    nom_colonne1 type,
    ...
    nom_colonnen type,

    [PRIMARY KEY (colonnes_cles_primaires)]
);
```

Création d'une table

Création :

```
CREATE TABLE nom_table (
    nom_colonne1 type,
    ...
    nom_colonnen type,

    [PRIMARY KEY (colonnes_cles_primaires)]
);
```

Dans le cas d'une clé primaire composée d'une seule colonne :

```
CREATE TABLE nom_table (
    nom_colonne1 type PRIMARY KEY,
    ...
    nom_colonnen type
);
```

Création d'une table

Autres propriétés de colonne

- `NOT NULL` : pour indiquer que ce champ doit toujours être rempli (par défaut le null est accepté)
- `AUTO_INCREMENT` : valeur à incrémenter automatiquement, généralement pour les clés primaires
- `DEFAULT` : pour indiquer une valeur par défaut si le champ n'est pas rempli
- `UNIQUE` : nous pouvons avoir d'autres colonnes que la clé primaire dont les valeurs sont uniques (exemple, numéro CNI, numéro PASSEPORT, numéro carte vitale, numéro carte étudiant...)
- `CHECK` : préciser une contrainte sur les valeurs acceptées

Suppression d'une table

Pour supprimer une table

```
DROP TABLE nom_table;
```

Suppression d'une table

Pour supprimer une table

```
DROP TABLE nom_table;
```

Pour supprimer toutes les données d'une table sans supprimer la table

```
TRUNCATE TABLE nom_table;
```

Modification d'une table (existante)

Ajouter une colonne (qui sera la dernière dans la table)

```
ALTER TABLE nom_table
ADD nom_colonne type propriété;
```

Modification d'une table (existante)

Ajouter une colonne (qui sera la dernière dans la table)

```
ALTER TABLE nom_table
ADD nom_colonne type propriété;
```

Ajouter une colonne à une position donnée

```
ALTER TABLE nom_table
ADD nom_colonne type propriété AFTER
   nom_colonne_existante;
```

Modification d'une table (existante)

Ajouter une colonne (qui sera la dernière dans la table)

```
ALTER TABLE nom_table
ADD nom_colonne type propriété;
```

Ajouter une colonne à une position donnée

```
ALTER TABLE nom_table
ADD nom_colonne type propriété AFTER
   nom_colonne_existante;
```

Supprimer une colonne

```
ALTER TABLE nom_table
DROP nom_colonne ;
```

Modification d'une table (existante)

Modifier une colonne (possible de la renommer)

```
ALTER TABLE nom_table
CHANGE ancien_nom nouveau_nom type propriété;
```

Modification d'une table (existante)

Modifier une colonne (possible de la renommer)

```
ALTER TABLE nom_table
CHANGE ancien_nom nouveau_nom type propriété;
```

Modifier une colonne (sans pouvoir la renommer)

```
ALTER TABLE nom_table
MODIFY nom_colonne type propriété;
```

Les contraintes sur tables : les clés étrangères

Au moment de la création de la table

```
CREATE TABLE  nom_table (
    nom_colonne1 type PRIMARY KEY,
    ...
    nom_colonnen type,
    [CONSTRAINT nom_contrainte] FOREIGN KEY (
       nom_colonne) REFERENCES table_dorigine(
       nom_colonne)

);
```

Les contraintes sur tables : les clés étrangères

En modifiant la table

```
ALTER TABLE nom_table
ADD CONSTRAINT nom_contrainte FOREIGN KEY (
   nom_colonne) REFERENCES table_dorigine(
   nom_colonne) ;
```

Les contraintes sur tables : les clés étrangères

En modifiant la table

```
ALTER TABLE nom_table
ADD CONSTRAINT nom_contrainte FOREIGN KEY (
   nom_colonne) REFERENCES table_dorigine(
   nom_colonne) ;
```

Attention

Il faut que les types des attributs référencés soient les mêmes.

Les contraintes sur tables : les clés étrangères

Pour supprimer une clé étrangère

```
ALTER TABLE nom_table
DROP FOREIGN KEY nom_contrainte
```

Les contraintes sur tables : les clés étrangères

Pour supprimer une clé étrangère

```
ALTER TABLE nom_table
DROP FOREIGN KEY nom_contrainte
```

Précisions sur la modification/suppression d'une valeur clé étrangère

```
ALTER TABLE nom_table
ADD CONSTRAINT nom_constraint FOREIGN KEY (
   nom_colonne) REFERENCES nom_table(nom_colonne)
     ON DELETE SET NULL
     ON UPDATE CASCADE;
```

Les contraintes sur tables : Les clés primaires

Nommer la contrainte clé primaire

```
CREATE TABLE nom_table (
    nom_colonne1 type,
    ...
    nom_colonnen type,
    CONSTRAINT nom_constraint PRIMARY KEY (
       colonnes_cles_primaires)
);
```

Les contraintes sur tables : Les clés primaires

Nommer la contrainte clé primaire

