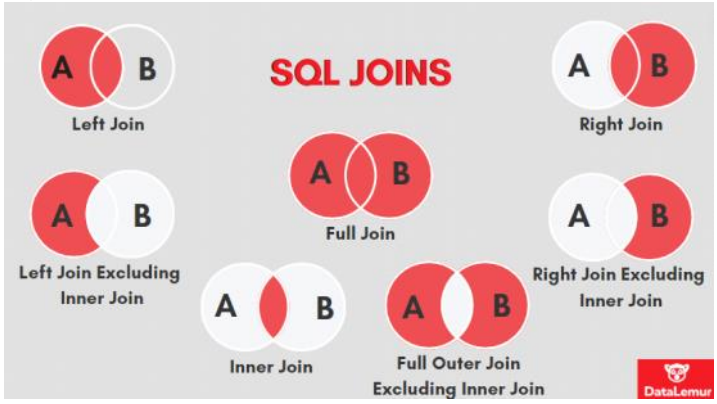
Joins

Saturday, January 4, 2025 9:55 AM

What is Join?

A `JOIN` clause is used to combine rows from two or more tables, based on a related column between them.

Types of join :

- 

Note :

Join :- cross product + condition statement(Select statement)
Common attributes are necessary

1. Natural Join :

- **Common Column(s) with the Same Name**:
  - Both tables must have at least one column with the same name.
  - Example:
    - **Table1**: emp_id, dept_id
    - **Table2**: dept_id, dept_name
    - dept_id is the common column
- **Same Data Type for Common Columns**:
  - The columns with the same name must have compatible data types (e.g., both INT, both VARCHAR, etc.).
  - Mismatched data types will cause an error.
- **Primary Key-Foreign Key Relationship (Recommended)**:
  - Typically, one table's common column is a primary key, and the other table's common column is a foreign key.
  - This ensures that rows in one table match rows in the other.
- **No Extra Columns with Unintended Matches**:
  - Avoid using **natural join** if multiple columns have the same name but unrelated meanings. This can cause incorrect joins.

```
SELECT *
FROM table1
NATURAL JOIN table2
WHERE condition;
```

2. Self Join :

Conditions for a Self Join:

1. **Table Must Have Related Data**:
   - The table must contain data where some rows can logically be related to others.
2. **Use of Aliases**:
   - Aliases are mandatory to differentiate between the two instances of the same table in the query.
3. **Valid Relationship**:
   - A column (or a combination of columns) must exist to define the relationship between rows in the table.
4. **Join Condition**:
   - You need a meaningful condition in the ON clause to specify how rows from one instance of the table relate to

rows in the other instance

```
SELECT a.column_name, b.column_name
FROM table_name a
JOIN table_name b
ON a.common_column = b.common_column;
```

3. Left join :A **LEFT JOIN** (or **LEFT OUTER JOIN**) retrieves all rows from the left table and the matching rows from the right table. If no match is found, NULL values are returned for columns from the right table.

   Conditions for a LEFT JOIN:

   1. **Two Tables with a Logical Relationship**:
      - The two tables should have a relationship that allows meaningful comparisons, typically involving a foreign key-primary key relationship.
   2. **Join Condition**:
      - A valid condition in the ON clause is required to define how rows from the left table relate to rows in the right table.
      - Example: left_table.common_column = right_table.common_column.
   3. **No Restrictions on Matching Rows**:
      - A LEFT JOIN always includes all rows from the left table, regardless of whether a match exists in the right table.
   4. **Nullable Columns from the Right Table**:
      - Columns from the right table may contain NULL values for rows that do not have a matching entry.
   5. **Filtering (Optional)**:
      - You can add a WHERE clause to filter rows further, but be cautious to avoid unintentionally converting the LEFT JOIN into an INNER JOIN (e.g., avoid filtering on NULLs directly).

```
SELECT Employees.emp_id, Employees.emp_name, Departments.dept_name
FROM Employees
LEFT JOIN Departments
ON Employees.dept_id = Departments.dept_id;
```

4] Right Join:A **RIGHT JOIN** (or **RIGHT OUTER JOIN**) retrieves all rows from the **right table** and the matching rows from the **left table**. If no match is found, NULL values are returned for columns from the left table.

Conditions for a RIGHT JOIN:

1. **Two Tables with a Logical Relationship**:
   - The two tables should have a meaningful relationship, often based on a primary key and foreign key.
2. **Join Condition**:
   - A valid condition in the ON clause is required to define how rows from the right table relate to rows in the left table.
   - Example: left_table.common_column = right_table.common_column.
3. **Include All Rows from the Right Table**:
   - All rows from the right table will be included in the result, regardless of whether they have matching rows in the left table.
4. **Nullable Columns from the Left Table**:
   - Columns from the left table may contain NULL values for rows that do not have a matching entry in the right table.
5. **Filtering (Optional)**:
   - A WHERE clause can be used to further filter the results but should not unintentionally filter out NULLs from the left table if they are needed.

```
SELECT Employees.emp_id, Employees.emp_name, Departments.dept_name
FROM Employees
RIGHT JOIN Departments
ON Employees.dept_id = Departments.dept_id;
```