



**ECOLE MAROCAINE DES  
SCIENCES DE L'INGENIEUR**  
*Membre de* **HONORIS UNITED UNIVERSITIES**

École Marocaine des Sciences de l'Ingénieur

**Filière: IIR**

# **Les Bases de Données**

**Prof. Zakaria KHATAR**

# **LANGAGE SQL :**

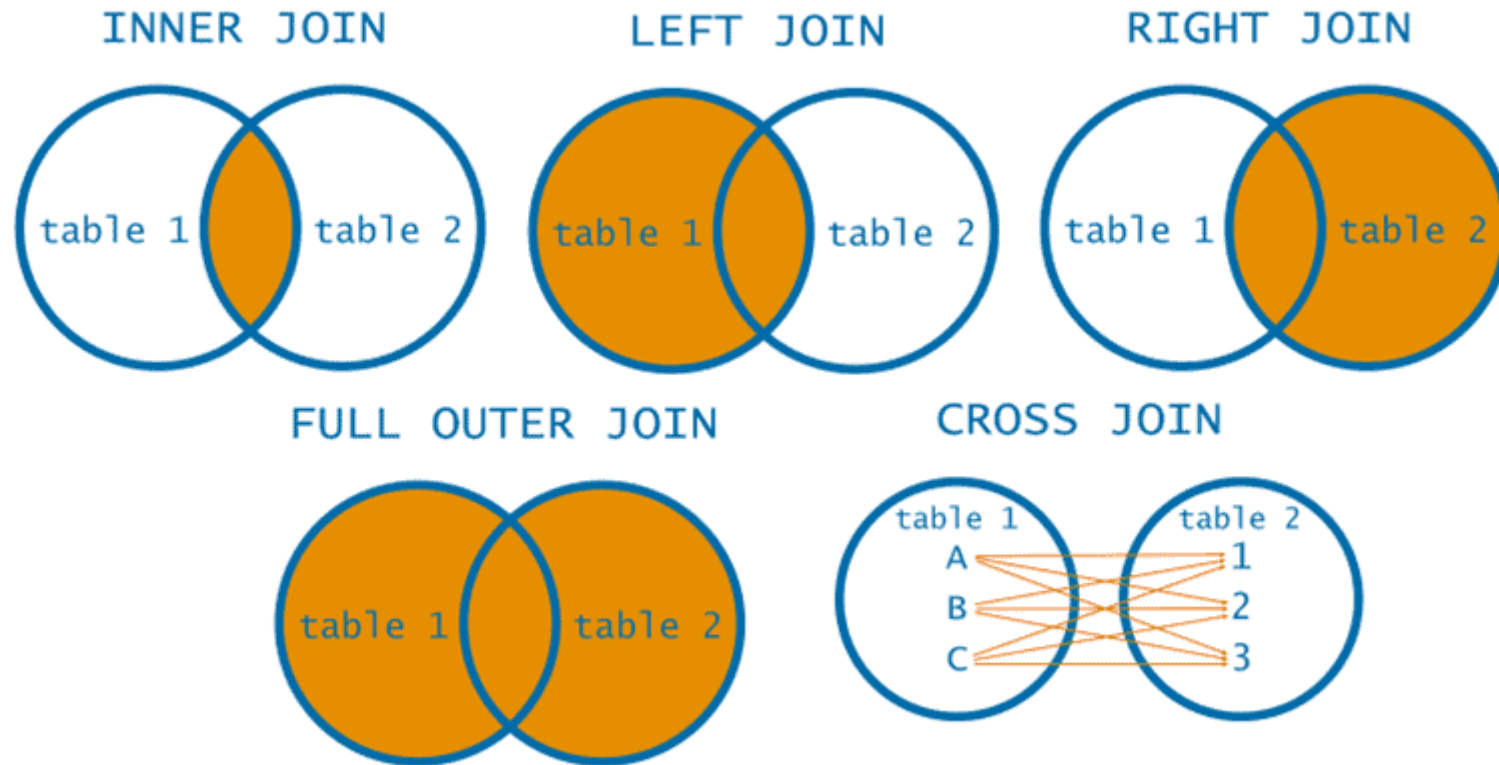
## **Langage d'interrogation des données (LID)**

