Les requêtes SQL de base

Qu'est-ce que SQL ?

SQL (**Structured Query Language**) est le langage standardisé utilisé pour interagir avec les bases de données relationnelles.

SQL permet de :

- **Interroger** les données (lecture)
- **Insérer** de nouvelles données
- **Modifier** des données existantes
- **Supprimer** des données

Il existe différents types de commandes SQL :

- **DML** (Data Manipulation Language) : Manipulation des données (SELECT, INSERT, UPDATE, DELETE)
- **DDL** (Data Definition Language) : Définition de la structure (CREATE, ALTER, DROP)
- **DCL** (Data Control Language) : Gestion des droits (GRANT, REVOKE)

> Dans cette page, nous nous concentrons sur les commandes **DML**, les plus utilisées au quotidien.

Contexte des exemples

Pour illustrer les requêtes SQL, nous utilisons deux tables inspirées de l'univers Star Wars :

Table `planetes` :

```
id | nom        | climat      | population
---|------------|-------------|--------------
1  | Tatooine   | aride       |       200 000
```

```
2  | Naboo      | tempéré     |     4 500 000
3  | Hoth       | glacial     |             0
4  | Coruscant  | urbain      | 1 000 000 000
5  | Endor      | forestier   |    30 000 000
```

Table `personnages` :

```
id | nom              | espece   | planete_id | cote
---|------------------|----------|------------|----------
1  | Luke Skywalker   | humain   | 1          | lumineux
2  | Leia Organa      | humain   | 2          | lumineux
3  | Dark Vador       | humain   | 1          | obscur
4  | Yoda             | inconnu  | NULL       | lumineux
5  | Chewbacca        | wookiee  | NULL       | lumineux
```

Lire des données : SELECT

La commande `SELECT` permet de lire des données dans une table.

Sélectionner toutes les colonnes

```
SELECT * FROM planetes;
```

Retourne toutes les lignes et toutes les colonnes de la table `planetes`.

Sélectionner des colonnes spécifiques

```
SELECT nom, climat FROM planetes;
```

Retourne uniquement les colonnes `nom` et `climat`.

Utiliser des alias

```
SELECT nom AS planete, population AS habitants FROM planetes;
```

Renomme les colonnes dans les résultats pour plus de clarté.

Trier les résultats : ORDER BY

```
SELECT nom, population FROM planetes ORDER BY population DESC;
```

Trie les planètes par population décroissante (`DESC`). Utiliser `ASC` pour un tri croissant.

Limiter le nombre de résultats : LIMIT

```
SELECT nom FROM planetes LIMIT 3;
```

Retourne uniquement les 3 premières planètes.

Filtrer avec WHERE

La clause `WHERE` permet de filtrer les résultats selon des conditions.

Opérateurs de comparaison

```
SELECT * FROM planetes WHERE climat = 'aride';
SELECT * FROM personnages WHERE planete_id != 1;
SELECT * FROM planetes WHERE population > 1000000;
```

Opérateurs disponibles : `=`, `!=`, `<`, `>`, `<=`, `>=`

Opérateurs logiques : AND, OR, NOT

```
SELECT * FROM personnages WHERE cote = 'lumineux' AND espece = 'humain';
SELECT * FROM planetes WHERE climat = 'aride' OR climat = 'glacial';
SELECT * FROM personnages WHERE NOT cote = 'obscur';
```

Recherche textuelle : LIKE

```
SELECT * FROM planetes WHERE nom LIKE '%oo%';
```

Retourne les planètes dont le nom contient "oo" (Tatooine, Naboo). Le symbole `%` représente n'importe quelle chaîne de caractères.

Listes de valeurs : IN

```
SELECT * FROM planetes WHERE nom IN ('Tatooine', 'Naboo', 'Hoth');
```

Retourne uniquement les planètes dont le nom est dans la liste.

Intervalles : BETWEEN

```
SELECT * FROM planetes WHERE population BETWEEN 100000 AND 5000000;
```

Retourne les planètes avec une population entre 100 000 et 5 000 000 habitants.

