5.

Base de données : Le SQL avancé

Mise à jour des tables et contraites

- Modifier / supprimer une table :

  ALTER TABLE *nom_table* ADD nom type;
  DROP TABLE *nom_table* ADD nom type;

- Ajout et modification de colonne / contrainte :

  ALTER TABLE *nom_table* ADD COLUMN *nom_col* type;
  ALTER TABLE *nom_table* ALTER COLUMN *nom_col* SET *new_val*;
  ALTER TABLE *nom_table* ADD CONSTRAINT *definition_constraint*;

- Mise à jour de données :

  UPDATE *nom_table* SET *nom_col* = *new_val*;

- Exemples :

  ALTER TABLE matable1 ADD COLUMN actif BOOLEAN DEFAULT True;
  UPDATE matable1 SET actif = False WHERE dernier_achat > 30;

Imports / Exports

- Il est possible d'importer et exporter la structure d'une base de données, avec ses données :
- Syntaxe Export Dump :

  pg_dump -U *user* -d *database* > *path*.pgsql (ou *path*.sql)
- Syntaxe Import :

  psql -U *user database* < *path*.pgsql (ou *path*.sql)

Requêtes avec fonctions

- Syntaxe générale :

  SELECT *function_name (column_name)* FROM *table_name*

- Fonctions principales :

  COUNT: Compte le nombre d'élément.
  AVG: Calcul une moyenne.
  MIN / MAX: Identifie le plus petit/grand.
  SUM: Calcul la somme.
  DISTINCT: Ne pas compter les doublons.

- Affichage / Regroupement :

  GROUP BY: Regrouper les valeurs identiques.
  HAVING « *condition* »: Limiter l'affichage

- Exemple :

  SELECT field1, COUNT(field3) FROM table1 GROUP BY field1 HAVING COUNT(field3) = 10;

Requêtes avec jointures (1/2)

- **Tables d'exemples :**

Nom	Prenom
BENRAMOS	Ahmed
RATO	William
VITEMPS	Corine

Prenom	Age
Ahmed	20
William	60
Corine	28

- **Jointure simple :**

```
SELECT A.Nom nom, A.Prenom prenom, B.Age age
    FROM table1 A, table2 B
    WHERE A.Prenom = B.Prenom
    GROUP BY A.Nom
```

- **Résultat :**

nom	prenom	age
BENRAMOS	Ahmed	20
RATO	William	60
VITEMPS	Corine	28

Requêtes avec jointures (2/2)

- **Tables d'exemples :**

Nom	Prenom
BENRAMOS	Ahmed
RATO	William
VITEMPS	Corine

Prenom	Age
Ahmed	20
William	60
Corine	28

- **Jointure simple :**

```
SELECT Nom AS nom, Prenom AS prenom, Age AS age
        FROM table1
        INNER JOIN table2
        ON table1.Prenom = table2.Prenom
```

- **Résultat :**

nom	prenom	age
BENRAMOS	Ahmed	20
RATO	William	60
VITEMPS	Corine	28

Création de vues

- Les vues:
  Une vue est une « **table virtuelle** » et permet de **pré-enregistré** des requêtes

- Syntaxe :
  CREATE VIEW *view_name* AS « *my_request* »

- Exemple :

```
CREATE VIEW utilisateur_majeur AS
        SELECT nom, prenom, age
        FROM utilisateur
        WHERE age > 18
```

- Requête :
  SELECT * FROM utilisateur_majeur

SQL: Les jointures avec JOIN

- 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.
- Il existe plusieurs types de jointure :

➢ INNER JOIN: Jointure quand la condition est vrai dans les 2 tables (PLUS COMMUN)

➢ LEFT JOIN: Jointure avec tous les enregistrements de la table gauche, même si la condition n'est pas vérifiée dans l'autre table.

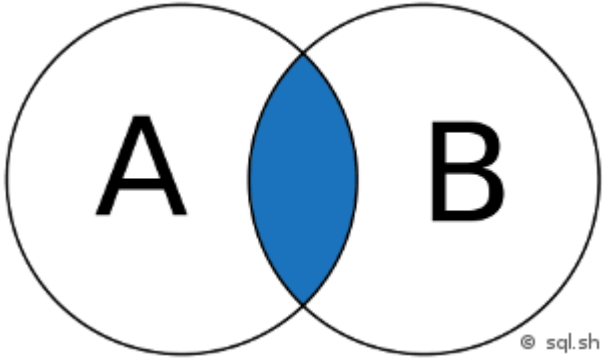
➢ RIGHT JOIN: Jointure avec tous les enregistrements de la table droite, même si la condition n'est pas vérifiée dans l'autre table.

➢ FULL JOIN: Jointure quand la condition est vrai dans au moins une des 2 tables

SQL: Les jointures INNER JOIN

- Voici la syntaxe de l'INNER JOIN :

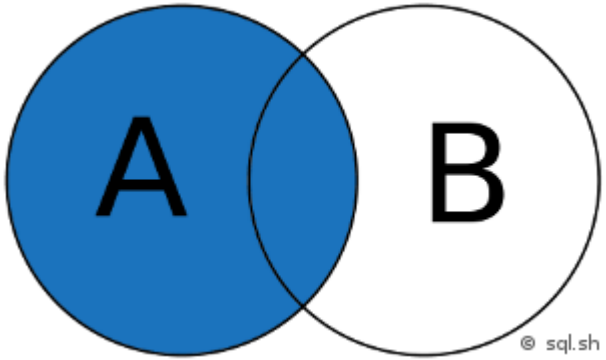
```
SELECT * FROM table1 INNER JOIN table2 ON table1.id = table2.fk_id;
```



SQL: Les jointures LEFT JOIN

- Voici la syntaxe du LEFT JOIN :

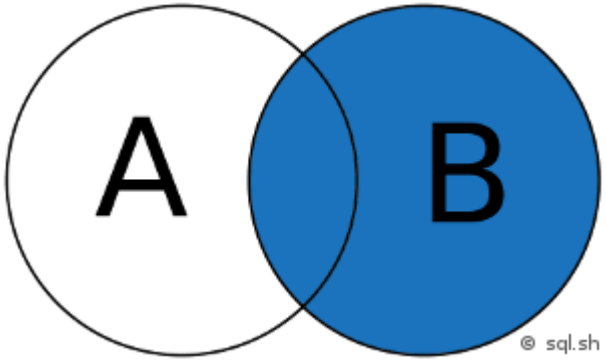
```
SELECT * FROM table1 LEFT JOIN table2 ON table1.id = table2.fk_id;
```



SQL: Les jointures RIGHT JOIN

- Voici la syntaxe du RIGHT JOIN :

```
SELECT * FROM table1 RIGHT JOIN table2 ON table1.id = table2.fk_id;
```



SQL: Les jointures FULL JOIN

- Voici la syntaxe du FULL JOIN :

```
SELECT * FROM table1 FULL JOIN table2 ON table1.id = table2.fk_id;
```