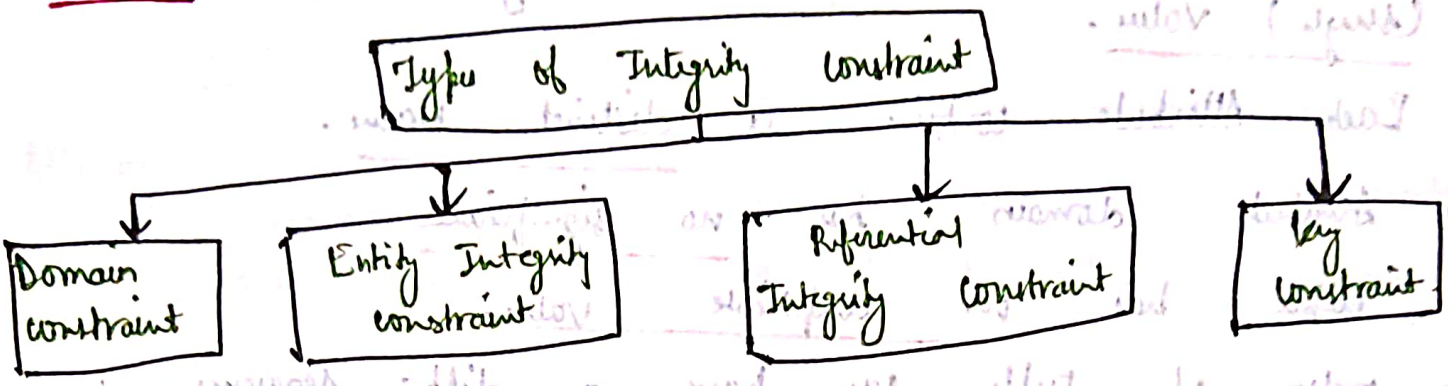
Integrity Constraints

Integrity constraints are a set of rules. It is used to maintain the quality of information.

Integrity constraints ensure that the data insertion, updating and other process have to be performed in such a way that data integrity is not affected.



1) **Domain constraint** (kis type ka data define karenge)

Domain constraints can be defined as the definition of a value set of values for a attribute.

The datatype of domain includes string, character, integer, time, date, etc. The value of the attribute must be available in the corresponding domain.

ID	Name — string	AGE — int
101	Aman	20
102	Arun	A → Not allowed string
103	32 → not allowed string	30

i) Entity integrity constraints

↳ The entity integrity constraint states that Primary Key value can't be null.

↳ This is because the primary key value is used to identify individual rows in relation and if the PK has a null value, then we can't identify those rows.

↳ A table can contain a null value other than the primary key field.

Emp-Id	Emp-Name	Salary
123	Komal	30000
NULL	Arun	40000

Not Allowed: (Never).

ii) Referential Integrity constraints (Foreign key concept).

↳ A referential integrity constraint is specified b/w two tables.

↳ In the Referential integrity constraints, if a foreign key in Table 1 refers to the PK on Table 2, then Every value of the foreign key in Table 1 must be null or be available in table 2.

4. Key constraints (Primary Key concept).

↳ Keys are the entity set that is used to identify an entity within its entity set uniquely.

↳ An Entity set can have multiple keys, but out of which one key will be Primary key.

↳ A Primary key can contain a unique value in the Relational table.

ID	Name	sem.	Age
101	Arun	1st	17
102	Ravi	2nd	81
(101)	Aman	3rd	24

Not allowed Because all value of PK must be unique

Relational Algebra.

