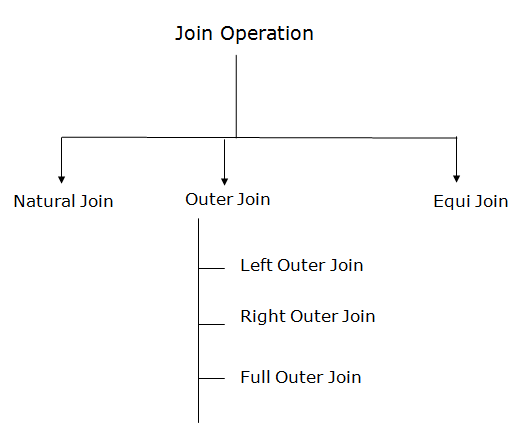
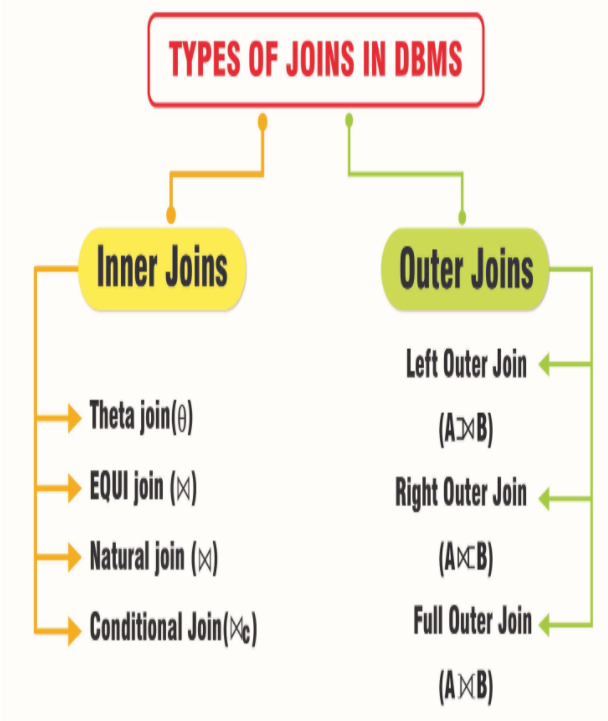
**Join Operations**

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.

**Types of Join operations:**  
   


* **Natural Join**

A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names. It is denoted by ⋈. Natural join does not use any comparison operator. It does not concatenate the way a Cartesian product does. We can perform a Natural Join only if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name and domain.

Natural join acts on those matching attributes where the values of attributes in both the relations are same.

EMPLOYEE

|  |  |
| --- | --- |
| **EMP\_CODE** | **EMP\_NAME** |
| 101 | Stephan |
| 102 | Jack |
| 103 | Harry |

SALARY

|  |  |
| --- | --- |
| **EMP\_CODE** | **SALARY** |
| 101 | 50000 |
| 102 | 30000 |
| 103 | 25000 |

**Example:** Let's use the above EMPLOYEE table and SALARY table:

**Input:** **Get the employee name with their salary name.**

∏EMP\_NAME, SALARY (EMPLOYEE ⋈ SALARY)

**Output:**

|  |  |
| --- | --- |
| **EMP\_NAME** | **SALARY** |
| Stephan | 50000 |
| Jack | 30000 |
| Harry | 25000 |

**Example:**

**Emp**  **Dep**

**(Name Id Dept\_name) (Dept\_name Manager)**

-------------------------------- ------------------------------

A 120 IT Sale Y

B 125 HR Prod Z

C 110 Sale IT A

D 111 IT

Emp ⋈ Dep

**Name Id Dept\_name Manager**

----------------------------------------------------

A 120 IT A

C 110 Sale Y

D 111 IT A

**Example:**

|  |  |
| --- | --- |
| **C** | |
| **Num** | **Square** |
| 2 | 4 |
| 3 | 9 |

|  |  |
| --- | --- |
| **D** | |
| **Num** | **Cube** |
| 2 | 8 |
| 3 | 27 |

C ⋈ D

|  |  |  |
| --- | --- | --- |
| **C ⋈ D** | | |
| **Num** | **Square** | **Cube** |
| 2 | 4 | 8 |
| 3 | 9 | 27 |