### **2<sup>ème</sup> Partie**

# Les jointures

## Les jointures

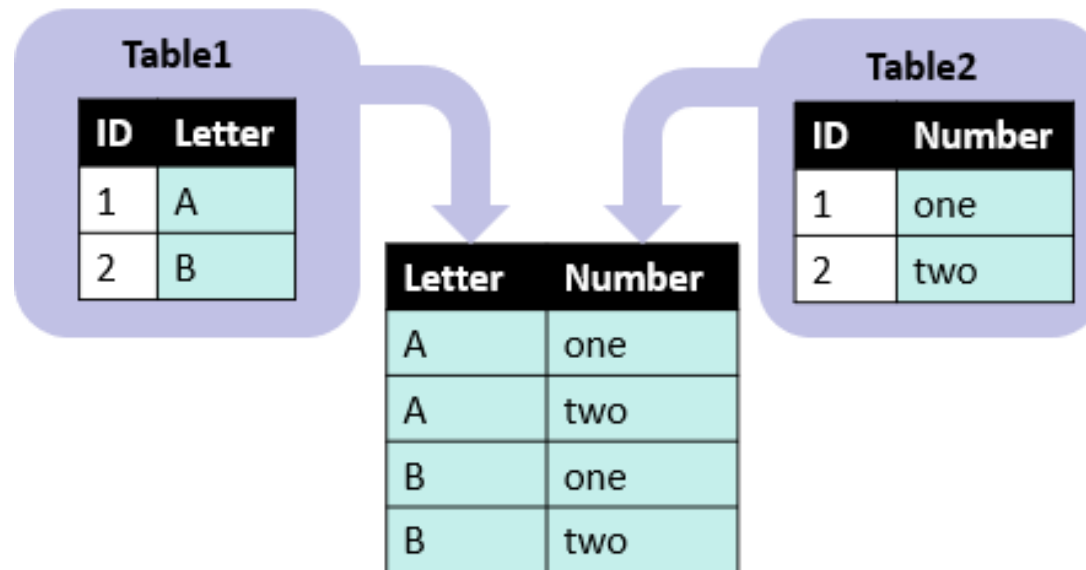
On utilise les jointures lorsque nous souhaitons extraire des données qui se trouvent dans différentes tables. Il existe différents types de jointures.



## 1) Produit cartésien (Jointures croisée ou Cross Join) :

Est une requête de sélection qui met en jeu **plusieurs tables**. Pour deux tables, la sélection consiste à afficher la première ligne de la première table avec toutes les lignes de la deuxième table, puis la deuxième ligne de la première table avec toutes les lignes de la deuxième table et ainsi de suite.

*Ce type de sélection implique beaucoup de redondances*



# LANGAGE D'INTERROGATION DES DONNÉES : LID

Exemple :

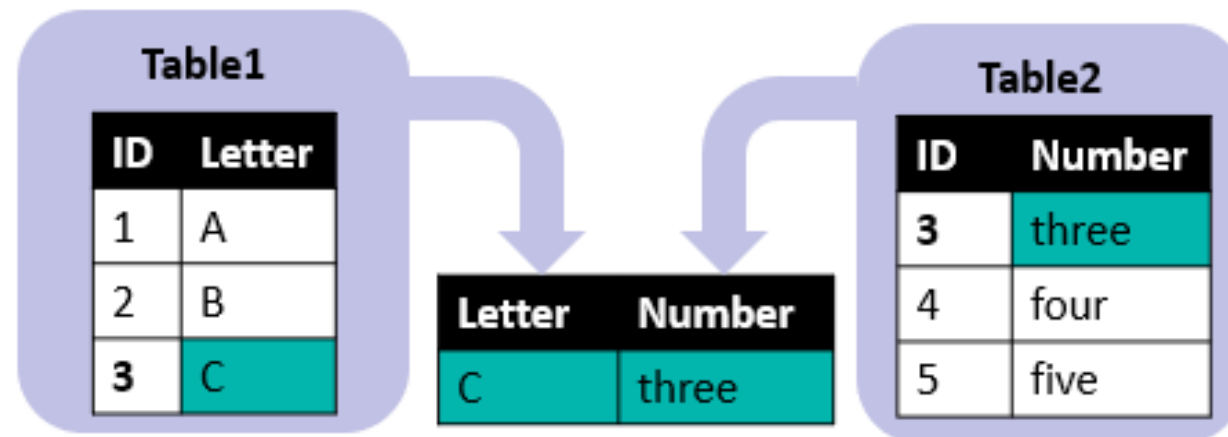
```
select FIRST_NAME, LAST_NAME, COUNTRY_NAME from EMPLOYEES, COUNTRIES;
```