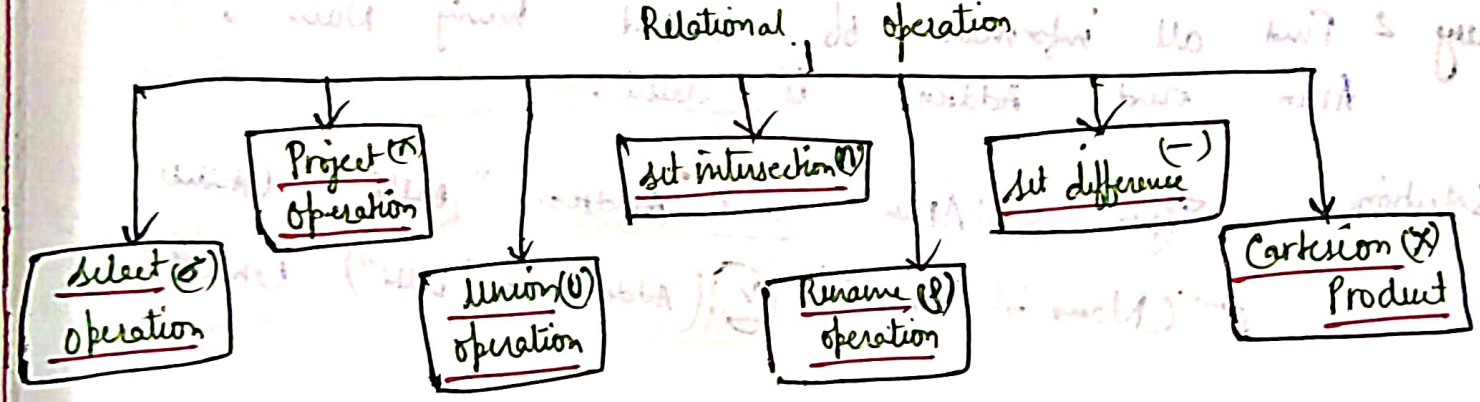
↳ Relational Algebra is a Procedural Query know language. (PQL)

↳ It gives a step by step Process to obtain the result of the Query.

↳ Relational Algebra mainly provides theoritical foundation for relational dB and SQL

↳ It uses operators to perform Queries.

Types of Relational operation



1. Select operation (σ)

↳ The select operation is used to select a subset of the tuple from a relation that satisfies a select condition.

↳ σ (sigma) is used to denote select operator.

↳ The select operator is (unary) such that it is applied to a single relation.

The general format of select operation is

$$\sigma_{<\text{select condition}>}(R)$$

where σ is used for selection prediction

R is used for Relation.

4 Select condition is the relational operators like =, ≠ ≥, ≤.

Ex :- student

Name	Rollno.	address
Aman	02	delhi
Arun	05	Rishikesh

just select ke jagah hai

Query 1 – Print all information of student having Roll is 04.

Solution :- σ Roll no = 04 (student)

Query 2- Find all information of student having Name is Arun and Address is delhi.

Solution :- σ (Name = "Arun" and Address = "Delhi") (student)

$= \sigma$ (Name = "Arun") $\wedge$ (Address = "Delhi") (student).

2. Project operation (π)

↳ Project operation selects certain columns from the table and discard the other columns.

↳ This operation shows the list of those attributes — that we wish to appear in the Result.

↳ Rest of the attributes are eliminated from the table.

the general format for project operation is

$$\pi < \text{Attribute list} > (R)$$

Where π is the signal used to represent the project operation

and attribute list is the list of attributes from the attributes of Relation R.

Ex = Student

Name	Roll no.	Address
Aman	02	Delhi
Karan	50	Katras
Arun	41	Rohor

Query 01 : Find student Name in the student table.

Solution : Π Name (student).

Query 02: Find student Name and Address list.

Solution Π Name, Address (student).

3 UNION operation (U)

↳ It performs binary union b/w two given Relation and is defin as

R U S

where R and S are either db relations or relation result set (temporary relation)

Union operation to be valid the following condition must hold —

↳ R and S must have the same number of Attributes

↳ Attributes domain must be Compatible.

↳ Duplicate tuples are automatically eliminated.

Ex

$$\Pi_{name}(student1) \cup \Pi_{name}(student2)$$

↓

Union (sara but not repeat any data).