|  |  |  |
| --- | --- | --- |
| **Courses** | | |
| **CID** | **Course** | **Dept** |
| CS01 | Database | CS |
| ME01 | Mechanics | ME |
| EE01 | Electronics | EE |

|  |  |
| --- | --- |
| **HoD** | |
| **Dept** | **Head** |
| CS | Alex |
| ME | Maya |
| **EE** | **Mira** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Courses ⋈ HoD** | | | |
| **Dept** | **CID** | **Course** | **Head** |
| CS | CS01 | Database | Alex |
| ME | ME01 | Mechanics | Maya |
| EE | EE01 | Electronics | Mira |

* **Outer Join**

The outer join operation is an extension of the Join operation. It is used to deal with missing information. Theta Join, Equijoin, and Natural Join are called inner joins. An inner join includes only those tuples with matching attributes and the rest are discarded in the resulting relation. Therefore, we need to use outer joins to include all the tuples from the participating relations in the resulting relation.

**Example:**

EMPLOYEE

|  |  |  |
| --- | --- | --- |
| **EMP\_NAME** | **STREET** | **CITY** |
| Ram | Civil line | Mumbai |
| Shyam | Park street | Kolkata |
| Ravi | M.G. Street | Delhi |
| Hari | Nehru nagar | Hyderabad |

FACT\_WORKERS

|  |  |  |
| --- | --- | --- |
| **EMP\_NAME** | **BRANCH** | **SALARY** |
| Ram | Infosys | 10000 |
| Shyam | Wipro | 20000 |
| Kuber | HCL | 30000 |
| Hari | TCS | 50000 |

**Input:**

(EMPLOYEE ⋈ FACT\_WORKERS)

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_NAME** | **STREET** | **CITY** | **BRANCH** | **SALARY** |
| Ram | Civil line | Mumbai | Infosys | 10000 |
| Shyam | Park street | Kolkata | Wipro | 20000 |
| Hari | Nehru nagar | Hyderabad | TCS | 50000 |

An outer join is basically of three types:

1. Left outer join
2. Right outer join
3. Full outer join

* **Left Outer Join**
* Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
* All the tuples from the Left relation, R, are included in the resulting relation.
* It is denoted by ⟕.
* In the left outer join, operation allows keeping all tuple in the left relation. However, if there is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.



**Example:** Using the above EMPLOYEE table and FACT\_WORKERS table

**Input:**

EMPLOYEE ⟕ FACT\_WORKERS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_NAME** | **STREET** | **CITY** | **BRANCH** | **SALARY** |
| Ram | Civil line | Mumbai | Infosys | 10000 |
| Shyam | Park street | Kolkata | Wipro | 20000 |
| Hari | Nehru street | Hyderabad | TCS | 50000 |
| Ravi | M.G. Street | Delhi | NULL | NULL |

**Example:**

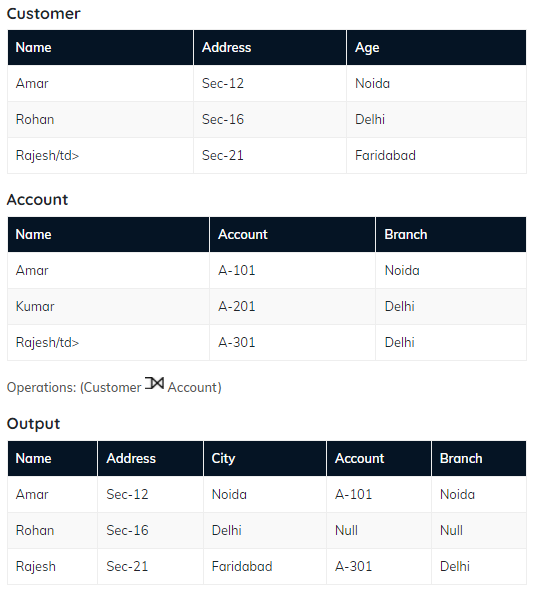
|  |  |
| --- | --- |
| **A** | |
| **Num** | **Square** |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |

|  |  |
| --- | --- |
| **B** | |
| **Num** | **Cube** |
| 2 | 8 |
| 3 | 18 |
| 5 | 75 |

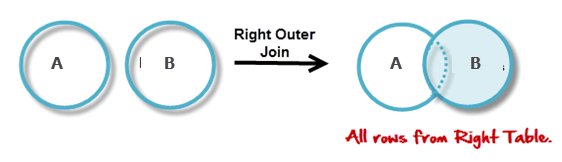
A [https://www.guru99.com/images/1/100518_0535_RelationalA5.png](https://www.guru99.com/images/1/100518_0535_RelationalA5.png) B

|  |  |  |
| --- | --- | --- |
| **A ⋈ B** | | |
| **Num** | **Square** | **Cube** |
| 2 | 4 | 8 |
| 3 | 9 | 18 |
| 4 | 16 | NULL |

**Example:**



* **Right Outer Join**
* Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
* All the tuples from the Right relation, R, are included in the resulting relation.
* It is denoted by ⟖.
* In the right outer join, operation allows keeping all tuple in the right relation. However, if there is no matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values.



**Example:** Using the above EMPLOYEE table and FACT\_WORKERS Relation

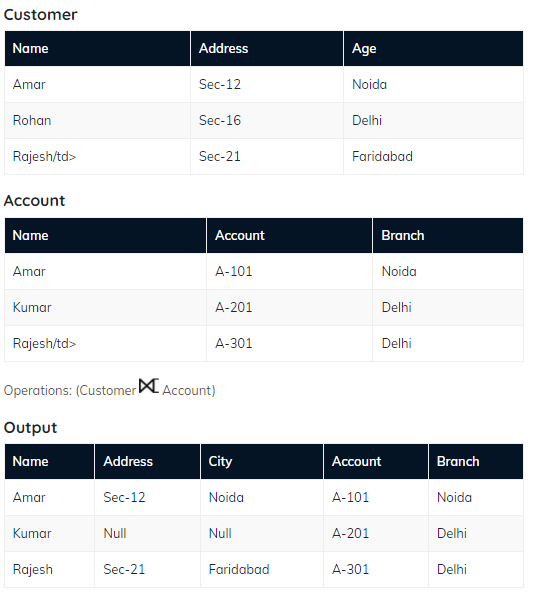
**Input:**

EMPLOYEE ⟖ FACT\_WORKERS

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_NAME** | **BRANCH** | **SALARY** | **STREET** | **CITY** |
| Ram | Infosys | 10000 | Civil line | Mumbai |
| Shyam | Wipro | 20000 | Park street | Kolkata |
| Hari | TCS | 50000 | Nehru street | Hyderabad |
| Kuber | HCL | 30000 | NULL | NULL |
| **Example:** **A ⋈ B** | | |
| **Num** | **Cube** | **Square** |
| 2 | 8 | 4 |
| 3 | 18 | 9 |
| 5 | 75 | - |

**Example:**



* **Full Outer Join**
* Full outer join is like a left or right join except that it contains all rows from both tables.
* All the tuples from both participating relations are included in the resulting relation. If there are no matching tuples for both relations, their respective unmatched attributes are made NULL.
* It is denoted by ⟗.

