Join

1. Theta join

A **theta join** allows for arbitrary comparison relationships (such as ≥). An equation is a **theta join** using the equality operator.

1. Natural join

A natural **join** is an equijoin on attributes that have the same name in each relationship

* Right join
* Left join
* Inner join

1. Rename operation
2. Assignment operation
3. Division operation
4. Additional operations
5. Set-intersection operation
6. Natural join operation

**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 Operator

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.

**DIVISION Operator**

The division operation will return a Relation R(X) that includes all tuples t[X] in R(Z) that appear in R1 in combination with every tuple from R2(Y), where Z = X ∪ Y. The division operator is symbolized by:

* R1(Z) ∻ R2(Y)

The division operator is the most difficult to implement in SQL as no SQL command is given for division operation. The division operator would be seen as the opposite of the Cartesian product operator; just as in standard math, the relation between division and multiplication. Therefore a series of current SQL commands have to be utilized in implementation of the division operator. An example of the SQL implementation of division operator:

SELECT surname, forenames

FROM employee X

WHERE NOT EXISTS

(SELECT ‘X’

FROM employee y

WHERE NOT EXISTS

(SELECT ‘X’

FROM employee z

WHERE x.empno = z.empno

AND y.surname = z.surname))

ORDER BY empno

**NATURAL JOIN Operator**

The NATURAL JOIN operation returns results that does not include the JOIN attributes of the second Relation B. It is not required that attributes with the same name be mentioned. The NATURAL JOIN operator is symbolized by:

* A \* ⟗<join condition>B, OR A \* ⟗(<join attributes 1>),

(<join attributes 2>)B

OR A \* B

SQL translation example where attribute dob is Date of Birth and empno is Employee Number:

SELECT A.dob, B.empno

FROM A

NATURAL JOIN B

//where depno =5

We can always use the ‘where’ clause to further restrict our output and stop a Cartesian product output.

Set Intersection Operation

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same. MySQL does not support INTERSECT operator.