Jointure SQL

Les jointures en SQL permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.

Exemple

En général, les jointures consistent à associer des lignes de 2 tables en associant l'égalité des valeurs d'une colonne d'une première table par rapport à la valeur d'une colonne d'une seconde table. Imaginons qu'une base de 2 données possède une table "utilisateur" et une autre table "adresse" qui contient les adresses de ces utilisateurs. Avec une jointure, il est possible d'obtenir les données de l'utilisateur et de son adresse en une seule requête.

On peut aussi imaginer qu'un site web possède une table pour les articles (titre, contenu, date de publication ...) et une autre pour les rédacteurs (nom, date d'inscription, date de naissance ...). Avec une jointure il est possible d'effectuer une seule recherche pour afficher un article et le nom du rédacteur. Cela évite d'avoir à afficher le nom du rédacteur dans la table "article".

Il y a d'autres cas de jointures, incluant des jointures sur la même table ou des jointure d'inégalité. Ces cas étant assez particulier et pas si simple à comprendre, ils ne seront pas élaboré sur cette page.

Types de jointures

Il y a plusieurs méthodes pour associer 2 tables ensemble. Voici la liste des différentes techniques qui sont utilisées :

- **INNER JOIN:** jointure interne pour retourner les enregistrements quand la condition est vrai dans les 2 tables. C'est l'une des jointures les plus communes.
- **CROSS JOIN:** jointure croisée permettant de faire le produit cartésien de 2 tables. En d'autres mots, permet de joindre chaque lignes d'une table avec chaque lignes d'une seconde table. Attention, le nombre de résultats est en général très élevé.
- **LEFT JOIN:** jointure externe pour retourner tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n'est pas vérifié dans l'autre table.
- **RIGHT JOIN :** jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite) même si la condition n'est pas vérifié dans l'autre table.

- **FULL JOIN :** jointure externe pour retourner les résultats quand la condition est vrai dans au moins une des 2 tables.
- **SELF JOIN :** permet d'effectuer une jointure d'une table avec elle-même comme si c'était une autre table.
- **NATURAL JOIN :** jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom entre les 2 tables SQL
- **UNION JOIN :** jointure d'union

Exemples de jointures

INNER JOIN

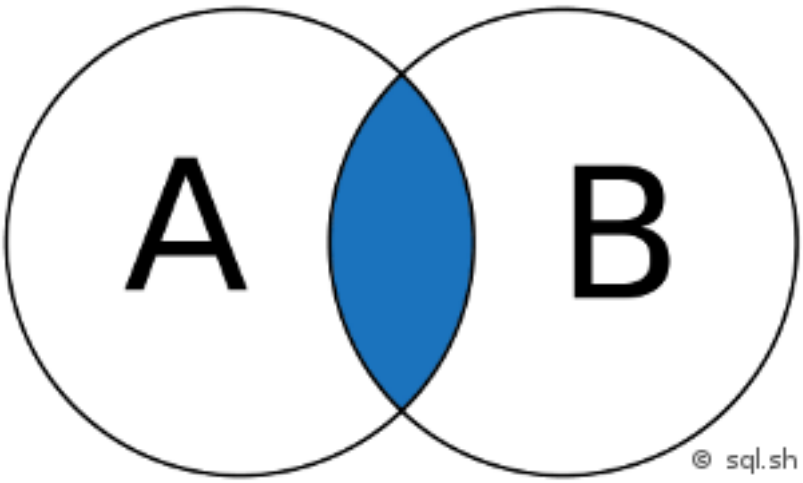

Figure 1: Intersection de 2 ensembles

```
SELECT *
FROM A
INNER JOIN B ON A.key = B.key
```

Exemple

Imaginons une application qui possède une table utilisateur ainsi qu'une table commande qui contient toutes les commandes effectuées par les utilisateurs.

Table utilisateur

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille

id	prenom	nom	email	ville
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

Pour afficher toutes les commandes associées aux utilisateurs, il est possible d'utiliser la requête suivante :

```
SELECT id, prenom, nom, date_achat, num_facture, prix_total
FROM utilisateur
INNER JOIN commande ON utilisateur.id = commande.utilisateur_id
```

Résultats

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35

LEFT JOIN

```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
```

Exemple

Imaginons une application contenant des utilisateurs et des commandes pour chacun de ces utilisateurs. La base de données de cette application contient une table pour les utilisateurs et sauvegarde leurs achats dans une seconde table. Les 2 tables sont reliées grâce à la colonne utilisateur_id de la table des commandes. Cela permet d'associer une commande à un utilisateur.

