Cours : Les Jointures en SQL

Propriétaire	Ⓜ Marine
Étiquettes	

Objectif : Comprendre et utiliser les différentes types de jointures en SQL pour interroger des bases de données relationnelles.

Introduction

Dans une base de données relationnelle, les données sont réparties dans plusieurs tables pour éviter les redondances et améliorer l'organisation. Les jointures permettent de combiner ces tables pour extraire des informations utiles.

Concept de Jointure

Une jointure relie des lignes de deux tables (ou plus) en fonction d'une condition basée sur une colonne commune, généralement une clé primaire et une clé étrangère.

Syntaxe de base :

```
SELECT colonnes
FROM table1
JOIN table2
ON table1.colonne_commune = table2.colonne_commune;
```

Les types de jointures

1. Jointure interne (INNER JOIN)

- **Description** : Retourne uniquement les lignes qui ont une correspondance dans les deux tables.
- **Exemple** : Trouver les livres empruntés par les clients.

```
SELECT c.nom, c.prenom, l.titre
FROM client c
INNER JOIN emprunt e ON c.idClient = e.idClient
INNER JOIN ligneEmprunt le ON e.noFiche = le.noFiche
INNER JOIN livre l ON le.refLivre = l.refLivre;
```

2. Jointure externe gauche (LEFT JOIN)

- **Description** : Retourne toutes les lignes de la table de gauche, même si aucune correspondance n’est trouvée dans la table de droite.
- **Exemple** : Afficher tous les clients, y compris ceux qui n'ont pas encore emprunté de livre.

```
SELECT c.nom, c.prenom, e.noFiche
FROM client c
LEFT JOIN emprunt e ON c.idClient = e.idClient;
```

3. Jointure externe droite (RIGHT JOIN)

- **Description** : Retourne toutes les lignes de la table de droite, même si aucune correspondance n’est trouvée dans la table de gauche.
- **Exemple** : Afficher tous les livres, même ceux qui n'ont pas encore été empruntés.

```
SELECT l.titre, le.noFiche
FROM livre l
RIGHT JOIN ligneEmprunt le ON l.refLivre = le.refLivre;
```

4. Jointure externe complète (FULL OUTER JOIN)

- **Description** : Combine les résultats des jointures gauche et droite. Retourne toutes les lignes des deux tables, avec `NULL` pour les colonnes sans correspondance.

- **Exemple** : Lister toutes les catégories de livres et les livres qui y sont associés.

```
SELECT c.nomCategorie, l.titre
FROM categorieLivre c
FULL OUTER JOIN livre l ON c.idCategorie = l.idCategorie;
```

⚠

Note : Certains SGBD comme MySQL ne prennent pas en charge directement `FULL OUTER JOIN`. On peut utiliser une union pour contourner cela :

5. Jointure croisée (CROSS JOIN)

- **Description** : Produit le produit cartésien des deux tables, c'est-à-dire toutes les combinaisons possibles entre les lignes des deux tables.
- **Exemple** : Associer chaque client à chaque livre (utile rarement, par exemple pour des tests).

```
SELECT c.nom, c.prenom, l.titre
FROM client c
CROSS JOIN livre l;
```

6. Self Join (Auto-jointure)

- **Description** : Joint une table avec elle-même, utile pour des relations hiérarchiques ou pour comparer des lignes d'une même table.
- **Exemple** : Trouver les clients ayant la même adresse.

```
SELECT c1.nom AS client1, c2.nom AS client2, c1.adresse
FROM client c1
INNER JOIN client c2 ON c1.adresse = c2.adresse AND c1.idClie
```

Exemple / démo :

Exemple pratique : Analyse des prêts de livres

Jeu de données

Tables utilisées :

- **client(idClient, nom, prenom, adresse)**
- **emprunt(noFiche, idClient, dateEmprunt)**
- **ligneEmprunt(noFiche, refLivre)**
- **livre(refLivre, titre, idCategorie, prix)**
- **categorieLivre(idCategorie, nomCategorie)**

Requêtes

1. **Lister tous les emprunts avec les noms des clients et les titres des livres :**

```
SELECT e.noFiche, c.nom, c.prenom, l.titre, e.dateEmprunt
FROM emprunt e
INNER JOIN client c ON e.idClient = c.idClient
INNER JOIN ligneEmprunt le ON e.noFiche = le.noFiche
INNER JOIN livre l ON le.refLivre = l.refLivre;
```

Bonnes pratiques

1. Utilisez des **alias** pour simplifier et rendre les requêtes plus lisibles.
2. Évitez les produits cartésiens accidentels en vérifiant toujours la condition de jointure dans une clause `ON`.
3. Testez vos requêtes sur un petit ensemble de données avant de les exécuter sur la base complète.