# Partie 3

# SUBQUERIES

*Cours*

Une sous-requête consiste à exécuter une requête à l'intérieur d'une autre requête. Une requête imbriquée est souvent utilisée au sein d'une clause WHERE ou de HAVING pour remplacer une ou plusieurs constante.

## Syntaxe

```
SELECT *
FROM `table`
WHERE `nom_colonne` IN (
    SELECT `colonne`
    FROM `table2`
    WHERE `condition
);
```

```
SELECT subtable.column FROM
(
SELECT * FROM table
WHERE condition) AS
subtable
```



TECH & CODE FACTORY

# Subqueries

*Exemples*

```
SELECT * FROM customers

WHERE city IN

 (SELECT city FROM customers

  WHERE postal_code LIKE
“6____”);
```

```
SELECT city, AVG(subtable.points)

FROM (SELECT city, points FROM

customers) AS subtable

GROUP BY subtable.city;
```

# Subqueries



Si vous souhaitez en savoir plus sur les subqueries je vous invite à regarder le lien suivant: https://www.javatpoint.com/mysql-subquery

# Jointures



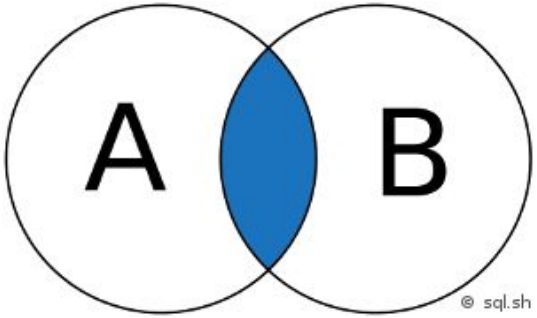
Les jointures en SQL permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.

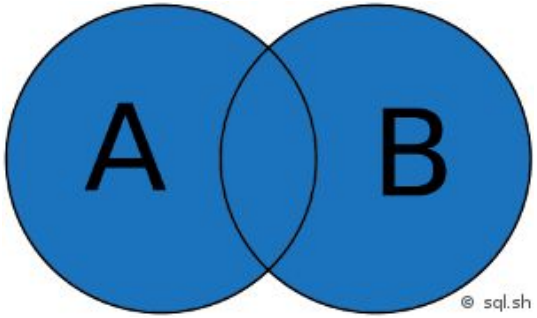
En général, les jointures consistent à associer des lignes de 2 tables en associant l'égalité des valeurs d'une colonne d'une première table par rapport à la valeur d'une colonne d'une seconde table. Imaginons qu'une base de 2 données possède une table "utilisateur" et une autre table "adresse" qui contient 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.

