SQL(Structured Query Language) **Enseignant : Mustapha Jaballah**
Niveau : 3ème SI
Matière STI

1. Types de Données

Types de données SQL

Type	Description
`INT`	Entier
`DECIMAL`	Réel (nombre à virgule)
`CHAR`	Chaîne de caractères de **longueur fixe**
`VARCHAR`	Chaîne de caractères de longueur variable (longueur max fixée)
`TEXT`	Chaîne de caractères de longueur variable (sans limite)
`DATE`	Date (`AAAA-MM-JJ`)
`TIME`	Temps (`HH:MM:SS`)
`DATETIME`	Date et Heure (`AAAA-MM-JJ HH:MM:SS`)

2. Contraintes d'Intégrité

Les contraintes garantissent la **validité** et la **cohérence** des données.

Contraintes disponibles

Contrainte	Rôle
`NOT NULL`	Interdit une valeur nulle
`DEFAULT`	Attribue une valeur par défaut
`CHECK`	Vérifie une condition sur les valeurs
`PRIMARY KEY`	Définit la clé primaire (identifiant unique)
`UNIQUE`	Impose l'unicité des valeurs d'une colonne
`FOREIGN KEY`	Définit une clé étrangère (lien vers une autre table)
`REFERENCES`	Fait référence à la clé primaire d'une autre table
`ON UPDATE CASCADE`	Met à jour automatiquement la clé étrangère si la clé primaire change
`ON DELETE CASCADE`	Supprime les enregistrements liés si l'enregistrement parent est supprimé

3. Opérateurs

3.1. Opérateurs de Comparaison

Comparaison

Op.	Signification	Op.	Signification
`=`	Égal à	`<>`	Différent de
`>`	Supérieur à	`<`	Inférieur à
`>=`	Supérieur ou égal	`<=`	Inférieur ou égal
`IN`	Dans un ensemble	`BETWEEN`	Entre deux valeurs
`LIKE`	Recherche de motif	`IS`	Teste la valeur NULL

3.2. Opérateurs Logiques

Logique

Opérateur	Description
`AND`	ET logique — les deux conditions doivent être vraies
`OR`	OU logique — au moins une condition doit être vraie
`NOT`	NON logique — inverse la condition

4. Fonctions SQL

4.1. Fonctions sur les Dates

Fonctions date

Fonction	Description
`DATE()`	Retourne la partie date d'une chaîne DATE ou DATETIME
`DAY()`	Retourne le jour du mois
`MONTH()`	Retourne le mois
`YEAR()`	Retourne l'année
`NOW()`	Retourne la date et l'heure courantes
`ADDDATE()`	Ajoute un nombre de jours ou un intervalle à une date
`DATEDIFF()`	Retourne la différence entre deux dates (en jours)

5. Langage de Définition des Données (LDD)

5.1. Gestion des Bases de Données

```
1 CREATE DATABASE nom_base;    -- Cree une base de donnees
2 DROP DATABASE nom_base;      -- Supprime une base de donnees
```

5.2. Création de Table

Syntaxe

```
CREATE TABLE nom_table (
    colonne1  type  [contrainte],
    colonne2  type  [contrainte],
    ...
    [CONSTRAINT nom contrainte, ...]
);
```

Exemple complet avec AUTO_INCREMENT et clé étrangère :

```
CREATE TABLE classe (
    id     INT AUTO_INCREMENT PRIMARY KEY,
    nom    VARCHAR(50) NOT NULL,
    niveau VARCHAR(20)
);

CREATE TABLE etudiant (
    id               INT AUTO_INCREMENT PRIMARY KEY,
    nom              VARCHAR(50) NOT NULL,
    prenom           VARCHAR(50) NOT NULL,
    date_naissance   DATE,
    email            VARCHAR(100) UNIQUE,
    note             DECIMAL CHECK (note >= 0 AND note <= 20),
    classe_id        INT,
    date_inscription DATETIME DEFAULT NOW(),
    FOREIGN KEY (classe_id) REFERENCES classe(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

5.3. Modification de Table (ALTER)

```
-- Ajouter une colonne
ALTER TABLE etudiant ADD telephone VARCHAR(15);

-- Modifier le type d'une colonne (* MODIFY selon le SGBD)
ALTER TABLE etudiant ALTER telephone VARCHAR(20);

-- Renommer une colonne (** CHANGE selon le SGBD)
ALTER TABLE etudiant RENAME telephone TO tel;

-- Supprimer une colonne
ALTER TABLE etudiant DROP tel;

-- Ajouter une contrainte
ALTER TABLE etudiant ADD CONSTRAINT ck_note CHECK (note >= 0);

-- Supprimer une contrainte
ALTER TABLE etudiant DROP CONSTRAINT ck_note;

-- Activer / Desactiver une contrainte
ALTER TABLE etudiant ENABLE  CONSTRAINT ck_note;
ALTER TABLE etudiant DISABLE CONSTRAINT ck_note;

```

```sql
-- Supprimer une table
DROP TABLE etudiant;
```

6. Langage de Manipulation des Données (LMD)

6.1. Sélection (SELECT)

Syntaxe complète du SELECT

