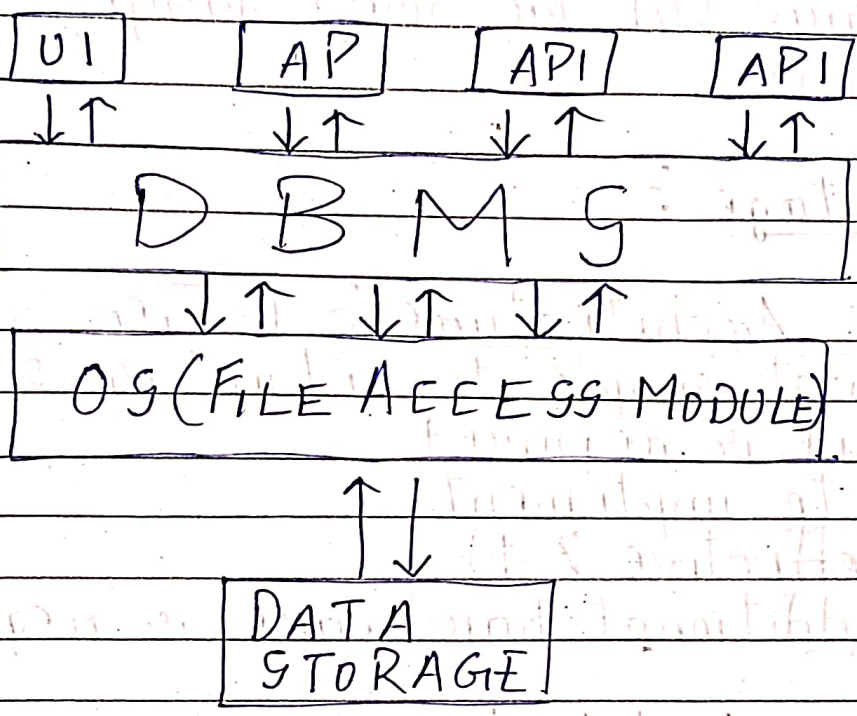
30/08/

Page No.
Date: / /

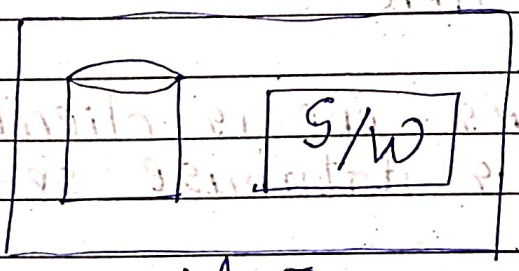
* DBMS → i) Collection of inter-related data.
ii) Set of software tools / program which access & process the data in an easy & effective manner.




* Key Features of DataBase

=> Data storage & retrieval
=> Concurrency control
=> Backup & recovery
=> Data integrity & security

* 1-Tier Architecture



Page No.
Date: / /

=> In 1-Tier Architecture, the database is directly available to the user.

=> Client & server present in same machine

=> To learn SQL or doing project in machine.

Advantage :-

→ Simple Architecture :- 1-Tier architecture is most simple to setup as single machine is required.
→ Easy to implement
→ Cost-effective
→ No additional hardware is required.

* 2-Tier Architecture

Advantage :-

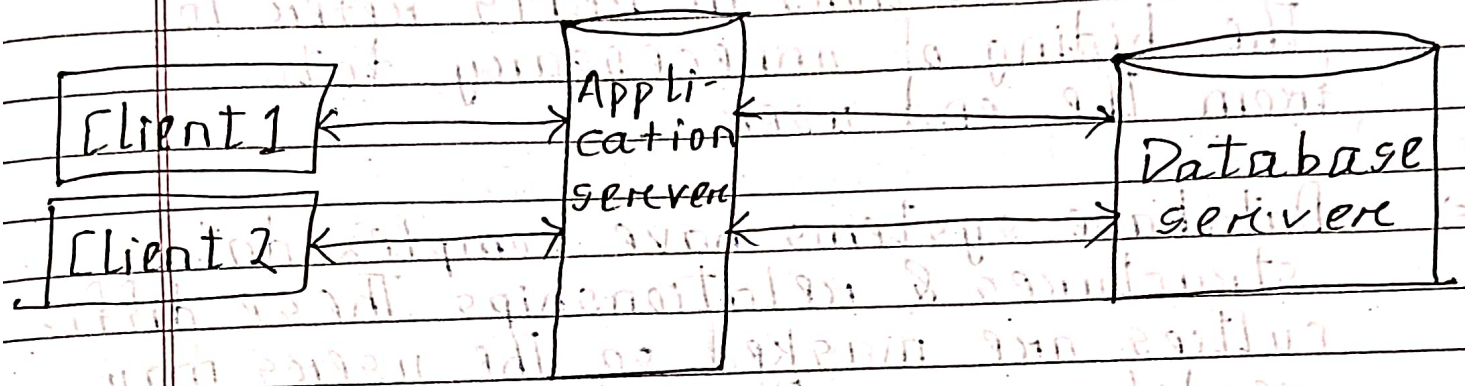
=> Easy to access :- 2 Tier architecture makes easy access to the database which makes fast retrieval
=> Scalable :- Can scale the database easily by adding clients or hardware.
=> Low cost than 3 Tier

=> Specifically 2 tiers - one is client layer & second is database server layer

Page No.
Date: / /

⇒ Client layer providing interface or API & fetch data from database API using 'jdbc', 'odbc' to connect with database.

* 3-Tier Architecture




Advantage :

→ Enhanced Scalability : Scalability is enhanced due to distributed deployment of application server.

→ Data integrity : 3-Tier architecture maintains data integrity. Since there is a middle layer bet^n client & server data corruption can be avoided.

Security : Improves security. This model prevent direct interaction of the client with server thereby reducing access to unauthorised data.

Page No.
Date: / /

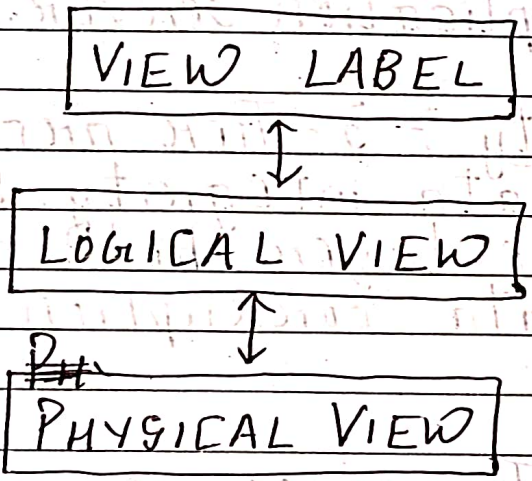
Disadvantages

→ Complex than 1-Tier & 2-Tier.

* Data Abstraction & Data Independence

=> Data abstractions in DBMS refer to the hiding of unnecessary data from the end user.

=> Database systems have complex data structures & relationships. These difficulties are masked so the users may readily access the data & only the relevant section of the database is made accessible through the data abstraction.



05/09/24

Page No.
Date: / /

## Physical / Internal Level

⇒ Lowest level of abstractions for DBMS.

How data is stored

- Data structures or storing data
- Database access mechanisms.
- Define physical storage.
- Data will be stored in block.
- B-Tree or Hash are defined in this level.

## Logical / Conceptual Level

⇒ Next higher level or intermediate label.

⇒ It describes which data are stored & how they are related.

- Define what table, what will be attributes & how the tables are connected.
- Primary key ($P_K$) & Foreign key ($F_K$) are defined.
- Internal data structures (DS) are hidden.

Page No.
Date: / /

## View / External Level

=> This is the top level.

=> Various views are present & it is the part of database.

=> Facilitates numerous views of a single database.

Eg. Student view, Faculty view, Accountant view.

## Data Independence

=> Primary goal of data abstractions in DBMS is to obtain data independence in order to save time & money while modifying or altering a database.

=> Ability to change the schema without impacting the programmers & applications to be rewritten.

=> Data is isolated from program so that changes to the data do not influence program.

=> Data independence are 2 types :-
a) Physical level
b) Logical level

a) Physical level Independence

=> It refers to the ability to change the physical schema without changing the logical / conceptual schema.

b) Logical level Independence

=> This feature is referred to has the ability to change the logical schema without changing the external schema or application progra

=> Any modifications to the conceptual representation of the data wouldn't affect user's perception of the data.

| FILE PROCESSING SYSTEM | DBMS |
|---|---|
| 1) The problem of duplicacy & inconsistency are present in FPS. | No redundancy & inconsistency. |
| 2) Accessing file is not easy. | Very easy in DBMS. |
| 3) No data independence in FPS. | Data independence are present in DBMS. |
| 4) Less Security | More Security. |

06/09/24

Page No.
Date: / /

## Advantages of Data Abstractions

=> It reduces the complexity for the user.

=> Increases the code reusability.

=> Increases the security aspect of the applications.

## Disadvantages of Data Abstractions

=> Might be confusing for developers as there are complexities at multiple level of the database.

=> Whenever extra layers is there, navigation becomes challenging.

## * Data Model

=> Data modelling is the consistency consists of the data & modelling of the data description.

=> Provides the conceptual tool for describing the design of database at each level of the abstractions.

## Types of Data Model

Page No.
Date: / /

1) Relational Data Model
2) E R Model
3) Object based Model
4) Semi-structured Model
5) Network Data Model
6) Hierarchial Data Model

## 1) Relational Data Model

=> Most widely used model by DBMS.

=> It use collection of tables for representing data & relationship.

=> Each table is a group of row & column.

## 2) ER Model (Entity Relationship)

=> High level Data Model.

=> Blue print of actual representation of data.

=> Logical representations of data as objects & relationships among them. These objects are known as entities

## 3) Object Based Data Model

=> Also known Object Oriented Table Data model.

=> Object based data model is an extension of E R Model with some encapsulation, object identity.

09/24

4) Semi Structured Data Model

=> Semi-structured model is the modified form of relational model.

=> Allows data specifications at places where the individual data item may have diff. set of attributes.

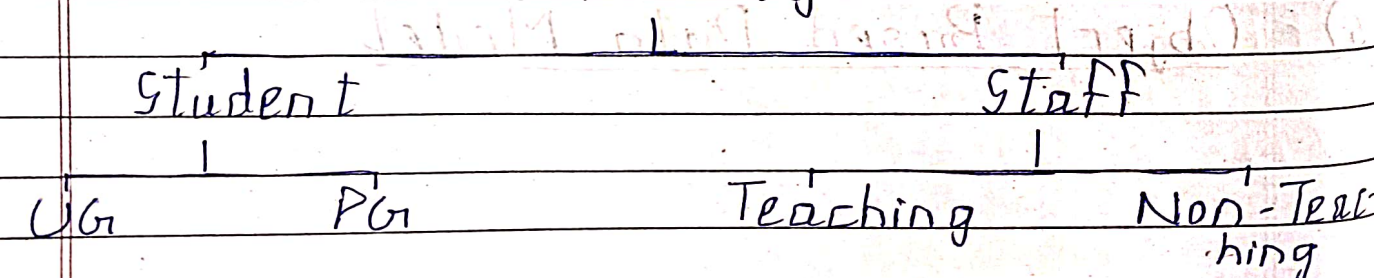
=> Represented by XML (Extended Mark Up Language)

5) Hierarchical Data Model

=> This is the old form of data model

=> This model use a tree like structure

=> Each entity has only one parent.