# Jointures

*Cours*

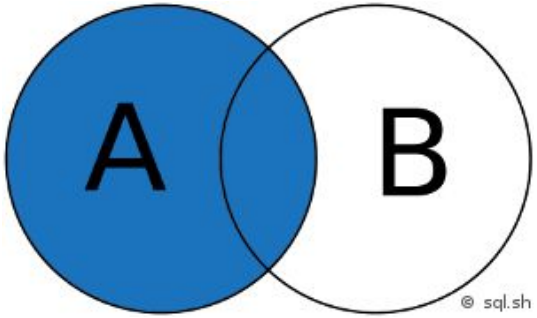
INNER JOIN

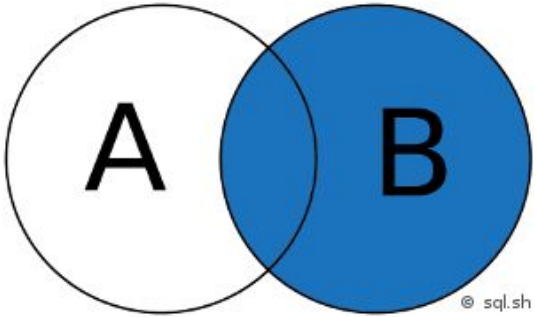



FULL JOIN

LEFT JOIN




RIGHT JOIN

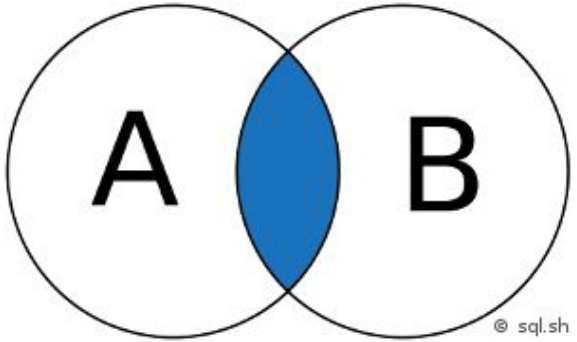


TECH & CODE FACTORY

# INNER JOIN



Cette commande retourne les enregistrements lorsqu'il y a au moins une ligne dans chaque colonne qui correspond à la condition.




## Syntaxe

```
SELECT *

FROM tableA

INNER JOIN tableB ON tableA.id = tableB.fk_id;
```

# INNER JOIN

*Exemples*

Nous voulons afficher toutes les informations des utilisateurs qui ont passé une commande

```
SELECT * FROM customers

INNER JOIN orders ON customers.customer_id = orders.customer_id;
```

L'utilisateur 12 n'est pas affiché car il n'a pas passé de commande

# INNER JOIN

*Exercices*

Affichez la liste des produits qui sont dans au moins une commande.

Affichez la liste des commandes avec leur statut

# LEFT JOIN



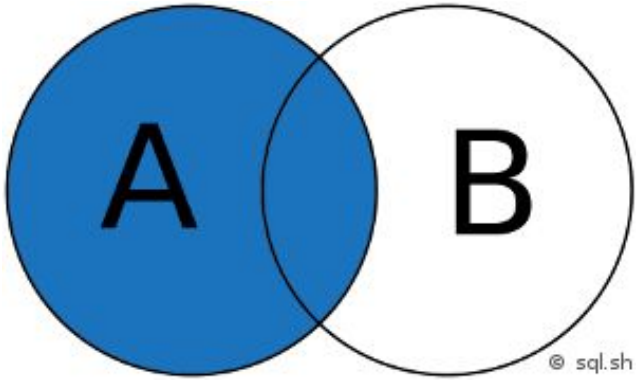
Dans le langage SQL, la commande LEFT JOIN (aussi appelée LEFT OUTER JOIN) permet de lister tous les résultats de la table de gauche même s'il n'y a pas de correspondance dans la deuxième table.

## Syntaxe

```
SELECT *

FROM tableA

LEFT JOIN tableB ON tableA.id = tableB.fk_id;
```



# LEFT JOIN

*Exemples*

Nous voulons afficher toutes les informations des utilisateurs et les informations de leur commandes s'ils en ont passé.

```
SELECT * FROM customers

LEFT JOIN orders ON customers.customer_id = orders.customer_id;
```

L’utilisateur 12 est bien affiché cette fois ci.

# LEFT JOIN

*Exercices*

Affichez la liste des produits et leurs éventuelles commandes

Affichez la liste des commandes et leur éventuel statut

# RIGHT JOIN



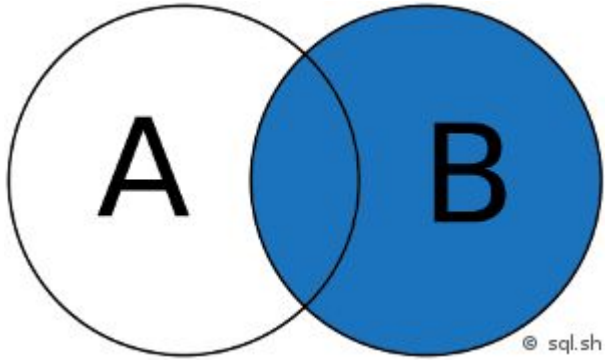
RIGHT JOIN est un type de jointure entre 2 tables qui permet de retourner tous les enregistrements de la table de droite même s'il n'y a pas de correspondance avec la table de gauche. S'il y a un enregistrement de la table de droite qui ne trouve pas de correspondance dans la table de gauche, alors les colonnes de la table de gauche auront NULL pour valeur.

