



Course Code: CL-2005	Course: Database Systems Lab
Instructor(s):	Sohail Ahmed Malik, Mr. Mubashir, Mr. Sameer Faisal, Ms. Mehak Mazhar, Ms. Fatima, Ms. Mahnoor Javed, Ms. Filza Akhlaq, Ms. Bushra Sattar, Ms. Syeda Ravia Ejaz, Ms. Yumna Asif

### Contents:

- Introduction to Joins
- Types of Joins
- Introduction to Outer Joins and Its Types
- Introduction to Set Operator
- Types of Set Operator
- Exercise

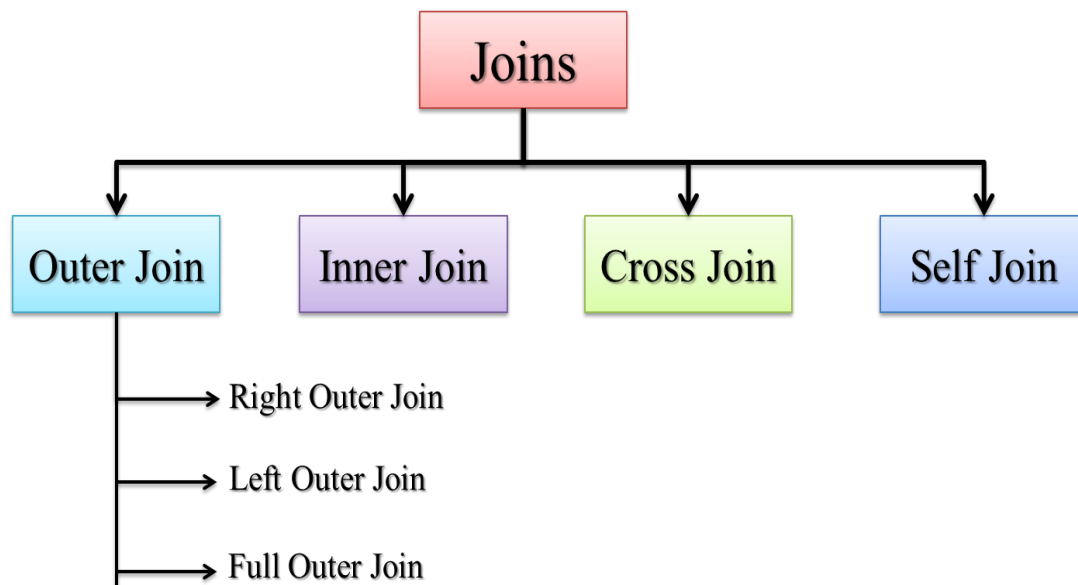
### INTRODUCTION TO JOIN

The JOIN keyword is used in an SQL statement to query data from two or more tables based on a relationship between certain columns in these tables.

### TYPES OF JOINS:

The following are the types of joins. They are:

- Cross Join / Cartesian Join
- Inner Join / Equity Join
- Outer Join
- Left Outer
- Right Outer
- Full Outer
- Self-Join



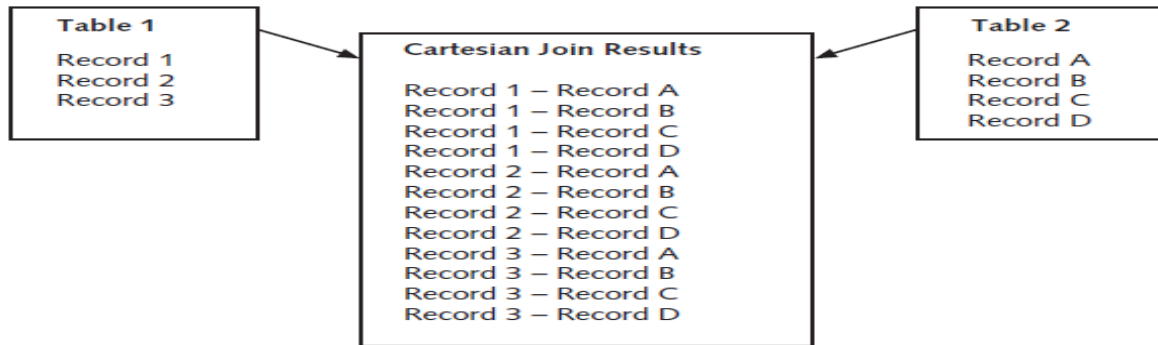
**Cross Join / Cartesian Join:**

In a Cartesian join, also called a Cartesian product or cross join, each record in the first table is matched with each record in the second table.