Teaching — Non-Teaching

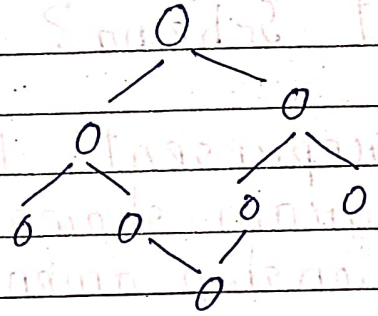
Faculty, Programmer — Executive, Non-executiv

6) Network Data Model

=> Modified version of Hierarchical data model.

=> Represented by using a graph.

=> Unlike hierarchical model, more than one parent can exist.



* Data Model Schema & Instance

=> Overall design of a database is called schema.

=> A database schema is the skeleton structure of the database. It represents logical view of data.

=> A schema contains objects like ~~table~~,

a) Table
b) Primary key
c) Foreign key
d) View
e) Columns
f) Datatype
g) Stored procedure
h) Triggers

Instance :-

⇒ The data which is stored in a database at a particular moment of time is called the instance of the database.

How to Represent Schema?

Schema can be represents by visuals diagram. The diagram shows database objects & relationship among them.

Why we have to draw schema and who will draw it.

Database schema drawn by database designers usually team leader or middle level managers to help programmers whose I/O will interact with database. The process of database creation is called data modelling.

| Student | Roll | Name | Course | DOB |
|---|---|---|---|---|

| College | College | Course name | Credit | Dept |
|---|---|---|---|---|

| Prerequisite | Course Id | Requirement |
|---|---|---|

today

## ER MODEL

=> ER Model stands for Entity Relationship model.

=> This model is used to define the data elements & relationships for a specified system.

=> The ~~be~~ diagram to represent ~~for~~ entity relationship is known as ER diagram.

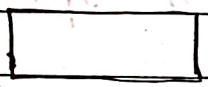
### Entity

It is a thing or object in real world. It is something which has a permanent existence in an enterprise and with an entity some set of attributes are connected depending on some value.

# ER Model

- Entity
  - Weak
  - Strong
- Attributes
  - Simple Vs Multivalue
  - Simple Vs Composite
  - Stored Vs Derived
  - key Vs Non-key
- Relation
  - 1 to 1
  - 1 to many
  - Many to 1
  - Many to many

## Symbols

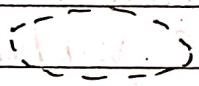
 → Entity

 → Attribute

 → Relationship

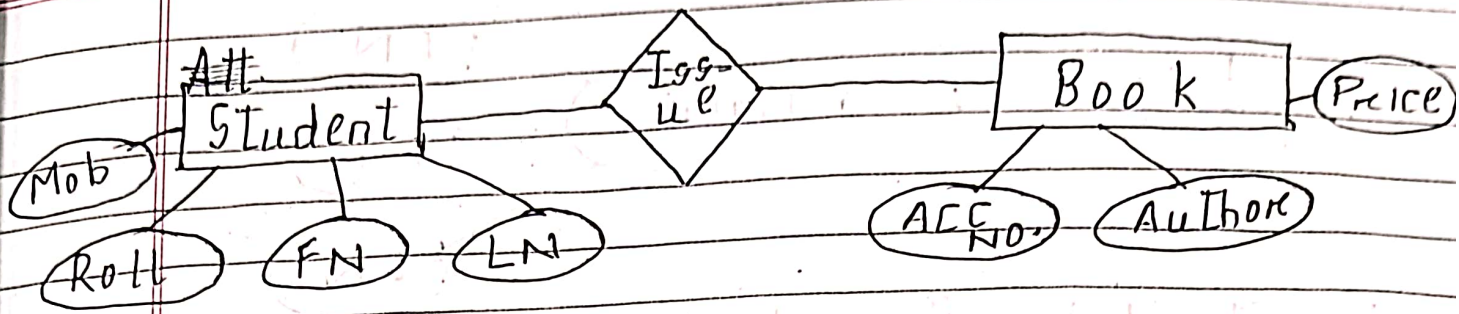
Lines → Link bet$^n$ Entity & Relationship

 → Multi valued Attribute

 → Derived attribute

 → Merging bet$^n$ relationship & entity

 → Weak entity

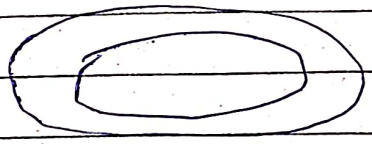


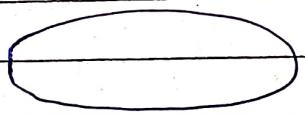

- Attributes

1) Single Vs Multivalue

=> Single → One value for that attribute.
Multiple value → More than one value for attribute.

=> Representation :-

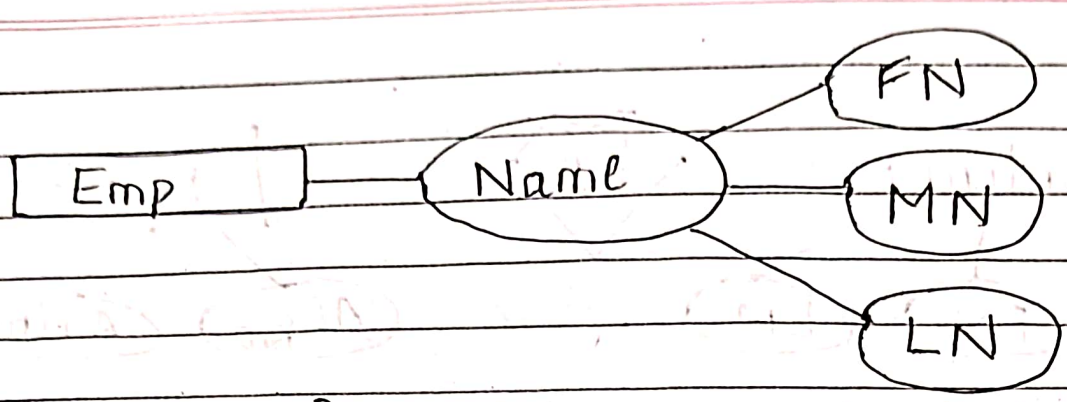
Multivalue 

Single 

2) Simple Vs Composite

=> Composite → Can be devided into further part.
Simple → Cannot be devided further.

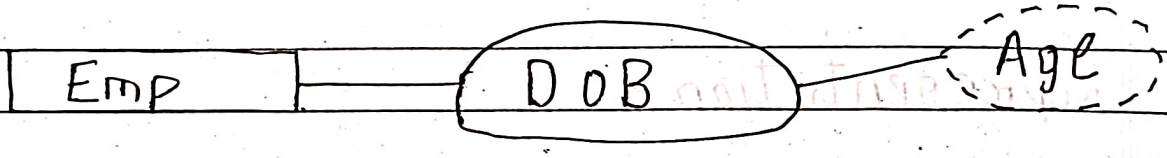
Eg. Name, Address → Composite
Gender, Salary → Simple




3) Stored Vs Derived

⇒ Stored → Data are explicitly entered & saved
Derived → Calculated from other attributes

Eg. DoB —— Age

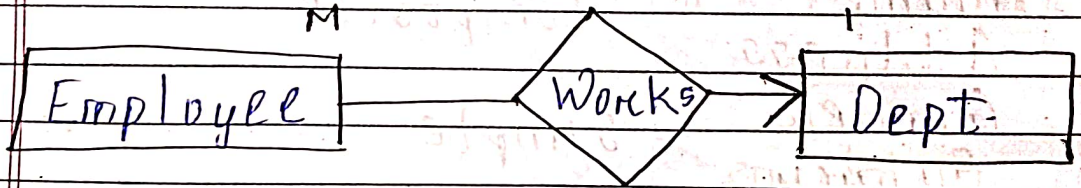



19/09/24

Relationship

⇒ Relationship can be defined as how 2 entities can be associated with each other

⇒ It can be one-to-one, one-to many, many-to-many & many-to-many.



20/09/24

Page No.
Date: / /

* Cardinality

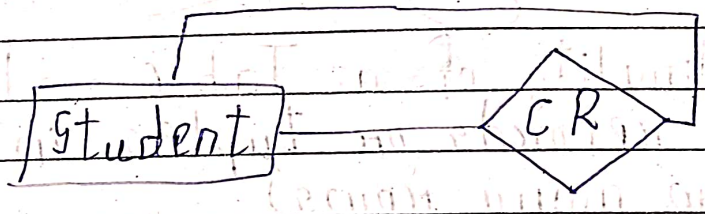
=> Mapping cardinality refers to the relationship bet^n 2 entities in a database.

=> It specifies how many instances of one entity can be linked to instance of another entity - 1:1, 1:m, m:1, m:n.

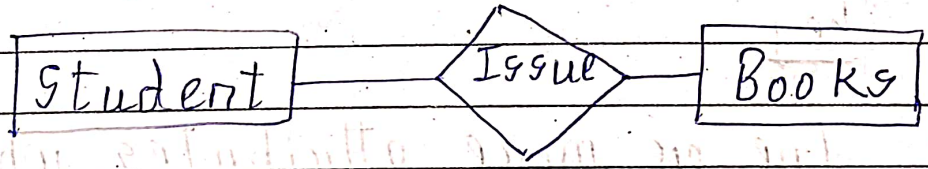
* Degree of a Relationship

=> It defines no. of entity that associate in a relationship.

