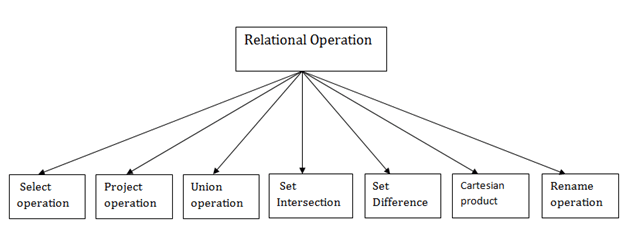
Relational Algebra is procedural query language, which takes Relation as input and generate relation as output. Relational algebra mainly provides theoretical foundation for relational databases and SQL.



### Select Operation:

It is denoted by sigma (σ).

Notation: σp(r)

Where:

σ is used for selection prediction  
r is used for relation  
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

Load table

|  |  |  |
| --- | --- | --- |
| BRANCH\_NAME | LOAN\_NO | AMOUNT |
| Perryride | L-15 | 1500 |
| Redwood | L-23 | 2000 |
| Perryride | L-15 | 1500 |
| Downtown | L-14 | 1500 |

σ BRANCH\_NAME="perryride" (LOAN)

Output:

|  |  |  |
| --- | --- | --- |
| BRANCH\_NAME | LOAN\_NO | AMOUNT |
| Perryride | L-15 | 1500 |
| Perryride | L-16 | 1300 |

σage > 17 (Student)

σage > 17 and gender = 'Male' (Student)

## **Project Operation (∏)**

used to project only a certain set of attributes of a relation.

∏A1, A2...(r)

Table : CUSTOMER

|  |  |  |
| --- | --- | --- |
| NAME | STREET | CITY |
| Jones | Main | Harrison |
| Smith | North | Rye |
| Hays | Main | Harrison |
| Curry | North | Rye |

∏ NAME, CITY (CUSTOMER)

Output:

|  |  |
| --- | --- |
| NAME | CITY |
| Jones | Harrison |
| Smith | Rye |
| Hays | Harrison |
| Curry | Rye |

∏Name, Age(Student)

Note : remove duplicate data from the columns.

## **Union Operation (∪)**

Notation:R ∪ S

A union operation must hold the following condition:

* R and S must have the attribute of the same number.
* Duplicate tuples are eliminated automatically.

DEPOSITOR RELATION

|  |  |
| --- | --- |
| CUSTOMER\_NAME | ACCOUNT\_NO |
| Johnson | A-101 |
| Smith | A-121 |
| Mayes | A-321 |
| Turner | A-176 |
| Johnson | A-273 |
| Jones | A-472 |
| Lindsay | A-284 |

BORROW RELATION

|  |  |
| --- | --- |
| CUSTOMER\_NAME | LOAN\_NO |
| Jones | L-17 |
| Smith | L-23 |
| Hayes | L-15 |
| Jackson | L-14 |
| Curry | L-93 |
| Smith | L-11 |
| Williams | L-17 |

Input:

1. ∏CUSTOMER\_NAME (BORROW) ∪ ∏ CUSTOMER\_NAME (DEPOSITOR)

|  |
| --- |
| CUSTOMER\_NAME |
| Johnson |
| Smith |
| Hayes |
| Turner |
| Jones |
| Lindsay |
| Jackson |
| Curry |
| Williams |
| Mayes |

example :   
∏Student(RegularClass) ∪ ∏Student(ExtraClass)

### **Set Intersection:**

1. Notation: R ∩ S

The set intersection operation contains all tuples that are in both R & S.

∏ CUSTOMER\_NAME (BORROW) ∩ ∏ CUSTOMER\_NAME (DEPOSITOR)

## **Set Difference (-)**

The set Difference operation contains all tuples that are in R but not in S.

Notation: R – S

∏ CUSTOMER\_NAME (BORROW) - ∏ CUSTOMER\_NAME (DEPOSITOR)

### Cartesian product

The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.

It is denoted by X.

Notation: E X D

EMPLOYEE

|  |  |  |
| --- | --- | --- |
| EMP\_ID | EMP\_NAME | EMP\_DEPT |
| 1 | Smith | A |
| 2 | Harry | C |
| 3 | John | B |

DEPARTMENT

|  |  |
| --- | --- |
| DEPT\_NO | DEPT\_NAME |
| A | Marketing |
| B | Sales |
| C | Legal |

Input: EMPLOYEE X DEPARTMENT

Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMP\_ID | EMP\_NAME | EMP\_DEPT | DEPT\_NO | DEPT\_NAME |
| 1 | Smith | A | A | Marketing |
| 1 | Smith | A | B | Sales |
| 1 | Smith | A | C | Legal |
| 2 | Harry | C | A | Marketing |
| 2 | Harry | C | B | Sales |
| 2 | Harry | C | C | Legal |
| 3 | John | B | A | Marketing |
| 3 | John | B | B | Sales |
| 3 | John | B | C | Legal |

### Rename Operation:

The rename operation is used to rename the output relation. It is denoted by rho(ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

ρ(STUDENT1, STUDENT)