	FIRST_NAME	LAST_NAME	COUNTRY_NAME
1	Donald	OConnell	Argentina
2	Donald	OConnell	Australia
3	Donald	OConnell	Belgium
4	Donald	OConnell	Brazil
5	Donald	OConnell	Canada
6	Donald	OConnell	Switzerland
7	Donald	OConnell	China
8	Donald	OConnell	Germany
9	Donald	OConnell	Denmark
10	Donald	OConnell	Egypt
11	Donald	OConnell	France

## 2) Jointure interne (ou jointure simple) :

La jointure interne **INNER JOIN**, est un type de jointure très commune pour lier plusieurs tables entre-elles dans une même requête.

Cette commande retourne **les lignes lorsqu'il y a au moins une ligne dans chaque colonne listé dans la condition**. Autrement dit, lorsque la condition est respectée dans les deux tables.



## 2) Jointure interne (ou jointure simple) :

La syntaxe :

```
SELECT * FROM table_1  
INNER JOIN table_2 ON table_1.colonne1 = table_2.colonne1;
```

Ou :

```
SELECT * FROM table_1, table_2  
WHERE table_1.colonne1 = table_2.colonne1;
```

Cette requête SQL va donc sélectionner grâce à la commande INNER JOIN les lignes des tables **table\_1** et **table\_2** lorsque les données de la colonne **colonne1** de la table **table\_1** est égal aux données de la colonne **colonne2** de la table **table\_2**.

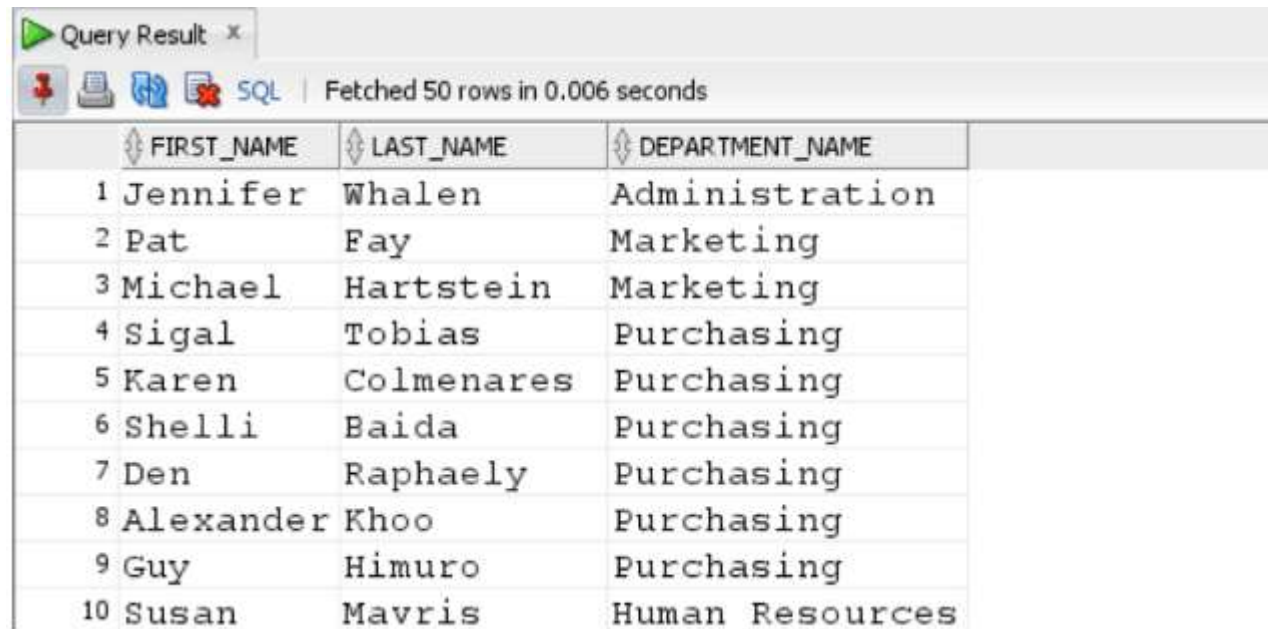


# LANGAGE D'INTERROGATION DES DONNÉES : LID

Exemple :

```
SELECT first_name, last_name, department_name FROM employees e  
INNER JOIN departments d ON e.department_id = d.department_id;
```

