SQL Joins

Use the SQL JOIN whenever multiple tables must be accessed through an SQL SELECT statement and no results should be returned if there is not a match between the JOINed tables.

Use an INNER JOIN when you want only records that are related in both tables.  
Use a LEFT JOIN when you want all records in the left table.  
Use a RIGHT JOIN when you want all records in the right table. Many never use RIGHT JOIN. They write thousands of queries with just INNER or LEFT.

JOIN| INNER JOIN

RIGHT JOIN | RIGHT OUTER JOIN

LEFT JOIN | LEFT OUTER JOIN

FULL JOIN | FULL OUTER JOIN

SELF JOIN (NOT A SYNTAX)

SELECT *column\_name(s)*  
FROM *table1 T1, table1 T2*  
WHERE *condition*;

**Table 1** − CUSTOMERS table is as follows.

+----+----------+-----+

| ID | NAME | AGE |

+----+----------+-----+

| 1 | Ramesh | 32 |

| 2 | Khilan | 25 |

| 3 | kaushik | 23 |

| 4 | Chaitali | 25 |

| 5 | Hardik | 27 |

| 6 | Komal | 22 |

+----+----------+-----+

SELECT *\**  
FROM CUSTOMERS *T1,* CUSTOMERS *T2*  
WHERE *T1.ID=T2.ID*;

+----+----------+-----+----+----------+-----+

| ID | NAME | AGE | ID | NAME | AGE |

+----+----------+-----+----+----------+-----+

| 1 | Ramesh | 32 | 1 | Ramesh | 32 |

| 2 | Khilan | 25 | 2 | Khilan | 25 |

| 3 | kaushik | 23 | 3 | kaushik | 23 |

| 4 | Chaitali | 25 | 4 | Chaitali | 25 |

| 5 | Hardik | 27 | 5 | Hardik | 27 |

| 6 | Komal | 22 | 6 | Komal | 22 |

+----+----------+-----+----+----------+-----+

CARTESIAN JOIN OR CROSS JOIN (NOT A SYNTAX)

**Table 1** − CUSTOMERS table is as follows.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

+----+----------+-----+-----------+----------+

Table 2: ORDERS Table is as follows −

+-----+---------------------+-------------+--------+

|OID | DATE | CUSTOMER\_ID | AMOUNT |

+-----+---------------------+-------------+--------+

| 102 | 2009-10-08 00:00:00 | 3 | 3000 |

| 100 | 2009-10-08 00:00:00 | 3 | 1500 |

| 101 | 2009-11-20 00:00:00 | 2 | 1560 |

| 103 | 2008-05-20 00:00:00 | 4 | 2060 |

+-----+---------------------+-------------+--------+

Now, let us join these two tables using CARTESIAN JOIN as follows −

SQL> SELECT ID, NAME, AMOUNT, DATE

FROM CUSTOMERS, ORDERS;

This would produce the following result −

+----+----------+--------+---------------------+

| ID | NAME | AMOUNT | DATE |

+----+----------+--------+---------------------+

| 1 | Ramesh | 3000 | 2009-10-08 00:00:00 |

| 1 | Ramesh | 1500 | 2009-10-08 00:00:00 |

| 1 | Ramesh | 1560 | 2009-11-20 00:00:00 |

| 1 | Ramesh | 2060 | 2008-05-20 00:00:00 |

| 2 | Khilan | 3000 | 2009-10-08 00:00:00 |

| 2 | Khilan | 1500 | 2009-10-08 00:00:00 |

| 2 | Khilan | 1560 | 2009-11-20 00:00:00 |

| 2 | Khilan | 2060 | 2008-05-20 00:00:00 |

| 3 | kaushik | 3000 | 2009-10-08 00:00:00 |

| 3 | kaushik | 1500 | 2009-10-08 00:00:00 |

| 3 | kaushik | 1560 | 2009-11-20 00:00:00 |

| 3 | kaushik | 2060 | 2008-05-20 00:00:00 |

| 4 | Chaitali | 3000 | 2009-10-08 00:00:00 |

| 4 | Chaitali | 1500 | 2009-10-08 00:00:00 |

| 4 | Chaitali | 1560 | 2009-11-20 00:00:00 |

| 4 | Chaitali | 2060 | 2008-05-20 00:00:00 |

+----+----------+--------+---------------------+

For INNER joins the order doesn't matter

For (LEFT, RIGHT or FULL) OUTER joins,the order matter. Outer Joins are not commutative ([a+b=b+a ie]. **a LEFT JOIN b** is not the same as**b LEFT JOIN a)** Outer joins are not associative either.

In these examples both (commutativity and associativity) properties are involved:

*a LEFT JOIN b*

*ON b.ab\_id = a.ab\_id*

*LEFT JOIN c*

*ON c.ac\_id = a.ac\_id*

The above query is equivalent to:

*a LEFT JOIN c*

*ON c.ac\_id = a.ac\_id*

*LEFT JOIN b*

*ON b.ab\_id = a.ab\_id*

but the below query is:

*a LEFT JOIN b*

*ON  b.ab\_id = a.ab\_id*

*LEFT JOIN c*

*ON  c.ac\_id = a.ac\_id*

*AND c.bc\_id = b.bc\_id*

Not equivalent to this query:

*a LEFT JOIN c*

*ON  c.ac\_id = a.ac\_id*

*LEFT JOIN b*

*ON  b.ab\_id = a.ab\_id*

*AND b.bc\_id = c.bc\_id*

Another associativity example is as follows. Assume this as (a LEFT JOIN b) LEFT JOIN c:

*a LEFT JOIN b*

*ON b.ab\_id = a.ab\_id          -- AB condition*

*LEFT JOIN c*

*ON c.bc\_id = b.bc\_id          -- BC condition*

Which is equivalent to a LEFT JOIN (b LEFT JOIN c):

*a LEFT JOIN*

*b LEFT JOIN c*

*ON c.bc\_id = b.bc\_id    -- BC condition*

*ON b.ab\_id = a.ab\_id        -- AB condition*



Interesting thought.

To use ***LEFT JOIN or NOT IN when writing a query?***

The answer is: ***It depends!***It all depends on what kind of data is and what kind query it is etc. In that case just for fun guess one option LEFT JOIN or NOT IN. If you need to refer the query which demonstrates the mentioned clauses, review following two queries for Join Better Performance.

Query 1:

SELECT ProductID

FROM Production.Product

WHERE ProductID

NOT IN (

SELECT ProductID

FROM Production.WorkOrder);

Query 2:

SELECT p.ProductID

FROM Production.Product p

LEFT JOIN Production.WorkOrder w ON p.ProductID = w.ProductID

WHERE w.ProductID IS NULL;