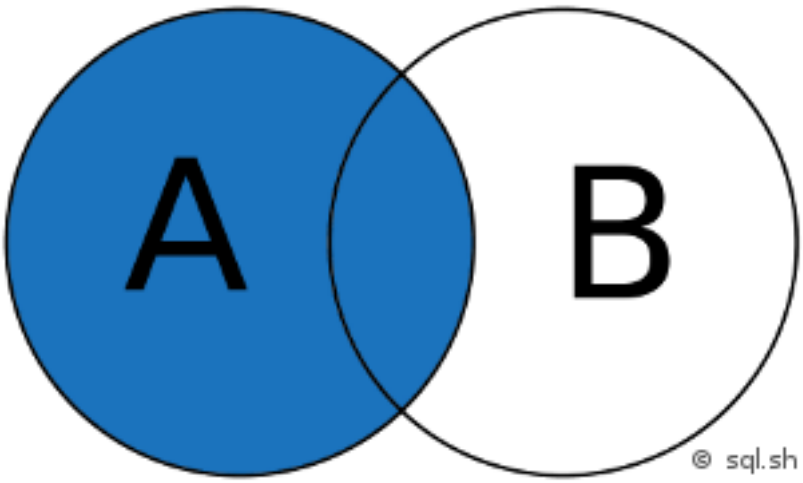

Figure 2: Jointure gauche (LEFT JOINT)

Table utilisateur :

id	prenom	nom	email	ville
1	Aimée	Marechal	aime.marechal@example.com	Paris
2	Esmée	Lefort	esmee.lefort@example.com	Lyon
3	Marine	Prevost	m.prevost@example.com	Lille
4	Luc	Rolland	lucrolland@example.com	Marseille

Table commande :

utilisateur_id	date_achat	num_facture	prix_total
1	2013-01-23	A00103	203.14
1	2013-02-14	A00104	124.00
2	2013-02-17	A00105	149.45
2	2013-02-21	A00106	235.35
5	2013-03-02	A00107	47.58

Pour lister tous les utilisateurs avec leurs commandes et afficher également les utilisateurs qui n'ont pas effectuées d'achats, il est possible d'utiliser la requête suivante:

```
SELECT *
FROM utilisateur
LEFT JOIN commande ON utilisateur.id = commande.utilisateur_id
```

Résultats :

id	prenom	nom	date_achat	num_facture	prix_total
1	Aimée	Marechal	2013-01-23	A00103	203.14
1	Aimée	Marechal	2013-02-14	A00104	124.00
2	Esmée	Lefort	2013-02-17	A00105	149.45
2	Esmée	Lefort	2013-02-21	A00106	235.35
3	Marine	Prevost	NULL	NULL	NULL
4	Luc	Rolland	NULL	NULL	NULL

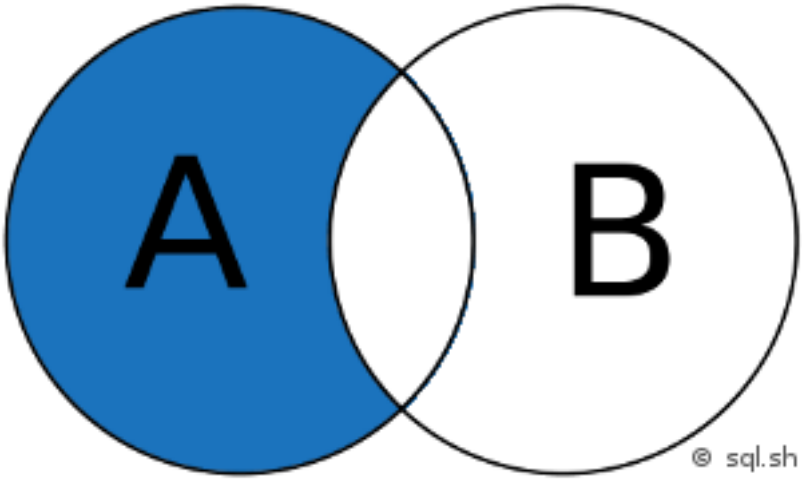

Figure 3: Jointure gauche (LEFT JOINT sans l'intersection B)

Exemple

```
SELECT id, prenom, nom, utilisateur_id
FROM utilisateur
LEFT JOIN commande ON utilisateur.id = commande.utilisateur_id
WHERE utilisateur_id IS NULL
```

Résultats

id	prenom	nom	utilisateur_id
3	Marine	Prevost	NULL
4	Luc	Rolland	NULL

RIGHT JOIN

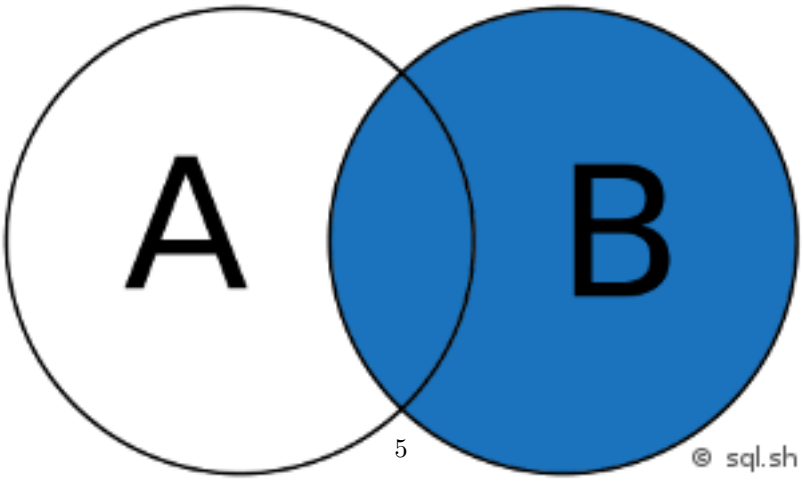

Figure 4: Jointure droite (RIGHT JOINT)

```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
```

