

## **Operations on Table in RDMS:**

The relational database model is based on the Relational Algebra. Which means that operations in the relational database model are based on Select, Project, Join, Intersection, Union, Difference, and Product. These are main operation in RDBMS on a table:

- Selection: Shows values for all rows found a table that meet a given criteria.
- Project: Shows values for all selected attributes.
- Join: Will combine information from one or more tables.
- Intersection: Shows all rows that are found in both tables.
- Union: Combines all rows from multiple tables and removes the duplicate rows.
- Difference: Shows all rows from one table that are not contained in another table.
- Product: Combines all rows from two or more tables (does contain duplicates).

### **1) Selection:**

To identify a set of tuples which is a part of a relation and to extract only these tuples out. The select operation selects tuples that satisfy a given predicate or condition.

- It is a unary operation defined on a single relation.
- It is denoted as  $\sigma$  (In SQL it's SELECT).

Necessary Condition for Selection:

- Needs a single table as its operand
- Yields values for all rows found in the table
- Can be used to list either all row values or it can yield only those row values that match a specified criterion
- Yields a horizontal subset of a table

Syntax for Selection operation:

**$\sigma$  <condition> Table\_name**

In SQL

**SELECT \*FROM Table\_name**

Example:

In Relational Algebra simply

**$\sigma$  Yr-pub=1992(Book)**

In SQL the same query goes link

**SELECT \*FROM Book WHERE Yr-pub=1992**

## 2) Projection:

In projection operation, returns its argument relation with certain attributes left out.

- It is a unary operation defined on a single relation
- It is denoted as  $\Pi$ .

Necessary Condition for Projection:

- Uses a single table as its operand
- Yields all values for selected attributes
- Yields a vertical subset of a table

Syntax

In Relation Algebra:

$\Pi$  <column\_names> Table\_name

In SQL

**SELECT** <column\_name1>,<column\_name2>,...,<column\_nameN> **FROM**  
**Table\_name**

Example:

In Relational Algebra simply

$\Pi$  book-name,Author (**Book**)

In SQL the same query goes like

SELECT book-name, Author FROM Book

## 3) JOIN :

It is a binary operation and a combination of certain selections and a Cartesian product into one operation.

- It is denoted as  $\bowtie$ .
- It is associative.

It forms a Cartesian product of its two arguments. Then performs a selection forcing equality on those attributes those appear in both the relations. And finally removes duplicates attributes.

$r(R)$ : r is a relation with attributes R.

$s(S)$ : s is a relation with attributes S.

If  $R \cap S = \Phi$ , they have no attributes in common then  $\mathbf{r \bowtie s = r \times s}$

Necessary Condition Join:

- Allows us to combine information from two tables

- Uses two table having a common attribute as its operands
- Join operation is considered as the real power behind the relational database implementations (RDBMS)
- JOIN allows the use of independent tables, linked by common attributes, resulting in minimal redundancy possible.
- Possible types of JOIN includes: NATURAL JOIN, EQUIJOIN, THETA JOIN, LEFT OUTER JOIN, and RIGHT OUTER JOIN

#### **a) NATURAL JOIN:**

- The operands must have common attribute(s) in order to yield the join outcome. Such common attributes are called join column(s)
- Links two tables by selecting only rows with common values in their join column(s)
- Works in a three-step process:

Step 1:

PRODUCT of the tables is created

Step 2.

SELECT is performed on Step 1 output to yield only the rows for which the join column values are equal.

Step 3.

PROJECT is performed on Step 2 results to yield a single copy of each attribute, thereby eliminating duplicate columns

Syntax:

In Relational Algebra

**Table\_name1** ⋈<sub>(JOIN CONDITION)</sub> **Table\_name2**

In SQL

**SELECT \* FROM Table\_name1 NATURAL JOIN Table\_name2 WHERE <Condition>**

#### **b) EQUI JOIN:**

- Links tables on the basis of an equality condition that compares specified columns (join columns) of each table
- Outcome does not eliminate duplicate columns. In other words, the result will show the join column more than once.
- In equijoin, the condition used to join the tables is equal to (=), thus the name equijoin.

Syntax:

In relational algebra

$$R \bowtie S = \sigma(R \times S)$$

In SQL

**SELECT column\_list FROM table1, table2 WHERE table1.column\_name = table2.column\_name**

### c) THETA JOIN:

While joining the tables using join condition having any other comparison operator other than equal to (=), the join is called theta join.

Syntax:

In relational algebra

$$R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$$

In SQL

**SELECT column\_list FROM table1, table2 WHERE table1.column\_name = table2.column\_name AND table1.column\_name = table2.column\_name**

### d) LEFT/RIGHT OUTER JOIN:

- In normal join operations, the outcome shows only the rows of the tables having matching values in the join columns while the rows having unmatched values are discarded.
- In case, if unmatched rows are required to be shown in the outcome (some columns in the result will be having NULL values), the resultant join is called an Outer Join.
- While taking an outer join between two tables A and B, there could be two possible outcomes.
  - If the result shows all the matching rows and the unmatched rows of A with the NULL values in the columns of B, the join is called Left outer join (allows the unmatched rows of left table to come).
  - If the result shows all the matching rows and the unmatched rows of B with the NULL values in the columns of A, the join is called Right outer join (allows the unmatched rows of right table to come).

Syntax:

In relational algebra

$$R \Join_{\theta} S$$

and

$$R \Join_{\theta} S$$

In SQL

**SELECT   *column\_name(s)*   FROM   *table1*   LEFT   JOIN   *table2*   ON**  
***table1.column\_name=table2.column\_name***  
**and**

**SELECT   *column\_name(s)*   FROM   *table1*   RIGHT   JOIN   *table2*   ON**  
***table1.column\_name=table2.column\_name***