1) Unary :-

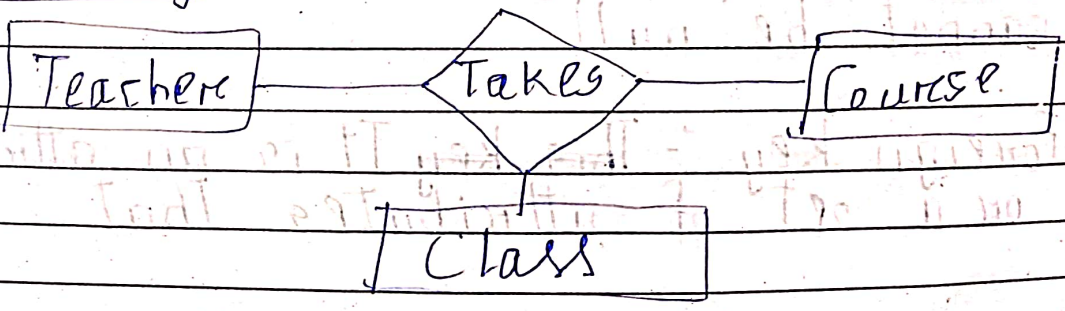



2) Binary :-




→ Relationship is connected with 2 entities

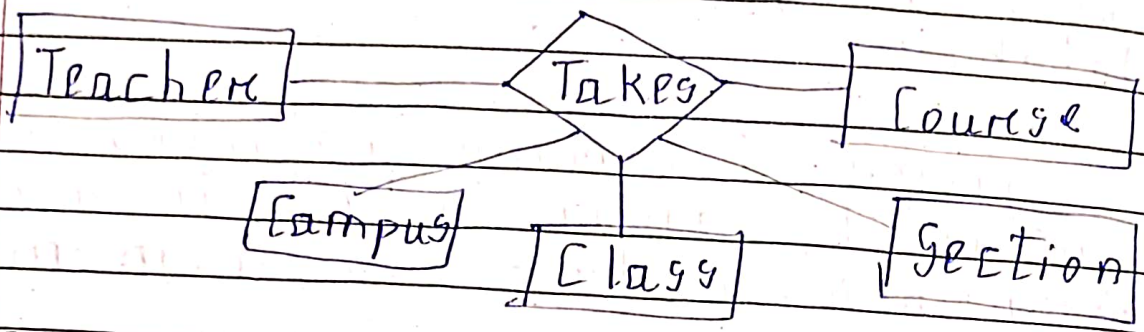
3) Ternary :-



Page No.
Date: / /

4) N-ARY :

n type of entities are connected with 1 relation




25/09/24

- Degree of a Table → How many attributes are there in a table. (column)

Degree = 4

- Cardinality of a Table → Refers to no. of records or tuples in a table (how many rows)
- key

⇒ One or more attributes which can uniquely identify tuple.

Primary key : This key uniquely identify each row & value of that attribute cannot be null.

Foreign key : ~~This key~~ It is an attribute or a set of attributes that

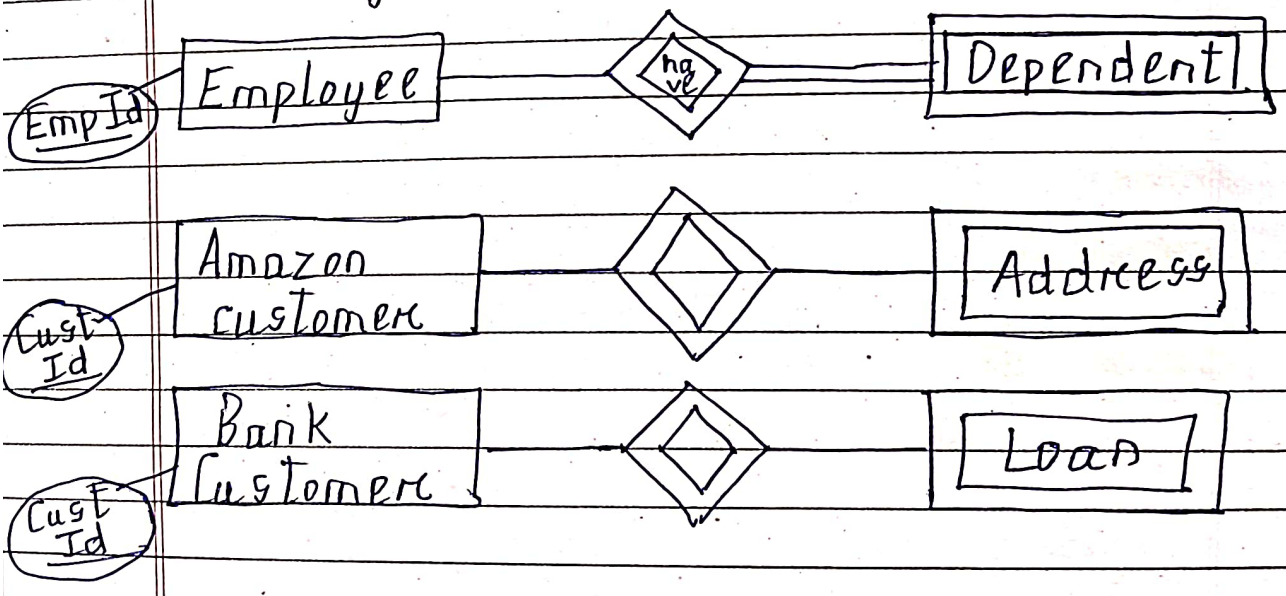
references to a primary key of some table or any relationship.

→ It maintains Referential integrity.

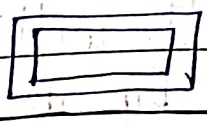
- Entity

Strong Entity :- It is ~~do~~ not dependent on any other entity in the schema. It will always have PK & represented by a rectangle.

Weak Entity :- It is dependent on strong entity. Unlike strong entity, weak entity does not have a PK. Relation bet$^n$ Weak entity & Strong entity is represented by double diamond. Weak entity represented by double rectangle.



Page No.
Date: / /

| | Strong Entity | Weak Entity |
|---|---|---|
| 1 | Has Pk | Has partial discrimination key |
| 2 | Not dependent on other entity | Depend on strong entity. |
| 3 | Represented by  | Represented by  |

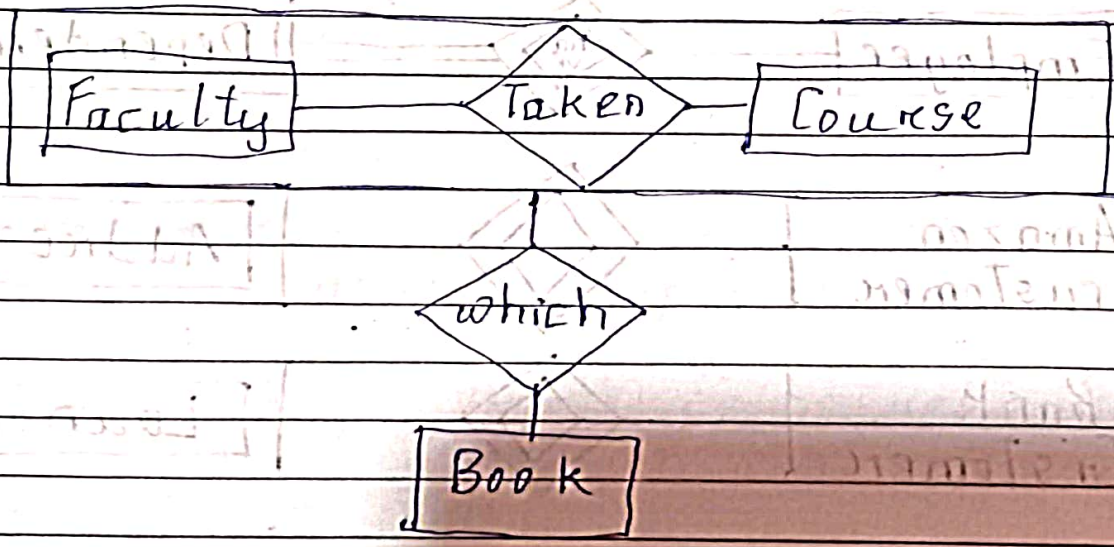
| | Strong Entity | Weak Entity |
|---|---|---|
| 1 | Has Pk | Has partial discrimination key |
| 2 | Not dependent on other entity | Depend on strong entity. |
| 3 | Represented by  | Represented by  |

03/10/24

## Extended ER Diagram

→ Generalisation
→ Specialization
→ Aggregation

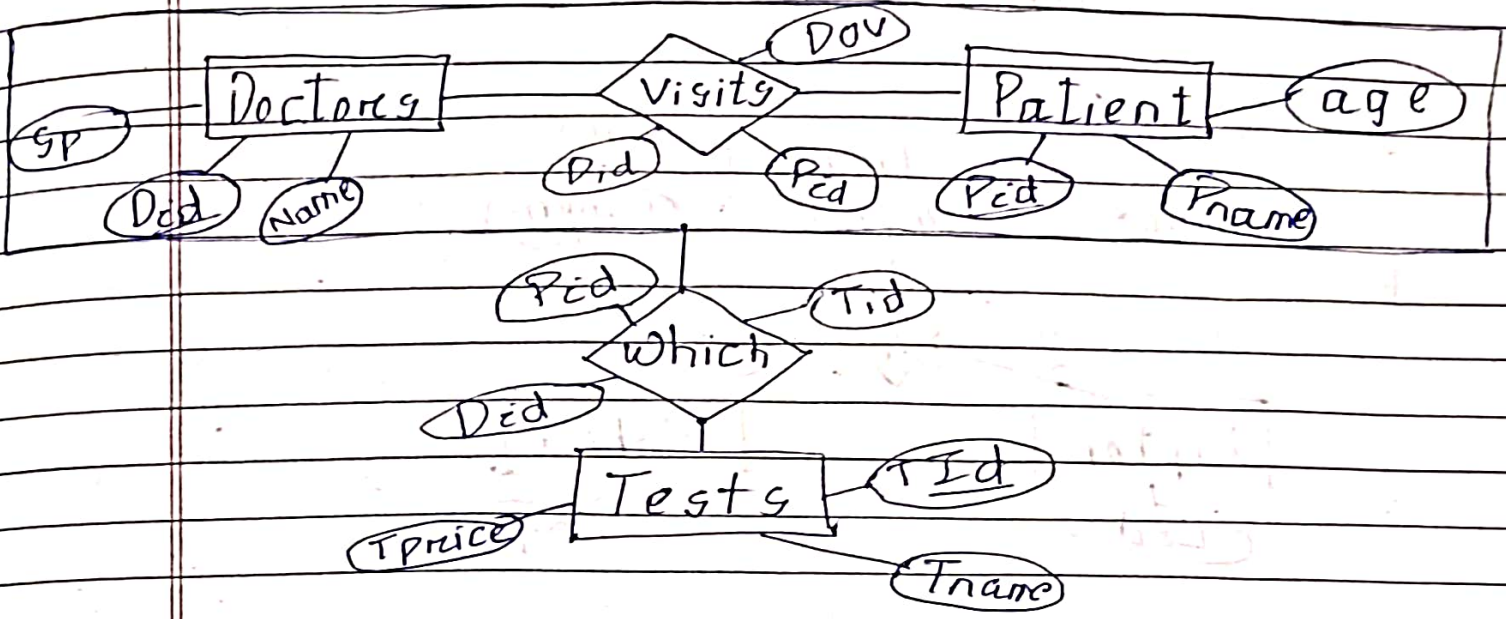
## Aggregation



Page No.
Date: / /

⇒ Aggregation in DBMS is a process of combining 2 or more entities do not make sense on their own without applying the aggregation process.

⇒ In order to create aggregation bet$^n$ 2 entities which cannot be used for its individual quantities rather a relationship is established & the resultant product is created into new entities.