4. Set Intersection ($\cap$)

↳ It performs binary Intersection b/w two given Relations and it is defin as:

$$R \cap S$$

where R and S are either dB relation or relation result set

Intersection (common)

Ex - $\Pi_{name}(student1) \cap \Pi_{name}(student2)$

5. Set Difference (−)

↳ The result of set differen operation is tuples, which are present in one relation but are not in the second relation.

$$R - S$$

Notation is to finds all the tuples that are present in R but not in S.

Ex $\Pi_{name}(student1) - \Pi_{name}(student2)$

Note

(Dono mein jo comman hai wo hat jayega aur jo sirf Pehle wale Matlab (R) mein Bacha hoga wo milega result humko).

6. Cartesian Product (X)

↳ Combins information of two diff. relations into one.

RXS.

where R and S are relations and their output will define as -

$$RXS = \{ qt \mid q \in R \text{ and } t \in S \}.$$

Ex

$\sigma_{\text{Name} = \text{'Kaul'}}$ (student 1 × student 2)

7 Rename operation (ρ) (Rho) Rho (ρ)

↳ Rename operation is used to rename the output of a relation rename operation is denoted with small Greek letter rho (ρ).

$\rho_x(E)$

where the Result of Expression E is saved with name of X.

Ex:- Query to rename the table student to employee and its attributes name, address, phone no.

$\rho_{\text{employee}(\text{name, address, phone no})}$ (student)

isse change ho jayega and new name employee ho jayega

4

Normalization

Functional Dependency

- The Functional Dependency is a relationship that exists b/w two attributes.
- It typically exists b/w the Primary key and non-key attributes within a table.

$$X \longrightarrow Y.$$

The left side of FD is known as a determinant
The Right side of the production is known as a dependent. If $t_1.x = t_2.x$ then $t_1.y = t_2.y$ (must).

For example :- Assume we have an employee table with Attributes. Emp-Id, Emp_Nam, Emp_Addres.

Here Emp-Id Attribute can uniquely identify the emp_nam attribute of employee table because if we known the emp_Id we can tell that employee name Associated with it.

Functional dependency can be written as:

Emp_id → Emp_Nam

we can say that emp_Name is functionally dependent on Emp_Id

Types of Functión dependency

- Trivial functional Dependency
- Non-trivial Functional dependency.

Trivial Functional dependency

- $A \rightarrow B$ has trivial functional dependency if B is a subset of A.
- The following dependicies are also Trivial like : $A \rightarrow A$, $B \rightarrow B$.

Ex ⇒ consider a table with two columns employee id and employee_Name.

{ Employee_id, Employee_Name } — employee_id

trivial Functional dependencies as employee id is a subset of { employee_id - employee_Name } Also employee id → Employe id and employee_Name

→ employee_Name are trivial dependencies too.

Non trivial Functional Dependency

- $A \rightarrow B$ has a non-trivial Functional dependency if B is not a subset of A.
- When A intersection B is NULL, then $A \rightarrow B$ is called as complete non-trivial.

Ex:- Consider a table with three column emp id, emp Name and emp-Age.

Emp_id → emp_Name is a Non trivial Functional dependency because emp Name is not a subset of emp_id

Similarly

emp id - emp_Name ⟶ emp_Age is non trivial because emp_Age is not a subset of emp_id - emp No.

* closure of a set of functional dependencies

Inference Rule (IR) — Armstrong's Axioms ...

↳ The inference rule is a type of assertion. It can apply a set of FD (Functional dependencies) to derive other FD.

↳ using the inference Rule, we can derive additional functional dependency from the initial set.

1. Reflexive Rule (IR1)

↳ In the Reflexive Rule, If Y is a subset of X, then X determines Y.

