

# DBMS - IV

---



A handwritten signature in black ink, appearing to read "Sath". It is positioned above a horizontal line.

---

---

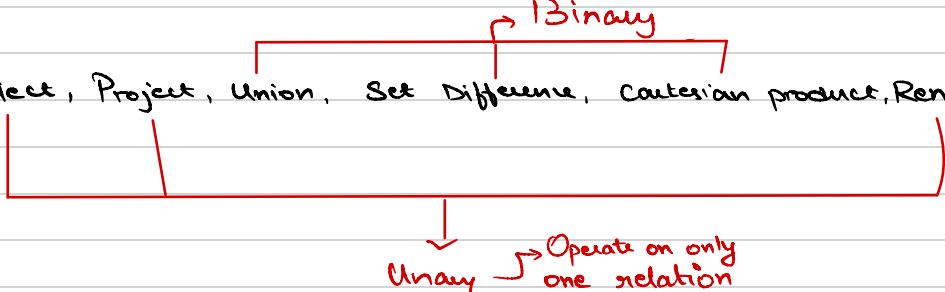
---



# RELATIONAL ALGEBRA.

- \* It is a **procedural Query Language**
- \* Consists of set of operations that take **2/more relations as input & produce a new result as output.**

- \* FUNDAMENTAL OPERATIONS → Select, Project, Union, Set Difference, Cartesian product, Rename



- \* ADDITIONAL OPERATIONS → Set Intersection, Natural Join, Assignment.

## SELECT ( $\sigma$ ) — SIGMA

- \* Used to select tuples that satisfy a given predicate.

↳ Conditional column

EXAMPLE:

ID	name	dept	salary
1	A	D	20
2	B	D	21
3	C	E	22

# Select instructors in 'D' dept:  
↓  
table name

$$\sigma_{dept} = "D" (instructor)$$

# Select instructors with salary more than 20

$$\sigma_{salary > 20} (instructor)$$

## PROJECT ( $\pi$ )

RollNo	Name	Age
1	A	20
2	B	21
3	A	19
:	:	:

# Retrieve roll.no from student

$$\pi_{RollNo} (\text{student})$$

$$\pi_{RollNo, Name} (\text{student})$$

# Name of student with RollNo = 2.

$$\pi_{Name} (\sigma_{RollNo=2} (\text{student}))$$

\* Projection = columns

\* Selection = Rows

# CARTESIAN / CROSS - PRODUCT

A	B	C
1	2	3
2	1	4

C	D	E
3	4	5
2	1	2

R <sub>1</sub> × R <sub>2</sub> ⇒	A	B	C	C	D	E
	1	2	3	3	4	5
	1	2	3	2	1	2
	2	1	4	3	4	5
	2	1	A	2	1	2

\* Cross-product / Joining can only be done if atleast One column is same.

## SET- DIFFERENCE

RollNo	Name	EmpNo	Name
1	A	9	E
2	B	1	A
3	C		

Student

Employee

- #1 No. of Columns must be same
- #2 Domain of Every column must be same

Student - Employee →

RollNo	Name
2	B
3	C

$$(\Pi_{\text{name}, \text{rollno}} (\text{Student}) - \Pi_{\text{name}, \text{rollno}} (\text{Employee}))$$

### OTHER RELATIONS

- #1 → Rename ( $\rho_x$ )  
↳  $x$  is the new name
- #2 → Assignment ( $\leftarrow$ )  
acts like '='.

## UNION:

RollNo	Name	EmpNo	Name
1	A	9	E
2	B	1	A
3	C		

Student

Employee

- # No. of columns must be same

- # Domain of Every Respective column is same.

$$(\Pi_{\text{name}, \text{rollno}} (\text{Student}) \cup (\Pi_{\text{name}, \text{rollno}} (\text{Emp})))$$

$$(\Pi_{\text{name}, \text{rollno}} (\text{Emp}))$$

# JOINS

- \* Cross join
- \* Natural join
- \* Conditional join
- \* Equijoin

## \* Self-Join

- \* Outer join

left IX  
Right IX  
full IX

## NATURAL JOIN: (X)

join = Cross product + Condition.

- \*  $\Pi_{\text{name}, \text{course\_id}}$  (instructor or teacher)

## EXTENDED RELATIONAL ALGEBRA.

### #1 Generalised projection:

\* Normal projection only displays columns.

