

5. Le langage SQL




Introduction

- **SQL** : **S**tructured **Q**uery **L**anguage
- Inventé chez IBM (centre de recherche d'Almaden en Californie), en 1974 par Astrahan & Chamberlin dans le cadre de System R
- Le langage SQL est normalisé
 - SQL2: adopté (SQL 92)
 - SQL3: adopté (SQL 99) → SQL-2008
- Standard d'accès aux bases de données relationnelles

SQL : Trois langages en un

- SQL = Langage de définition de données (LDD)
 - CREATE TABLE
 - ALTER TABLE
 - DROP TABLE
- SQL = Langage de manipulation de données (LMD)
 - INSERT INTO
 - UPDATE
 - DELETE FROM
- SQL = Langage de requêtes (LMD) /ou LID Langage d'Intérogation de données
 - SELECT ... FROM ... WHERE ...
 - Sélection
 - Projection
 - Jointure
 - Les agrégats

Terminologie

- Relation  Table
- Tuple  Ligne
- Attribut  Colonne

SQL (LDD)

Un langage de définition de données

Types de données

- Une base de données contient des **tables**
- Une table est organisée en **colonnes**
- Une colonne stocke des **données**

- Les données sont séparées en plusieurs **types** !

Type des colonnes (en MySQL)

- Numériques

- NUMERIC : idem DECIMAL //valeur exacte
- DECIMAL. Possibilité DECIMAL(M,D) M chiffre au total //valeur exacte
- INTEGER
 - TINYINT 1 octet (de -128 à 127)
 - SMALLINT 2 octets (de -32768 à 32767)
 - MEDIUMINT 3 octets (de -8388608 à 8388607)
 - INT 4 octets (de -2147483648 à 2147483647)
 - BIGINT 8 octets (de -9223372036854775808 à 9223372036854775807)
 - Possibilité de donner la taille de l'affichage : INT(6) => 674 s'affiche 000674
 - Possibilité de spécifier UNSIGNED
 - INT UNSIGNED => de 0 à 4294967296
- FLOAT : 4 octets par défaut. Possibilité d'écrire FLOAT(P) //valeur approchée
- REAL : 8 octets (4 octets dans d'autres SGBD) //valeur approchée
- DOUBLE : 8 octets //valeur approchée
 - 56,6789 → MySQL stockera 56,67890000000000000001

Type des colonnes (en MySQL)

■ Date et Heure

■ DATETIME

- AAAA-MM-JJ HH:MM:SS
- de 1000-01-01 00:00:00 à '9999-12-31 23:59:59

■ DATE

- AAAA-MM-JJ
- de 1000-01-01 à 9999-12-31

■ TIMESTAMP

- Date sans séparateur AAAAMMMJJHHMMSS

■ TIME

- HH:MM:SS (ou HHH:MM:SS)
- de -838:59:59 à 838:59:59

■ YEAR

- YYYY
- de 1901 à 2155

Type des colonnes (en MySQL)

■ Chaînes

■ CHAR(n) $1 \leq n \leq 255$

■ VARCHAR(n) $1 \leq n \leq 255$

Exemple :

	CHAR(4)		VARCHAR(4)	
Valeur	Stocké e	Taille	Stocké e	Taille
"	' '	4 octets	"	1 octets
'ab'	'ab '	4 octets	'ab'	3 octets
'abcd'	'abcd'	4 octets	'abcd'	5 octets
'abcde f'	'abcd'	4 octets	'abcd'	5 octets

Type des colonnes (en MySQL)

■ Chaînes

- TINYBLOB Taille < 2^8 caractères
- BLOB Taille < 2^8 caractères
- MEDIUMBLOB Taille < 2^{24} caractères
- LONGBLOB Taille < 2^{32} caractères

- TINYTEXT Taille < 2^8 caractères
- TEXT Taille < 2^8 caractères
- MEDIUMTEXT Taille < 2^{24} caractères
- LONGTEXT Taille < 2^{32} caractères

Les tris faits sur les BLOB tiennent compte de la casse, contrairement aux tris faits sur les TEXT.

Type des colonnes (en MySQL)

■ ENUM //valeur décimal

- Enumération
- ENUM("un", "deux", "trois")
- Valeurs possibles : "" , "un", "deux", "trois"
- Au plus 65535 éléments

■ SET //valeur binaire

- Ensemble
- SET("un", "deux")
- Valeurs possibles : "" , "un", "deux", "un,deux"
- Au plus 64 éléments

Type des colonnes (en MySQL)

- Dans quelles situations faut-il utiliser ENUM ou SET ?

