BTS
BREVET DE TECHNICIEN SUPÉRIEUR
LYCÉE QUALIFIANT MOHAMED V
ESSAOUIRA

Modélisation de données et bases de données

Le langage SQL

Fatiha BENDAIDA

LMD

Langage de Manipulation des Données

- Le **LMD** est un sous-ensemble du SQL (*Structured Query Language*) utilisé pour **interagir avec les données stockées** dans une base de données relationnelle. Il permet d'effectuer des opérations de **lecture**, **insertion**, **modification** et **suppression** des enregistrements.

Ajoute de nouveaux enregistrements dans une table.

- Syntaxe:
  - Spécifiez à la fois les noms des colonnes et les valeurs à insérer :

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

  - Si vous ajoutez des valeurs pour toutes les colonnes de la table, il n'est pas nécessaire de spécifier les noms des colonnes dans la requête SQL.

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Ajoute de nouveaux enregistrements dans une table.

- Exemple:
  - Spécifiez à la fois les noms des colonnes et les valeurs à insérer :

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen
21', 'Stavanger', '4006', 'Norway');
```

  - Insérer plusieurs lignes

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode,
Country)
VALUES
('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'),
('Greasy Burger', 'Per Olsen', 'Gateveien 15', 'Sandnes', '4306', 'Norway'),
('Tasty Tee', 'Finn Egan', 'Streetroad 19B', 'Liverpool', 'L1 0AA', 'UK');
```

Modifie des enregistrements existants.

- Syntaxe:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- Exemple:

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

Supprimer des enregistrements d’une table.

- Syntaxe:

```
DELETE FROM table_name WHERE condition;
```

- Exemple:

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

La requête SELECT

- La requête SELECT: permet de récupère des données depuis une ou plusieurs tables.

Lister tous les enregistrements d’une table.

- Syntaxe:

```
SELECT * FROM table_name;
```

- Exemple:

```
SELECT * FROM Customers;
```

Récupération de colonnes spécifiques depuis une table

- Syntaxe:

```
SELECT column1, column2, ...
FROM table_name;
```

- Exemple:

```
SELECT CustomerName, City FROM Customers;
```

SELECT DISTINCT

- La commande SELECT DISTINCT est utilisée pour renvoyer uniquement des **valeurs distinctes** (différentes).

- Syntaxe:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

- Exemple:

```
SELECT DISTINCT Country FROM Customers;
```

La clause WHERE

- La clause WHERE est utilisée pour filtrer les enregistrements.
- Syntaxe:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Exemples:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

```
SELECT * FROM Customers
WHERE CustomerID > 80;
```

ORDER BY

- Le mot-clé ORDER BY permet de trier l'ensemble des résultats par ordre croissant ou décroissant :
- Syntaxe:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

- Exemples:

```
SELECT * FROM Products
ORDER BY Price;
```

```
SELECT * FROM Products
ORDER BY ProductName DESC;
```

ORDER BY

- Exemples :

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

L’operateur AND

- Syntaxe:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

- Exemples:

```
SELECT *
FROM Customers
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';
```

```
SELECT * FROM Customers
WHERE Country = 'Brazil'
AND City = 'Rio de Janeiro'
AND CustomerID > 50;
```

L’operateur OR

- Syntaxe:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

- Exemples:

```
SELECT *
FROM Customers
WHERE Country = 'Germany' OR Country = 'Spain';
```

```
SELECT * FROM Customers
WHERE City = 'Berlin' OR Country = 'Norway';
```

Combinant Les operateurs OR et AND

- Exemple:

```
SELECT * FROM Customers
WHERE Country = 'Spain' AND ( City = 'Berlin' OR Country = 'Norway’);
```

L’operateur NOT

- Syntaxe:

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

- Exemples:

```
SELECT * FROM Customers
WHERE NOT Country = 'Spain';
```

```
SELECT * FROM Customers
WHERE NOT CustomerID > 50;
```

Les fonctions d'agrégation

- Une fonction d'agrégation est une fonction qui effectue un calcul sur un ensemble de valeurs et renvoie une seule valeur.

✓MIN() - Retourne la **valeur minimale** d'une colonne sélectionnée.

✓MAX() - Retourne la **valeur maximale** d'une colonne sélectionnée.

✓COUNT() - Compte le **nombre de lignes** dans un ensemble de résultats.

✓SUM() - Calcule la **somme totale** d'une colonne numérique.

✓AVG() - Calcule la **moyenne arithmétique** d'une colonne numérique.