Résultat :



The screenshot shows a 'Query Result' window with a toolbar containing icons for a pin, print, refresh, and SQL. Below the toolbar, it states 'Fetched 50 rows in 0.006 seconds'. The table below has three columns: FIRST\_NAME, LAST\_NAME, and DEPARTMENT\_NAME. The data is as follows:

	FIRST_NAME	LAST_NAME	DEPARTMENT_NAME
1	Jennifer	Whalen	Administration
2	Pat	Fay	Marketing
3	Michael	Hartstein	Marketing
4	Sigal	Tobias	Purchasing
5	Karen	Colmenares	Purchasing
6	Shelli	Baida	Purchasing
7	Den	Raphaely	Purchasing
8	Alexander	Khoo	Purchasing
9	Guy	Himuro	Purchasing
10	Susan	Mavris	Human Resources

### 2-1) Jointure interne en utilisant la clause USING :

On peut réécrire la même requête que précédemment, en insérant la clause **USING** dans la requête au lieu de **ON** :

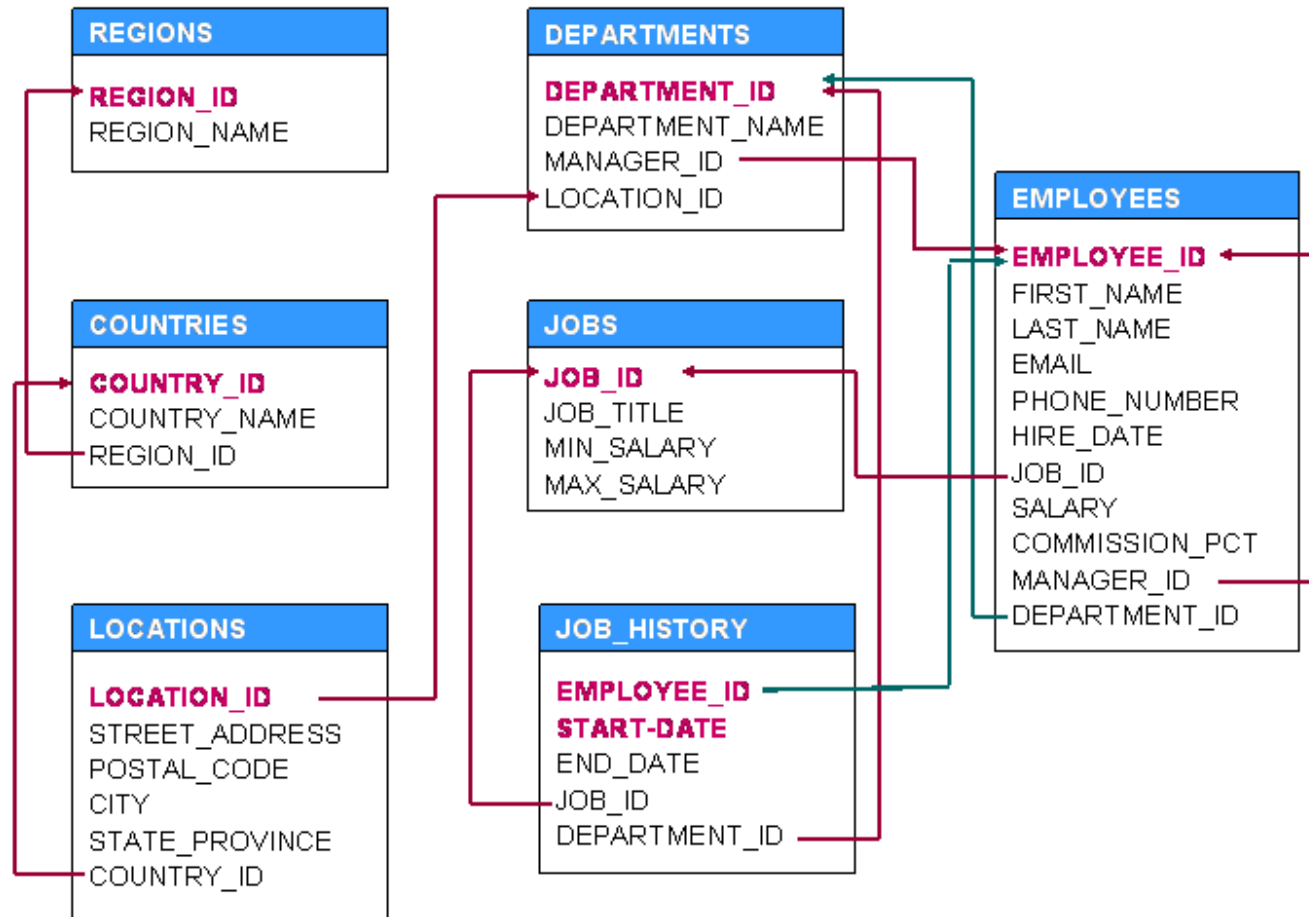
```
SELECT first_name, last_name, department_name FROM employees  
INNER JOIN departments USING (department_id);
```

