FORMATION

IT - Digital - Management

m2iformation.fr

Langage SQL

Les fondamentaux avec Postgresql

Nehemie “Alfred” BALUKIDI
CTO/Software Engineering & Data|ML/AI

1.

Base de données : les concepts

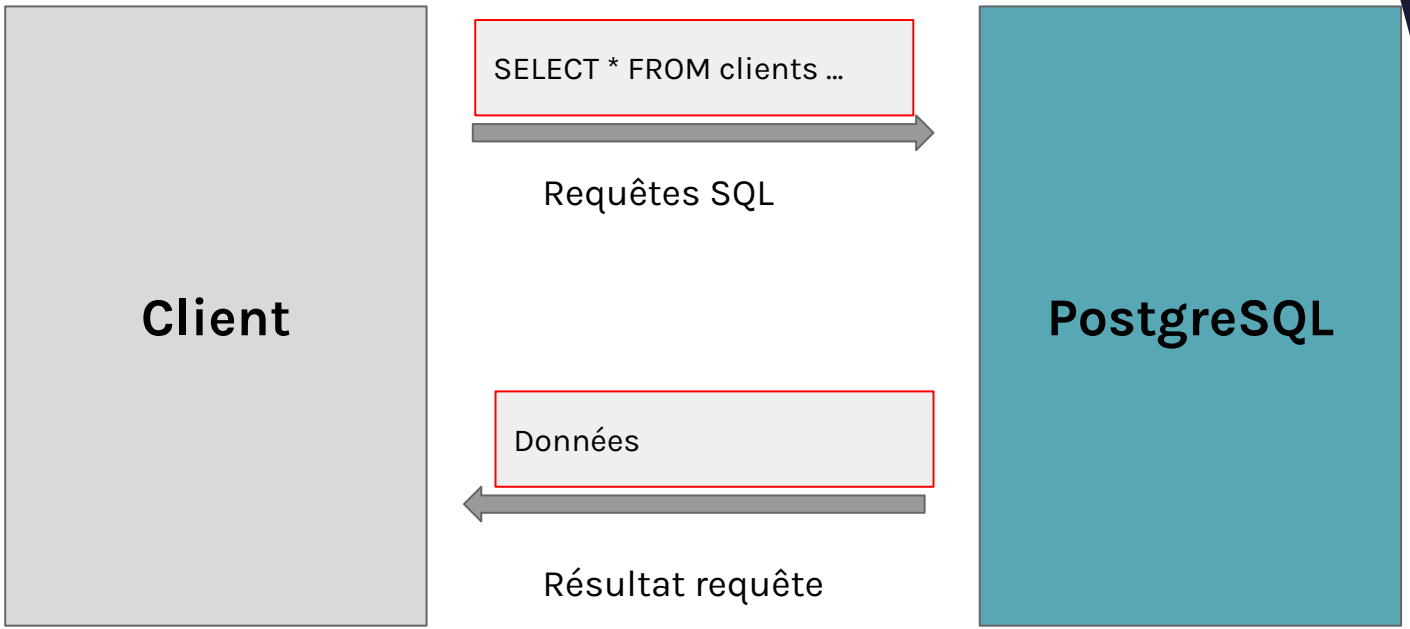
Base de données

Les bases de données servent à la **collecte** et à **l'enregistrement d'informations**.