$$(\# \text{ rows in Table 1}) * (\# \text{ rows in Table 2})$$

**Syntax for Cross Join/Cartesian Join:**

```
SELECT * FROM TABLE1, TABLE2;
```

**ISO Standard:**

```
SELECT * FROM TABLE1 CROSS JOIN TABLE2;
```

**Inner Join / Equality Joins:**

If the join contains an equality condition, it is also called Equi Join, Natural Join, Inner Join.

**Syntax For Inner Join:**

```
SELECT TABLE1_COLUMN, TABLE2_COLUMN
FROM TABLE1 T1, TABLE2 T2
WHERE T1.TABLE1_COLUMN = T2.TABLE2_COLUMN.
```

**Example**

To retrieve the employee name, their job and department name, we need to extract data from two tables, EMP and DEPT:

```
SELECT E.ENAME, E.JOB, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO = D.DEPTNO;
```

**The SQL-1999 standard:**

```
SELECT ENAME, JOB, DNAME FROM EMP NATURAL JOIN DEPT;
```

**Using Clause:**

No matter how many common columns are available in the tables, NATURALJOIN will join with all the common columns. Use USING clause to join with specified columns.

**Syntax for Using Clause:**

```
SELECT TABLE1_COLUMN, TABLE2_COLUMN FROM TABLE1 JOIN TABLE2 USING(TABLE2_COLUMN,
TABLE1_COLUMN)
```

**Example**

```
SELECT EMPNO, ENAME, MGR, DNAME FROM EMP JOIN DEPT USING (DEPTNO,MGR);
```

**Self-Join:**

When a table is joined to itself then it is called as Self join or in less words we can just say “joining a table to itself is called self-join”.

**Syntax for Self-join:**

```
SELECT T1.TABLE1_COLUMN,T2. TABLE2_COLUMN FROM TABLE T1,TABLE T2 WHERE T1.COLUMN =
T2.COLUMN;
```

Example

```
SELECT WORKER.ENAME, MANAGER.ENAME FROM EMP_WORKER, EMP_MANAGER WHERE
WORKER.MGR = MANAGER.EMPNO;
```

**INTRODUCTION TO OUTER JOIN & ITS TYPES**

Use Outer join to return records which don't have direct match.

In outer join operation, all records from the source table are included in the result even though they don't satisfy the join condition.

**Syntax for Outer Join:**

**SELECT** column names from both tables **FROM** table name 1 **LEFT|RIGHT|FULL OUTER JOIN** table name 2 **ON** condition.

**Types of Outer Joins:**

Outer joins are classified into three types:

1. Left Outer Join
2. Right Outer Join
3. Full Outer Join

**Left Outer Join:**

The left outer join produces a table that contains the matched data from the two tables, as well as the remaining rows of the left table and null from the columns of the right table.

**Syntax for Left Outer Join:**

```
SELECT T1.TABLE1_COLUMN, T2.TABLE2_COLUMN FROM TABLE1 T1, TABLE2 T2
WHERE T1.TABLE1_COLUMN = T2. TABLE2_COLUMN(+);
```

Example

```
SELECT E.ENAME, D.DEPTNO, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO = D.DEPTNO (+);
```

**NOTE:** The outer join operator appears on only that side that has information missing.

**The SQL-1999 standard:**

```
SELECT T1.TABLE1_COLUMN, T2.TABLE2_COLUMN FROM TABLE1 T1 LEFT OUTER JOIN TABLE2 T2 ON
T1.TABLE1_COLUMN = T2. TABLE2_COLUMN;
```

Example

```
SELECT E.ENAME, D.DEPTNO, D.DNAME FROM EMP E LEFT OUTER JOIN DEPT D ON (E.DEPTNO = D.DEPTNO);
```

**Right Outer Join:**

The right outer join returns a table with the matched data from the two tables being joined, then the remaining rows of the right table and null for the remaining left table's columns.

