DBMS - IV

RELATIONAL ALGEBRA.

* It is a procedural Query language
* Consists of sets 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
  - Binary: Union, Set Difference, Cartesian product
  - Unary: Select, Project, Rename → Operate on only one relation
* ADDITIONAL OPERATIONS → Set Intersection, Natural Join, Assignment.

SELECT (σ) — 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 :)

Instructors → table name

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

Select instructors with salary more than 20

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

PROJECT (Π)

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

Retrieve roll.no from student

$$\Pi_{RollNo}(student)$$

$$\Pi_{RollNo, Name}(student)$$

* Projection = columns
* Selection = Rows

Name of student with RollNo = 2.

$$\Pi_{name}(\sigma_{RollNo = 2}(students))$$

CARTESIAN / CROSS-PRODUCT

A	B	C
1	2	3
2	1	4

R_1

C	D	E
3	4	5
2	1	2

R_2

$R_1 \times R_2 \Rightarrow$

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	4	2	1	2

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

SET-DIFFERENCE

RollNo	Name
1	A
2	B
3	C

Student

EmpNo	Name
7	E
1	A

Employee

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

Student - Employee →

RollNo	Name
2	B
3	C

$(\Pi_{\substack{name, \\ RollNo}}(Student) - \Pi_{\substack{name, \\ RollNo}}(employee))$

OTHER RELATIONS

#1 → Rename (ρ_x)
↳ x is the new name

#2 → Assignment (←)
acts like '='.

UNION:

RollNo	Name
1	A
2	B
3	C

Student

EmpNo	Name
7	E
1	A

Employee

RollNo	Name
1	A
2	B
3	C
7	E

No. of columns must be same
Domain of Every Respective Column is same.

$(\Pi_{\substack{name, RollNo}}(Student)) \cup (\Pi_{name, Roll}(Emp))$

JOINS

* Cross join
* Natural join
* Conditional join
* Equijoin
* Self - Join
* Outer join → Left ⟕ / Right ⟖ / full ⟗

NATURAL - JOIN: (⋈)

join = Cross product + Condition.

* $\Pi_{name, course\_id}$ (instructor ⋈ teaches)

EXTENDED RELATIONAL ALGEBRA.

#1 Generalised projection:

* Normal projection only displays columns.
* whereas, Generalised projection enables arithmetic operation on projected columns.

r

A	B	C
a	1	5
a	2	6
b	3	7
b	4	8

$\Pi_{A, C-B}(r)$

#2 Aggregation Values:

* Takes multiple values & returns single value as result.

- avg
- min
- max
- sum
- count

→ distinct: unique

Relation. (E)

$G_1, G_2 G_3 \ldots G_n \; g_{f(A_1) \ldots f(A_n)}$

grouping — aggregation arithmetic

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 deletes other important data.

NORMALIZATION

* It is a database design Technique to reduce Data Redundancy & prevent uncertainities 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 → Y → Dependent
  - X → Determinant.
  - ↳ X is a Set of attribute that is capable of determining value of Y.

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 → Y holds true.

2) AUGMENTATION: If X → Y is valid, then XZ → YZ is also valid

3) TRANSITIVITY: If X → Y and Y → Z are valid, then X → 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.

* $ABC \rightarrow D$ ✓

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

  $AC \not\rightarrow D$

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 → B, B → D, C → B}

$A^+ \Rightarrow ABD$
$B^+ \Rightarrow BD$
$C^+ \rightarrow CB$
$D^+ \rightarrow D$

PRACTICE CLOSURE AND CANDIDATE KEY PROBLEMS.

NORMALIZATION forms:

I FIRST NORMAL FORM:

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

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

→

Roll	Name	Subj
1	A	C
2	B	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.
  - Transitive: 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.