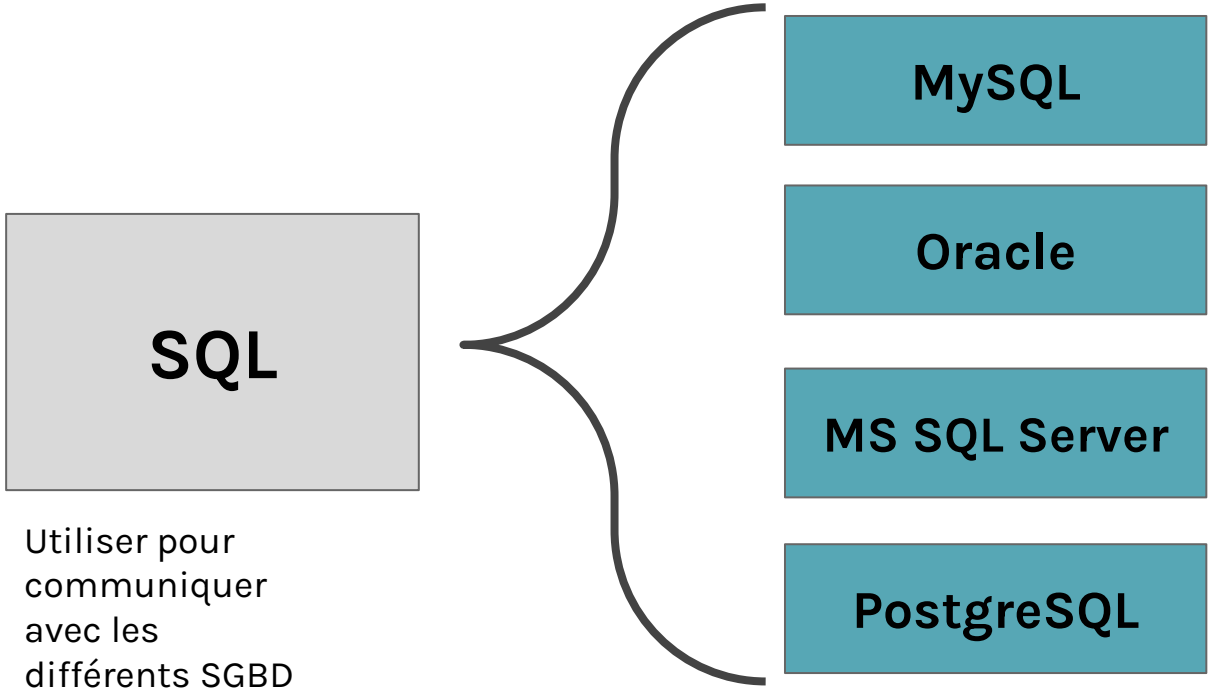
Pour enregistrer une information, vous devez décider quoi enregistrer : c'est le **domaine fonctionnel** de votre projet.

C’est grâce aux SGBD ou système de gestion de base de données que nous allons enregistrer et manipuler les données grâce au langage SQL.

Communiquer un SGBD



Communiquer un SGBD



Utiliser pour communiquer avec les différents SGBD

Base de données

Il existe plusieurs natures des SGBD,mais les plus connues sont:

- SGBD relationnels : les données sont représentées dans différents tableaux pouvant être liés entre eux

- SGBD NoSQL : Les données sont organisées dans d'autres structures(clé-valeur, graph,document,etc.)

Base de données : Modélisation

Avant de stocker les données, nous devons bien comprendre le domaine fonctionnel, ressortir les liens qui existent entre les données.

Pour y arriver nous allons modéliser le domaine.

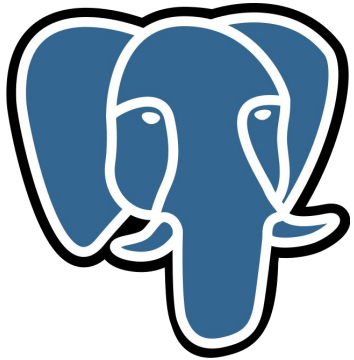
Installation de PostgreSQL

Windows : https://www.postgresql.org/download/windows/

MacOS :

https://postgresapp.com/

https://www.pgadmin.org/download/



2.