Generalization & Sp

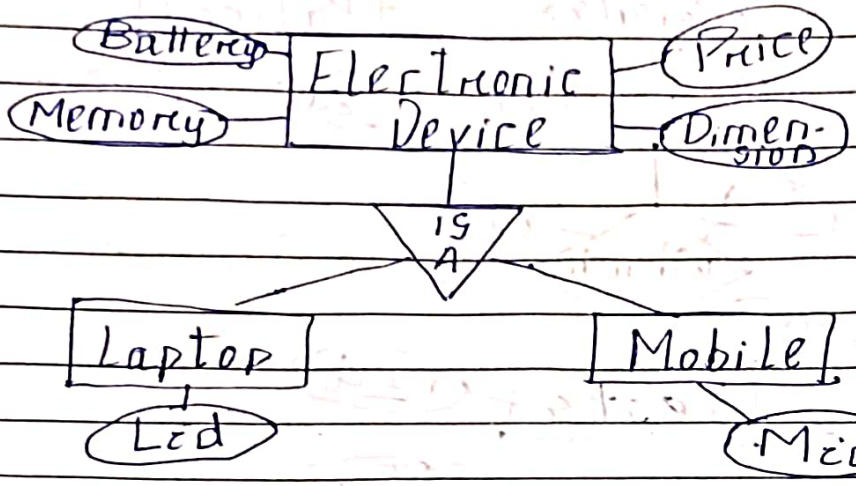
It is known as extended ER-diagram

For generalization, it is a bottom up approach

- In generalization, lower level func's are combined to form higher level entities
- In the generalized process, we simplify the entities.

Mobile (Mobile-id, Dimension, Memory, Battery, Price)

Laptop (Laptop-id, Dimension, Memory, Battery, Price)




Rule Table → Leaf node

- Specialization means app. of generalization
- Broken into smaller parts to simplify it further.
- It is a top down approach.

Page No.
Date / /

# Relational Algebra

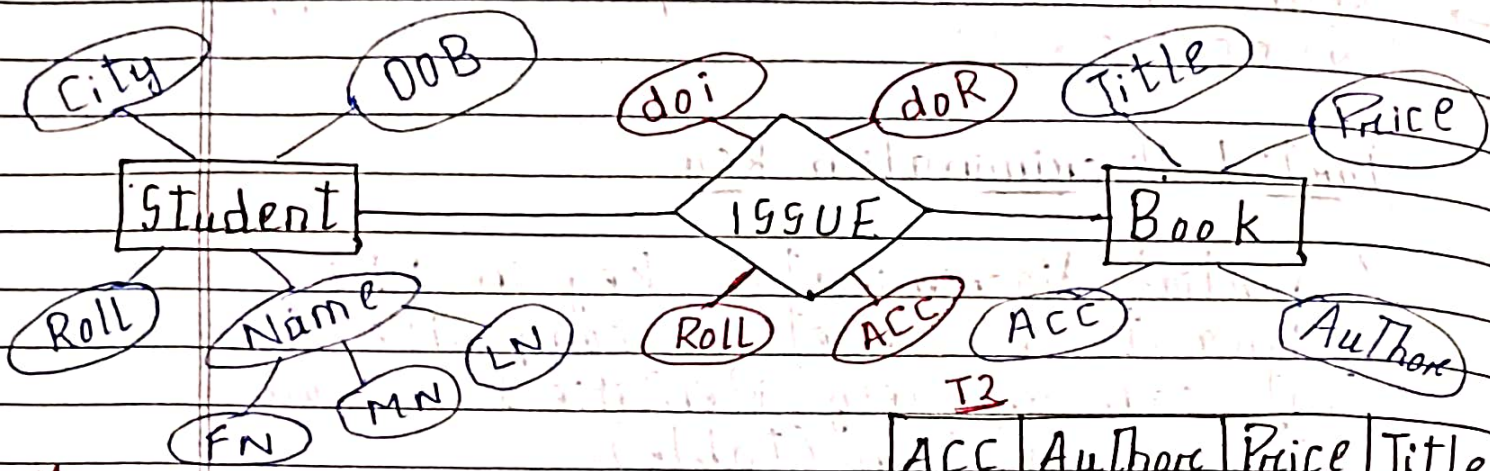
=> It is a procedural language for manipulating relations.

## Fundamental Operator

Unary → Select, Project, Rename
Binary → Union, set diff, cartesian product

## Secondary Operator

Intersection, Natural join, Division & Assignment




T1

| Roll | FN | MN | LN | City | DOB |
|---|---|---|---|---|---|
| | | | | | |

T2

| ACC | Author | Price | Title |
|---|---|---|---|
| | | | |

T3

| Roll | ACC | doi | doR |
|---|---|---|---|
| | | | |

1) Select Operator ($\sigma$)

Chooses the subset of rows from the relation that satisfies the given cond" mentioned in the syntax of selection.

select * from student (query)

Relation $\longrightarrow$ $\sigma$ (Student)

select * from student where city = "kolkata".

Relation $\longrightarrow$ $\sigma_{city = \text{"kolkata"}}$ (Student)

select * from books where price > 500

Relation $\longrightarrow$ $\sigma_{price > 500}$ (Book)

Note 'Select' in query selects the column from the given table but the 'select' ($\sigma$) operator selects the row of the same.

2) Projection Operator ($\pi$)

Chooses the specific column from a table while excluding the other columns from a selected table.

select *roll from issue where doi > 1-Sep-2024

Relation $\longrightarrow$ $\pi_{roll}\left(\sigma_{doi > 1\text{-}Sep\text{-}2024}(issue)\right)$

select roll from student where city = "kolkata" or city = "BBSR"

Relation $\longrightarrow$ $\pi_{roll}\left(\sigma_{city = \text{"kolkata"} \vee city = \text{"BBSR"}}(Student)\right)$

3) Join Operator ($\bowtie$)

Combine the tables according to the cond$^n$.

Eg.

A

| P | Q | R |
|---|---|---|
| $P_1$ | $Q_1$ | $R_1$ |
| $P_2$ | $Q_2$ | $R_2$ |

B

| M | N |
|---|---|
| $M_1$ | $N_1$ |
| $M_2$ | $N_2$ |

5 column — A $\bowtie$ B

| P | Q | R | M | N |
|---|---|---|---|---|
| $P_1$ | $Q_1$ | $R_1$ | $M_1$ | $N_1$ |
| $P_1$ | $Q_1$ | $R_1$ | $M_2$ | $N_2$ |
| $P_2$ | $Q_2$ | $R_2$ | $M_1$ | $M_1$ |
| $P_2$ | $Q_2$ | $R_2$ | $M_2$ | $N_2$ |

6 rows

Find FN, city of student who issued books after 1 Sep 2024

A- $\pi_{FN, city, issue.roll}\left(\sigma_{doi > 1\ Sep\ 2024 \wedge student.roll = issue.roll}(Student \bowtie Issue)\right)$

29/10/24

1) Find the student roll no & books they have issued

A- select roll, ~~from~~ ~~issue~~ ACC from issue

$\pi_{roll, ACC}$ (ISSUE)

2) Find out the roll no. who's issue the book A0010.

A- select roll from issue where ACC = "A0010".

$\pi_{roll}\left(\sigma_{ACC = "A0010"}(ISSUE)\right)$

3) Find the roll no, & ACC who issue the books after 1 Sep 2024.

A- select roll, ACC from issue where doi > 1 Sep 2024

$\pi_{roll, ACC}\left(\sigma_{doi > '1\ Sep\ 2024}(ISSUE)\right)$

Page No.
Date : / /

4) Find out ACC, title of the books which author name is kanit kar.

A- select ACC, title from book where author = "kanit kar"

$$\Pi_{ACC, title}\left(\sigma_{Author = \text{"kanit kar"}}(Book)\right)$$

5) Find roll, FN who issue the book A0010.

A- select student.roll, FN from Student, issue where student.roll = issue.roll and ACC = "A0010"

$$\Pi_{student.roll,\ FN}\left(\sigma_{ACC = \text{"A0010"} \wedge\ student.roll = issue.roll}(STUDENT \times ISSUE)\right)$$

6) Find student roll & city who issue book after 1st Sep 2024.

A- select student.roll, city from student, issue where doi > 1 Sep 2024 and student.roll = issue.roll

$$\Pi_{student.roll,\ city}\left(\sigma_{student.roll = issue.roll\ \wedge\ issue.ACC = b}(Student \times Issue)\right)$$

7) Find student roll, city who issue the book of kanit kar.

A- select student.roll, city from student, issue, book where student.roll = issue.roll and issue ACC = book.ACC and author = "kanit kar"

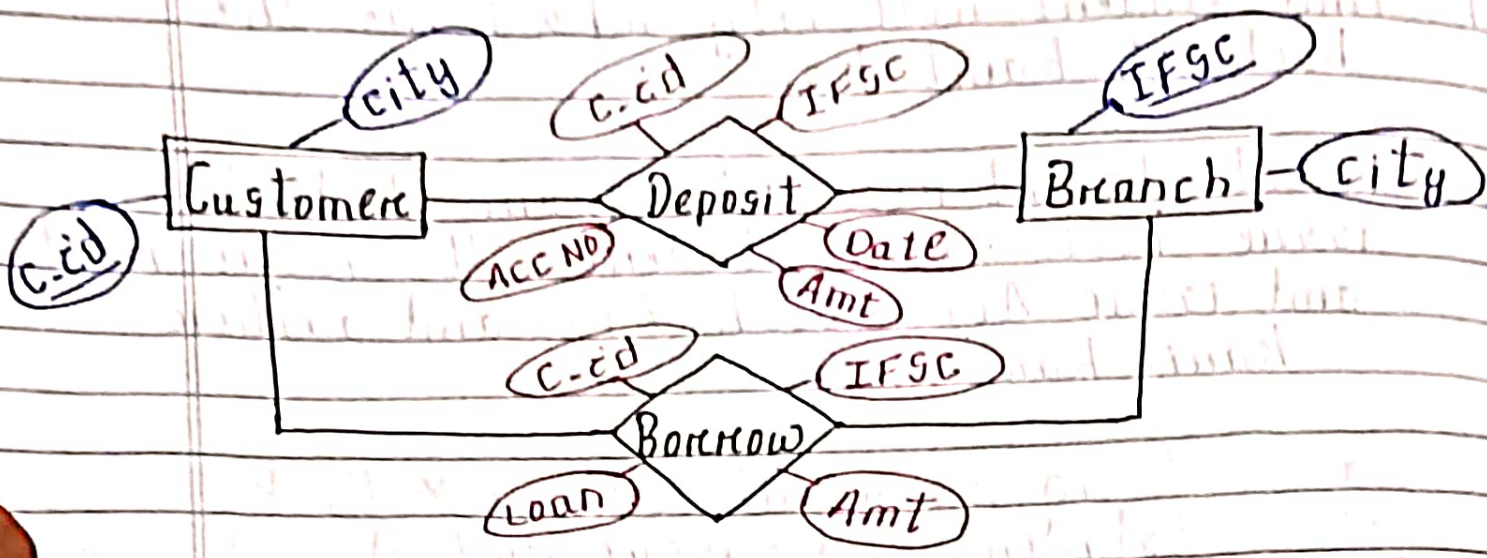
$$\Pi_{\text{student.roll, city}}\left(\sigma_{\text{student.roll} = \text{issue.roll} \wedge \text{issue.ACC} = \text{book.ACC} \wedge \text{author} = \text{"kanit kar"}}(\text{Student} \times \text{Issue} \times \text{Book})\right)$$

8) Find the author of the book which is issued by students of Calcutta & Jamshedpur.

