**Join**

A SQL join clause combines records from two or more tables in a relational database. It creates a set that can be saved as a table or used as it is. A JOIN is a means for combining fields from two tables (or more) by using values common to each.

**Join** is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

SQL Join is used for combining column from two or more tables by using values common to both tables.

**Theta (θ) Join**

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol **θ**.

**Notation**

**R1 ⋈θ R2**

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.

## Natural Join (⋈)

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.

There are different types of joins available in SQL:

[INNER JOIN:](http://www.tutorialspoint.com/sql/sql-inner-joins.htm) returns rows when there is a match in both tables.

[LEFT JOIN:](http://www.tutorialspoint.com/sql/sql-left-joins.htm) returns all rows from the left table, even if there are no matches in the right table.

[RIGHT JOIN:](http://www.tutorialspoint.com/sql/sql-right-joins.htm) returns all rows from the right table, even if there are no matches in the left table.

[FULL JOIN:](http://www.tutorialspoint.com/sql/sql-full-joins.htm) returns rows when there is a match in one of the tables.

[SELF JOIN:](http://www.tutorialspoint.com/sql/sql-self-joins.htm) is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

[CARTESIAN JOIN:](http://www.tutorialspoint.com/sql/sql-cartesian-joins.htm) returns the Cartesian product of the sets of records from the two or more joined tables.

**Rename Operation (ρ)**

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.

**Notation** − *ρ* x (E)

Where the result of expression **E** is saved with name of **x**.

Additional operations are −

* Set intersection
* Assignment
* Natural join

## Assignment Operation

The assignment operator is one of the most intuitive to use. It assigns a value to a variable. The only confusion in using this operator could stem from its overloading. All RDBMS overload this operator with an additional function — comparison — in the SQL.

The equals operator (=) is used as an assignment in the following SQL query that updates the price (PROD\_PRICE\_N) column in the PRODUCT table, raising the existing prices by 2 percent:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 (10 row(s) affected)

And the same operator would be used for comparing values when used, for example, in theWHERE clause of an SQL statement:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 WHERE prod\_id\_n = 1880 (1 row(s)

affected)

This statement assigns a 2 percent increase to a product whose ID is 1880; in the same query, the equals operator (=) is used in its assignment and comparison capacity at the same time.

**The Intersect Operation**

The **intersect** operation acts as intersection of a set: it outputs a relation with thew tuples common to both the relation. It automatically eliminates duplicates. For example, if it were the case that 4 sections of ECE-101 were taught in the Fall 2009 semester and 2 sections of ECE-101 were taught in the Spring 2010 semester, then there would be only 1 tuple with ECE-101 in the result.

If we want to retain all duplicates, we must write intersect all in place of intersect.

Example: To find the set of all courses taught in the Fall 2009 as well as in Spring 2010 we write:

(**select** *course id*

**from** *section*

**where** *semester* = ’Fall’ **and** *year*= 2009)

**intersect**

(**select** *course id*

**from** *section*

**where** *semester* = ’Spring’ **and** *year*= 2010);

**Division operation**

The division operator of relational algebra, “÷”, is defined as follows. Let *r* (*R*) and *s*(*S*) be relations, and let *S* ⊆ *R*; that is, every attribute of schema *S* is also in schema *R*. Then *r* ÷ *s* is a relation on schema *R* − *S* (that is, on the schema containing all attributes of schema *R* that are not in schema *S*). A tuple *t* is in *r* ÷ *s* if and only if both of two conditions hold:

* *t* is in *\_R*−*S*(*r* )
* For every tuple *ts* in *s*, there is a tuple *tr* in *r* satisfying both of the following:

1. *tr* [*S*] = *ts* [*S*]
2. *tr* [*R* − *S*] = *t*

**Natural Join Operation**

The **natural join** operation operates on two relations and produces a relation as the result. Unlike the Cartesian product of two relations, which concatenates each tuple of the first relation nwith every tuple of the second, natural join considers only those pairs of tuples with the same value on those attributes that appear in the schemas of both relations.