Pour utiliser la clause **USING**, la clé étrangère et la clé primaire des deux tables doivent avoir le même nom.

# LANGAGE D'INTERROGATION DES DONNÉES : LID

## Exemple :

Si je veux afficher les employés de 'United Kingdom', donc je dois joindre plusieurs tables :



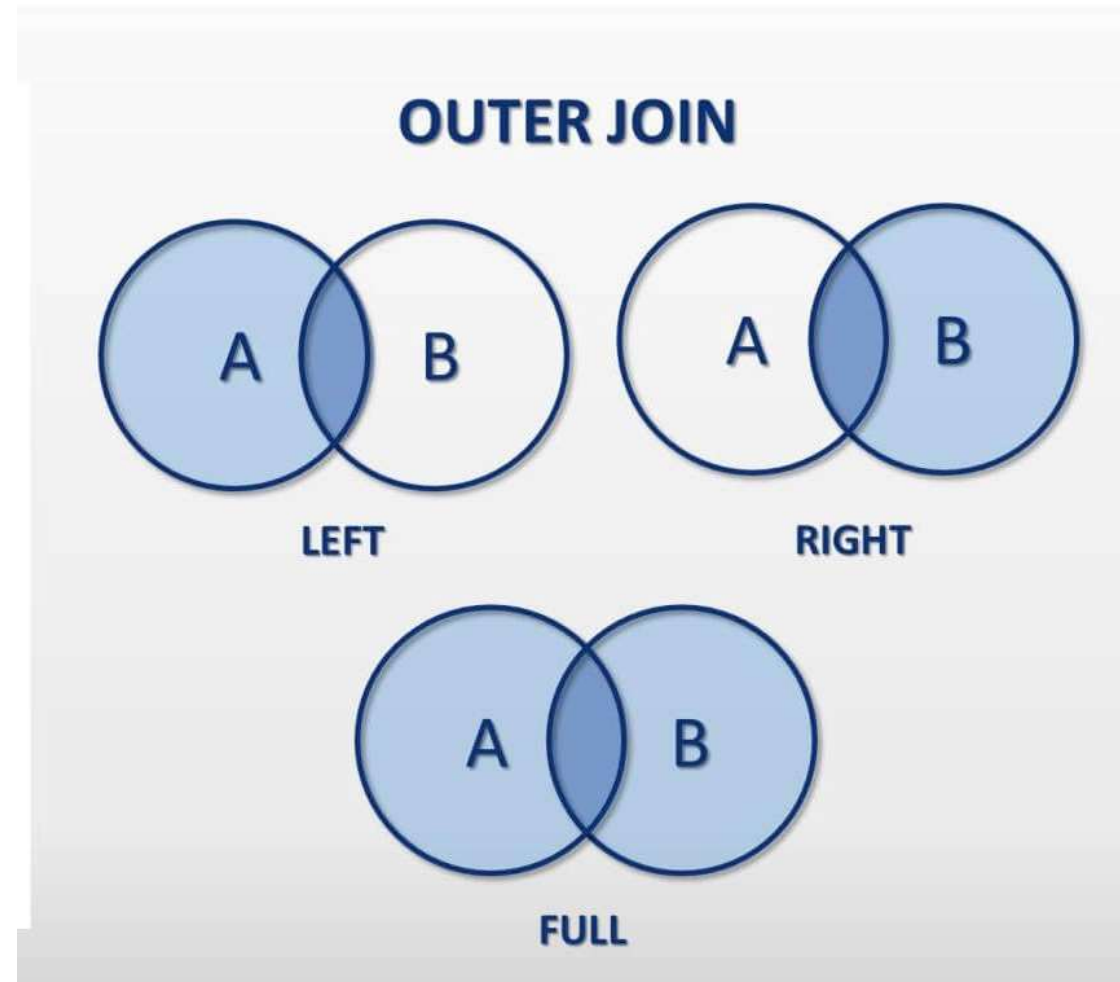
## Exemple :

```
SELECT first_name, last_name, country_name FROM employees
INNER JOIN departments USING (department_id)
INNER JOIN locations USING (location_id)
INNER JOIN countries USING (country_id)
WHERE country_name='United Kingdom'
```