JAMAIS !!

- il faut toujours éviter autant que possible les fonctionnalités propres à un seul SGBD.

Un langage de définition de données

Commandes pour Créer et supprimer une base de données:

- **CREATE DATABASE** *nom_base*: créer une base de données,
 - **CREATE DATABASE** *bibliotheque* **CHARACTER SET** *'utf8'* :
créer une base de données et encoder les tables en UTF-8
-
- **DROP DATABASE** *bibliotheque* : supprimer la base de données,
 - **DROP DATABASE IF EXISTS** *bibliotheque* ;
-

Utilisation d'une base de données

- **USE** *bibliotheque* ;

Un langage de définition de données

Commandes pour créer, modifier et supprimer les éléments du schéma:

- **CREATE TABLE** : créer une table (une relation),
- **CREATE VIEW** : créer une vue particulière sur les données à partir d'un SELECT,
- **DROP {TABLE | VIEW}** : supprimer une table ou une vue,
- **ALTER {TABLE | VIEW}** : modifier une table ou une vue.

CREATE TABLE

Commande créant une table en donnant son **nom**, ses **attributs** et ses **contraintes**:

```
CREATE TABLE [IF NOT EXISTS] nom_table (  
    colonne1 description_colonne1,  
    [colonne2 description_colonne2,  
    colonne3 description_colonne3,  
    ...,]  
    [PRIMARY KEY (colonne_clé_primaire)]  
)  
[ENGINE=moteur];
```

Les moteurs de tables

- Les moteurs de tables sont une spécificité de MySQL. Ce sont des moteurs de stockage. Cela permet de gérer différemment les tables selon l'utilité qu'on en a.
- Les deux moteurs les plus connus sont **MyISAM** et **InnoDB**.
- **MyISAM** : C'est le moteur par défaut. Les commandes sont particulièrement rapides sur les tables utilisant ce moteur. Cependant, il ne gère pas certaines fonctionnalités importantes comme **les clés étrangères**.
- **InnoDB** : Plus lent et plus gourmand en ressources que MyISAM, ce moteur **gère les clés étrangères**

CREATE TABLE

Exemples:

```
CREATE TABLE Emprunteur(  
  id SMALLINT UNSIGNED NOT NULL  
    AUTO_INCREMENT,  
  nom VARCHAR(20) NOT NULL,  
  prenom VARCHAR(15) NOT NULL,  
  annee_insc YEAR DEFAULT 2021,  
  PRIMARY KEY (id)  
)  
ENGINE=INNODB;
```

CREATE TABLE

Exemples: Autre possibilité

```
CREATE TABLE Emprunteur(  
    id SMALLINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    nom VARCHAR(20) NOT NULL,  
    prenom VARCHAR(15) NOT NULL,  
    annee_insc YEAR DEFAULT 2021,  
)  
ENGINE=INNODB;
```

Vérifications

Deux commandes pour vérifier la création des tables :

- **SHOW TABLES;**
 - liste les tables de la base de données
- **DESCRIBE Emprunteur;**
 - liste les colonnes de la table avec leurs caractéristiques

DROP TABLE

- **DROP TABLE** : Supprimer une table
 - supprime la table et tout son contenu
- **DROP TABLE** nom_table [**CASCADE CONSTRAINTS**];
- **CASCADE CONSTRAINTS**
 - Supprime toutes les contraintes référençant une clé primaire (primary key) ou une clé unique (UNIQUE) de cette table
 - Si on cherche à détruire une table dont certains attributs sont référencés sans spécifier CASCADE CONSTRAINT, on a un message d'erreur.

ALTER TABLE

- Modifier la définition d'une table:
 - Changer le nom de la table
mot clé : **RENAME**
 - Ajouter une colonne ou une contrainte
mot clé : **ADD**
 - Modifier une colonne ou une contrainte
mot clé : **MODIFY/CHANGE**
 - Supprimer une colonne ou une contrainte
mot clé : **DROP**
 - renommer une colonne ou une contrainte
mot clé : **RENAME**

ALTER TABLE

Syntaxe :

ALTER TABLE nom-table

```
{  RENAME TO nouveau-nom-table
  | ADD (( nom-col type-col [DEFAULT valeur] [contrainte-col]))*
  | MODIFY (nom-col [type-col] [DEFAULT valeur] [contrainte-col])*
  | DROP COLUMN nom-col [CASCADE CONSTRAINTS]
  | RENAME COLUMN old-name TO new-name
};
```