**Syntax for Right Outer Join:**

```
SELECT T1.TABLE1_COLUMN, T2.TABLE2_COLUMN FROM TABLE1 T1, TABLE2 T2
WHERE T1.TABLE1_COLUMN (+)= T2. TABLE2_COLUMN;
```

Example:

```
SELECT E.ENAME, D.DEPTNO, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO(+) = D.DEPTNO;
```

**SQL-1999 standard:**

```
SELECT T1.TABLE1_COLUMN, T2.TABLE2_COLUMN FROM TABLE1 T1 RIGHT OUTER JOIN TABLE2 T2 ON
T1.TABLE1_COLUMN = T2. TABLE2_COLUMN;
```

Example:

```
SELECT E.ENAME, D.DEPTNO, D.DNAME FROM EMP E RIGHT OUTER JOIN DEPT D ON (E.DEPTNO =
D.DEPTNO);
```

**Full Outer Join:**

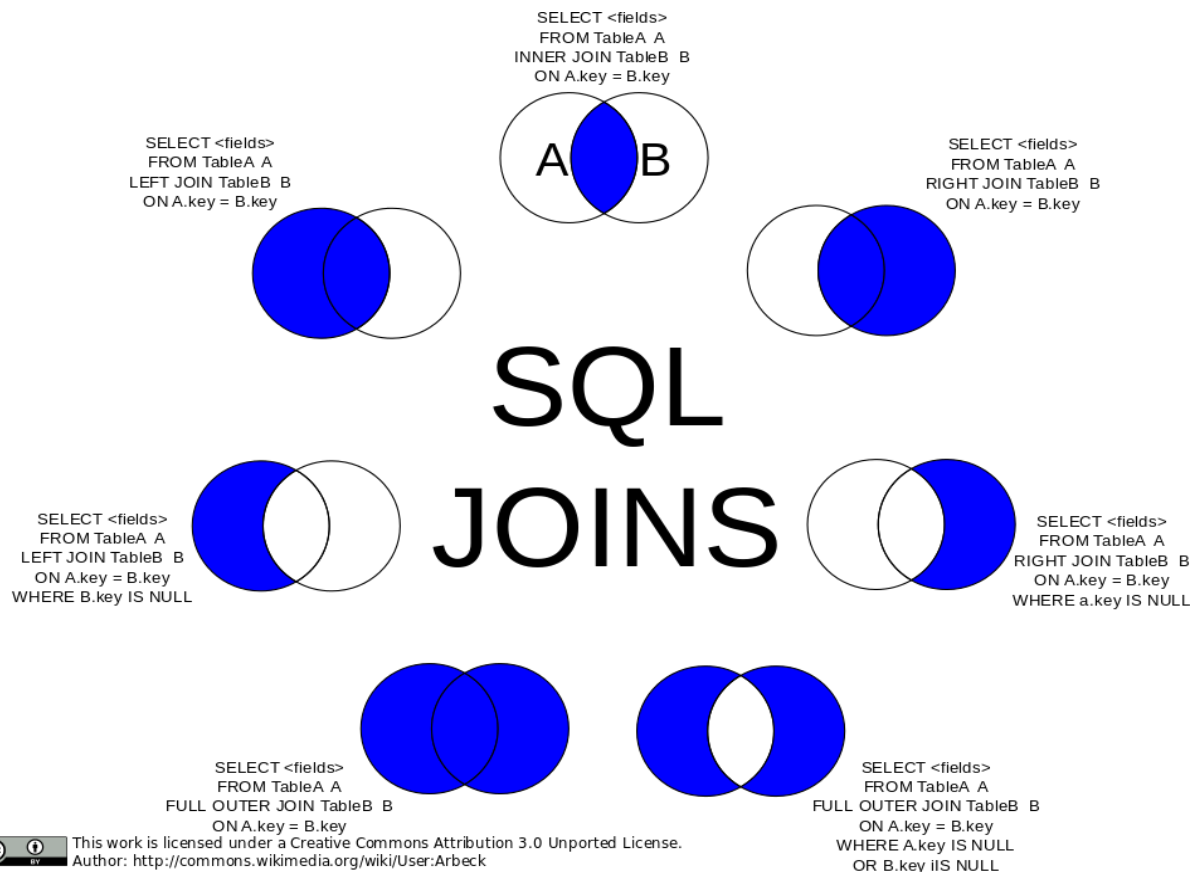
The full outer join returns a table with the matched data of two table then remaining rows of both left table and then the right table.