## Résultat :

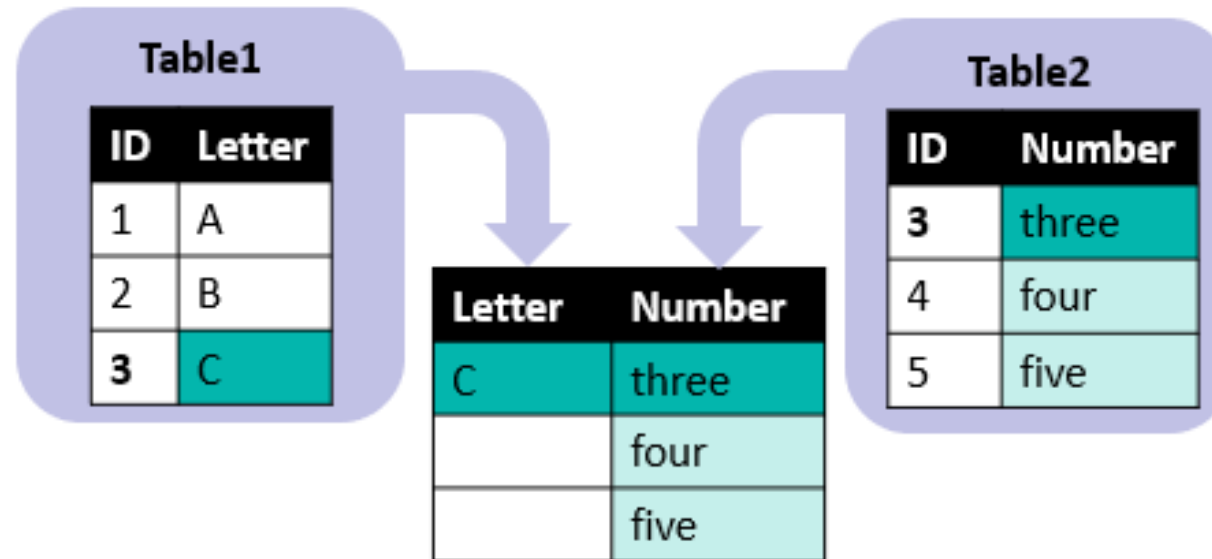
	FIRST_NAME	LAST_NAME	COUNTRY_NAME
24	Amit	Banda	United Kingdom
25	Lisa	Ozer	United Kingdom
26	Harrison	Bloom	United Kingdom
27	Tayler	Fox	United Kingdom
28	William	Smith	United Kingdom
29	Elizabeth	Bates	United Kingdom

## 3) Jointure externe (OUTER JOIN) :



## 3-1) Jointure externe droite (RIGHT OUTER JOIN)

Dans la jointure externe droite, des lignes de table à droite de la jointure seront ramenés même si ceux-ci n'ont pas d'occurrences dans la première table.



