# Thứ tự thực hiện của các thành phần trong một câu query

1. **FROM**: Xác định bảng nguồn dữ liệu.
2. **WHERE**: Lọc các bản ghi.
3. **GROUP BY**: Nhóm các bản ghi.
4. **HAVING**: Lọc các nhóm bản ghi.
5. **SELECT**: Chọn các cột để hiển thị.
6. **DISTINCT**: Loại bỏ các bản ghi trùng lặp.
7. **ORDER BY**: Sắp xếp kết quả.
8. **LIMIT/OFFSET**: Giới hạn số lượng bản ghi trả về.

# Các loại JOIN

* **INNER JOIN**: Chỉ trả về các bản ghi có sự phù hợp giữa các bảng.

SELECT \* FROM customers c

INNER JOIN transactions t ON c.CustomerId = t. CustomerId;

* **LEFT JOIN**: Trả về tất cả các bản ghi từ bảng bên trái và các bản ghi phù hợp từ bảng bên phải. Nếu không có sự phù hợp, các cột từ bảng bên phải sẽ có giá trị NULL.

SELECT \* FROM customers c

LEFT JOIN transactions t ON c.CustomerId = t. CustomerId;

* **RIGHT JOIN**: Ngược lại với LEFT JOIN, trả về tất cả các bản ghi từ bảng bên phải.

SELECT \* FROM customers c

RIGHT JOIN transactions t ON c.CustomerId = t. CustomerId;

* **FULL JOIN**: Trả về tất cả các bản ghi từ cả hai bảng, với giá trị NULL cho các bản ghi không phù hợp

SELECT \* FROM customers c

FULL JOIN transactions t ON c.CustomerId = t. CustomerId;

# Subquery

* Subquery (câu truy vấn con) là một truy vấn nằm trong một câu truy vấn khác.
* Ví dụ:

SELECT \* FROM customers *c*

WHERE *c*.CustomerId IN (SELECT CustomerId FROM transactions *t* WHERE *t*.TransactionId = 1);

# CTE (Common Table Expression)

* CTE là một truy vấn tạm thời, có thể được sử dụng trong các câu lệnh SELECT, INSERT, UPDATE, DELETE.
* Ví dụ:

WITH CTE AS (

SELECT CustomerId, COUNT(\*) as count

FROM customers

GROUP BY CustomerId

)

SELECT \*

FROM CTE

WHERE count > 1;

# Ranking: ROW\_NUMBER, RANK, DENSE\_RANK

1. **ROW\_NUMBER**: Gán một số thứ tự duy nhất cho mỗi bản ghi trong nhóm.

SELECT \*, ROW\_NUMBER() OVER (ORDER BY CustomerId) AS row\_num

FROM customers;

1. **RANK**: Gán thứ tự nhưng có thể có các bản ghi có cùng giá trị, và nếu có bản ghi trùng, thứ tự tiếp theo sẽ bị bỏ qua.

SELECT \*, RANK() OVER (ORDER BY CustomerId) AS rank\_num

FROM customers;

1. **DENSE\_RANK**: Tương tự như RANK nhưng không bỏ qua thứ tự.

SELECT \*, DENSE\_RANK() OVER (ORDER BY CustomerId) AS dense\_rank\_num

FROM customers;

# VIEW

VIEW là một bảng ảo được tạo ra từ một câu truy vấn. Có thể thực hiện các thao tác CRUD trên VIEW, nhưng có một số hạn chế:

1. **SELECT**: Thực hiện được bình thường.
2. **INSERT/UPDATE/DELETE**: Có thể thực hiện nhưng phải đảm bảo VIEW đáp ứng một số điều kiện (ví dụ: không có GROUP BY, không có DISTINCT, ...).

CREATE VIEW customers\_view AS SELECT customerId, Name FROM customers;

UPDATE customers\_view SET customerId = 1 WHERE Name = 'OK';

# PROCEDURE

* Mục đích khi tạo ra một PROCEDURE là để thực hiện một chuỗi các lệnh SQL, giúp quản lý logic phức tạp và có thể tái sử dụng.
* Để sử dụng PROCEDURE trong ứng dụng Java, có thể sử dụng JDBC:

CallableStatement stmt = connection.prepareCall("{call procedure\_name(?, ?)}");

stmt.setInt(1, parameter1);

stmt.setString(2, parameter2);

stmt.execute();