Valeurs nulles : IS NULL / IS NOT NULL

```
SELECT * FROM personnages WHERE planete_id IS NULL;
SELECT * FROM personnages WHERE planete_id IS NOT NULL;
```

Retourne les personnages sans planète d'origine (ou avec une planète d'origine).

Modifier les données

Insérer des données : INSERT

La commande `INSERT` ajoute de nouvelles lignes dans une table.

```
INSERT INTO planetes (nom, climat, population)
VALUES ('Dagobah', 'marécageux', 0);
```

Insère une nouvelle planète dans la table.

Insertion multiple :

```
INSERT INTO personnages (nom, espece, planete_id, cote) VALUES
('Obi-Wan Kenobi', 'humain', 1, 'lumineux'),
('Han Solo', 'humain', NULL, 'lumineux');
```

Modifier des données : UPDATE

La commande `UPDATE` modifie des lignes existantes.

```
UPDATE planetes SET population = 250000 WHERE nom = 'Tatooine';
```

Met à jour la population de Tatooine.

Modifier plusieurs colonnes :

```
UPDATE planetes SET climat = 'désertique', population = 200500
WHERE nom = 'Tatooine';
```

Supprimer des données : DELETE

La commande `DELETE` supprime des lignes d'une table.

```
DELETE FROM personnages WHERE nom = 'Dark Vador';
```

Supprime le personnage Dark Vador de la table.

> **ATTENTION !** Toujours utiliser `WHERE` avec `UPDATE` et `DELETE` !
> Sans `WHERE`, toutes les lignes de la table seront modifiées ou supprimées.
>
> Exemple dangereux : `DELETE FROM personnages;` supprimera TOUS les personnages !

Les transactions

Une **transaction** est un ensemble d'opérations SQL qui doivent être exécutées comme une unité indivisible : soit toutes réussissent, soit aucune n'est appliquée.

Principe

Par défaut, MariaDB est en mode **autocommit** : chaque requête est validée immédiatement. Les transactions permettent de regrouper plusieurs opérations et de les valider (ou annuler) ensemble.

- `START TRANSACTION` : démarre une transaction
- `COMMIT` : valide toutes les modifications
- `ROLLBACK` : annule toutes les modifications depuis le début de la transaction

Exemple : transfert de population

Imaginons un transfert de population entre deux planètes. Les deux opérations doivent réussir ensemble :

```
START TRANSACTION;

-- Retirer 10 000 habitants de Naboo
UPDATE planetes SET population = population - 10000 WHERE nom = 'Naboo';

-- Ajouter 10 000 habitants sur Tatooine
UPDATE planetes SET population = population + 10000 WHERE nom = 'Tatooine';
```

```
-- Tout s'est bien passé : on valide
COMMIT;
```

Si une erreur survient, on peut annuler :

```
START TRANSACTION;

UPDATE planetes SET population = population - 10000 WHERE nom = 'Naboo';
UPDATE planetes SET population = population + 10000 WHERE nom = 'Tatooine';

-- Oups, erreur détectée : on annule tout
ROLLBACK;
```

> Après un `ROLLBACK`, les données sont dans le même état qu'avant le `START TRANSACTION`.

Quand utiliser les transactions ?

- Modifications sur **plusieurs tables liées**
- Opérations qui doivent être **cohérentes ensemble** (transferts, réservations)
- **Tests de requêtes** : démarrer une transaction, tester, puis `ROLLBACK` pour annuler

Bonnes pratiques

- **Toujours tester avec SELECT** : Avant un `UPDATE` ou `DELETE`, lancez d'abord un `SELECT` avec le même `WHERE` pour vérifier les lignes affectées.
- **Utiliser LIMIT lors des tests** : Ajouter `LIMIT 10` pour tester les requêtes sans surcharger la base.
- **Utiliser les transactions** : Pour sécuriser les modifications importantes.
- **Sauvegarder avant modification** : Toujours effectuer une sauvegarde avant des modifications massives.

> Exemple de test avant suppression :
> `SELECT * FROM personnages WHERE cote = 'obscur';`
> Vérifier les résultats, puis :
> `DELETE FROM personnages WHERE cote = 'obscur';`

Revision #4
Created 1 January 2026 13:26:22 by Thibaud FRICHET
Updated 16 January 2026 08:15:19 by Thibaud FRICHET