\* whereas, Generalised projection enables arithmetic Operation on projected columns.

r	A	B	C	$\Pi_{A, C-B}(r)$
a	1	5		
a	2	6		
b	3	7		
b	4	8		

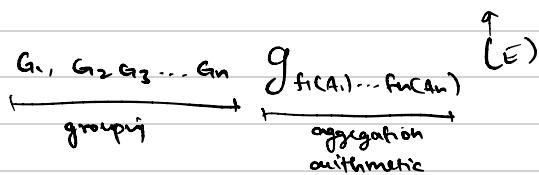
### #2 Aggregation Values:

\* Takes multiple values & returns single value as result.

- avg  
- min  
- max  
- sum  
- count

{ - distinct: unique }

Relation.



Cause for  
Normalization.

## Anomalies in DB Design

### INSERT Anomaly

\* Can't add Data to DB due to missing information.

### UPDATE Anomaly

\* Updating same data in multiple places leading to inconsistency.

### DELETE Anomaly

\* Deleting a Record, accidentally delete other important data.

## NORMALIZATION

- \* It is a database design Technique to reduce Data Redundancy & prevent uncertainties like INSERT, UPDATE and DELETE Anomalies.
- \* Normalization rules divides larger tables into smaller ones & links them using relationships.

## Functional Dependencies

- \* It is a constraint that specifies relationship b/w 2 sets of attributes.

\*  $X \rightarrow Y$  → Dependent

↳  $X$  is a set of attributes that is capable of determining value of  $Y$ .  
Determinant.

# Ex:	Name	RollNo	CGPA
A		1	9.1
A		2	7.2
B		3	9.1
C		4	8.4

\* RollNo can uniquely determine CGPA, NAME

## Armstrong's Axioms/properties.

1) REFLEXIVITY: If  $Y$  is a subset of  $X$ ,  
then  $X \rightarrow Y$  holds true.

2) AUGMENTATION: If  $X \rightarrow Y$  is valid,  
then  $XZ \rightarrow YZ$  is also valid

3) TRANSITIVITY: If  $X \rightarrow Y$  and  $Y \rightarrow Z$  are valid,  
then  $X \rightarrow Z$  is valid

## TYPES OF F.D:

- #1 Fully FD
- #2 Partial FD
- #3 Trivial FD
- #4 Non-trivial FD
- #5 Multi-valued FD
- #6 Transitive FD.

## Fully FD:

\*  $X \rightarrow Y$  is fully FD if  $Y$  cannot be determined by any proper-subset of  $X$ .

\*  $ABC \rightarrow D$

$A \not\rightarrow D$	$AC \not\rightarrow D$
$B \not\rightarrow D$	
$C \not\rightarrow D$	
$AB \not\rightarrow D$	
$BC \not\rightarrow D$	

## Partial FD:

\*  $X \rightarrow Y$  is partially FD if  $Y$  can be determined by any proper-subset of  $X$ .

## Trivial:

\*  $A \rightarrow B$  is trivial FD if  $B$  is subset of  $A$ .

## Non-Trivial:

\*  $A \rightarrow B$  is non-trivial FD if  $B \not\subset A$ .

## Multi-valued:

\* Multiple independent attributes in a table.

## Closure:

\* It is the set of attributes that can be functionally determined from it.

\* Closure of  $X$  is  $X^+$

# Eg:  $R(A, B, C, D)$

FD:  $\{A \rightarrow B, B \rightarrow D, C \rightarrow B\}$

$A^+ \rightarrow ABD$

$B^+ \rightarrow BD$

$C^+ \rightarrow CD$

$D^+ \rightarrow D$

PRACTICE CLOSURE  
AND CANDIDATE KEY  
PROBLEMS.

# NORMALIZATION forms:

## I FIRST NORMAL FORM:

- \* Should contain only atomic valued column
- \* All columns are unique and values are of same domain.

Roll	Name	Subj		Roll	Name	Subj
1	A	C		1	A	C
2	B	C++, C		2	B	C
3	C	C#		2	B	C++
				3	C	C#

## II SECOND NORMAL FORM

- Condition 1 → It should be 1NF
- Condition 2 → No Partial dependency.

## III THIRD NORMAL FORM

- Condition 1 → It should be 2NF
  - Condition 2 → No Transitive Partial dependency.  
*Attribute depending on other attribute and not on composite key*
- \* 3NF aims on removing data duplication and improve data integrity.

## IV BOYCE-CODD NORMAL FORM.

- Stricter version of 3NF
- Condition 1 → Should be 3NF
- Condition 2 → FD, LHS should be Superkey.

## V FOURTH NORMAL FORM

- Condition 1 → BCNF
- Condition 2 → No Multi-valued Dependency.