Cours : Gestion des Jointures et Sous-requêtes en SQL

Introduction aux Jointures et Sous-requêtes

Lorsqu'on travaille avec des bases de données relationnelles, il est fréquent de devoir interroger plusieurs tables pour obtenir des informations complètes. Les **jointures** et les **sous-requêtes** sont deux techniques SQL puissantes qui permettent de combiner et d'extraire des données provenant de plusieurs tables.

- **Les jointures** permettent de relier des données provenant de tables différentes en fonction d'une colonne partagée.
- **Les sous-requêtes** permettent d'imbriquer une requête à l'intérieur d'une autre pour obtenir des résultats plus complexes.

Ce cours vous guidera à travers l'utilisation des jointures et des sous-requêtes pour manipuler efficacement les données de plusieurs tables.

---

1. Les Jointures (JOINs)

Les jointures permettent de combiner des données provenant de deux ou plusieurs tables en fonction d'une colonne commune. Il existe plusieurs types de jointures en SQL :

1.1. Jointure Interne (INNER JOIN)

La **jointure interne** renvoie uniquement les lignes où il y a une correspondance dans les deux tables.

Syntaxe de base :

```
SELECT colonne1, colonne2, ...
FROM table1
INNER JOIN table2
ON table1.colonne_commune = table2.colonne_commune;
```

- **table1** et **table2** : Les deux tables à joindre.
- **colonne_commune** : La colonne sur laquelle les tables sont jointes.

Exemple :

```
SELECT Employes.Nom, Employes.Prenom, Departements.Nom_departement
FROM Employes
INNER JOIN Departements
ON Employes.Dept_ID = Departements.Dept_ID;
```

Cette requête retourne les noms et prénoms des employés ainsi que le nom de leur département, uniquement pour les employés ayant un département attribué.

1.2. Jointure Externe (LEFT JOIN ou RIGHT JOIN)

Une **jointure externe** permet de récupérer toutes les lignes de la première table, même si elles n'ont pas de correspondance dans la deuxième table.

- **LEFT JOIN** : Retourne toutes les lignes de la table à gauche et les correspondances de la table à droite.
- **RIGHT JOIN** : Retourne toutes les lignes de la table à droite et les correspondances de la table à gauche.

Syntaxe de base (LEFT JOIN) :

```
SELECT colonne1, colonne2, ...
FROM table1
LEFT JOIN table2
ON table1.colonne_commune = table2.colonne_commune;
```

Exemple :

```
SELECT Employes.Nom, Employes.Prenom, Departements.Nom_departement
FROM Employes
LEFT JOIN Departements
ON Employes.Dept_ID = Departements.Dept_ID;
```

Cette requête retourne tous les employés, y compris ceux qui ne sont affectés à aucun département (les départements seront alors `NULL`).

1.3. Jointure Croisée (CROSS JOIN)

La **jointure croisée** produit un produit cartésien entre les deux tables, c'est-à-dire qu'elle retourne toutes les combinaisons possibles entre les lignes des deux tables.

Syntaxe de base :

```
SELECT colonne1, colonne2
FROM table1
CROSS JOIN table2;
```

Exemple :

```
SELECT Employes.Nom, Produits.Nom_produit
FROM Employes
CROSS JOIN Produits;
```

Cette requête retourne toutes les combinaisons possibles entre les employés et les produits.

---

2. Les Sous-requêtes (Subqueries)

Les sous-requêtes sont des requêtes imbriquées à l'intérieur d'une autre requête SQL. Elles peuvent être utilisées dans les clauses `SELECT`, `FROM`, `WHERE`, et d'autres.

2.1. Sous-requête dans la clause WHERE

Une sous-requête dans la clause `WHERE` permet de filtrer les résultats en fonction d'une valeur qui est le résultat d'une autre requête.

Syntaxe de base :

```
SELECT colonne1, colonne2
FROM table
WHERE colonne IN (SELECT colonne FROM autre_table WHERE condition);
```

Exemple :

```
SELECT Nom, Prenom
FROM Employes
WHERE Dept_ID IN (SELECT Dept_ID FROM Departements WHERE Nom_departement =
'Marketing');
```

Cette requête retourne les employés qui travaillent dans le département "Marketing".

2.2. Sous-requête dans la clause SELECT

Une sous-requête dans la clause `SELECT` permet de renvoyer une valeur calculée pour chaque ligne dans le résultat de la requête principale.

Syntaxe de base :

```
SELECT colonne1, (SELECT colonne2 FROM autre_table WHERE condition) AS Alias
FROM table;
```

Exemple :

```
SELECT Nom, Prenom, (SELECT Nom_departement FROM Departements WHERE Dept_ID =
Employes.Dept_ID) AS Département
FROM Employes;
```

Cette requête retourne les noms et prénoms des employés ainsi que le nom de leur département, en utilisant une sous-requête pour obtenir le nom du département.

2.3. Sous-requête dans la clause FROM

Une sous-requête dans la clause `FROM` permet de traiter les résultats d'une requête comme une table temporaire.

Syntaxe de base :

```
SELECT colonne1, colonne2
FROM (SELECT colonne1, colonne2 FROM table WHERE condition) AS Alias;
```

Exemple :

```
SELECT Nom, Prenom
FROM (SELECT Nom, Prenom, Dept_ID FROM Employes WHERE Age > 30) AS
Employes_Adultes
WHERE Dept_ID = 2;
```

Cette requête retourne les employés âgés de plus de 30 ans et qui appartiennent au département avec l'ID 2.

Conclusion

Les jointures et sous-requêtes sont des éléments essentiels pour effectuer des requêtes SQL complexes. Les jointures permettent de combiner les données de plusieurs tables en fonction de colonnes communes, tandis que les sous-requêtes permettent de filtrer, transformer et manipuler les données de manière plus flexible.

En maîtrisant ces techniques, vous serez en mesure d'effectuer des requêtes très puissantes et de récupérer des informations de manière efficace à partir de bases de données relationnelles.