Cours : Les requêtes SQL - Sélection, Insertion et Mise à jour des données

Introduction aux requêtes SQL

SQL (Structured Query Language) est un langage standard utilisé pour interagir avec des bases de données relationnelles. Les requêtes SQL permettent de manipuler et de gérer les données en effectuant diverses opérations telles que la sélection, l'insertion, la mise à jour et la suppression des informations dans les bases de données. Ce cours se concentrera sur les requêtes SQL fondamentales qui vous permettront de sélectionner, insérer et mettre à jour les données dans une base de données.

Les principales commandes SQL que nous allons aborder sont les suivantes :

1. **Sélectionner les données (SELECT)**
2. **Insérer des données (INSERT)**
3. **Mettre à jour les données (UPDATE)**

---

1. Sélectionner les données avec la commande SELECT

La commande `SELECT` est utilisée pour récupérer des données à partir d'une ou plusieurs tables d'une base de données. Elle permet de spécifier les colonnes à sélectionner et les critères de filtrage.

Syntaxe de base :

```
SELECT colonne1, colonne2, ...
FROM table
WHERE condition;
```

- **colonne1, colonne2, ...** : Ce sont les colonnes que vous souhaitez récupérer.
- **table** : Le nom de la table d'où proviennent les données.
- **condition** : Une condition qui limite les résultats, par exemple une clause `WHERE`.

Exemples :

- **Sélectionner toutes les données d'une table :**
- `SELECT * FROM Employes;`

  Cette requête retourne toutes les colonnes de la table `Employes`.

- **Sélectionner des colonnes spécifiques :**
- `SELECT Nom, Prenom, Age FROM Employes;`

Cette requête retourne uniquement les colonnes `Nom`, `Prenom` et `Age` de la table `Employes`.

- **Filtrer les résultats avec une condition :**
- `SELECT * FROM Employes WHERE Age > 30;`

  Cette requête sélectionne tous les employés dont l'âge est supérieur à 30 ans.

- **Utiliser des opérateurs de comparaison :**
- `SELECT * FROM Employes WHERE Salaire BETWEEN 2500 AND 3500;`

  Cette requête retourne les employés ayant un salaire compris entre 2500 et 3500.

- **Trier les résultats :**
- `SELECT * FROM Employes ORDER BY Nom ASC;`

  Cette requête trie les résultats par ordre croissant de la colonne `Nom`.

---

2. Insérer des données avec la commande INSERT

La commande `INSERT INTO` est utilisée pour ajouter de nouvelles lignes de données dans une table. Vous pouvez spécifier les colonnes auxquelles les valeurs doivent être insérées ou laisser SQL insérer des valeurs dans toutes les colonnes.

Syntaxe de base :

```
INSERT INTO table (colonne1, colonne2, ...)
VALUES (valeur1, valeur2, ...);
```

- **table** : Le nom de la table dans laquelle vous souhaitez insérer des données.
- **colonne1, colonne2, ...** : Les colonnes dans lesquelles vous souhaitez insérer des valeurs.
- **valeur1, valeur2, ...** : Les valeurs correspondant à chaque colonne.

Exemples :

- **Insérer une seule ligne de données :**
- `INSERT INTO Employes (ID, Nom, Prenom, Age, Salaire)`
- `VALUES (1, 'Dupont', 'Pierre', 30, 2500.00);`

  Cette requête insère un nouvel employé avec les informations spécifiées dans la table `Employes`.

- **Insérer plusieurs lignes de données :**
- `INSERT INTO Employes (ID, Nom, Prenom, Age, Salaire)`
- `VALUES`
- `    (2, 'Martin', 'Claire', 28, 2800.00),`

```
    (3, 'Lemoine', 'Jean', 35, 3200.00),
    (4, 'Durand', 'Sophie', 40, 3000.00);
```

Cette requête insère trois nouveaux employés dans la table `Employes`.

---

3. Mettre à jour les données avec la commande UPDATE

La commande `UPDATE` permet de modifier les données existantes dans une table. Elle met à jour une ou plusieurs colonnes pour un ou plusieurs enregistrements qui répondent à une condition spécifique.

Syntaxe de base :

```
UPDATE table
SET colonne1 = valeur1, colonne2 = valeur2, ...
WHERE condition;
```

- **table** : Le nom de la table dans laquelle les données doivent être mises à jour.
- **colonne1, colonne2, ...** : Les colonnes que vous souhaitez mettre à jour.
- **valeur1, valeur2, ...** : Les nouvelles valeurs à attribuer aux colonnes.
- **condition** : La condition qui permet de cibler les lignes spécifiques à mettre à jour.

Exemples :

- **Mettre à jour une seule colonne :**

```
UPDATE Employes
SET Salaire = 2700.00
WHERE ID = 1;
```

Cette requête met à jour le salaire de l'employé ayant l'ID 1 pour qu'il passe à 2700.00.

- **Mettre à jour plusieurs colonnes :**

```
UPDATE Employes
SET Salaire = 2800.00, Age = 31
WHERE ID = 1;
```

Cette requête met à jour le salaire et l'âge de l'employé ayant l'ID 1.

- **Mettre à jour plusieurs lignes de données :**

```
UPDATE Employes
SET Salaire = Salaire + 200
WHERE Age > 30;
```

Cette requête augmente le salaire de tous les employés de plus de 30 ans de 200.

Conclusion

Les requêtes SQL de sélection, d'insertion et de mise à jour sont essentielles pour gérer les données dans une base de données relationnelle. Vous avez appris à :

- Sélectionner des données spécifiques à partir de tables.
- Ajouter de nouvelles données à une table.
- Mettre à jour les données existantes selon des critères définis.

La maîtrise de ces commandes SQL fondamentales est cruciale pour travailler efficacement avec les bases de données et manipuler les informations de manière flexible et puissante. Vous pouvez maintenant appliquer ces connaissances pour interroger et manipuler les données de vos bases de données relationnelles.