Les jointures SQL

Pourquoi les jointures ?

Dans une base de données relationnelle, les données sont réparties dans plusieurs tables pour éviter la redondance et maintenir la cohérence.

Les **jointures** permettent de **relier les données de plusieurs tables** dans une seule requête, en se basant sur les relations entre elles (clés étrangères).

Exemple : pour afficher les personnages avec le nom de leur planète d'origine, il faut combiner les tables `personnages` et `planetes`.

Rappel du contexte

Nous utilisons les mêmes tables Star Wars :

Table `planetes` :

```
id | nom        | climat      | population
---|------------|-------------|-----------
1  | Tatooine   | aride       | 200000
2  | Naboo      | tempéré     | 4500000
3  | Hoth       | glacial     | 0
```

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
```

La colonne `planete_id` dans la table `personnages` fait référence à l'`id` de la table `planetes` .

INNER JOIN

L'`INNER JOIN` retourne uniquement les lignes qui ont une **correspondance exacte** dans les deux tables.

Syntaxe

```
SELECT colonnes
FROM table1
INNER JOIN table2 ON table1.colonne = table2.colonne;
```

Exemple : Personnages avec leur planète

```
SELECT personnages.nom, planetes.nom AS planete
FROM personnages
INNER JOIN planetes ON personnages.planete_id = planetes.id;
```

Résultat :

```
nom              | planete
-----------------|----------
Luke Skywalker   | Tatooine
Leia Organa      | Naboo
Dark Vador       | Tatooine
```

> Yoda et Chewbacca ne sont **pas** dans les résultats car leur `planete_id` est `NULL`. L'`INNER JOIN` exclut les lignes sans correspondance.

Utiliser des alias de tables

Pour simplifier les requêtes, on peut utiliser des **alias de tables** :

```
SELECT p.nom, pl.nom AS planete, pl.climat
FROM personnages p
INNER JOIN planetes pl ON p.planete_id = pl.id
WHERE pl.climat = 'aride';
```

Résultat : Uniquement les personnages originaires de planètes arides (Luke et Vador de Tatooine).

LEFT JOIN (OUTER)

Le `LEFT JOIN` retourne **toutes les lignes de la table de gauche**, même si elles n'ont pas de correspondance dans la table de droite.

Pour les lignes sans correspondance, les colonnes de la table de droite contiennent `NULL`.

Syntaxe

```
SELECT colonnes
FROM table1
LEFT JOIN table2 ON table1.colonne = table2.colonne;
```

Exemple : Tous les personnages, avec ou sans planète

```
SELECT personnages.nom, planetes.nom AS planete
FROM personnages
LEFT JOIN planetes ON personnages.planete_id = planetes.id;
```

Résultat :

```
nom              | planete
-----------------|----------
Luke Skywalker   | Tatooine
Leia Organa      | Naboo
Dark Vador       | Tatooine
Yoda             | NULL
Chewbacca        | NULL
```

Cette fois, Yoda et Chewbacca **apparaissent** dans les résultats, avec `NULL` pour leur planète.

Différence INNER vs LEFT

La différence fondamentale :

- **INNER JOIN** : Uniquement les correspondances exactes (3 lignes dans notre exemple)

- **LEFT JOIN** : Toutes les lignes de la table de gauche + correspondances (5 lignes dans notre exemple)

> Utilisez `LEFT JOIN` si vous ne voulez perdre aucune ligne de votre table principale, même sans correspondance.
> Utilisez `INNER JOIN` si vous ne voulez que les lignes ayant une correspondance complète.

Filtrer les valeurs NULL

On peut utiliser un `LEFT JOIN` pour trouver les personnages **sans** planète d'origine :

```
SELECT personnages.nom
FROM personnages
LEFT JOIN planetes ON personnages.planete_id = planetes.id
WHERE planetes.id IS NULL;
```

Résultat : Yoda et Chewbacca

Jointures multiples

On peut chaîner plusieurs jointures dans une même requête.

Exemple avec une troisième table `vaisseaux` (hypothétique) :

```
SELECT p.nom, pl.nom AS planete, v.nom AS vaisseau
FROM personnages p
INNER JOIN planetes pl ON p.planete_id = pl.id
LEFT JOIN vaisseaux v ON p.id = v.pilote_id;
```

Cet exemple combine un `INNER JOIN` et un `LEFT JOIN` pour afficher les personnages, leur planète (obligatoire) et leur vaisseau (optionnel).

GROUP BY et fonctions d'agrégation

Les jointures peuvent être combinées avec `GROUP BY` pour regrouper et compter les données.

Fonctions d'agrégation

SQL propose des fonctions pour calculer des statistiques :

- `COUNT()` : Compter les lignes
- `SUM()` : Somme des valeurs
- `AVG()` : Moyenne des valeurs
- `MIN()` / `MAX()` : Valeur minimale / maximale

Exemple : Compter les personnages par planète

```
SELECT planetes.nom, COUNT(personnages.id) AS nombre_personnages
FROM planetes
LEFT JOIN personnages ON planetes.id = personnages.planete_id
GROUP BY planetes.nom;
```

Résultat :

```
nom         | nombre_personnages
------------|--------------------
Tatooine    | 2
Naboo       | 1
Hoth        | 0
```

Exemple : Personnages par côté de la Force

```
SELECT cote, COUNT(*) AS nombre
FROM personnages
GROUP BY cote;
```

Résultat :

```
cote       | nombre
-----------|--------
lumineux   | 4
obscur     | 1
```

> `GROUP BY` est un sujet à part entière qui sera approfondi dans une section dédiée du cours.

Bonnes pratiques

- **Utiliser des alias** : Simplifiez les requêtes avec des alias de tables (`p`, `pl`, etc.)
- **Tester avec LIMIT** : Ajoutez `LIMIT 10` pour tester vos jointures sur de gros volumes
- **Attention aux performances** : Les jointures multiples sur de grosses tables peuvent être lentes. Assurez-vous que les colonnes utilisées dans `ON` sont indexées
- **Comprendre INNER vs LEFT** : Choisissez le bon type de jointure selon vos besoins (correspondances exactes vs données complètes)

> Toujours vérifier le nombre de résultats attendus. Une jointure mal écrite peut produire des doublons ou des résultats inattendus.

Récapitulatif

Type de jointure	Description	Cas d'usage
INNER JOIN	Uniquement les correspondances exactes	Quand on ne veut que les données complètes
LEFT JOIN	Toutes les lignes de gauche + correspondances	Quand on veut garder toutes les données de la table principale

> Les jointures sont essentielles pour exploiter la puissance des bases de données relationnelles. Maîtriser `INNER JOIN` et `LEFT JOIN` couvre 90% des besoins quotidiens.

Revision #1
Created 1 January 2026 13:29:31 by Thibaud FRICHET
Updated 1 January 2026 13:29:31 by Thibaud FRICHET