La fonction Min()

- Syntaxe:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

- Exemples:

```
SELECT MIN(Price)
FROM Products;
```

Utilisation d’alias

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

La fonction Max()

- Syntaxe:

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

- Exemples:

```
SELECT MAX(Price)
FROM Products;
```

Utilisation d’alias

```
SELECT MAX(Price) AS LargestPrice
FROM Products;
```

La fonction count()

- Syntaxe:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

- Exemples:

```
SELECT COUNT(*)
FROM Products;
```

```
SELECT COUNT(ProductID)
FROM Products
WHERE Price > 20;
```

```
SELECT COUNT(DISTINCT Price)
FROM Products;
```

La fonction SUM()

- Syntaxe:

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

- Exemples:

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

```
SELECT SUM(Quantity)
FROM OrderDetails
WHERE ProductId = 11;
```

La fonction SUM()

- En utilisant les alias:

```
SELECT SUM(Quantity) AS total
FROM OrderDetails;
```

- En utilisant une expression:

```
SELECT SUM(Quantity * 10)
FROM OrderDetails;
```

La fonction AVG()

- Syntaxe:

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

- Exemples:

```
SELECT AVG(Price)
FROM Products;
```

```
SELECT AVG(Price)
FROM Products
WHERE CategoryID = 1;
```

La fonction AVG()

- En utilisant les alias:

```
SELECT AVG(Price) AS [average price]
FROM Products;
```

L’operateur LIKE

- **Filtrer** des données textuelles selon un **modèle partiel**.
- Syntaxe:

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

- Exemples:
  - Sélectionnez tous les clients qui commencent par la lettre « a » :

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

L’operateur LIKE

- Exemples:
  - Renvoyer tous les clients d'une ville qui commence par “L” suivi d'un caractère générique, puis de “nd” et de deux caractères génériques

```
SELECT * FROM Customers
WHERE city LIKE 'L_nd__';
```

  - Renvoyer tous les clients d'une ville contenant la lettre « L » :

```
SELECT * FROM Customers
WHERE city LIKE '%L%';
```

L'operateur LIKE

- Exemples:
  - Retourner tous les clients qui commencent par "La" :

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'La%';
```

  - Retourner tous les clients qui commencent par "a" ou qui commencent par "b" :

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%' OR CustomerName LIKE 'b%';
```

  - Retourner tous les clients dont le nom se termine par "a" :

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%a';
```

L’operateur IN

- L'opérateur **IN** permet de spécifier plusieurs valeurs dans une clause WHERE.
- Syntaxe:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

- Exemple:

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

L’operateur NOT IN

- Exemple:

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

L’operateur BETWEEN

- Syntaxe:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

- Exemples:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

L’operateur Between

- Exemples:

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
```

```
SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon
Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

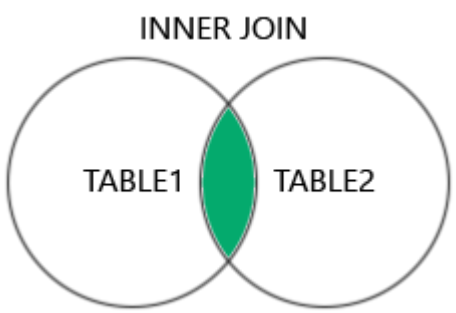
La jointure

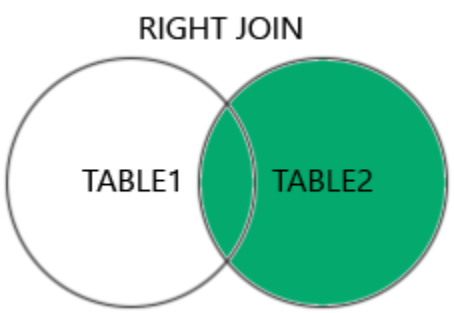
La clause JOIN

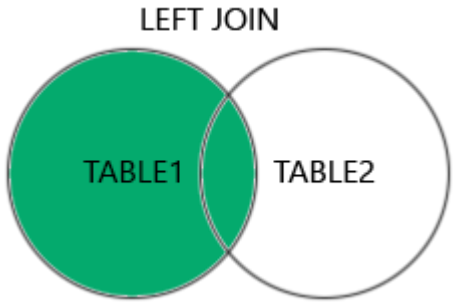
JOIN est utilisée pour combiner des lignes provenant de deux tables ou plus, sur la base d'une colonne liée entre elles.

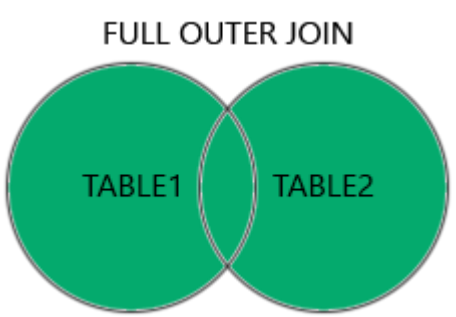
- Exemple :

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```



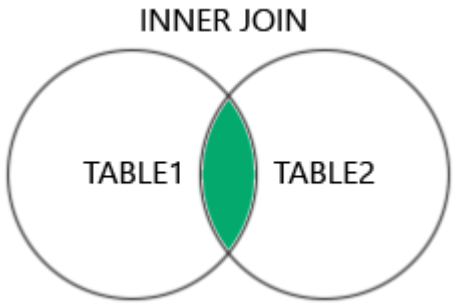






SQL INNER JOIN

Le mot-clé INNER JOIN/JOIN sélectionne les enregistrements dont les valeurs correspondent dans les deux tables.



- Exemple:

```
SELECT ProductID, ProductName, CategoryName
FROM Products
INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;
```

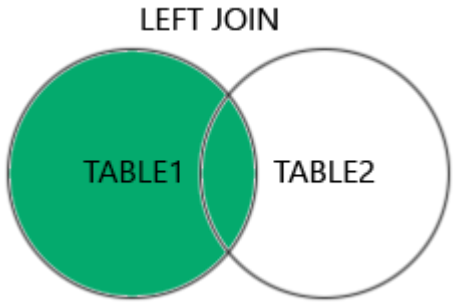
SQL INNER JOIN

- Exemple:

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

SQL LEFT JOIN

Le mot-clé LEFT JOIN renvoie tous les enregistrements de la table de gauche (table1) et les enregistrements correspondants de la table de droite (table2).

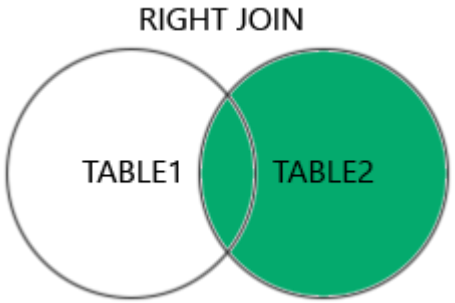


- Exemple:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SQL RIGHT JOIN

Le mot-clé RIGHT JOIN renvoie tous les enregistrements de la table de droite (table2) et les enregistrements correspondants de la table de gauche (table1).

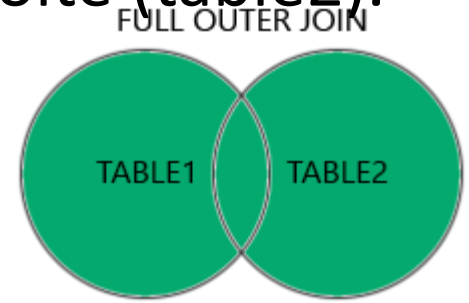


- Exemple:

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

SQL FULL OUTER JOIN

Le mot-clé FULL OUTER JOIN renvoie tous les enregistrements lorsqu'il existe une correspondance dans les enregistrements des tables de gauche (table1) ou de droite (table2).



- Exemple:

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

GROUP BY

- Syntaxe:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

- Exemples:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

```
SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders
 FROM Orders
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID
GROUP BY ShipperName;
```

La clause HAVING

La clause HAVING a été ajoutée à SQL parce que le mot clé WHERE ne peut pas être utilisé avec les fonctions d'agrégation.

- Syntaxe:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

- Exemples:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

L’operateur ANY et ALL

- Les opérateurs ANY et ALL vous permettent d'effectuer une comparaison entre une valeur de colonne unique et une plage d'autres valeurs.

L’operateur ANY

- ANY signifie que la condition sera vraie si l'opération est vraie pour n'importe quelle valeur de la plage
- Syntaxe:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

- Exemple :

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY
  (SELECT ProductID
  FROM OrderDetails
  WHERE Quantity = 10);
```

L’operateur ALL

- L'opérateur ALL :
  - renvoie une valeur booléenne comme résultat
  - renvoie TRUE si TOUTES les valeurs de la sous-requête remplissent la condition
  - est utilisé avec les instructions SELECT, WHERE et HAVING

L’operateur ALL

- Syntaxe:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

- Exemple :

```
SELECT ProductName
FROM Products
WHERE ProductID = ALL
  (SELECT ProductID
  FROM OrderDetails
  WHERE Quantity = 10);
```