If $y \subseteq x$ then $x \rightarrow y$

Ex

$x = \{a, b, c, d, e\}$

$y = \{a, b, d\}$

2. Argumentation Rule (IR2)

↳ The Argumentation is also called as a Partial dependency

In Argumentation if X determines Y, then XZ determines YZ for any Z.

if $x \rightarrow y$ then $xz \rightarrow yz$

Ex

for R(ABCD)

if $A \rightarrow B$ then

$AC \rightarrow BC$

3. Transitive Rule (IR3)

↳ In the transitive Rule, if X determine Y and Y determine Z, then X must also determine Z.

If $X \to Y$ and $Y \to Z$ then $X \to Z$

4. Union Rule (IR4)

↳ Union Rule says if X determine Y and X determine Z then X must also determine Y and Z.

If $X \to Y$ and $X \to Z$ then $X \to YZ$.

5. Decomposition Rule (IR5)

↳ Decomposition Rule is also known as project rule. It is the reverse of union rule. This Rule says if X determine Y and Z then X determine Y and X determine Z.

If $X \to YZ$ then $X \to Y$ and $X \to Z$.

6. Pseudo transitive Rule (IR6)

↳ In Pseudo transitive Rule, If X determine Y and YZ determine W, then XZ determine W.

If $X \to Y$ and $YZ \to W$ then $XZ \to W$.

7. Composition :- If $x \to y$ & $A \to B$ then $XA \to yB$ ~~then XA → yB~~

4

Normalization

↳ Normalisation is the Process of organising the data in the DB.

↳ It is used to minimize the redundancy (repeatation) from a relation or set of relation.

↳ It is used to eliminate undesirable characteristics like insertion, update and deletion Anomalies.

↳ It divides the larger table into smaller and links them using relationship.

↳ The normal form is used to reduce redundancy from the DB table.

↳ It is the process of making the table free from insertion, updation and deletion anamolies. / save data also.

* When we normalize the DB we have 4 goals:

N of

1. Arranging data into logical groupings such that Each group describes a small part of the whole

2. Minimizing the amount of duplicate data, called redundancy, stored in a DB.

3. Organising the data such that, when you modify it you make the changes only in one place.

4. Building a DB in which you can Access and manipulate the data quickly and efficiently without compromising the integrity of the data in storage.

Types of Normal forms

1NF 2NF 3NF BCNF 4NF 5NF

1NF (First normal form)

- A Relation is in 1NF if it contains an Atomic value (single)
- It eliminate Repeating groups.

2NF (Second Normal Form)

- A Relation will be in 2NF if it is in 1NF and all non key attributes are fully functional dependent on the Primary key.

3NF (Third Normal Form)

- A Relation will be in 3NF if it is in 2NF and no transition dependency exists.
- It eliminate transition dependency.

BCNF (Boyce codd's normal Form)

- A stronger definition of 3NF is known as Boyce Codd's normal Form.
- It is also called 3.5 NF.

4NF (Fourth Normal Form)

- A Relation will be in Fourth normal form if it is in Boyce Codd's Normal Form and has no multivalued dependency.

4 ↳ Eliminate Multi-Valued dependency.

5NF (Fifth Normal Form)

↳ A Relation is in 5NF if it is in 4NF and does not contain any join dependencies joining should be lossless
↳ eliminate join dependency.

* Advantages of Normalization

↳ It helps to minimize data redundancy (repeatation).
↳ Greater overall database organisation. (better) / proper.
↳ Data consistency within the DB. (No loss of data).
↳ much more flexible DB design. (better design of DB)
↳ Fewer indexes per table ensure Faster maintanence tasks (index rebuilds). ↳ Access.
↳ Relizes the option of Joining only the tables that are needed.
↳ A better handle on DB security
↳ Increase storage efficiency
↳ Speed up data Access.

* Disadvantages of Normalization.

↳ You cannot start building the DB before knowing what the user needs.

↳ Requires much CPU, memory and I/O process than normalized data gives reduced DB performance.