Ajout et suppression d'une colonne

ALTER TABLE nom_table

ADD [COLUMN] nom_colonne description_colonne;

■ Exemple :

ALTER TABLE Emprunteur

ADD COLUMN date_emprunt **DATE NOT NULL** ;

Ajout et suppression d'une colonne

ALTER TABLE nom_table

DROP [COLUMN] nom_colonne;

■ Exemple :

ALTER TABLE Emprunteur

DROP COLUMN date_emprunt ;

Modification d'une colonne

```
ALTER TABLE nom_table
```

```
CHANGE ancien_nom nouveau_nom description_colonne;
```

■ Exemple :

```
ALTER TABLE Emprunteur
```

```
CHANGE nom nom_famille VARCHAR(10) NOT NULL ;
```

Changement du type de données

ALTER TABLE nom_table

CHANGE ancien_nom nouveau_nom description_colonne;

Ou

ALTER TABLE nom_table

MODIFY nom_colonne description_colonne;

Des exemples pour illustrer :

ALTER TABLE Emprunteur

CHANGE nom nom_famille **VARCHAR**(10) **NOT NULL** ;

→ Changement du type + changement du nom

ALTER TABLE Emprunteur

CHANGE id id **BIGINT NOT NULL** ;

→ Changement du type sans renommer

ALTER TABLE Emprunteur

MODIFY id **BIGINT NOT NULL AUTO_INCREMENT**;

→ Ajout de l'auto-incrémentation

ALTER TABLE Emprunteur

MODIFY nom **VARCHAR**(30) **NOT NULL DEFAULT** 'Anonyme';

→ Changement de la description

```
CREATE TABLE Emprunteur(  
  id SMALLINT UNSIGNED  
    AUTO_INCREMENT PRIMARY KEY,  
  nom VARCHAR(20) NOT NULL,  
  prenom VARCHAR(15) NOT NULL,  
  annee_insc YEAR DEFAULT 2021,  
)  
ENGINE=INNODB;
```

Renommer une table

- ... **RENAME TO** nouveau-nom-table

- Exemple :

ALTER TABLE Emprunteur **RENAME TO** Emprunteurs ;

Les clé étrangères

```
CREATE TABLE [IF NOT EXISTS] Nom_table (  
    colonne1 description_colonne1,  
    [colonne2 description_colonne2,  
    colonne3 description_colonne3,  
    ...],  
    [ [CONSTRAINT [symbole_contrainte]] FOREIGN  
    KEY (colonne(s)_clé_étrangère) REFERENCES  
    table_référence (colonne(s)_référence)]  
)  
[ENGINE=moteur];
```

Exemple

- On imagine les tables **Client** et **Commande**,
- pour créer la table **Commande** avec une **clé étrangère** ayant pour **référence la colonne numero** de la table **Client**, on utilisera :

```
CREATE TABLE Commande (  
    numero INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,  
    client INT UNSIGNED NOT NULL,  
    produit VARCHAR(40),  
    quantite SMALLINT DEFAULT 1,  
    CONSTRAINT fk_client_numero          -- On donne un nom à notre clé  
    FOREIGN KEY (client)                 -- Colonne sur laquelle on crée la clé  
    REFERENCES Client(numero)           -- Colonne de référence  
)  
ENGINE=InnoDB;                          -- MyISAM interdit, je le rappelle encore une fois  
!
```

Après création de la table

ALTER TABLE Commande

ADD CONSTRAINT fk_client_numero **FOREIGN KEY**
(client) **REFERENCES** Client(numero);

Suppression d'une clé étrangère

ALTER TABLE nom_table

DROP FOREIGN KEY symbole_contrainte;

Petit TP

Créer la base de données et les différentes tables de ce schéma relationnel:

- Personnes(PersonneID, Nom, Age, Adresse);
- Commandes(CommandeID, NumCommande, PersonneID);



Petit TP

■ Solution:

```
CREATE DATABASE gestion;  
USE gestion;
```

```
CREATE TABLE Personnes (  
    PersonneID int AUTO_INCREMENT PRIMARY KEY,  
    Nom VARCHAR(20) NOT NULL,  
    Age int,  
    Adresse VARCHAR(100)  
);
```

```
CREATE TABLE Commandes (  
    CommandeID int AUTO_INCREMENT PRIMARY KEY,  
    NumCommande int NOT NULL,  
    PersonneID int,  
    FOREIGN KEY (PersonneID) REFERENCES Personnes(PersonneID)  
);
```