```
CREATE TABLE nom_table (
    nom_colonne1 type,
    ...
    nom_colonnen type,
    CONSTRAINT nom_constraint PRIMARY KEY (
       colonnes_cles_primaires)
);
```

Exemple

```
CREATE TABLE employee(
    cin char(6),
    prenom char(20),
    nom char(20),
    CONSTRAINT pk_employee PRIMARY KEY(nom,prenom)
);
```

Les contraintes sur tables : Les clés primaires

Supprimer une clé primaire

```
ALTER TABLE nom_table
DROP PRIMARY KEY;
```

Les contraintes sur tables : Les clés primaires

Supprimer une clé primaire

```
ALTER TABLE nom_table
DROP PRIMARY KEY;
```

Exemple

```
ALTER TABLE employee
DROP PRIMARY KEY ;
```

Les contraintes sur tables : Les clés primaires

Déclarer une clé primaire pour une table déjà existante

```
ALTER TABLE nom_table
ADD [CONSTRAINT [nom_contrainte]] PRIMARY KEY (
   colonnes_cles_primaires);
```

Les contraintes sur tables : Les clés primaires

Déclarer une clé primaire pour une table déjà existante

```
ALTER TABLE nom_table
ADD [CONSTRAINT [nom_contrainte]] PRIMARY KEY (
   colonnes_cles_primaires);
```

Exemple

```
ALTER TABLE employee
ADD CONSTRAINT pk_employee PRIMARY KEY (cin);
```

Pourquoi nommer les contraintes ?

Explication

- Les tables peuvent avoir plusieurs contraintes
- Une contrainte non-nommée aura un nom attribué par le SGBD
- En cas de violation d'une contrainte, le SGBD affiche un message d'erreur + le nom de la contrainte violée
- Si on ne connait pas le nom de la contrainte, on ne saura pas la source du problème

Quelques exemples pratiques

AUTO_INCREMENT

- Par défaut, il commence de 1.
- On peut toujours changer sa valeur initiale : AUTO_INCREMENT = 100

Quelques exemples pratiques

AUTO_INCREMENT

- Par défaut, il commence de 1.
- On peut toujours changer sa valeur initiale : AUTO_INCREMENT = 100

```
CREATE TABLE etudiant(
   num int PRIMARY KEY AUTO_INCREMENT,
   age int
);

ALTER TABLE etudiant AUTO_INCREMENT = 100;
```

Quelques exemples pratiques

Pour rendre une colonne `auto-increment` après création de la table

```
ALTER TABLE etudiant
MODIFY COLUMN num int AUTO_INCREMENT;
```

Quelques exemples pratiques

Exemple de CHECK

```
CREATE TABLE etudiant(
   num int,
   age int,
   CHECK (age < 150)
);
```

Quelques exemples pratiques

Consulter la liste des tables d'une base de données

```
SHOW TABLES;
```

Quelques exemples pratiques

Consulter la liste des tables d'une base de données

```
SHOW TABLES;
```

Afficher le schéma d'une table (la description)

```
DESC[RIBE] nom_table;
```

Quelques exemples pratiques

Consulter la liste des tables d'une base de données

```
SHOW TABLES;
```

Afficher le schéma d'une table (la description)

```
DESC[RIBE] nom_table;
```

ou

```
EXPLAIN nom_table;
```

Quelques exemples pratiques

Consulter la liste des tables d'une base de données

```
SHOW TABLES;
```

Afficher le schéma d'une table (la description)

```
DESC[RIBE] nom_table;
```

ou

```
EXPLAIN nom_table;
```

ou aussi

```
SHOW COLUMNS FROM nom_table;
```

Quelques exemples pratiques

Afficher le nom de la base de données courante

```
SELECT DATABASE();
```

Quelques exemples pratiques

Afficher le nom de la base de données courante

```
SELECT DATABASE();
```

Afficher les détails sur ma base de données

```
USE INFORMATION_SCHEMA;
SELECT TABLE_NAME,
       COLUMN_NAME,
       CONSTRAINT_NAME,
       REFERENCED_TABLE_NAME,
       REFERENCED_COLUMN_NAME
FROM KEY_COLUMN_USAGE
WHERE TABLE_SCHEMA = "nom_BD"
      AND TABLE_NAME = "nom_table"
      AND REFERENCED_COLUMN_NAME IS NOT NULL;
```

Quelques exemples pratiques

Pour changer le moteur d'une table

```
ALTER TABLE nom_table
engine = InnoDB;
```

Quelques exemples pratiques

Pour changer le moteur d'une table

```
ALTER TABLE nom_table
engine = InnoDB;
```

Pour changer l'encodage d'une table

```
ALTER TABLE nom_table
charset = utf8;
```

Quelques exemples pratiques

Pour changer le moteur d'une table

```
ALTER TABLE nom_table
engine = InnoDB;
```

Pour changer l'encodage d'une table

```
ALTER TABLE nom_table
charset = utf8;
```

Pour consulter le code de création d'une table

```
SHOW CREATE TABLE nom_table;
```