PAGE No.
DATE / /

QB - I

(1) What is data abstraction and data independence

Ans:

(1) Data abstraction: Data abstraction refers to the process of hiding the complex details of data storage and presenting only the essential features to the users.

It helps simplify the interaction b/w users and the database system by focusing on the high-level structure and functionalities rather than low-level implementation details.

- Data abstraction access in three levels.

(1) physical or internal level: describe how the data is stored in the database
eg: File storage, indexing
This level is hidden from the user

(2) Logical or conceptual level: defines what data is stored in the database.
and relationship b/w those data.
This level is typically used by DBMS developers and database administrators.

(3) View or external level: This level represents how users interact with data. It provides a specific view data relavent to individual users or user groups (eg: a subset of the data)

PAGE No.
DATE / /

2. Data Independence:

data independences refers to the ability to change the schema at one level of the database without affecting the schema at higher levels.

In short, it ensures that changes in data representation or structure do not affect the overall database functionality and application that uses the data.

- There are two types of data independences:

(1) physical data independences: The capacity to change the physical storage structures or devices (eg. disk format) without affecting the logical schema

This is considered more difficult to achieve.

(2) Logical data independences: The ability to change the logical schema without affecting the external schema or applications that access the data.

This is more difficult to achieve than physical data independences, but it offers better flexibility for the users and administrators.

PAGE No.
DATE / /

Q) explain types of user in database management system?

Ans:

A database User is defined as a person who interact with data daily, updating, reading, and modifying the given data.

In DBMS, different types of users interact with the database based on their roles and responsibilities.

- These users can be categorized as follows:

(1) Database Administrator (DBA)

- The DBA is responsible for managing and maintaining the database.
- task includes:
  - database design and creation
  - User account management
  - Backup and recovery
  - ensuring security and integrity of the DB

(2) Applications programmers:

- These users develops application programs that interact with DB
- They write SQL queries, procedures, and functions to retrieve, insert, update, or delete data.

(3) End Users (Final Users who interact with the DB)

- end users are those who directly use application or interfaces to retrieve, or modify data.

(1) Casual Users:

Occasionally access the database using query language.

eg: A manager who checks reports

(2) Naive Users:

Regular user who interact with application

eg: Bank customers using an ATM

(3) Sophisticated Users:

Have advanced knowledge of DB queries - use analytical tools, reporting tools, SQL queries to fetch specific data.

eg: Data scientists

(4) Testers:

The database applications for bugs, performance issue and security vulnerabilities.

- validate data integrity and functionality

(5) External Users:

- Users who access the database remotely through API or web services

(Q3) define DBA, discussed role and responsibilities

Ans:

- Database Administrater (DBA):

- The database administrater is responsible for the overall planning of company's data resources, for the design data, and for the day to day operational aspects of data management

- A database administrater is a person responsible for the installation, configuration, up-gradition, maintances and monitoring database in an organization.

- The DBA is crucial in handling the technical aspects of the database while ensuring that users can access and use it effectively.

- Roles of database administrater:

(1) The DBA needs to performs many roles to keep the dB up and running
(2) system administrater / designer
(3) DBA need to manage DBMS software and server
(4) DBA also responsible for deciding on the storage and access methods
(5) DBA performs all data fields updates or adding new fields into dB
(6) Database Developer / programmer
(7) System Analyst
(8) DBA needs to take care of system crashes by planning proper recovery procedures.

- Responsibilities of DBA

(1) Designing overall dB Schema
(2) selecting and installing db software and hardware
(3) Designing Recovery procedures
(4) The DBA select suitable DBMS software like oracle, SQL Server or MySql
(5) Designing Authorization / Access control
(6) operation management
(7) investigation of errors found in data
(8) initiation and control of all periodic dump data
(9) in order to take care of crashes DBA needs to design the system recovery procedures and also specifying techniques for monitoring database performances.