```sql
SELECT [DISTINCT] expression [, col, ...] [[AS] alias]
FROM   table1 [[AS] alias] [, table2, ...]
[WHERE    condition]
[GROUP BY critere]
[HAVING   condition]
[ORDER BY expression [ASC | DESC]];
```

Clause	Rôle
`WHERE`	Filtre les lignes selon une condition
`ORDER BY`	Trie le résultat (ASC par défaut, DESC pour décroissant)
`DISTINCT`	Élimine les doublons dans le résultat

Exemples :

```sql
-- Selection simple et alias
SELECT nom AS NomFamille, prenom AS Prenom FROM etudiant;

-- Filtres et operateurs
SELECT * FROM etudiant WHERE note BETWEEN 10 AND 15;
SELECT * FROM etudiant WHERE nom IN ('Dupont', 'Martin');
SELECT * FROM etudiant WHERE email LIKE '%@email.com';
SELECT * FROM etudiant WHERE date_naissance IS NULL;

-- Tri
SELECT * FROM etudiant ORDER BY nom ASC, prenom DESC;

-- Fonctions de date
SELECT nom,
       DAY(date_naissance)   AS Jour,
       MONTH(date_naissance) AS Mois,
       YEAR(date_naissance)  AS Annee,
       DATEDIFF(NOW(), date_naissance) AS AgeEnJours
FROM etudiant;

-- Jointure avec alias de table
SELECT e.nom, e.prenom, c.nom AS Classe
FROM etudiant e, classe c
WHERE e.classe_id = c.id;
```

6.2. Insertion (INSERT)

```sql
INSERT INTO etudiant (nom, prenom, date_naissance, email, note)
VALUES ('Dupont', 'Jean', '2000-05-15', 'jean.dupont@email.com', 15.5);

-- Avec valeur par defaut pour date_inscription (NOW())
INSERT INTO etudiant (nom, prenom, email)
VALUES ('Martin', 'Marie', 'marie.martin@email.com');
```

6.3. Mise à Jour (UPDATE)

```sql
-- Mise a jour simple
UPDATE etudiant SET note = 16.5 WHERE id = 1;

-- Mise a jour avec expression
UPDATE etudiant SET note = note + 1 WHERE note < 18;

-- Mise a jour de plusieurs colonnes
UPDATE etudiant
SET classe_id = 2, date_inscription = NOW()
WHERE nom = 'Martin' AND prenom = 'Marie';
```

6.4. Suppression (DELETE)

```sql
-- Suppression avec condition
DELETE FROM etudiant WHERE note < 5;

-- Suppression de tous les enregistrements
DELETE FROM etudiant;
```

7. Clé Primaire Composite et Clés Étrangères

Il est possible de définir une **clé primaire composée de deux colonnes** qui sont **simultanément des clés étrangères**. C'est le cas typique d'une **table d'association** (relation plusieurs-à-plusieurs).

Principe

La table `inscription` relie un `etudiant` à une `classe`.
Les deux colonnes `etudiant_id` et `classe_id` forment ensemble la clé primaire,
et chacune est une clé étrangère vers sa table respective.

```sql
-- Tables parentes
CREATE TABLE etudiant (
    id     INT AUTO_INCREMENT PRIMARY KEY,
    nom    VARCHAR(50) NOT NULL,
    prenom VARCHAR(50) NOT NULL
);

CREATE TABLE classe (
    id     INT AUTO_INCREMENT PRIMARY KEY,
    nom    VARCHAR(50) NOT NULL,
    niveau VARCHAR(20)
);

-- Table d'association : cle primaire composite
```

```sql
-- etudiant_id et classe_id sont a la fois :
--   * la cle primaire (ensemble)
--   * des cles etrangeres (chacune vers sa table)
CREATE TABLE inscription (
    etudiant_id      INT,
    classe_id        INT,
    date_inscription DATE,
    PRIMARY KEY (etudiant_id, classe_id),
    FOREIGN KEY (etudiant_id) REFERENCES etudiant(id)
        ON DELETE CASCADE,
    FOREIGN KEY (classe_id)   REFERENCES classe(id)
        ON UPDATE CASCADE
);
```

Points clés à retenir

Point	Explication
PRIMARY KEY composite	Définie après les colonnes avec `PRIMARY KEY (col1, col2)`
Unicité	La combinaison `(etudiant_id, classe_id)` doit être unique
Double rôle	Chaque colonne est à la fois clé primaire **et** clé étrangère
Cascade	Chaque clé étrangère a sa propre règle `ON DELETE / ON UPDATE`

8. Récapitulatif — Modèle Complet

```sql
-- Modele complet illustrant toutes les contraintes
CREATE TABLE categorie (
    id  INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL
);

CREATE TABLE produit (
    id           INT AUTO_INCREMENT PRIMARY KEY,
    nom          VARCHAR(50) NOT NULL,          -- NOT NULL
    age          INT DEFAULT 0,                 -- DEFAULT
    email        VARCHAR(100) UNIQUE,           -- UNIQUE
    note         DECIMAL CHECK (note >= 0 AND note <= 20), -- CHECK
    categorie_id INT,
    FOREIGN KEY (categorie_id) REFERENCES categorie(id)
        ON UPDATE CASCADE                       -- Cascade mise a jour
        ON DELETE CASCADE                       -- Cascade suppression
);

-- Requete avec jointure et tri
SELECT p.nom, c.nom AS Categorie
FROM produit p, categorie c
WHERE p.categorie_id = c.id
  AND p.note IS NOT NULL
ORDER BY p.note DESC;
```