Syntaxe :

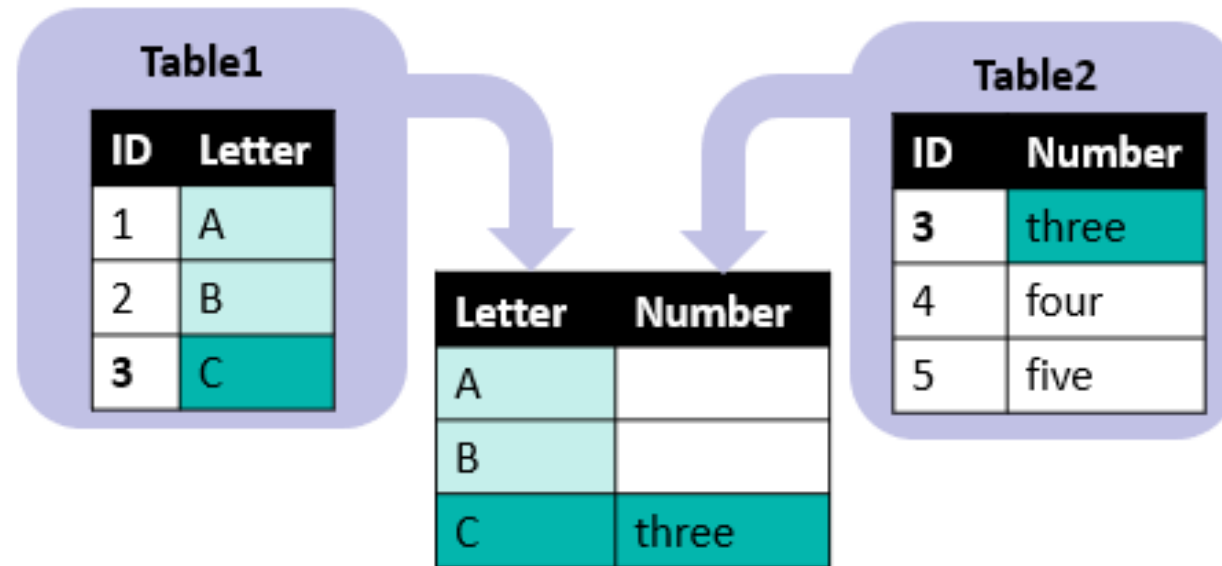
```
SELECT * FROM table_1  
RIGHT OUTER JOIN table_2 ON table_1.colonne1 = table_2.colonne1;
```

Ou, Lorsque la clé étrangère et la clé primaire des deux tables portent le même nom, on peut utiliser la clause **USING**.

```
SELECT * FROM table_1  
RIGHT OUTER JOIN table_2 USING (colonne1);
```

## 3-2) Jointure externe gauche (LEFT OUTER JOIN)

Dans la jointure externe gauche des lignes de table à gauche de la jointure seront ramenés même si ceux-ci n'ont pas d'occurrences dans la deuxième table.





Syntaxe :

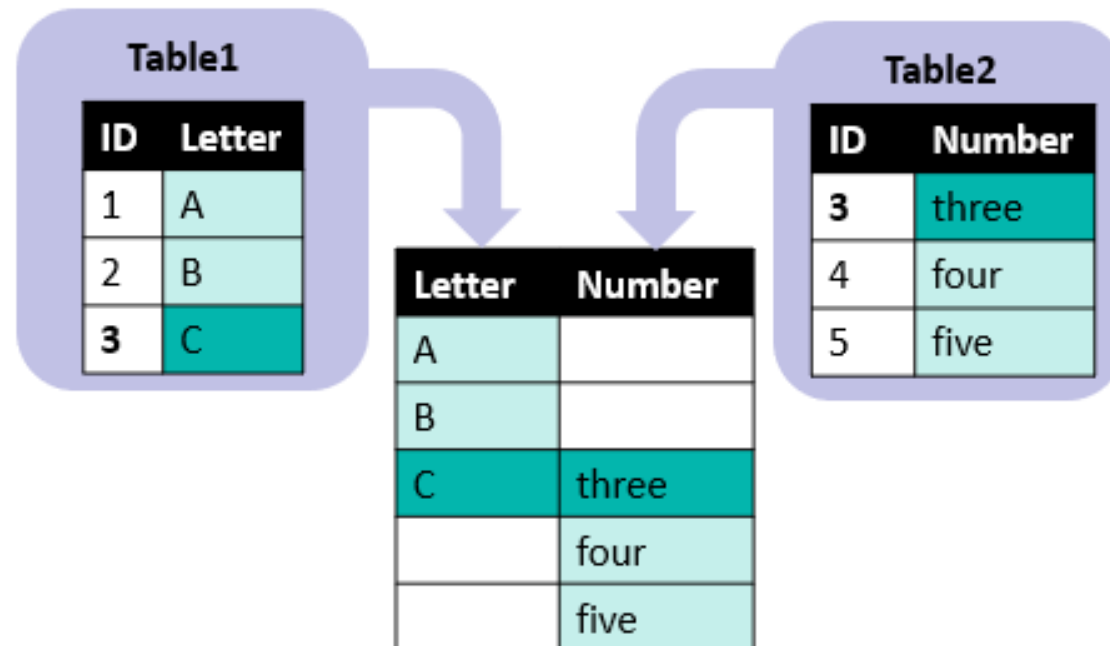
```
SELECT * FROM table_1  
LEFT OUTER JOIN table_2 ON table_1.colonne1 = table_2.colonne1;
```