**Syntax for Full Outer Join:**

```
SELECT T1. TABLE1_COLUMN, T2. TABLE2_COLUMN FROM TABLE1 T1 FULL OUTER JOIN TABLE2 T2 ON
T1.TABLE1_COLUMN = T2. TABLE2_COLUMN;
```

**Example**

```
SELECT E.ENAME, D.DEPTNO, D.DNAME FROM EMP E FULL OUTER JOIN DEPT DON (E.DEPTNO = D.DEPTNO);
```

**Conclusion for Joins:****INTRODUCTION TO SET OPERATOR**

Set operators are used to join the results of two (or more) SELECT statements. The SET operators available in Oracle 11g are UNION, UNION ALL, INTERSECT and MINUS.

All of the SET operators have the same order of precedence. Instead, Oracle evaluates queries from left to right or top to bottom during execution. If parentheses are used explicitly, the order may change because parentheses take precedence over dangling operators.

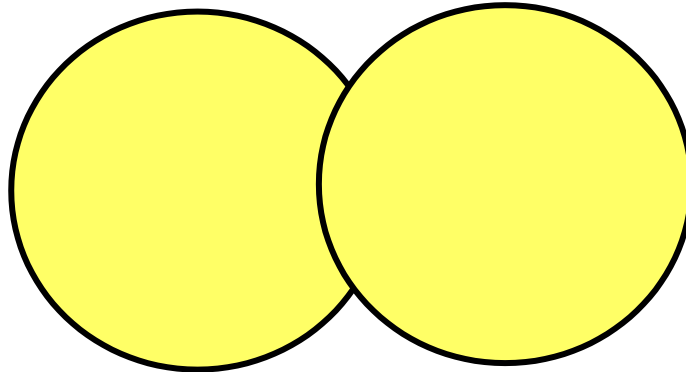
**TYPES OF SET OPERATOR**

The following are the types of operators that are used for set in oracle. They are:

1. Union
2. Union all
3. Intersect
4. Minus

**Union Operator:**

The SQL Union function joins the results of two or more SQL SELECT queries. The number of datatypes and columns in both tables on which the UNION operation is performed must be the same in order to perform the union operation. The duplicate rows are removed from the result of the union operation.



Diagrammatic view of Union operator

**Syntax for Union Operator**

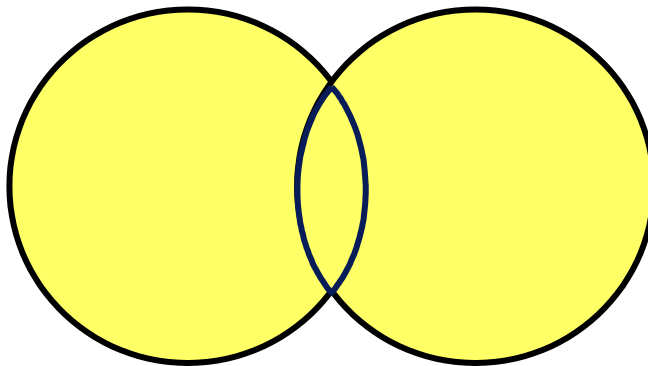
```
SELECT TABLE1_COLUMN, TABLE2_COLUMN FROM TABLE1 UNION SELECT TABLE1_COLUMN_ID,  
TABLE2_COLUMN_ID FROM TABLE2
```