## Syntaxe

```
SELECT *

FROM tableA

RIGHT JOIN tableB ON tableA.id =
tableB.fk_id;
```



# RIGHT JOIN

*Exemples*

Que retourne cette query ?

```
SELECT * FROM orders

RIGHT JOIN order_statuses

ON orders.status = order_statuses.order_status_id;
```

# RIGHT JOIN

*Exercices*

Affichez tous les produits et leurs commandes éventuelles associés

# FULL JOIN



La commande FULL JOIN permet de combiner les résultats des 2 tables, les associer entre eux grâce à une condition et remplir avec des valeurs NULL si la condition n'est pas respectée.

## Syntaxe

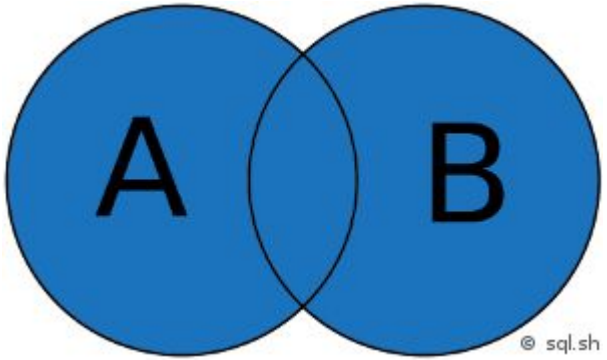
```
SELECT * FROM tableA

LEFT JOIN tableB ON tableA.id = tableB.id

UNION

SELECT * FROM tableA

RIGHT JOIN tableB ON tableA.id = tableB.id
```



# FULL JOIN

*Exemples*

```
SELECT * FROM customers

LEFT JOIN orders ON customers.customer_id = orders.customer_id

UNION

SELECT * FROM customers

RIGHT JOIN orders ON customers.customer_id = orders.customer_id;
```

# FULL JOIN

*Exercices*

Faites un full join entre orders et orders_items

Faites un full join entre products et orders_items

# A vous de jouer !

*Exercices globaux*

Y a t'il des tables dans sql_store dont le résultat est le même en inner join et full join?

Y a t'il des tables dans sql_store dont le résultat est le même en full join et right join/left join ?

Joignez les commandes avec les informations clients et leur statut ensemble

# SELF JOIN

*Exemples*

Un SELF JOIN correspond à une jointure d'une table avec elle-même. Ce type de requête n'est pas si commun mais très pratique dans le cas où une table lie des informations avec des enregistrements de la même table.

Ici la jointure est effectuée avec un LEFT JOIN, mais il est aussi possible de l'effectuer avec d'autres types de jointures.

| id | prenom | nom | email | manager_id |
|---|---|---|---|---|
| 1 | Sebastien | Martin | s.martin@example.com | NULL |
| 2 | Gustave | Dubois | g.dubois@example.com | NULL |
| 3 | Georgette | Leroy | g.leroy@example.com | 1 |
| 4 | Gregory | Roux | g.roux@example.com | 2 |

# SELF JOIN

*Exemples*

```
SELECT u1.u_id, u1.u_nom, u2.u_id, u2.u_nom

FROM utilisateur as u1

LEFT OUTER JOIN utilisateur as u2 ON u2.u_manager_id = u1.u_id
```

| u1_id | u1_prenom | u1_nom | u1_email | u1_manager_id | u2_prenom | u2_nom |
|---|---|---|---|---|---|---|
| 1 | Sebastien | Martin | s.martin@example.com | NULL | NULL | NULL |
| 2 | Gustave | Dubois | g.dubois@example.com | NULL | NULL | NULL |
| 3 | Georgette | Leroy | g.leroy@example.com | 1 | Sebastien | Martin |
| 4 | Gregory | Roux | g.roux@example.com | 2 | Gustave | Dubois |

# NATURAL JOIN



La commande NATURAL JOIN permet de faire une jointure naturelle entre 2 tables. Cette jointure s'effectue à la condition qu'il y ai des colonnes du même nom et de même type dans les 2 tables. Le résultat d'une jointure naturelle est la création d'un tableau avec autant de lignes qu'il y a de paires correspondant à l'association des colonnes de même nom.