**Example:** Using the above EMPLOYEE table and FACT\_WORKERS table

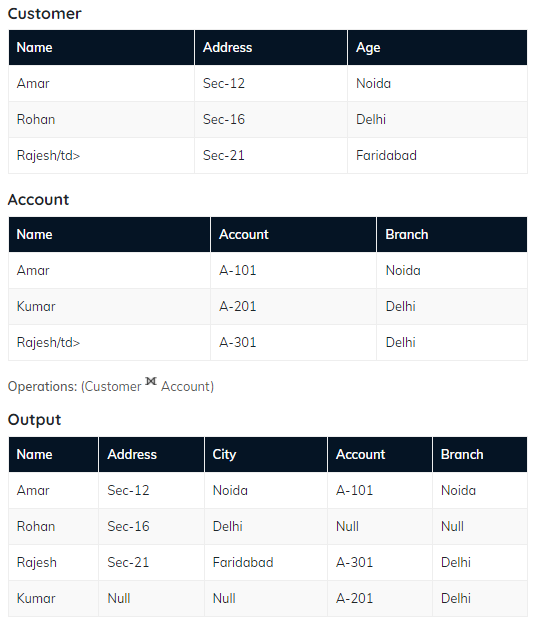
**Input:**

EMPLOYEE ⟗ FACT\_WORKERS

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_NAME** | **STREET** | **CITY** | **BRANCH** | **SALARY** |
| Ram | Civil line | Mumbai | Infosys | 10000 |
| Shyam | Park street | Kolkata | Wipro | 20000 |
| Hari | Nehru street | Hyderabad | TCS | 50000 |
| Ravi | M.G. Street | Delhi | NULL | NULL |
| Kuber | NULL | NULL | HCL | 30000 |
| **Example:** **A ⋈ B** | | |
| **Num** | **Square** | **Cube** |
| 2 | 4 | 8 |
| 3 | 9 | 18 |
| 4 | 16 | - |
| 5 | - | 75 |

**Example:**



* **Inner Join**

In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded. Let's study various types of Inner Joins:

* **Theta Join**

The general case of JOIN operation is called a Theta join. It is denoted by symbol **θ.** Theta join can use any conditions in the selection criteria.

**Example**

A ⋈θ B

R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don’t have anything in common, that is R1 ∩ R2 = Φ.

Theta join can use all kinds of comparison operators.

|  |  |  |
| --- | --- | --- |
| **Student** | | |
| **SID** | **Name** | **Std** |
| 101 | Alex | 10 |
| 102 | Maria | 11 |

|  |  |
| --- | --- |
| **Subjects** | |
| **Class** | **Subject** |
| 10 | Math |
| 10 | English |
| 11 | Music |
| 11 | Sports |

Student\_Detail −

STUDENT ⋈Student.Std = Subject.Class SUBJECT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student\_detail** | | | | |
| **SID** | **Name** | **Std** | **Class** | **Subject** |
| 101 | Alex | 10 | 10 | Math |
| 101 | Alex | 10 | 10 | English |
| 102 | Maria | 11 | 11 | Music |
| 102 | Maria | 11 | 11 | Sports |

* **Equi Join**

It is also known as an inner join. It is the most common join. It is based on matched data as per the equality condition. The equi join uses the comparison operator (=).

**Example:**

CUSTOMER RELATION

|  |  |
| --- | --- |
| **CLASS\_ID** | **NAME** |
| 1 | John |
| 2 | Harry |
| 3 | Jackson |

PRODUCT

|  |  |
| --- | --- |
| **PRODUCT\_ID** | **CITY** |
| 1 | Delhi |
| 2 | Mumbai |
| 3 | Noida |

**Input:**

CUSTOMER ⋈ PRODUCT

**Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| **CLASS\_ID** | **NAME** | **PRODUCT\_ID** | **CITY** |
| 1 | John | 1 | Delhi |
| 2 | Harry | 2 | Mumbai |
| 3 | Harry | 3 | Noida |

When a theta join uses only equivalence condition, it becomes a equi join.

EQUI join is the most difficult operations to implement efficiently using SQL in an RDBMS and one reason why RDBMS have essential performance problems.

**Example:**