↳ Requires more joins to get the desired result; A poorly-written Query can bring the DB down.

↳ It is very time-consuming and difficult to normalize relations of a higher degree

↳ Careless decomposition may lead to a bad DB design, leading to serious problems.

↳ Cost high

4. First Normal Form (1NF)

↳ A Relation will be 1NF if it contain an atomic (Individual) (single) value.

↳ It states that an attribute of a table cannot hold multiple values. It must hold only single value attributes.

↳ It disallow the multivalued attribute, composite attribute and their combinations.

Ex. Relation student is not in 1NF because of multivalued attribute Stud_Phon → No duplicate rows allowed.

↳ No ordering to rows and columns.
→ Each column have unique name

Stud_Id	Stu_Name	Stud_Phon (Multivalued Not Allowed)	Stud_Branch
11	Kamal	8442891 24, 7124489 91	IT
12	Karan	8234 58 910	CS
13	Ravi	8910568888, 9431349154	ES
14	Ram	8217 498125	ME
15	Komal	6203377224	IT

The decomposition of the student table into 1NF is

Convert it into 1NF

Stud_Id	Stud_Name	Stud_Phon	Stud_Branch
11	Kamal	8442891 24	IT
11	Kaml	7124489 91	IT
12	Karan	8234 58910	CS
13	Ravi	8910568888,	ES
13	Ravi	9431349154	ES
15	Komal	6203377224	IT
14	Ram	8217498145	ME

Second Normal Form (2NF) (Most important)

↳ In the 2NF, relation must be in 1NF - 1

↳ No attributes of the table should be functionally dependent on only one part of a Concatenated PK. (dependent on Both ($AB \rightarrow C$) $A \rightarrow C$ and $B \rightarrow C$)).

↳ All non-key attributes are fully functional dependent on the PK.

Ex> 2NF is the based the concept of full functional dependency $X \rightarrow Y$ is a fully function Dependency (FFD) if removed of any attribute (A) from X means that the dependency does not hold any more:

In the Book order table such as.

order No.	title	Qty	Unit Price.
1	Computer Network	1	200
1	Java	1	275
1	DBMS	2	295
2	Multimedia	1	300
2	Data structure	1	190
3	DBMS	1	295
3	Multimedia	2	300
3	Computer Network	5	280

It is not in 2NF because it hold Partial function Dependency and Fully function dependency.

title $\rightarrow$ Unit Price
PFD (Partial functi dependency.

and order No, Title $\rightarrow$ Qty } FFD

4

Order No $\xrightarrow{\not\rightarrow}$ Qty

Title $\xrightarrow{\not\rightarrow}$ Qty.

Now we will convert the given table in 2NF to decompose the given table into two sub table such as

Order Master table → Fully Functional dependency (FFD).

Order No	Title	Qty.
1	Computer Network	1
1	Java	1
1	DBMS	2
2	Multimedia	1
2	Data structure	1
3	Multimedi	2
3	Computer Netwrk	5

Book Master table → Remove Redundancy. → 2NF.

Title	Unit Price
Computer Network	250
Java	275
DBMS	295
Multimedia	300
Data structure	190

Third Normal Form (3NF)

- A relation is in 3NF if it is in 2NF and no non Prime attribute functionally dependent on other non-prime Attributes.
- A relation is in 3NF if it is in 2NF be transitively functionally dependent on the Prime key.
- 3NF is used to achieve the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal Form.
- A relation is in third normal Form if it holds atleast one of the following conditions for Every non-trivial function dependency $X \rightarrow Y$.
  1. X is super key.
  2. Y is a Prime attribute such as Each element of Y is part of some candidate key.

Eg → employee_department location table

Emp_Num	E_Name	Sal	Dept_Nam	Dept location
101	Vishal	7000	Account	102
102	Amit	5000	Sales	104
102	Anuj	10000	Account	102
104	Vikas	8000	Sales	104
105	Sanjay	6000	Store	106