Ou, Lorsque la clé étrangère et la clé primaire des deux tables portent le même nom, on peut utiliser la clause **USING**.

```
SELECT * FROM table_1  
LEFT OUTER JOIN table_2 USING (colonne1);
```

## 3-3) Jointure externe FULL OUTER JOIN

Un **FULL OUTER JOIN** est une combinaison de **LEFT OUTER JOIN** et **RIGHT OUTER JOIN**, qui permet de faire une jointure entre 2 tables. L'utilisation de cette commande 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 table_1  
FULL OUTER JOIN table_2 ON table_1.colonne1 = table_2.colonne1;
```

Ou, Lorsque la clé étrangère et la clé primaire des deux tables portent le même nom, on peut utiliser la clause **USING**.

```
SELECT * FROM table_1  
FULL OUTER JOIN table_2 USING (colonne1);
```

## Les Requêtes Imbriquées (sous-requêtes)

## 4) Les requêtes imbriquées (sous-requêtes)

On appelle **une requête imbriquée** ou **une sous requête**, une requête qui se trouve à l'intérieur d'une autre requête.

- Une requête imbriquée peut renvoyer comme résultat une colonne ou une table.
- On peut utiliser des requêtes imbriquées sur autant de niveaux que l'on veut. Le plus important est de maîtriser ce que vous faites.



## 4) Les requêtes imbriquées (sous-requêtes)

### Syntaxe :

Requête imbriquée dans la clause WHERE d'une requête externe:

**Requête  
externe**

```
SELECT ...  
FROM ...  
WHERE [Expression] Opérateur (SELECT ...  
                                FROM ...  
                                WHERE ...);
```

**Requête  
imbriquée**

## 4 -1) Les opérateurs

Opérateur	Description
<b>IN</b>	Vérifie si une valeur est dans un ensemble de valeurs renvoyées par une sous-requête.
<b>EXISTS</b>	Vérifie si une sous-requête renvoie au moins une ligne.
<b>ANY</b>	Vérifie si une condition est vraie pour n'importe quelle valeur dans un ensemble renvoyé par une sous-requête.
<b>ALL</b>	Vérifie si une condition est vraie pour toutes les valeurs dans un ensemble renvoyé par une sous-requête.

## 4 -1) Les opérateurs

### Exemples :

Opérateur	Exemple d'utilisation	Exemple dans le schéma HR
IN	Voir quels employés travaillent à New York.	<b>SELECT</b> * <b>FROM</b> employees <b>WHERE</b> department_id <b>IN</b> ( <b>SELECT</b> department_id <b>FROM</b> departments <b>WHERE</b> location = 'New York');
EXISTS	Trouver si certains départements ont des employés.	<b>SELECT</b> * <b>FROM</b> departments <b>WHERE EXISTS</b> ( <b>SELECT</b> * <b>FROM</b> employees <b>WHERE</b> employees.department_id = departments.department_id);
ANY	Chercher des employés gagnant plus qu'un employé du marketing.	<b>SELECT</b> * <b>FROM</b> employees <b>WHERE</b> salary > <b>ANY</b> ( <b>SELECT</b> salary <b>FROM</b> employees <b>WHERE</b> department_id = 5);
ALL	Identifier les employés qui gagnent plus que tous ceux du marketing.	<b>SELECT</b> * <b>FROM</b> employees <b>WHERE</b> salary > <b>ALL</b> ( <b>SELECT</b> salary <b>FROM</b> employees <b>WHERE</b> department_id = 5);