Base de données : modélisation

Modélisation

➔ **Pourquoi modéliser ?**

- Avoir une représentation graphique de la structure
- Connaître les propriétés attendues d'une données
- Connaître les relations entre les données
-

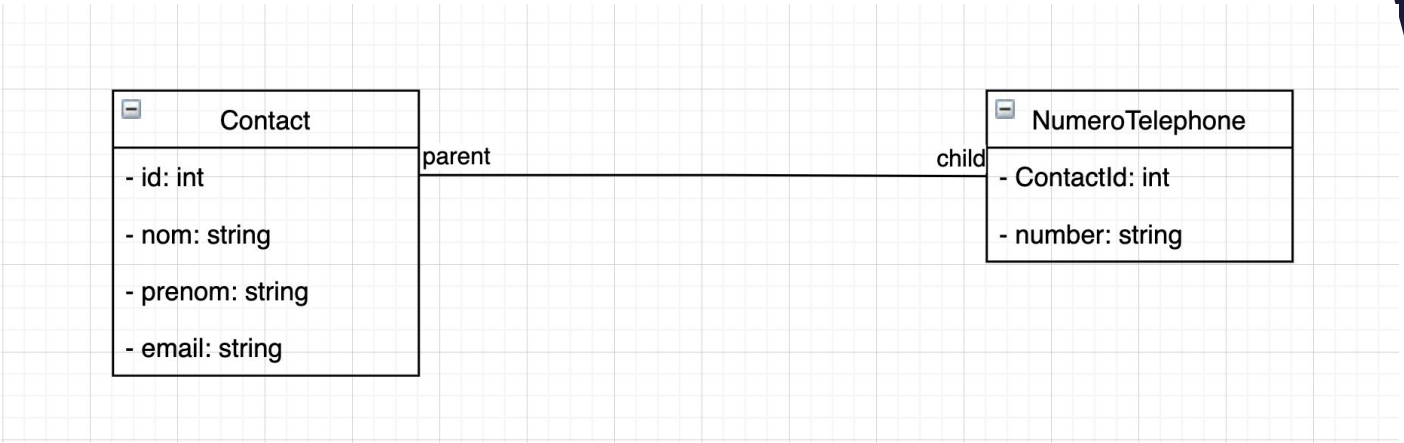
➔ **Comment modéliser ?**

- Effectuer un design conceptuel
- Insérer des cardinalités
- Effectuer un modèle logique

Modélisation : les règles à respecter

- **Normalisation des tables**
  - Ne contient pas d'espace, d'accents ni de caractères spéciaux
  - Tout doit être écrit en lowercase (minuscule)
  - Les espaces sont remplacés par des underscores : "_"
- **Les relations**
  - Les relations "1,n" : l'identifiant de la table mère est dupliqué en clef étrangère de la table fille
  - Les relations "n, n":
  - Effectuer un modèle logique
- **Un identifiant**
  - Un champ d'identification unique est obligatoire, la clef primaire (souvent nommée "id").

Modélisation : 1er Cas 1 à n



Modélisation : 2ème n à n

Modéliser un système dans une école qui permet de stocker des informations sur les étudiants ainsi que les cours auxquels ceux-ci sont inscrits.

N.B : Ici, nous n’allons pas tenir compte de la relation entre la formation suivie et les cours.

Modélisation : 3ème Cas 1 à 1

Modéliser la relation entre un utilisateur et son compte.

3.

Base de données : Le SQL

3.1.

Data Definition Language(DDL)

Créer et supprimer une table

➔ **Création d'une table**

- CREATE TABLE <nom_table>(<nom_colonne> <type_colonne> <contraintes>, …)
- CREATE TABLE IF NOT EXISTS <nom_table>(…)

Ex : **CREATE TABLE IF NOT EXISTS products(id ISERIAL PRIMARY KEY, email VARCHAR(50),…);**

➔ **Suppression d'une table**

- DROP TABLE <nom_table>
- DROP TABLE IF EXISTS <nom_table>

Ex : **DROP TABLE IF EXISTS products;**

Modifier une table

➔ **Ajouter une colonne**

- ALTER TABLE <nom_table> ADD COLUMN <nom_colonne> TYPE <type_colonne> <contraintes>

  Ex : **ALTER TABLE users ADD COLUMN password VACHAR(255) NOT NULL;**

➔ **Modification d'une colonne de table**

- ALTER TABLE <nom_table> ALTER COLUMN <nom_colonne> TYPE <type_colonne> <contraintes>

  Ex :

  ALTER TABLE users ALTER COLUMN email TYPE CHAR(10);
  ALTER TABLE users ALTER COLUMN email SET NOT NULL;
  ALTER TABLE users ALTER ADD UNIQUE(email);

Modifier une table(suite)

➔ **Renommer une colonne sans spécifier de nouveau type**

- ALTER TABLE <nom_table> RENAME COLUMN <ancien_nom> TO <nouveau_nom>

  Ex : **ALTER TABLE users RENAME COLUMN email_address TO email VARCHAR(50);**

3.2.