A- select book.author from student, issue, book where student.roll = issue.roll and issue.ACC = book.ACC and city = "Calcutta" and city = "Jamshedpur"

$$\Pi_{\text{book.author}}\left(\sigma_{\text{student.roll} = \text{issue.roll} \wedge \text{issue.ACC} = \text{book.ACC} \wedge \text{city} = \text{"CALCUTTA"} \vee \text{city} = \text{"JAMSHEDPUR"}}(\text{Student} \times \text{Issue} \times \text{Book})\right)$$

01/11/24




Total Tables

1) Deposit (C-cid, IFSC, ACC NO, Date, Amt)
2) Branch (IFSC, city)
3) Customer (C-cid, City)
4) Borrow (C-cid, IFSC, loan, Amt)

1) List all data from deposit.

A- $\sigma$ (Deposit)

2) List all data from borrow where amount is greater than 4000.

A- $\sigma_{amt > 4000}$ (Borrow)

3) List the ACC no & amt from deposit

A- $\pi_{ACC\,no,\,amt}$ (deposit)

4) List the ACC No. & C-id from deposit who deposited more than 2000

A- $\Pi_{\text{ACC no., C-id}} \left( \begin{matrix} \sigma\ (\text{Deposit}) \\ \text{amt} > 2000 \end{matrix} \right)$

5) Find the branch name in city BBSR & CALCUTTA.

A- $\Pi_{\text{IFSC}} \left( \begin{matrix} \sigma\ (\text{Branch}) \\ \text{city} = \text{'BBSR'} \vee \\ \text{city} = \text{'CALCUTTA'} \end{matrix} \right)$

6) Find the customer info where customer name is Anil.

A- $\sigma\ (\text{Customer})$
$\text{C-cid} = \text{'Anil'}$

28/11/24

7) Find out all the customer name, who are living in same city where is branch city exist.

A- $\Pi_{\text{C.C\_cid}} \left( \begin{matrix} \sigma\ (\text{Customer} \times \text{Deposit} \times \text{Branch}) \\ \text{C.C-cid} = \text{d.c-cid} \wedge \\ \text{d.IFSC} = \text{b.IFSC} \wedge \\ \text{c.city} = \text{b.city} \end{matrix} \right)$

8) List customer name, amount where branch city : Mumbai.

A- $\Pi_{C.name, amt}\left(\sigma_{deposit.b\text{-}name = branch.b\text{-}name \wedge branch.city = 'MUMBAI'}(Deposit \times Branch)\right)$

select c_name, amt from Deposit, Branch where deposit.b_name = branch.b_name and branch.city = 'MUMBAI'.

9) Find out customers who have deposited but never taken loan.

A- $\Pi_{C\_cid}(Deposit) - \Pi_{C\_cid}(Borrow)$

Self Join

Find out all the emp cid who are working in same dept. where ~~Chinmay~~ Chinmay Sir is working.

A- $\Pi_{M.Empcid}\left(\sigma_{M.dept = N.dept \wedge N.Empcid = 'CM' \wedge M.Empcid != N.Empcid}(\rho_M(Emp) \times \rho_N(Emp))\right)$

Set Operations
- Union
- Intersection
- Minus

## Union

=> The union operation is used to combine data from 2 relations.

=> Denoted by union (∪) symbol.

$R_1 \cup R_2$ is a relation that includes all tuples that are either present in $R_1$ or $R_2$ or in both.

## Intersection

=> The intersection operation is used to identify the rows that are common to 2 relations.

=> Denoted by (∩) symbol.

$R_1 \cap R_2$ is a relation that includes all tuples that are present both in $R_1$ & $R_2$.

## Difference

=> The difference operation is used to identify the rows that are in one relation & not in another.

=> Denoted as (−) symbol.

$R_1 - R_2$ is a relation that includes all tuples that are in $R_1$, but not in $R_2$

Division Operator (÷)

=> The division operation creates a new relation by selecting the rows in one relation that match every row in another relation.

=> Denoted by (÷) symbol.

=> Division operator is suited to queries that include the phrase every or all as part of the cond$^n$

| Roll | Ccid |
|---|---|
| 1 | $C_2$ |
| 2 | $C_3$ |
| 3 | $C_1$ |
| 3 | $C_2$ |
| 4 | $C_3$ |
| 3 | $C_3$ |
| 3 | $C_4$ |

| Ccid | CName |
|---|---|
| $C_1$ | C |
| $C_2$ | C++ |
| $C_3$ | JAVA |
| $C_4$ | PHP |

Q- List the roll no. of the student who have registered in all the courses.

A- $\frac{\Pi_{roll,\,cid}(\sigma(Student))}{\Pi_{cid}(\sigma(course))}$

## Rename Operator

Denoted by $\rho$.

$$\sigma_{C.city = 'Mumbai'}\left(\rho_{C}(Customer)\right)$$

Q- Find out the customer name who stays in the same city where Anil stays.

A-

| CName | City |
|---|---|
| Barun | Delhi |
| Anil | BBSR |
| Ram | BBSR |

$\rho_M$

| CName | City |
|---|---|
| Barun | Delhi |
| Barun | Delhi |
| Barun | Delhi |
| Anil | BBSR |
| Anil | BBSR |
| Anil | BBSR |
| Ram | BBSR |
| Ram | BBSR |
| Ram | BBSR |

$\rho_N$

| CName | City |
|---|---|
| Barun | Delhi |
| Anil | BBSR |
| Ram | BBSR |
| Barun | Delhi |
| Anil | BBSR |
| Ram | BBSR |
| Barun | Delhi |
| Anil | BBSR |
| Ram | BBSR |

$$\Pi_{M.CName}\left(\sigma_{M.city = N.city \wedge N.Cname = 'ANIL' \wedge M.CName != N.CName}\left(\rho_{M}(Customer) \times \rho_{N}(Customer)\right)\right)$$

13/11/24

Page No.
Date / /

* Super key

=> It is a comb$^{n}$ of one or more than one attribute using which we can identify one entity among a entity set.

| Roll | Class | Sec | Name | DoB |
|---|---|---|---|---|
| 1 | 5 | A | AB | 1993 |
| 2 | 6 | B | CD | 1997 |
| 3 | 1 | A | EF | 1998 |
| 4 | 5 | B | GH | 1997 |

$S_K$ = {Roll, {Roll, Class, Sec}, {Roll, Class}
{Roll, Class, Sec, Name, DoB}

* Candidate key

Minimal subset of a super key.

$C_K$ = {Roll, Class, Sec} is the subset of Super key
$C_K$ = {Roll}
$C_K$ = {Name}

"ALL THE CANDIDATE KEY ARE SUPER KEY BUT ALL SUPER KEY ARE NOT SUPER KEY"

14/11/24

# FUNCTIONAL DEPENDENCY AND NORMALISATION

=> A functional dependency occurs when one attribute uniquely ~~identifies~~ determines another attribute within a relation.

=> Describes how attributes in a table relate to each other.

A → B (A determines B)

| Roll | Name | Marks | Dept | Course |
|---|---|---|---|---|
| 1 | a | 78 | CS | C1 |
| 2 | b | 60 | EE | C1 |
| 3 | a | 78 | CS | C2 |
| 4 | b | 60 | EE | C3 |
| 5 | c | 80 | IT | C3 |
| 6 | d | 80 | EC | C2 |

Roll → Name (Roll determines Name) ✓
Name → Roll (Name is not unique) ✗
Roll → Marks ✓
Dept → Course ✗
Marks → Dept ✗
Roll, No → Marks ✓

* Types of Functional Dependency

1) Trivial FD

=> A dependent is always a subset of the determinant.

i.e. If $X \rightarrow Y$ and Y is subset of X then it is called Trivial FD.

Eg. $\underbrace{\{Roll, Name\}}_{X} \rightarrow \underbrace{Name}_{Y}$

The dependent is a subset of the determinant set.

Eg. Roll → Roll

2) Non-Trivial FD

=> The dependent is strictly not a subset of the determinant.

i.e. If $X \rightarrow Y$ and Y is not a subset of X then it is called non-Trivial FD.

Eg. $\underbrace{\{Roll, Name\}}_{X} \rightarrow \underbrace{Marks}_{Y}$

The dependent is not subset of the determinant set.

21/11/24

Page No.
Date: / /

## * Armstrong Axioms (Inference Rules)

Primary

### 1) Reflexivity

If $A \rightarrow B$ Then $A \rightarrow A$.

### 2) Transitivity

If $A \rightarrow B$ and $B \rightarrow C$
Then $A \rightarrow C$

### 3) Augmentation

If $X \rightarrow Y$ and $A$ is an attribute
Then $XA \rightarrow YA$

Secondary

### 4) Union

If $X \rightarrow Y$ and $X \rightarrow Z$
Then $X \rightarrow YZ$

### 5) Decomposition / Split

If $X \rightarrow YZ$ Then $X \rightarrow Y$ and $X \rightarrow Z$.
(Vice-versa is not true)

### 6) Pseudo transitivity

If $X \rightarrow Y$ and $YZ \rightarrow A$
Then $XZ \rightarrow A$

7) Composition

If $X \rightarrow Y$ and $A \rightarrow B$
Then $XA \rightarrow YB$

Eg. R(ABCDE)

$A \rightarrow B$ , $B \rightarrow D$ , $E \rightarrow A$

$SK = (ABCDE)^+$
$(ACE)^+$
↓
CK

$(AB)^+ = \{A, B, D\}$ SK X

$(CD)^+ = \{C, D\}$ SK X

$(ABCD)^+ = \{A, B, C, D\}$ SK X

$(CE)^+ = \{C, E, A, B, D\}$ SK ✓

↓
CK / C E _ _ _

$2^3$ SK

How to ~~Method~~ Find Candidate key ?

Page No :
Date : / /

R(ABCDEF)
A → B  C → D  D → E

**Rule 1** Attribute not present in the right hand side in this dependency.
ACF is not present in RHS
So, essential attribute is A, C, F.

**Rule 2** Remaining attribute are non-essential.
B, D, E is non-essential.

**Case ①** If All Essential Attribute Can Determine All Remaining Non-Essential Then Comb$^N$ Of Essential = Ck.
$(ACF)^+ = \{A, B, C, D, E, F\}$
↓
Ck

**Case ②** If Essential Attribute Cannot Be Determined The Set Of Non-Essential Then Comb$^N$ Of Essential & Particular Non-Essential = Ck.