Q) explain the concept of data independence and give diff b/w logical and physical data independence

(for this ques follow Q1)

Ans:

S	physical data independence	Logical data independence
(1)	it mainly concern about how data is stored in the system	it mainly concerned about the structure or the changing data defines.
(2)	it is easy to retrieves	it is difficult to retrieves because the data is mainly dependent on the logical structure data.
(3)	It is concerned with the internal schema	it is concerned with the conceptual schema.
(4)	Any change at the physical level, doesn't require to change at the application level	the change in the logical level requires a change at the application level.
(5)	Storage structure, indexing, compression, file organization	Table structure, relationships, attributes, constraints.
(6)	As compared to the logical independences it is easy to achieve physical data independences.	As compared to the physical independences it is not easy to achieve logical data independences

example: change in compression techniques, hashing algorithm and storage devices

example: Add / modify or delete a new attributes.

Q) define entity, types of entity with example.

Ans:

- entity:

- entity is anything in real world which may have physical or logical existence.

- ~~An entity is anything in real world with its physical existence.~~

example: Student, Faculty, subject having independent physical existence

- entity set:

entity set is collection of entities of same types

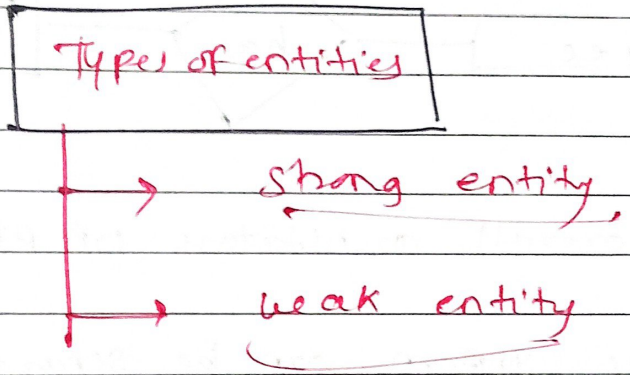
example: Student entity set contain all Students in college database.

- entity types:

- entity set is collection of entities with same attributes

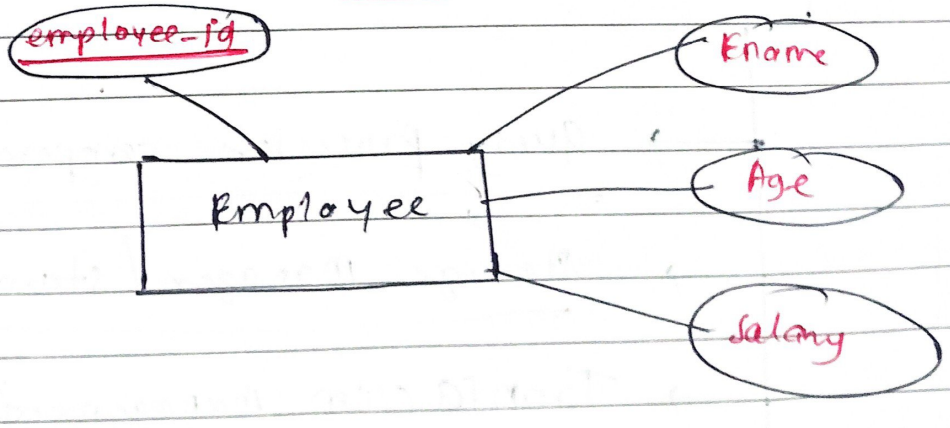
- As in student table, each row is an entity and have same attributes
  In other words we can say a student table is an entity type:

- The types of entities are:



(a) Strong entity: entity type which has its own key attributes by which we can identify specific entity uniquely is called as strong entity type.

Example:

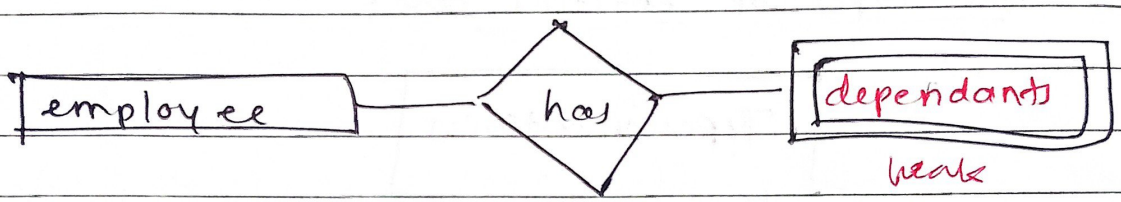


Strong entity type is represented by single rectangle.