Data Manipulation Language(DML)

SQL: Les requêtes (CRUD)

- **Lire des données**
  - SELECT <nom_colonne> FROM <nom_table>
  - SELECT <nom_colonne> FROM <nom_table> WHERE <condition>
  - SELECT <nom_colonnes> FROM <nom_table> where <colonne> (NOT)

    IN (‘st1’,’st2’)
  - SELECT <nom_colonnes> FROM <nom_table> where <colonne>

    BETWEEN 1 AND 2
  - SELECT <nom_colonnes> FROM <nom_table> where <colonne> <> 2

SQL: Les requêtes (CRUD)

- **Ajouter des données**
  - INSERT INTO <nom_table>(col1,col2) VALUES (‘valeur1’, ‘valeur2’, ….)
- **Modifier des données**
  - UPDATE <nom_table> SET <nom_colonne> = ‘valeur’ WHERE <condition>
- **Supprimer des données**
  - DELETE FROM <nom_table> WHERE <condition>

SQL: Extraire des données spécifiques

- **LIKE**
  - SELECT <liste_colonnes> FROM <nom_table> WHERE <nom_colonne> LIKE “%fin_chaine”
  - SELECT <liste_colonnes> FROM <nom_table> WHERE <nom_colonne> LIKE “debut_chaine%”
  - SELECT <liste_colonnes> FROM <nom_table> WHERE <nom_colonne> LIKE “%chaine%”
- **ORDER BY**
  - SELECT <liste_colonnes> FROM <nom_table> ORDER BY <nom_colonne> ASC
  - SELECT <liste_colonnes> FROM <nom_table> ORDER BY <nom_colonne> DESC
- **DISTINCT**
  - SELECT DISTINCT(<colonne>) FROM <nom_table>

SQL: Travailler avec les strings

- **CONCAT : Concatenation des chaines**
  - SELECT CONCAT(<col1>,<col2>) from <nom_table>
  - SELECT CONCAT(<col1>," ",<col2>) as “<alias>” from <nom_table>

- **UPPER : Mettre une chaine en majuscule**
  - SELECT UPPER(<colonne>) FROM <nom_table>

- **LOWER : Mettre une chaîne en minuscule**
  - SELECT LOWER(<colonne>) FROM <nom_table>

- **SUBSTR : Sous chaîne**
  - SELECT SUBSTR(<colonne>,<debut>,<fin>) FROM <nom_table>

SQL: Aggregation

- **COUNT : Retourne le nombre des lignes**
  - SELECT COUNT(*) from <nom_table>
- **MIN : Retourner la valeur minimale**
  - SELECT MIN(<colonne>) FROM <nom_table>
- **MAX : Retourner la valeur maximale**
  - SELECT MAX(<colonne>) FROM <nom_table>
- **SUM : Calcul la somme**
  - SELECT SUM(<colonne>) FROM <nom_table>
- **AVG : Calcul la moyenne**
  - SELECT AVG(<colonne>) FROM <nom_table>