Page No.
Date: / /

Q- R(ABCDE)
AB → C  C → D  B → E

A- Essential → {A, B}

$(AB)^+$ = {A, B, C, D, E}
↓
CK

5/11/24

Q- R(ABCDE)
A → B  B → C  C → D  D → E

A- Essential → {A}

CK → $A^+$ = {A, B, C, D, E}
↓
SK

Q- R(ABCDEFG)
A → BC  BC → DE  D → F  CF → G

A- Essential → {A}

SK → $A^+$ = {A, B, C, D, E, F, G}
↑
SK  $2^6$ → SK

Q- R(ABCD)
AB → CD  D → A

Page No.
Date: / /

A- Essential → {B}

$B^+$ = {B} Not a super key

$C_K$ → $(AB)^+$ = {A, B, C, D}

↓

$S_K$

Prime attribute = {A, B}

$C_K$ → $(DB)^+$ = {B, D, A, C}

↓

$S_K$

Prime attribute = {A, D, B}

Q- R (A B C D E)

A → B D → E

A- Essential → {A, C, D}

$C_K$ → $(ACD)^+$ = {A, C, D, B, E}

↓

$S_K$

Prime attribute = {A, C, D} One $C_K$

Q- R (A B C D)

A → B B → C C → A

A- Essential → {D}

$D^+$ = {D} Not super key

$C_K \rightarrow (AD)^+ = \{A, D, B, C\}$

↓

$S_K$

Prime Attribute = {A D}

$C_K \rightarrow (CD)^+ = \{C, D, A, B\}$

↓

$S_K$

Prime Attribute = {A D C}

$C_K \rightarrow (BD)^+ = \{B, D, C, A\}$

↓

$S_K$

Prime Attribute = {A D C B}

Q- R (A B C D)

$AB \rightarrow CD \quad D \rightarrow B \quad C \rightarrow A$

A- Essential → X

$ABCD^+ \rightarrow S_K$

$C_K \rightarrow AB^+ = \{A, B, C, D\}$

↓

$S_K$

$C_K \rightarrow CD^+ = \{C, D, A, B\}$

↓

$S_K$

~~$CB^+ = \{C, B, D, A\}$~~

$C_K \rightarrow AD^+ = \{A, D, B, C\}$

↓

$S_K$

Page No.
Date: / /

$C_K \rightarrow DC^+ = \{D, C, B, A\}$

$\downarrow$

$S_K$

Primder Attribute = {A, B, C, D}

## NORMALIZATION

| Roll | SName | Credits | Dept-name | Building | Room |
|---|---|---|---|---|---|
| 1 | Rahul | 6 | CSE | C-15 | 11 |
| 2 | Amit | 7 | CSE | C-15 | 11 |
| 3 | Bipasha | 8 | IT | C-14 | 10 |
| 4 | Rupa | 6 | IT | C-14 | 10 |
| 5 | Aniket | 6 | EEE | C-13 | 11 |

Normalization is a process of organizing the data in database.

It is used to minimize the redundancy from a relation or set of relations.

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

## WHY DO WE NEED NORMALIZATION?

The main reason for normalizing the relations is removing certain anomalies. Failure to eliminate anomalies leads to data redundancy & can cause data integrity.

## Data modification Anomalies

i) Insertion Anomaly :- It refers to when one cannot insert a new tuple into a relationship due to lack of data.

ii) Deletion Anomaly :- It refers to a situation where the deletion of data results in the unintended loss of some other important data.

iii) Updation Anomaly :- The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

## Types of Normal Forms

6 types :- i) 1 NF
ii) 2 NF
iii) 3 NF
iv) BCNF
v) 4 NF
vi) 5 NF

Page No.
Date: / /

1) First Normal Form (1NF)

A relation is in 1NF if every attribute in that relation is single valued attribute.

Rules :-

① Atomic values for each table.

② Attribute type should be same.

③ No ordering of tuples or columns.

④ No duplicate records are there.

2) Second Normal Form (2NF)

Rules :-

① It must be in 1NF.

② No partial dependency in relation R.

NOTE A functional dependency $X \rightarrow Y$ is said to be in partial dependency, if Y can be determined by any proper subset of X.

* In 2NF, every non-prime attribute must not partially dependent on each CK of the table.

SUBSET OF $C_K$ ⟶ Non-prime

2NF ✗✗

Page No.
Date: / /

R(A B C D E F)

A → B  B → C  C → D  D → E

Essential → {A, F}

$C_K$

$S_K$ → $(AF)^+$ = {A, F, B, C, D, E}

$A^+$  $F^+$ {F} ✗

{A, B, C, D, E} ✗

Prime → {A F}

Non-Prime → {B C D E}

A → B

∵ As subset of $C_K$ can determine a non-prime attribute.

Hence, it is not 2NF.

3) Third Normal Form (3NF)

⇒ The relational schema R in 3NF if

i) FD is non-trivial ~~and X → A~~

ii) X → A  X is super key OR A is prime attribute

Page No.
Date: / /

4) BCNF (Boyce Codd Normal Form)

→ Every relation is in BCNF if :

i) It is in 3NF.

ii) X → A (X is a super key)

Q- R (A B C D)
AB → C AB → D C → A B → D

A-

| RHS | Not in RHS |
|---|---|
| A C D | B |

$B^+ = \{B, D\}$ → $S_K$ ✗✗

$C_K$ → $(AB)^+ = \{A, B, C, D\}$
↓
$S_K$

Prime attribute = {A, B}

$C_K$ → $(CB)^+ = \{C, B, A, D\}$
↓
$S_K$

Prime = {A, B, C}
Non-prime = {D}

AB → C BCNF (∵ AB is $S_K$)

AB → D BCNF (∵ AB is $S_K$)

$C \rightarrow A$ 3NF (∵ A is prime)

$B \rightarrow D$ 1NF (∵ D is non-prime

∴ The relation R is in 1NF

Q- R(ABCD)
AB → CD  D → A
$C_K$ AB  DB

A- Prime = {A, B, D}
Non-prime = {C}

AB → CD  BCNF
D → A  3NF

∴ The relation R is in 3NF

Q- R(ABCDE)
A → B  BC → E  ED → A

A-

| RHS | Not in RHS |
|---|---|
| B, E, A | CD |

$(CD)^+ = \{C, D\}$ $S_K$ ✗✗

$C_K \rightarrow (ACD)^+ = \{A, C, D, B, E\}$

$C_K \rightarrow (ECD)^+ = \{B, C, D, E, A\}$

$K \rightarrow (BCD)^+ = \{B, C, D, E, A\}$

Page No.
Date: / /

Prime = { A, B, C, E }
~~Non prime = { B }~~

* If all the attributes are prime, then the relation is in 3NF.

Q- R(A B C)
A → B  B → A  B → C

A- (ABC)⁺ = { A, B, C },

A⁺ = { A, B, C } CK

B⁺ = { A, B, C } CK

| | |
|---|---|
| A → B | BCNF |
| B → A | BCNF |
| B → C | BCNF |

∴ The relation R is in BCNF.

Q- R(A B C D E F G)
AB → C  AC → B  AB → E
B → D  BC → A  E → F

R₁ (ABC)  R₂ (ABCD)  R₃ (ABCEF)

R(ABC)

$AB \rightarrow C$ $AC \rightarrow B$ $BC \rightarrow A$

| RHS | Not in RHS |
|---|---|
| AB | C |

$\{C\}^+ = \{C\}$

$\{AC\}^+ = \{A, C, B\}$

| RHS | Not in RHS |
|---|---|
| A, B, C | X |

$\{ABC\}^+ = \{A, B, C\}$ SK

$\{AB\}^+ = \{A, B, C\}$ SK

$\{AC\}^+ = \{A, C, B\}$ SK

$\{BC\}^+ = \{B, C, A\}$ SK

Prime $\rightarrow \{A, B, C\}$

$\boxed{AB \rightarrow C}$ BCNF

$\boxed{AC \rightarrow B}$ BCNF

$\boxed{BC \rightarrow A}$ BCNF

∴ The relation R is in BCNF

Page No.
Date: / /

# DECOMPOSITION

The process of breaking up or dividing a single relation into 2 or more sub relation. is called Decomposition.

It is of 2 types :-
i) Lossless
ii) Dependency preserving.

## i) Lossless

It is a decomposition of relation R into relations $R_1$ & $R_2$ such that, if we perform natural join of $R_1$ & $R_2$, it will return the original relation R.

$$R \Leftrightarrow R_1 \times R_2$$

| a | b | c |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 5 | 3 |
| 3 | 3 | 3 |

| A | B |
|---|---|
| 1 | 2 |
| 2 | 5 |
| 3 | 3 |

| B | C |
|---|---|
| 2 | 1 |
| 5 | 3 |
| 3 | 3 |

## How to Detect if R is lossy OR lossless

RULE 1 Union of both the sub relations containing all attributes that are present in parent relation R.

R(ABC)

$R_1$ (AB) $R_2$ (BC)

$R_1 \cup R_2$ = ABC => R

Rule 2 Intersection of $R_1$ & $R_2$ sub relations cannot be null. $\{\Phi\}$

Rule 3 $R_1 \cap R_2 = \{val\}$
↳ Must be $\{K$ of $R_1$ or $R_2$ or $R_1\}$

Q- $R(ABCD)$
$A \to B$ $C \to D$

$R_1(AB)$ $R_2(CD)$
Rule 1 ✓
Rule 2 ✗
∴ Lossy

Q- $R(ABCD)$
$R_1$
$R_1(ABC)$ $R_2(BD)$
$R_3$ $R_4$
$(AB)$ $(BC)$
$A \to B$ $B \to C$ $C \to D$ $D \to B$

$R_1 \cup R_2$ ✓
$R_1 \cap R_2 = \{B\}$ ✓

$B^+\} = \{B, C, D\}$

∴ This is lossless.

Page No.
Date: / /

## Lossy Decomposition

⇒ It is a decomposition of relation R into relation $R_1$ & $R_2$ such that if we perform natural join of $R_1$ & $R_2$, it will not return the original relation R but also some additional information with it.

R (A B C)

R(AC) R(BC)

original R

| a | b | c |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 5 | 3 |
| 3 | 3 | 3 |

AC

| a | c |
|---|---|
| 1 | 1 |
| 2 | 3 |
| 3 | 3 |

BC

| b | c |
|---|---|
| 2 | 1 |
| 5 | 3 |
| 3 | 3 |

AC X BC

| a | b | c |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 5 | 3 |
| 2 | 3 | 3 |
| 3 | 3 | 3 |
| 3 | 5 | 3 |

original ≠ AC X BC

Page No.
Date: / /

$R_2(ABCD)$

$AB \to C$ $AC \to B$ $B \to D$ $BC \to A$

A- $(AB)^+ \to \{A, B, C, D\}$ SK

$(BC)^+ \to \{B, C, A, D\}$ SK

$(AC)^+ \to \{A, C, B, D\}$ SK

Prime = {A, B, C}

Non-prime = D

$\boxed{AB \to C}$ BCNF

$\boxed{AC \to B}$ BCNF

$\boxed{B \to D}$ 3NF

$\boxed{BC \to A}$ BCNF

∴ The relation $R_2$ is 3NF.

12/12/24

# TRANSACTION AND CONCURRENCY CONTROL

Page No.
Date: / /

Transactions (Tx) is a set of operations used to perform a logical unit of work.

Money transfer from A to B

| | | | |
|---|---|---|---|
| Buffer | 1000 | READ (A) | TRANSACTION |
| | 950 | A = A - 50 | |
| | 500 | READ (B) | |
| | 550 | B = B + 50 | |
| RAM | 950 | WRITE (A) | |
| | 550 | WRITE (B) | |
| (Write in DataBase) | | COMMIT | |

READ : It is used to read the value of ~~the~~ X from the database & store it in a buffer in the main memory for further actions.

