La sélection de données dans MariaDB

Rappels

- 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.
- La commande `INSERT` utilise la syntaxe suivante:

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

- La commande `UPDATE` utilise la syntaxe suivante:

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

- La commande `DELETE` utilise la syntaxe suivante:

```
DELETE FROM nom_table WHERE filtre
```

- Les commandes `UPDATE` et `DELETE` utilisent toutes les deux une clause `WHERE` avec un filtre.
- Un filtre peut utiliser des opérateurs logiques (`AND`, `OR`, `NOT`, `XOR`) et des opérateurs de comparaison (ex: `=`, `!=`, `<`, `<=`, `>`, `>=`, etc).
- La commande `SELECT` utilise aussi un filtre, avec les mêmes opérateurs.

Table utilisée dans les exemples

Pour les exemples suivants, nous allons utiliser la table "livre" créée par la requête suivante:

```
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
);
```

Sélection de données (SELECT)

Liste de sélection

Dans sa forme la plus simple, la commande SELECT utilise la syntaxe suivante:

```
SELECT * FROM nom_table;
```

L'astérisque (*) signifie "tous les champs". Cette requête permet donc de récupérer les valeurs de tous les champs de toutes les lignes d'une table. Si on exécutait cette commande dans un client SQL, nous verrions s'afficher la table au complet.

On peut aussi préciser la liste des champs à récupérer au lieu de l'astérisque. Par exemple, si on veut afficher l'ISBN et le titre de tous les livres, nous ferons:

```
SELECT isbn, titre FROM livre;
```

Il est possible qu'une requête SELECT retourne plusieurs résultats identiques. Par exemple, la requête SELECT titre FROM livre pourrait retourner plusieurs fois le même titre, puisque plusieurs livres dans la table peuvent avoir le même titre. On peut ajouter le mot-clé DISTINCT pour éliminer les doublons:

```
SELECT DISTINCT titre FROM livre;
```

Filtre

On peut filtrer les lignes qui seront récupérées à l'aide d'une clause WHERE. Celle-ci fonctionne de la même façon que pour les requêtes UPDATE et DELETE. Par exemple, si on veut récupérer seulement les livres de l'éditeur #1:

```
SELECT * FROM livre WHERE id_editeur = 1;
```

Comme pour UPDATE et DELETE, on peut formuler des conditions de filtre plus complexes à l'aide d'opérateurs logiques et d'opérateurs de comparaison. Par exemple, si on veut récupérer les titres des livres de l'éditeur #1 qui coûtent 20$ ou moins:

```

```

```sql
SELECT titre FROM livre WHERE id_editeur = 1 AND prix <= 20;
```

Voici quelques opérateurs de comparaison supplémentaires que nous n'avons pas vus précédemment:

- `BETWEEN x AND y` : la valeur est située entre x et y, c'est-à-dire à la fois supérieure ou égale à x (`>= x`) et inférieure ou égale à y (`<= y`)
- `IN` et `NOT IN` : la valeur est comprise dans une liste (ex: `IN ('abc', 'def', 'ghi')`) ou n'est pas comprise dans une liste (`NOT IN`)
- `IS` et `IS NOT` : pour valider une valeur booléenne (`IS TRUE` ou `IS FALSE`), ou vérifier si une valeur est nulle (`IS NULL` ou `IS NOT NULL`)

La liste complète des opérateurs de comparaison est disponible ici.

L'opérateur `LIKE`

L'opérateur `LIKE` permet d'appliquer un filtre partiel sur une chaîne de caractères. Voici par exemple une requête permettant de récupérer tous les livres dont le titre commence par "Harry Potter" :

```sql
SELECT *
    FROM livre
    WHERE titre LIKE 'Harry Potter%';
```

On peut placer le caractère `%` n'importe où dans l'expression de recherche (qu'on appelle un modèle ou *pattern*). Celui-ci permet de trouver toutes les valeurs pour lesquelles il peut remplacer 0, 1 ou plusieurs caractères. Le caractère _ correspond pours a part à un caractère unique.

Prenons par exemple la liste de numéros de téléphone suivante:

- 911
- 811
- 511
- 819-564-6350
- (819) 564-6350
- (514) 872 0311
- 555-555-1819

Voici quelques exemples de modèles pouvant être utilisés avec `LIKE` avec les numéros qui seraient retournés:

Modèle	Numéros retournés
`8%`	811 819-564-6350
`___` (trois fois « _ »)	911 811 511

Modèle	Numéros retournés
`(%)%`	(819) 564-6350 (514) 872 0311
`___-___-____`	819-564-6350
`(819)%`	(819) 564-6350
`%819%`	819-564-6350 (819) 564-6350 555-555-1819

Il existe aussi un opérateur `NOT LIKE`.

Clause de tri (`ORDER BY`)

Une requête de sélection peut contenir une clause `ORDER BY` permettant de trier (ordonner) les données selon les valeurs d'une ou plusieurs colonnes. Par exemple, pour récupérer tous les livres en ordre alphabétique de titre:

```
SELECT *
    FROM livre
    ORDER BY titre;
```

Les mots-clés `ASC` et `DESC` permettent respectivement d'ordonner les résultats en ordre ascendant ou descendant. L'ordre ascendant est utilisé par défaut.

Par exemple, pour ordonner les livres en ordre descendant de prix (du plus cher au moins cher):

```
SELECT *
    FROM livre
    ORDER BY prix DESC;
```

On peut aussi utiliser plusieurs champs de tri, en les séparant par des virgules. Le tri se fera d'abord selon le premier champ, puis le deuxième, etc. Par exemple:

```
SELECT *
    FROM livre
    ORDER BY prix DESC, titre ASC;
```

Il n'est pas obligatoire pour les champs de tri utilisés d'être présents dans la liste de sélection. Par exemple, cette requête est tout à fait valide:

```

```

```
SELECT isbn, titre, prix
    FROM livre
    ORDER BY date_parution DESC;
```

Finalement, la clause `ORDER BY` peut bien entendu être combinée à une clause `WHERE`. La clause `WHERE` doit cependant apparaître en premier.

```
SELECT isbn, titre, prix
    FROM livre
    WHERE id_editeur = 1
    ORDER BY titre;
```

Clause `LIMIT`

La clause `LIMIT` permet de retourner seulement un certain nombre de résultats. On la combine souvent avec la clause `ORDER BY` (mais cela n'est pas obligatoire). Par exemple, la requête suivante permet d'obtenir les 10 derniers livres parus:

```
SELECT *
    FROM livre
    ORDER BY date_parution DESC
    LIMIT 10;
```

Il est aussi possible d'exclure un certain nombre de résultats à partir du début de ceux-ci. Imaginez par exemple que vous voulez afficher la liste de tous les livres sur une page Web. S'il y a beaucoup de livres, vous voudrez probablement implémenter une pagination, c'est-à-dire afficher seulement un certain nombre de livres (disons 50) suivi d'un bouton "Suivant" pour afficher les 50 livres suivants. Pour récupérer les livres de la page 3 depuis la base de données, vous devrez faire une requête qui récupère 50 livres en excluant les 100 premiers. Voici comment vous pourriez faire cela avec MariaDB:

```
SELECT *
    FROM livre
    ORDER BY titre
    LIMIT 100, 50;
```

Références

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