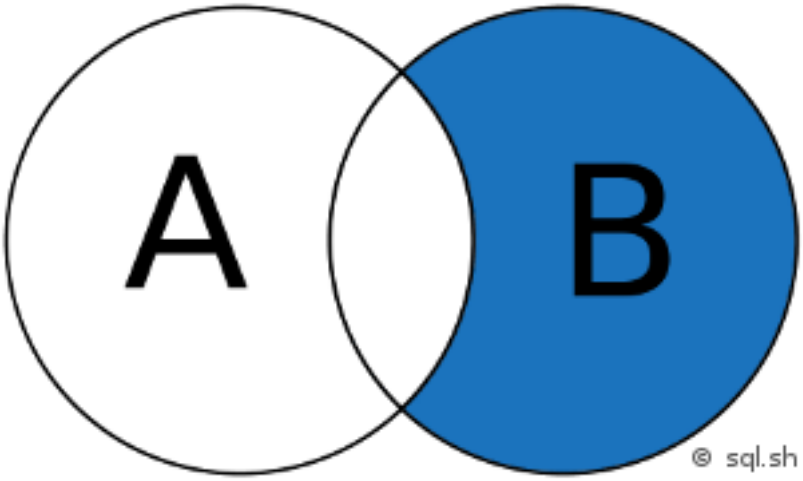

Figure 5: Jointure droite (RIGHT JOINT sans l'intersection A)

FULL JOIN

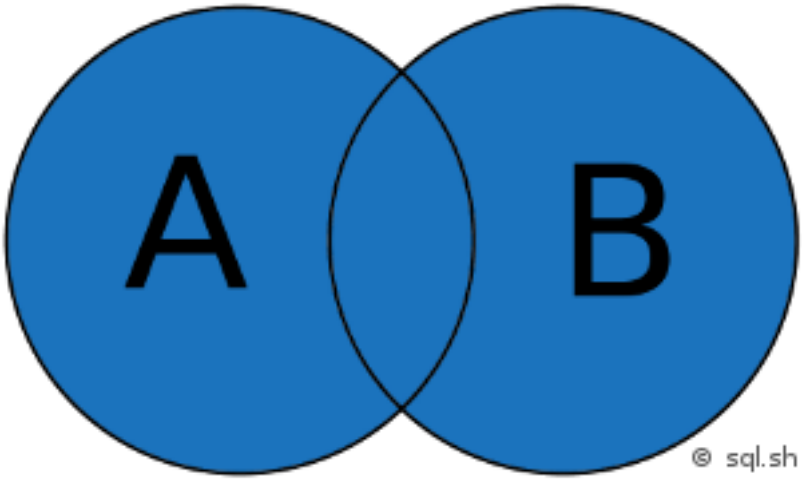

Figure 6: Union de 2 ensembles

```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
```

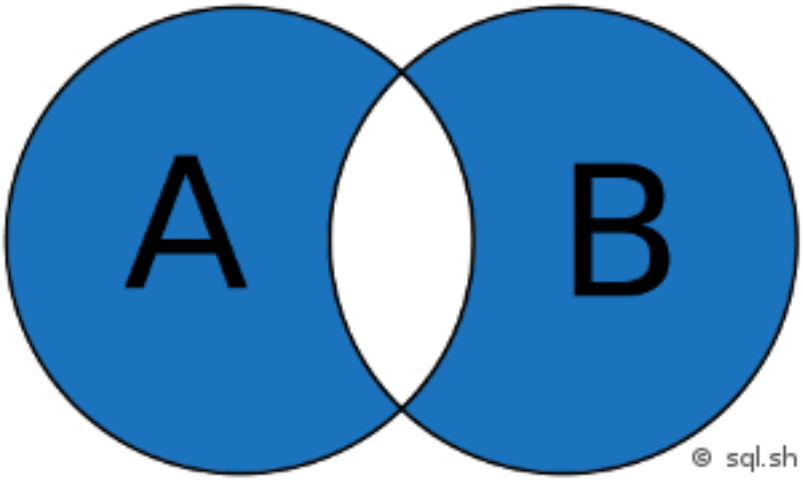

Figure 7: Jointure pleine (FULL JOINT sans intersection)

FULL JOIN (sans intersection)

```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```