WRITE : It is used to write the value to the database from the buffer in the main memory. For a write operation to be performed, first a read operation is performed to bring its value in buffer.

COMMIT : It is used to save all changes made during the transaction permanently to the database.

ROLLBACK : It is used to undo all changes made during the transaction.

* ACID Properties of Transaction

ACID
- ATOMICITY
- CONSISTENCY
- ISOLATION
- DURABILITY

1) Atomicity -

- Do all operations & COMMIT Else ROLLBACK.
- FAIL transactions cannot be resumed. Just RESTART
- A transaction is an indivisible unit. The recovery subsystem is handling atomicity

2) Consistency

- Before Tx start & after the Tx completed, sum of amount will be saved.

- Consistency maintained by software in DBMS or application developers.

3) Isolation

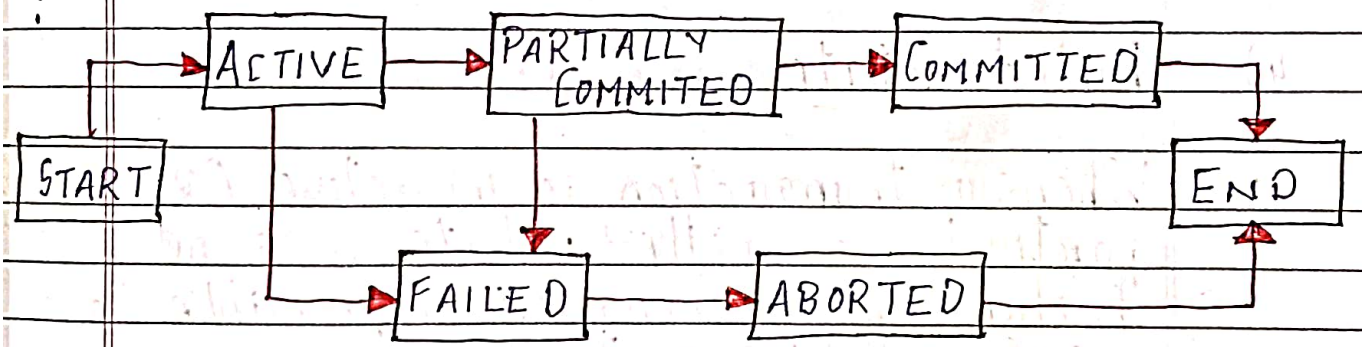
- It is the responsibility of the concurrency control subsystem to ensure isolation where each Tr can execute independently so that partial effect of each Tr should not be visible to other Tr.

4) Durability

- It is the responsibility of the recovery manager to record permanently the effect of successfully completed Tr & must not be lost due to subsequent failure.

05/12/24

## Transaction Life Cycle



Page No.
Date: / /

1) Active State

=> 1st state of transaction life cycle.

=> A transaction is called in an active state as long as the transactions are getting executed.

=> All changes are stored in buffer

2) Partially Committed State

=> After the execution of instructions when it is stored in intermediate buffer then it is in partially committed state.

3) Committed State

=> All changes are stored in database permanently from the buffer.

4) Failed State

=> When a transaction is in active OR partially committed state & some failure may occur & impossible to continue the execution

Page No.
Date: / /

5) Aborted State

=> After the transaction is in failed state all changes are made by transaction to be undone. OR Rollback.

=> After rollback completely transactions enter into aborted state.

6) Terminated State

=> Last state of transaction life cycle.

=> After committed or aborted transaction will be end of life cycle OR terminated state.

## Concurrency Problem

When multiple transactions execute concurrently in an uncontrol OR unrestricted manner then there might be several problems :-

i) Dirty Read
ii) Unrepeatable Read
iii) Lost Update
iv) Phantom Read

ii) Dirty Read

| $T_1$ | $T_2$ | |
|---|---|---|
| R(A) | | |
| | R(A) | 1000 |
| | A = A-50 | 950 |
| | W(A) | 950 |
| ROLL BACK | Commit | |
| $T_1$ FAIL | | |

=> There is always a chance that uncommitted transaction might ROLLBACK.

=> Uncommitted transaction might make another transaction read a value that doesn't even exist.

=> Lead inconsistency in DBMS.

ii) Unrepeatable Read

| $T_1$ | | $T_2$ |
|---|---|---|
| 1000 | R(X) | |
| 900 | A = A-100 | R(X) - 1000 |
| 900 | W(A) | |
| | | R(X) - 900 |

This problem occurs when a transaction read diff. value even it is not updated.

iii) Lost Update

| | $T_1$ | $T_2$ | |
|---|---|---|---|
| 1000 | R(A) | | |
| | | R(A) | 950 |
| 950 | A = A - 50 | | |
| | | A = A - 100 | 850 |
| 950 | W(A) | | |
| | | W(A) | 850 |
| | COMMIT | | |
| | | COMMIT | |

In concurrent transaction multiple transaction executed concurrently & updates from one or more transactions getting lost.

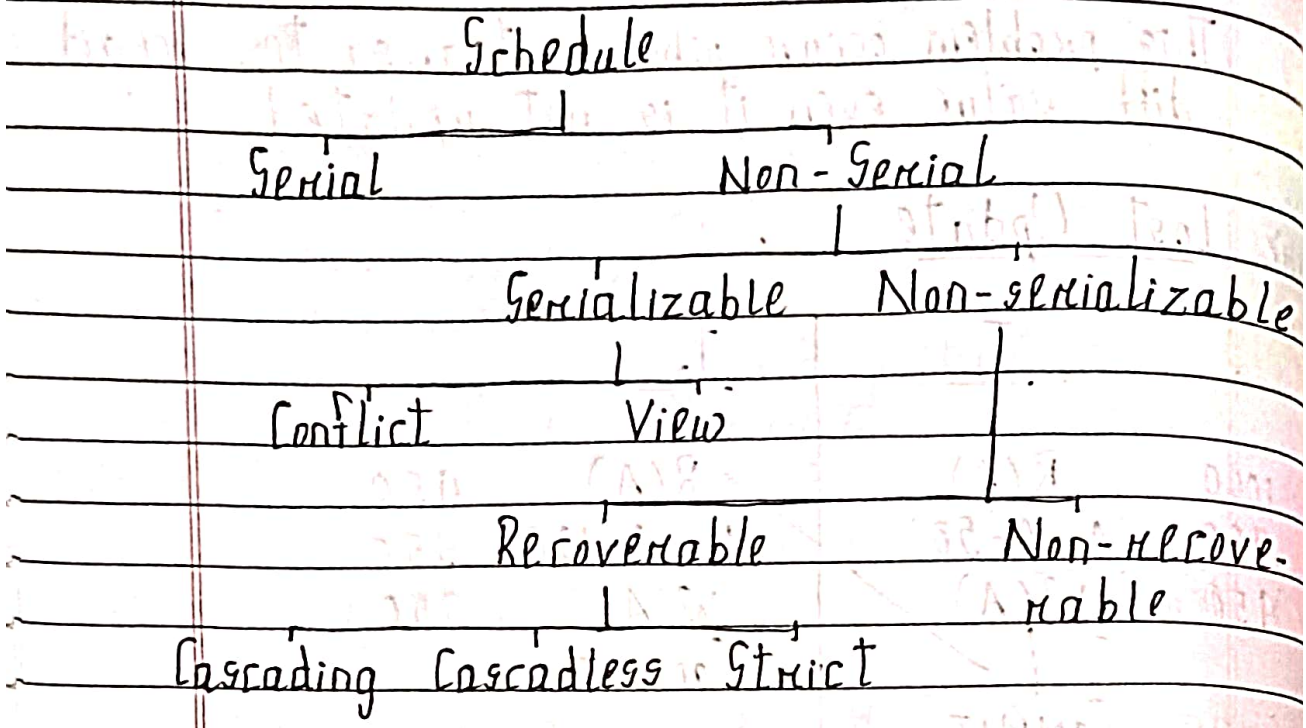
iv) Phantom Read

| $T_1$ | $T_2$ |
|---|---|
| R(A) | |
| | R(A) |
| delete | |
| | A = A - 50 |

The problem occurs when a transaction read some variable from the buffer & the same variables are updated from other Trs which doesn't exist.

06/12/24

Page No.
Date: / /




## Schedule

A sequence of operations by a set of concurrent transactions that preserves the order of operations in each of the individual ~~oper~~ transactions.

It is of 2 types :-
i) Serial
ii) Non-serial

i) ## Serial Schedule

A schedule where the operations of each transactions are executed concurrently without any interleived operations from other transactions

Advantages

Consistent
Recovarable

Disadvantages

Waiting Time More.
Efficiency Less.

ii) Non-Serial Schedule

A non-serial schedule perform the operations concurrently in an inter-leaved manner.

Disadvantages :- i) Not always consistent & recovarable

Advantages :- i) Efficiency more
ii) Waiting Time less

It is further divided into 2 types :-
a) Serializable
b) Non-Serializable

a) Serializable

The objective of serializability is to find non-serial schedule that allows transaction to execute concurrently

without interfering one another & thereby produce a database state that could be produce by a serial execution.
That means a non-serial sche is serializable & correct if it produces same result as a serial execution.

It is of 2 types :-
i) Conflict
ii) View

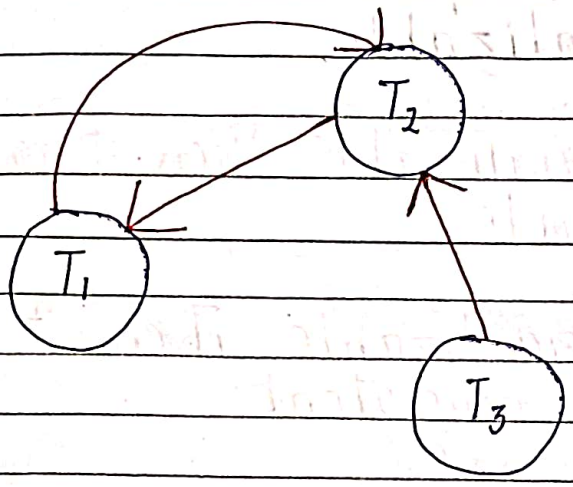
i) Conflict

If a given non-serial schedule can be converted into a serial schedule by swapping its non-conflicting operations then it is called conflict serializable schedule

Both the operations belong to different transactions & both are on the same data.

