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.

I. Jointure ?

Joindre deux tables, c'est pouvoir afficher les informations de ces tables tout en **gardant la cohérence des données**. Par exemple, une table "utilisateur" et une table "ville" qui contient les adresses de ces utilisateurs. Avec une jointure, il est possible d'obtenir les données de l'utilisateur et de la ville dans laquelle il habite dans le même document (le résultat de la requête. **Garder la cohérence**, c'est ne pas afficher n'importe quelle ville en face des utilisateurs.

Il existe des types différents de jointure : la jointure interne, l'auto-jointure, la jointure droite, gauche et la jointure naturelle sont les plus importantes.

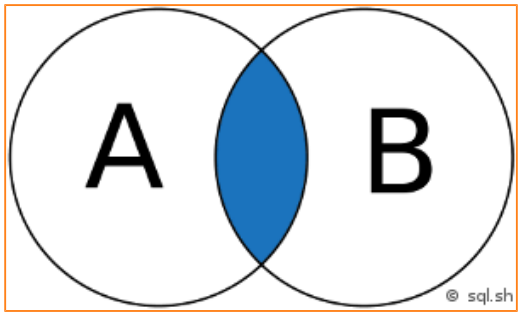
II. Types De Jointures

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

- **NATURAL JOIN :** jointure naturelle sur des champs (FK et PK) qui portent le même dans les 2 tables
- **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.
- **LEFT JOIN (ou LEFT OUTER 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 (ou RIGHT OUTER 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 (ou FULL OUTER 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. *C'est utile quand on a une table VILLE qui peut être utilisée comme arrivée ou comme départ pour un vol, par exemple.*
- **CROSS JOIN :** jointure croisée. C'est 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. C'est aussi ce qui se passe quand on ne fait PAS de jointure …

III. Exemples De Jointures

INNER JOIN



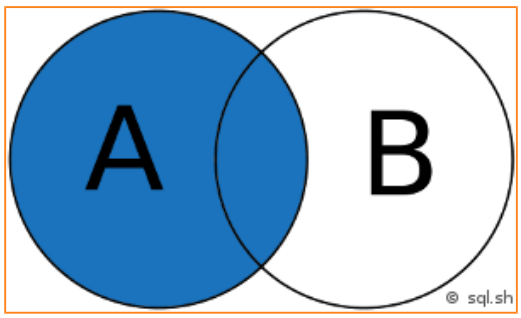
Intersection de 2 ensembles

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

Equivaut à

```
SELECT *
FROM A, B
WHERE A.key = B.key ;
```

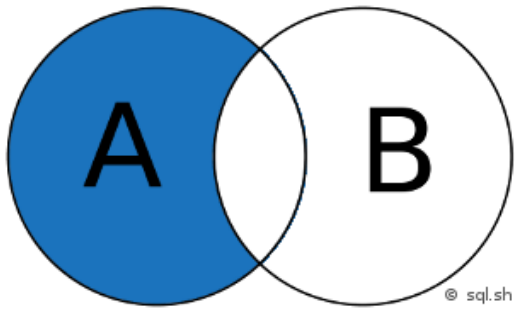
LEFT JOIN



Jointure gauche (LEFT JOINT)

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

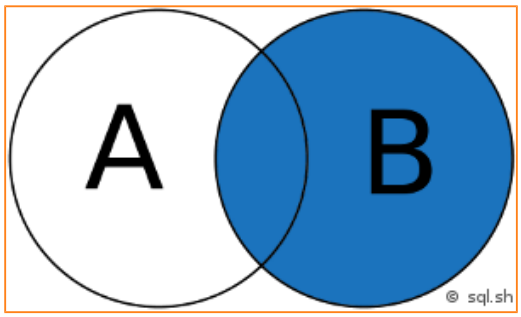
LEFT JOIN (sans l'intersection de B)



Jointure gauche (LEFT JOINT sans l'intersection B)

```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
WHERE B.key IS NULL ;
```

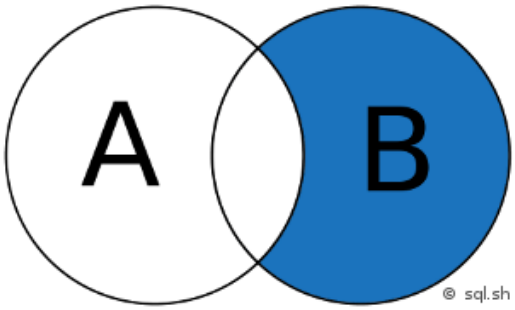
RIGHT JOIN



Jointure droite (RIGHT JOINT)

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

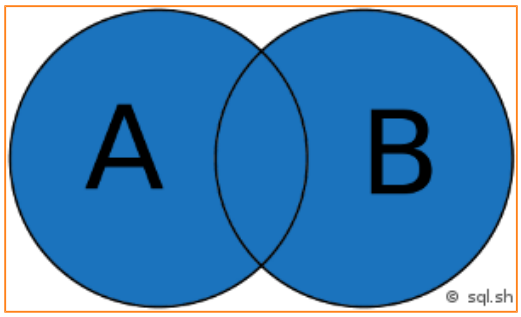
RIGHT JOIN (sans l'intersection de A)



Jointure droite (RIGHT JOINT sans l'intersection A)

```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
WHERE B.key IS NULL ;
```

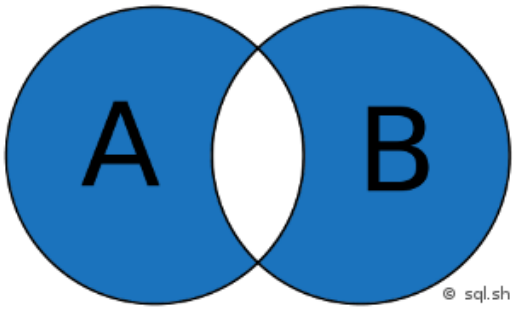
FULL JOIN



Union de 2 ensembles

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

FULL JOIN (sans intersection)



Jointure pleine (FULL JOINT sans intersection)

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

Source: sql.sh