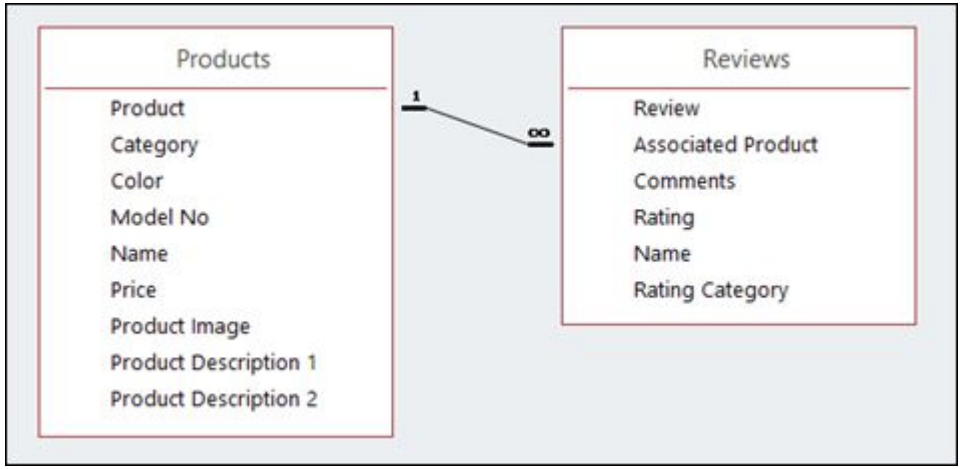
4.

Associations & Jointures

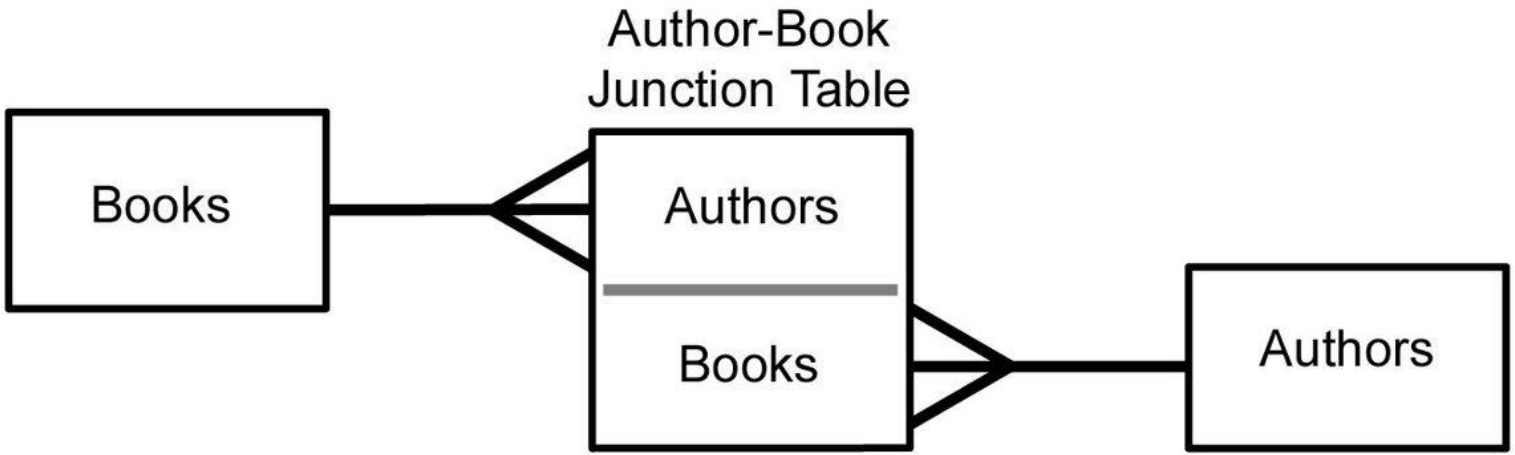
4.1.

