L'insertion, la modification et la suppression de données dans MariaDB

Rappels

- Le langage SQL est composé d'instructions dont chacune appartient à l'une des catégories suivantes:
    - Langage de définition de données (DDL pour *Data Definition Language*)
    - Langage de manipulation de données (DML pour *Data Manipulation Language*)
    - Langage de contrôle de données (DCL pour *Data Control Language*)
    - Langage de contrôle des transactions (TCL pour *Transaction Control Language*)
- Une requête SQL commence toujours par un verbe et se termine toujours par un point-virgule (`;`).
- Une requête de type DDL est formée de mots-clés et d'identifiants.
- Un mot-clé est un mot qui fait partie du langage SQL, tandis qu'un identifiant est un nom donné par l'utilisateur à un élément de la base de données, par exemple: un nom de table, un nom d'attribut, etc.
- Le DDL de MariaDB comprend les commandes `CREATE`, `ALTER` et `DROP`.

DML

- Alors que le DDL concerne la structure des données, le DML, lui, concerne les donneés stockées à l'aide de cette structure.
- Le DML de MariaDB comprend quatre commandes SQL:
    - `INSERT` qui permet d'insérer de nouvelles données dans une table;
    - `SELECT` qui permet de sélectionner (lire) des données dans une table;
    - `UPDATE` qui permet de modifier des données existantes d'une table;
    - `DELETE` qui permet de supprimer des données dans une table.
- On dit parfois que le DML concerne les opérations de type **CRUD** (**C**reate, **R**ead, **U**pdate, **D**elete). Attention cependant, il ne faut pas confondre le *Create* dans *CRUD* avec la commande `CREATE` du DDL! On parle ici de création de données, donc d'insertion (commande **INSERT**). L'opération *Read* correspond pour sa part à la commande `SELECT`.
- Une requête DML est composée principalement de mots-clés, d'identifiants et de valeurs.

Base de données utilisée dans les exemples

Pour les exemples suivants, nous allons utiliser la base de données "librairie" créée par les requêtes suivantes:

```
DROP DATABASE IF EXISTS librairie;
```

```sql
CREATE DATABASE librairie;

USE librairie;

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

CREATE TABLE langue (
    code CHAR(2) PRIMARY KEY,
    nom VARCHAR(20) NOT NULL UNIQUE
);

CREATE TABLE livre (
    isbn CHAR(13) PRIMARY KEY,
    titre VARCHAR(50) NOT NULL,
    id_editeur INT,
    date_parution DATE NOT NULL,
    description TEXT,
    code_langue CHAR(2),
    prix DECIMAL(5,2) UNSIGNED,
        -- prix: Maximum de 5 chiffres dont 2 après la virgule
    FOREIGN KEY (id_editeur) REFERENCES editeur(id)
        ON DELETE CASCADE
        ON UPDATE RESTRICT,
    FOREIGN KEY (code_langue) REFERENCES langue(code)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE auteur (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50) NOT NULL,
    prenom VARCHAR(50) NOT NULL
);

CREATE TABLE auteur_livre (
    id_auteur INT NOT NULL,
    isbn_livre CHAR(13) NOT NULL,
    PRIMARY KEY (id_auteur, isbn_livre),
    FOREIGN KEY (id_auteur) REFERENCES auteur(id)
        ON DELETE CASCADE
        ON UPDATE RESTRICT,
    FOREIGN KEY (isbn_livre) REFERENCES livre(isbn)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Insertion de données (`INSERT`)

La commande `INSERT` utilise la syntaxe suivante:

```
INSERT INTO nom_table(nom_champ_1, nom_champ_2, ...)
    VALUES(valeur_champ_1, valeur_champ_2, ...);
```

Par exemple, pour ajouter l'éditeur "Gallimard Jeunesse" à la table `editeur`:

```
INSERT INTO editeur(nom) VALUES('Gallimard Jeunesse');
```

Dans l'exemple ci-dessus, on peut remarquer deux choses:

- Nous n'avons pas assigné de valeur au champ `id`, puisque celui-ci obtiendra une valeur automatiquement.
    - Il est à noter qu'on peut toujours laisser des champs vides lors d'une requête d'insertion, à condition que ces champs n'aient pas une contrainte de non-nullité (`NOT NULL`).
- Nous avons placé `Gallimard Jeunesse` entre apostrophes (`''`). Cela est toujours nécessaire pour les chaînes de caractères.

On peut aussi insérer plusieurs entrées à la fois dans une table:

```
INSERT INTO langue(code, nom)
    VALUES ('fr', 'Français'), ('en', 'Anglais'), ('es', 'Espagnol');
```

Ici, nous avons rempli tous les champs des trois entrées que nous avons ajoutées. Nous aurions donc pu omettre les noms des champs, à condition de respecter l'ordre dans lequel ils ont été définis lors de la création de la table:

```
INSERT INTO langue
    VALUES ('fr', 'Français'), ('en', 'Anglais'), ('es', 'Espagnol');
```

Voici un exemple plus complexe avec différents types de données:

```
/*
 Avant d'insérer un livre, il faut connaître l'ID qui a été généré pour son
éditeur. Aux fins de l'exemple, supposons que l'éditeur « Gallimard Jeunesse » a
la valeur 1 comme ID.
*/
INSERT INTO livre(isbn, titre, id_editeur, date_parution, description,
code_langue, prix)
    VALUES(
        '0747532699',
        'Harry Potter à l''école des sorciers',
        1,
        '1998-10-09',
        CONCAT( -- Concaténer (coller ensemble) des chaînes de caractères
```

```
            'Orphelin vivant chez son oncle et sa tante qui ne l\'aiment guère, '
            'Harry découvre qu\'il est magicien. Il voit son existence ',
            'bouleversée par l\'arrivée d\'un géant, Hagrid, qui l\'emmène ',
            'à l\'école pour sorciers de Poudlard.'
        ),
        'fr',
        16.95
    );
```

Syntaxe alternative

On peut aussi utiliser la syntaxe suivante pour effectuer une insertion:

```
INSERT INTO livre
    SET titre = 'WordPress pour les nuls',
        isbn = '9782412090121',
        date_parution = '2023-09-22',
        id_editeur = 2,
        prix = 49.95;
```

Modifier des données (UPDATE)

La commande `UPDATE` utilise la syntaxe suivante:

```
UPDATE nom_table
    SET nom_champ = nouvelle_valeur, nom_champ = nouvelle_valeur,...
    WHERE filtre
```

Le filtre de la clause `WHERE` indique quelle(s) entrée(s) de la table doivent être modifiées. **Si on ne met pas de clause `WHERE`, toutes les entrées de la table seront modifiées.**

Modifier une seule entrée

Lorsqu'on veut modifier une seule entrée de la table, on utilise généralement la clé primaire de cette entrée dans notre filtre. Par exemple:

```
-- Changer le titre du livre "WordPress pour les nuls" pour "WordPress pour les
nuls 6e édition"
UPDATE livre
    SET titre = 'WordPress pour les nuls 6e édition'
    WHERE isbn = '9782412090121';
```

Rien ne nous empêche cependant d'utiliser n'importe quel autre champ de la table:

```sql
-- Changer le nom de l'éditeur "Pearson" pour "Pearson Education France"
UPDATE editeur SET nom = 'Pearson Education France' WHERE nom = 'Pearson';

-- Donner la langue "Français" à tous les livres dont la langue est inconnue
UPDATE livre SET code_langue = 'fr' WHERE code_langue IS NULL;
```

On peut modifier autant de champs qu'on veut:

```sql
-- Donner le prix 16.95, l'éditeur "Gallimard Jeunesse" ainsi qu'une description
au livre "Harry Potter et la Chambre des secrets"
UPDATE livre
    SET id_editeur = 1,
        prix = 16.95,
        description = CONCAT(
            'Une rentrée fracassante en voiture volante, une étrange '
            'malédication qui s\'abat sur les élèves, cette deuxième année à ',
            'l\'école des sorciers ne s\'annonce pas de tout repos ! ',
            'Entre les cours de potions magiques, les matches de ',
            'Quidditch et les combats de mauvais sorts, ',
            'Harry et ses amis Ron et Hermione trouveront-ils ',
            'le temps de percer le mystère de la Chambre des Secrets ?'
        )
    WHERE isbn = '0747538492';
```

Modifier plusieurs entrées à la fois

Si on veut modifier plusieurs entrées de la table en même temps, la syntaxe est exactement la même. Il suffit d'utiliser un filtre qui s'applique à plusieurs entrées. Par exemple:

```sql
-- Changer le prix de tous les livres de l'éditeur #4 pour 10$
UPDATE livre SET prix = 10 WHERE id_editeur = 4;
```

Supprimer des données (`DELETE`)

La commande `DELETE` utilise la syntaxe suivante:

```sql
DELETE FROM nom_table WHERE filtre
```

Par exemple:

```sql
-- Supprimer la langue "Espagnol"
DELETE FROM langue WHERE code = 'es';
```

Les filtres

Vous aurez remarqué qu'autant la commande `UPDATE` que la commande `DELETE` utilisent des filtres. La commande `SELECT`, que nous verrons au prochain cours, utilise aussi des filtres. Or, pour toutes ces commandes, il est possible d'utiliser des filtres plus complexes. En voici un exemple:

```
-- Supprimer les livres dont l'ISBN est 0747532699 ou 0747538492
DELETE FROM livre WHERE isbn = '0747532699' OR isbn = '0747538492';
```

Dans ce filtre, nous avons combiné deux **conditions** à l'aide de l'**opérateur logique** `OR`. Voici les opérateurs logiques disponibles sur MariaDB:

Opérateur logique	**Description**	**Exemple**
`OR` ou `||`	Le "ou logique". La combinaison des deux conditions est vraie si au moins une des conditions est vraie.	`WHERE langue = 'fr' OR id_editeur = 4`
`AND` ou `&&`	Le "et logique". La combinaison des deux conditions est vraie si les deux conditions sont vraies.	`WHERE prix = 19.99 AND id_editeur = 4`
`XOR`	Le "ou exclusif". La combinaison des deux conditions est vraie si seulement une des deux conditions est vraie. Très rarement utilisé.	`WHERE langue = 'fr' XOR id_editeur = 4`
`NOT` ou `!`	L'opérateur de négation. Permet de filtrer sur les données qui ne respectent PAS la condition.	`WHERE NOT (langue = 'fr' OR id_editeur = 4)`

Il est aussi possible de combiner plus de deux conditions à l'aide d'opérateurs logiques, par exemple:

```
DELETE FROM livre
    WHERE   isbn = '0747532699'
            OR isbn = '0747538492'
            OR id_editeur = 3;
```

Attention cependant si vous mélangez plusieurs opérateurs (ex: des `OR` et des `AND`), car il existe un ordre de priorité entre eux. Il vaut donc mieux dans ce cas ajouter des parenthèses pour éviter toute confusion:

```
DELETE FROM livre
    WHERE (id_editeur = 1 AND prix = 19.99)
        OR (id_editeur = 2 AND prix = 29.99)
```

En plus des opérateurs logiques, il existe aussi plusieurs **opérateurs de comparaison**. Nos exemples précédents ont tous utilisé l'opérateur de comparaison `=`, qui signifie « est égal à », mais il en existe d'autres. En voici quelques-uns:

- `!=` ou `<>`: est différent de
- `<` : est plus petit que
- `<=` : est plus petit ou égal à
- `>` : est plus grand que
- `>=` : est plus grand ou égal à

Par exemple:

```
-- Supprimer les livres en anglais qui coûtent moins de 20$
DELETE FROM livre WHERE code_langue = 'en' AND prix < 20;
```

Voir cette page de la documentation de MariaDB pour la liste complète des opérateurs de comparaison.

Références

- Documentation officielle du DML de MariaDB
- Tutoriel sur MySQL de W3Schools