**Example**

```
SELECT employee_id, job_id FROM employees UNION SELECT employee_id, job_id FROM job_history;
```

**Union All Operator:**

With one exception, UNION and UNION ALL operate in a similar manner. UNION ALL, on the other hand, returns the result set without eliminating duplication or sorting the data.



Diagrammatic view of Union all operator

**Syntax for Union All Operator**

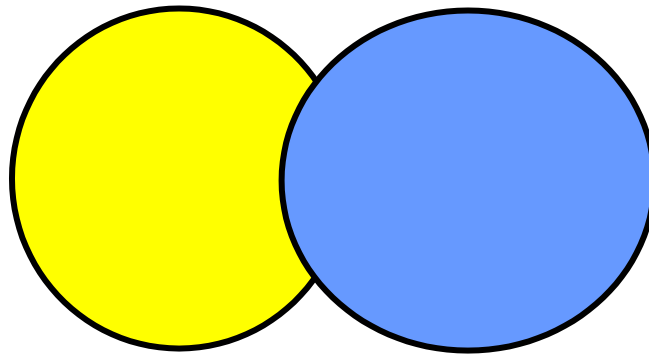
```
SELECT TABLE1_COLUMN, TABLE2_COLUMN FROM TABLE1 UNION ALL SELECT TABLE1_COLUMN_ID,  
TABLE2_COLUMN_ID FROM TABLE2
```

**Example**

```
SELECT employee_id, job_id, department_id FROM employees UNION ALL  
SELECT employee_id, job_id, department_id FROM job_history
```

**Intersect Operator**

It's used to join two SELECT statements together. The common rows from both SELECT statements are returned by the Intersect procedure. The number of datatypes and columns in the Intersect operation must be the same. There are no duplicates, and the data is arranged in ascending order by default.



Diagrammatic view of Intersect operator

Syntax for Intersect Operator

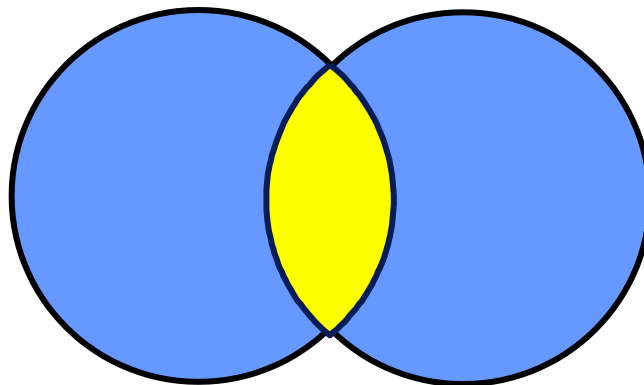
```
SELECT TABLE1_COLUMN, TABLE2_COLUMN FROM TABLE1 INTERSECT SELECT TABLE1_COLUMN_ID,  
TABLE2_COLUMN_ID FROM TABLE2
```

Example

```
SELECT employee_id, job_id FROM employees INTERSECT SELECT employee_id, job_id FROM job_history;
```

Minus Operator

It combines the results of two SELECT statements into a single statement. The minus operator is used to show rows that are present in the first query but not in the second. There are no duplicates, and the data is sorted ascending by default.



Diagrammatic view of Minus operator

Syntax for Minus Operator

```
SELECT TABLE1_COLUMN, TABLE2_COLUMN FROM TABLE1 MINUS SELECT TABLE1_COLUMN_ID,  
TABLE2_COLUMN_ID FROM TABLE2
```

Example

```
SELECT employee_id, job_id FROM employees MINUS SELECT employee_id, job_id FROM job_history
```

**How to implement joins as set operator?**