Associations

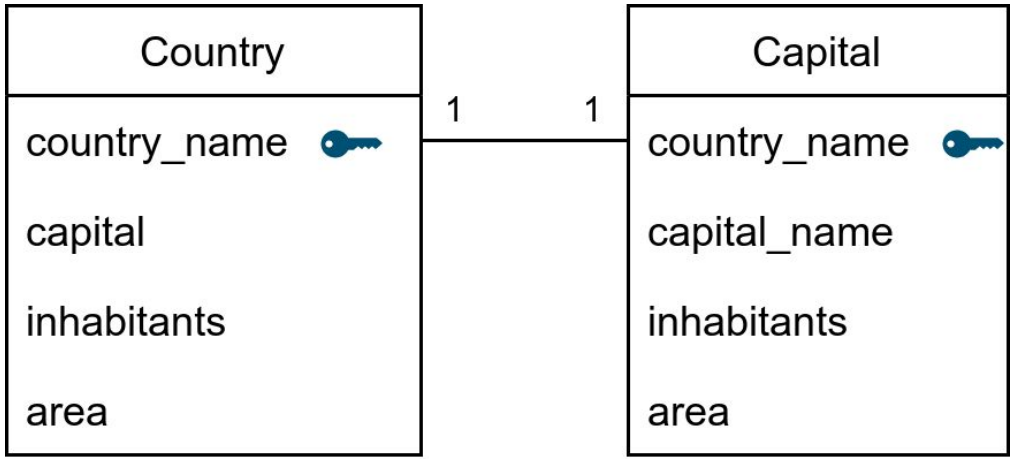
SQL - Les jointures : 1 à plusieurs



SQL - Les jointures : plusieurs à plusieurs



SQL - Les jointures : 1 à 1

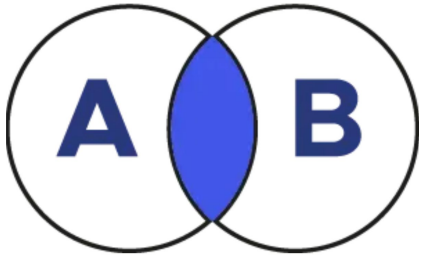


4.2.

SQL - Les jointures

SQL - Les jointures

Le JOIN ou INNER JOIN

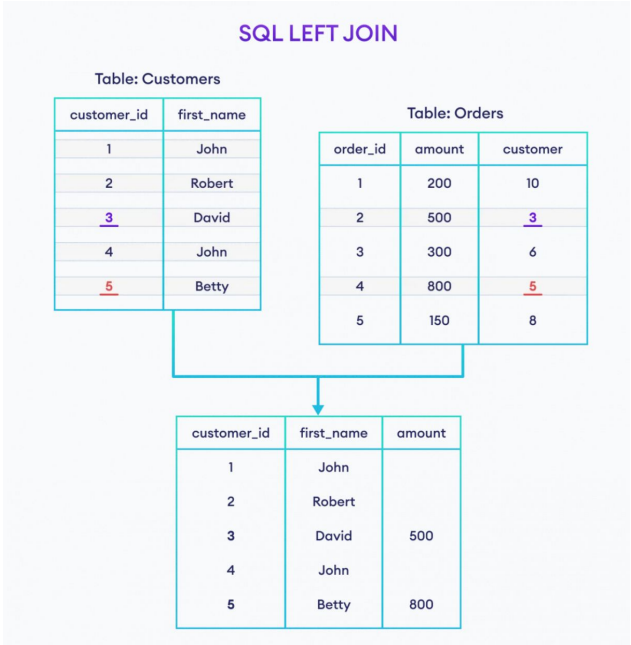


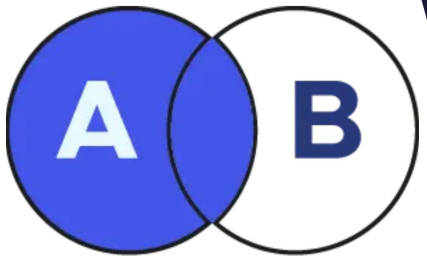
Intersection de deux ensembles

```sql
SELECT c.customer_id, c.first_name, o.amount
FROM customers c
INNER JOIN orders o ON c.customer_id = o.customer_id;
```