S → $R_1(A)$ $R_2(A)$ $R_1(B)$ $R_2(B)$ $R_3(B)$ $W_1(A)$ $W_2(B)$

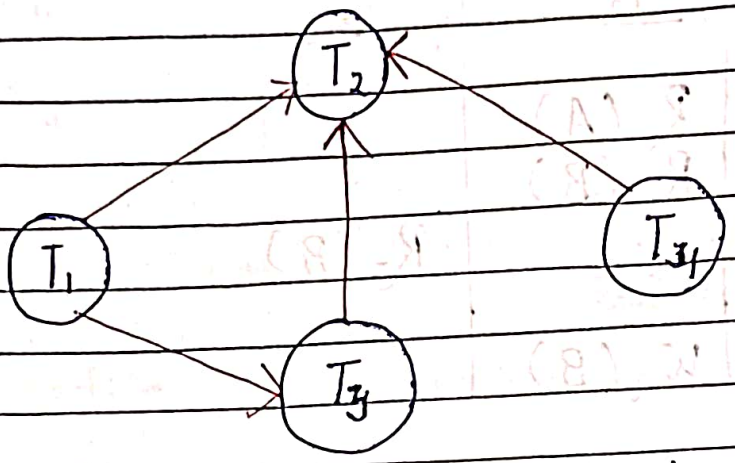
| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| $R_1(A)$ | $R_2(A)$ | |
| $R_1(B)$ | $R_2(B)$ | |
| | | $R_3(B)$ |
| $W_1(A)$ | | |
| | $W_2(B)$ | |




There is a cycle is present. So schedule S is not conflict serializable.

09/12/24

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| | R(A) | | R(A) |
| | | R(A) | |
| W(B) | | | |
| | W(A) | | |
| | | R(B) | |
| | W(B) | | |




There is no cycle present. So, it is conflict serializable.

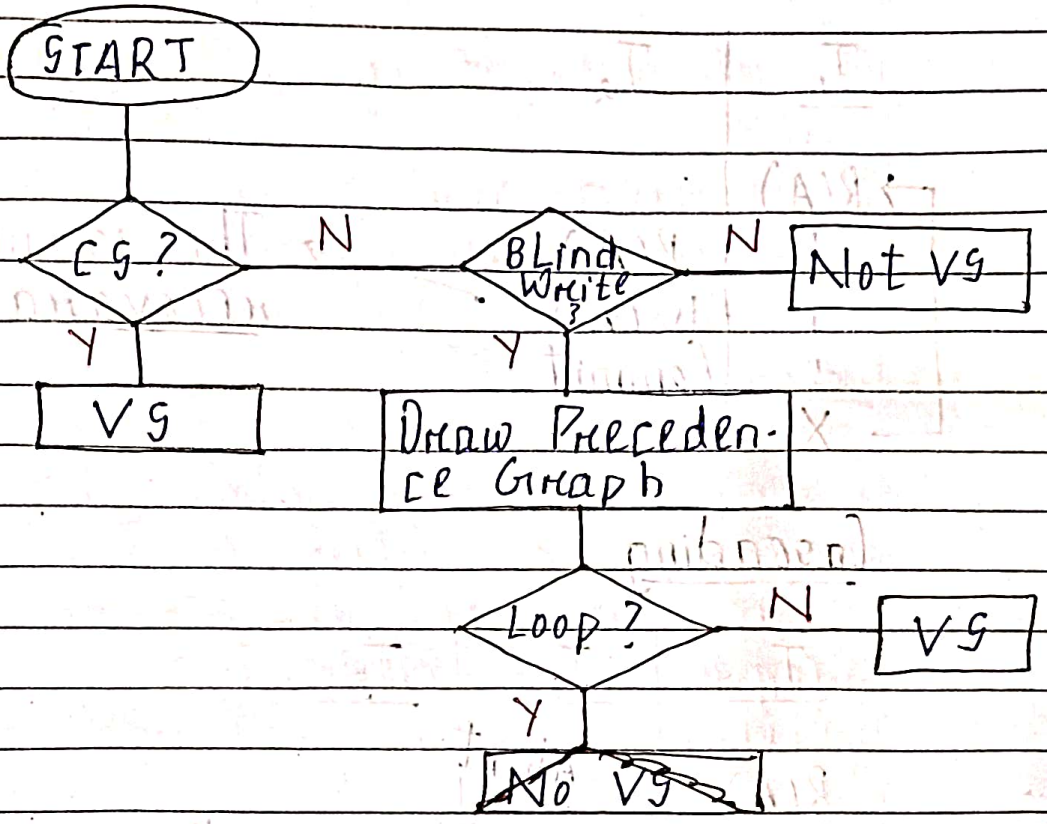
Note | If conflict serializable then it is in view Serializable.

If conflict serializable then it is Recoverable & consistent.

iii) View

If a given schedule is found to be view equivalent to some serial schedule then it is called view serializable.

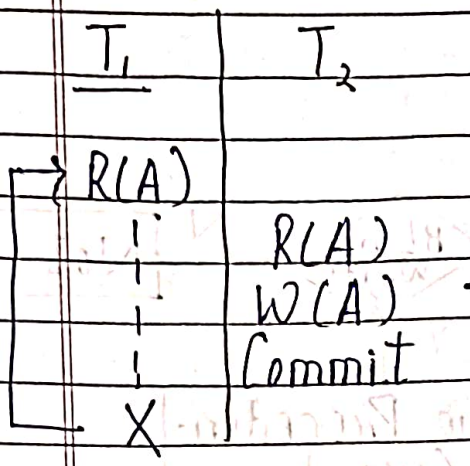
ALL CONFLICT SERIALIZABLE ARE VIEW SERIALIZABLE BUT ALL VIEW SERIALIZABLE ARE NOT CONFLICT SERIALIZABLE.




A transaction is said to be blind write when the user directly write the value from buffer without reading a value from database.

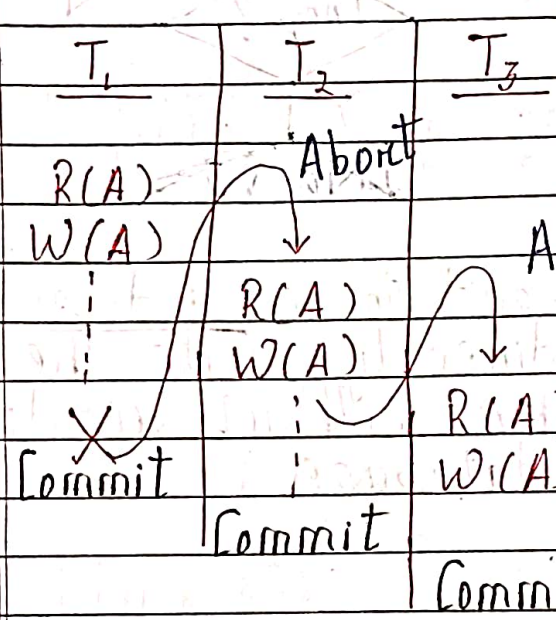
| $T_1$ | $T_2$ |
|---|---|
| R(A) | |
| W(A) | |
| | R(A) |
| | W(A) |
| Commit | |
| | Commit |

This is recovarable as the 1st transaction occurs first & committed first.




→ This is not recoverable

## Cascading




If in a schedule, failure of one transaction causes several other dependent transaction to roll back or abort. Then, such a schedule is called cascading schedule. (Cascading ROLLBACK)

Page No.
Date: / /

## Cascadless

If in a schedule, a transaction is not allowed to read a data item until last transaction has written is committed or aborted schedule then it is called cascadless schedule.

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| R(A) | | W(A) |
| W(A) | | |
| Commit | | |
| | R(A) | |
| | W(A) | |
| | Commit | |
| | | R(A) |
| | | W(A) |
| | | Commit |

## Strict

If in a schedule a transaction is neither allowed to read or write a data item until the last transaction that has written is committed or aborted.

It doesn't support blind write.

11/12/24

Page No.
Date: / /

## Objective of Concurrency Control

1) Serializability
2) Recoverability

## Locking

A procedure used to control concurrent access to data when one transaction is accessing the database. A lock may deny access to other transactions prevent incorrect results or inconsistent data.

Lock may be 2 types :- (S/W)

a) Shared [S(A)]
b) Exclusive [X(A)]

## Shared

If a transaction has a shared lock on a data it can read a data but it cannot write or update the data. Only read operation is possible.

## Exclusive

If a transaction has an exclusive lock on a data, it can read & update the data.

| $T_2$ \ $T_1$ | Unlock | Shared | Exclusive |
|---|---|---|---|
| Unlock | | ✓ | ✓ |
| Shared | ✓ | ✓ | X |
| Exclusive | ✓ | X | X |

R R ✓
R W X
W R X
W W X

## Advantages of Locking

Data consistency :- Lock can help to ensure it by preventing multiple users from modifying the same data simultaneously.

By controlling access to shared resource locking can help to prevent data conflict.

Isolation :- locking can ensure the transaction are executed in isolation from other transaction.

## 2 Phase locking Protocol

According to this rule, every transaction are divided into 2 phase :-

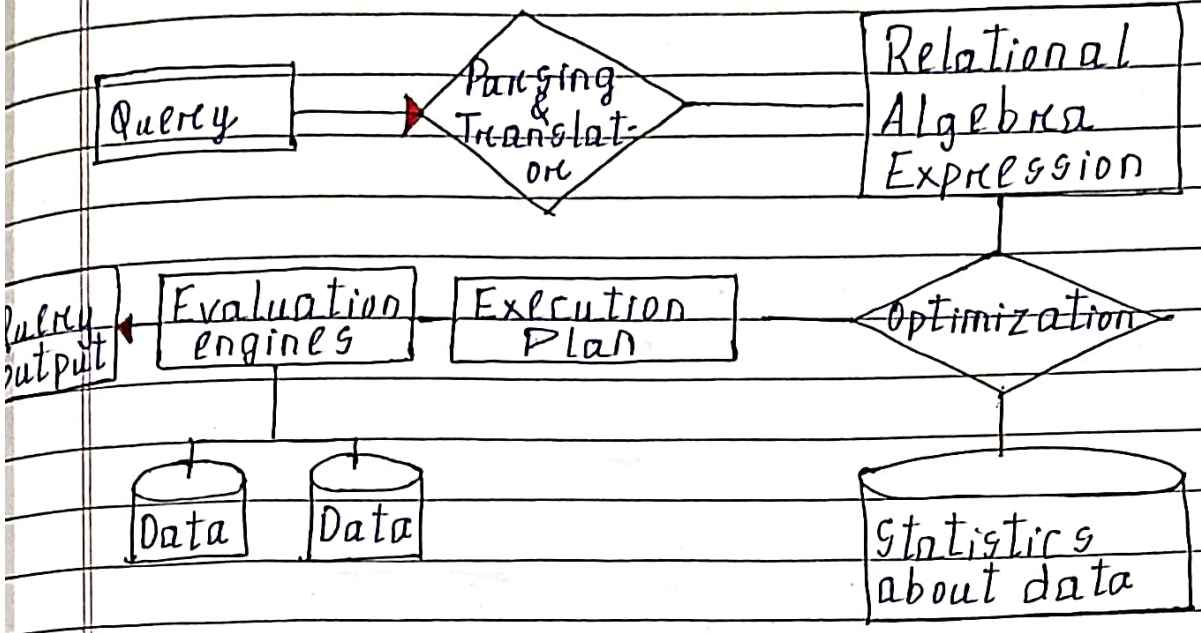
i) Growing Phase :- In this phase Tx acquire all the locks needed but cannot release any lock.

Page No.
Date: / /

Shrinking Phase :- In this phase, it releases all the locks but cannot acquire any new lock.

12/12/24

# Query Processing And Optimization




select salary from employee
where salary > 75k

$$\pi_{sal}\left(\sigma_{sal>75k}(emp)\right)$$