## Syntaxe

```
SELECT *

FROM table1

NATURAL JOIN table2
```

# NATURAL JOIN

*Exemples*

```
SELECT *

FROM customers

NATURAL JOIN orders;
```

# CROSS JOIN



Dans le langage SQL, la commande CROSS JOIN est un type de jointure sur 2 tables SQL qui permet de retourner le produit cartésien. Autrement dit, cela permet de retourner chaque ligne d'une table avec chaque ligne d'une autre table. Ainsi effectuer le produit cartésien d'une table A qui contient 30 résultats avec une table B de 40 résultats va produire 1200 résultats (30 x 40 = 1200). En général la commande CROSS JOIN est combinée avec la commande WHERE pour filtrer les résultats qui respectent certaines conditions.

**Attention**, le nombre de résultat peut facilement être très élevé. S'il est effectué sur des tables avec beaucoup d'enregistrements, cela peut ralentir sensiblement le serveur.

# CROSS JOIN

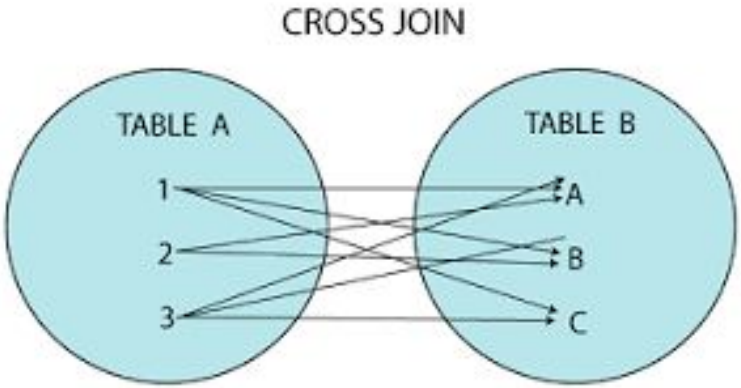


## Syntaxe

```
SELECT *

FROM tableA

CROSS JOIN tableB
```



# CROSS JOIN

*Exemples*

Que retourne cette query ?

```
SELECT * FROM customers

CROSS JOIN products;
```

# CROSS JOIN - NATURAL JOIN

*Exercices*

Effectuer un cross join entre les commandes et leur statut

Effectuer un natural join entre orders et order_items

# A vous de jouer !

*Exercices globaux*

1. Affichez les téléphones des clients dont la commande est en acheminement
2. Combien de commandes sont dans chaque statut
3. Affichez le nombre de commandes passés pour chaque utilisateur
4. Affichez le prix total pour chaque commande
5. Combien de recette avons nous fait depuis le lancement de notre magasin ?
6. Affichez combien chaque client à dépensé dans notre magasin
7. Afficher pour chaque client le nombre total d'articles acheté (toutes commandes confondues)
8. Quel est l'article le plus acheté du magasin?
9. Affichez tous les clients ayant commandé au moins 2 articles différents
10. Combien est ce que les clients dépensent en moyenne dans notre magasin
11. Combien de commandes est ce que les clients font en moyenne sur notre magasin
12. Affichez les adresses des clients qui ont dépensé + de 100€ sur notre store et dont la commande n'a pas encore été envoyée
13. Affichez pour chaque produit son stock futur (on prend en compte toutes les commandes)
14. Afficher les produits à restocker urgemment (dont le stock futur <10)
15. Affichez l'article préféré de (le plus acheté) pour chaque client (Attention: difficile !)

# Bonus

*Exercices globaux*

- Trouver comment faire un minus en mySQL
- Trouver comment faire un intersect en  mySQL
- Trouver comment faire un left join sans l'intersection
- Trouver comment faire un right join sans l'intersection
- Trouver comment faire un full join sans l'intersection
- Quelle est la différence entre un INNER JOIN et un UNION ?