SQL - Les jointures

Le LEFT JOIN



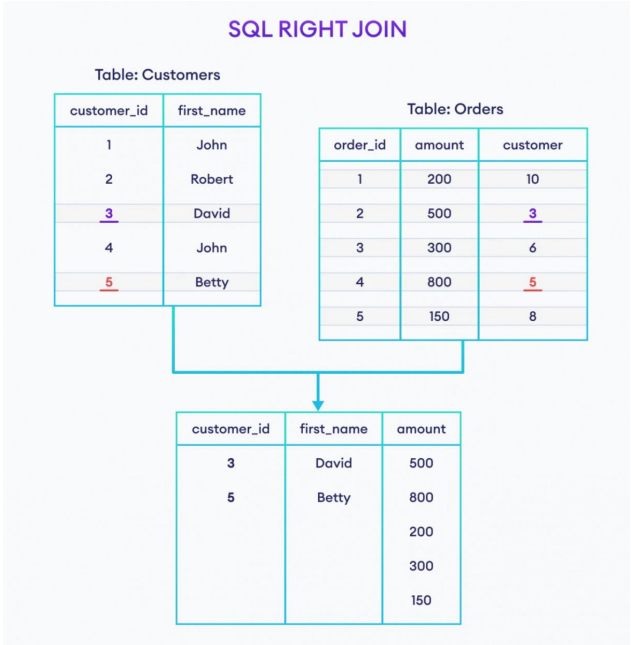


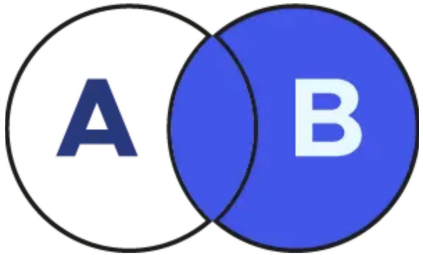
Jointure Gauche (Left Join)

```sql
SELECT c.customer_id, c.first_name, o.amount
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

SQL - Les jointures

Le RIGHT JOIN





Jointure droite (Right Join)

```sql
SELECT c.customer_id, c.first_name, o.amount
FROM customers c
RIGHT JOIN orders o ON c.customer_id = o.customer_id;
```

SQL: GROUP BY

L'instruction **GROUP BY** est souvent utilisée avec les fonctions d'agrégation **(COUNT(), MAX(), MIN(), SUM(), AVG())** pour regrouper l'ensemble des résultats par une ou plusieurs colonnes.

```
SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country;
```

TP 1

Soit la base de données relationnelle des vols quotidiens d'une compagnie aérienne qui aimerait stocker des informations sur les avions, pilotes et vols.

Modéliser et implémenter cette base de données en sachant que:

- Un pilote peut être affecté à plusieurs vols
- Un avion peut être affecté à plusieurs vols
- Pour un vol, on doit pour connaître : le numéro de vol, le pilote, l'avion,ville de départ, ville de destination,heure de départ, heure d'arrivé
- Pour un pilote, on doit connaitre : Le nom,prénom et l'adresse
- Pour l'avion, on doit connaitre : Le fabricant, le modèle,la localité et la capacité

TP 2

A partir du TP précédent :

1. Insérer les avions suivants dans la table Avion :(111,BOEING, 787, 300, NAIROBI), (112,BOEING,737,250,SIDNEY), (113,AIRBUS, 320,220,MILAN),(114,AVIC, 250,150,PEKIN),
2. Afficher tous les avions
3. Afficher tous les avions par ordre croissant sur le fabricant
4. Afficher les noms et les capacités des avions
5. Afficher les localités des avions sans redondance
6. Afficher les avions dans la localité et NAIROBI ou MILAN
7. Modifier la capacité de l'avion numéro 113, la nouvelle capacité et 400
8. Supprimer les avions dans la capacité et inférieure à 200

TP 2(suite)

9. Afficher la capacité maximale, minimale, moyenne des avions
10. Afficher les données des avions dont la capacité est la plus basse
11. Afficher les données des avions dont la capacité et supérieure à la capacité moyenne
12. Afficher le nom et l'adresse des pilotes assurant les vols IT100 et IT104
13. Afficher les numéros des pilotes qui sont en service
14. Afficher les numéros des pilotes qui ne sont pas en service
15. Afficher les noms des pilotes qui conduisent un AIRBUS

Ressources

- https://www.w3schools.com/SQL/sql_check.asp
- https://www.geeksforgeeks.org/difference-between-where-and-having-clause-in-sql/
- https://www.javatpoint.com/where-vs-having#:~:text=The%20main%20difference%20between%20them,filtering%20values%20from%20a%20group.

THE END.