(b) Weak entity type: entity type which cannot have distinct key from its attributes and takes help from corresponding strong entity is called weak entity type.

- Weak entity is represented by double rectangle

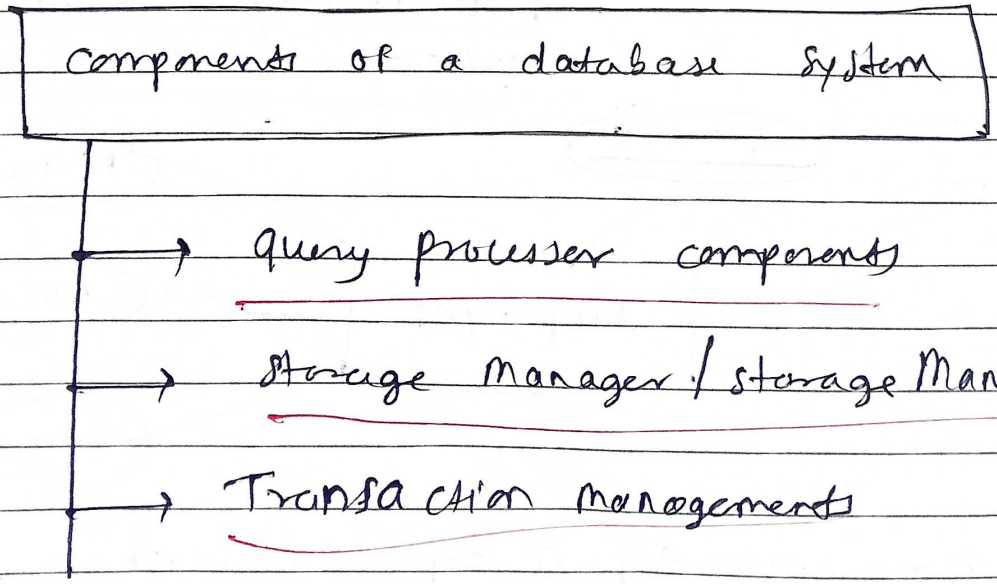
example:



(9) describe overall architecture of DBMS

Ans:

A database system can be separated into two different modules that deal with all operation of the overall system.

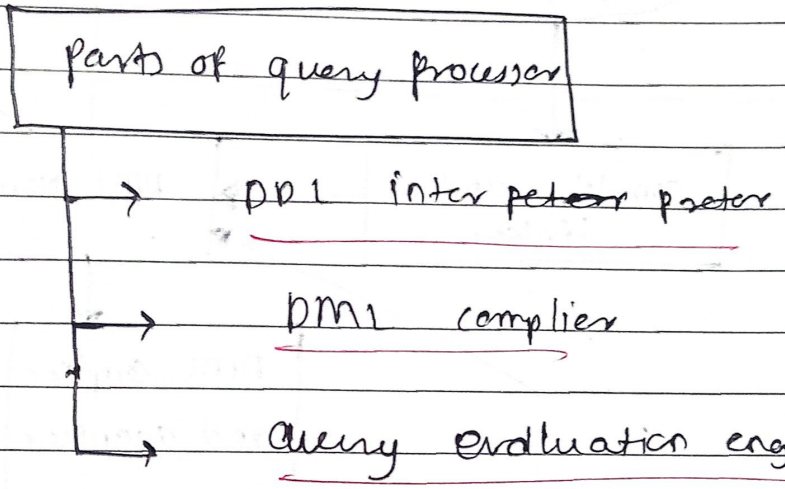


PAGE No.
DATE / /

1) query processor components.

The query processor will accepts query from user and solves it by accessing the database.

- Parts of query processor

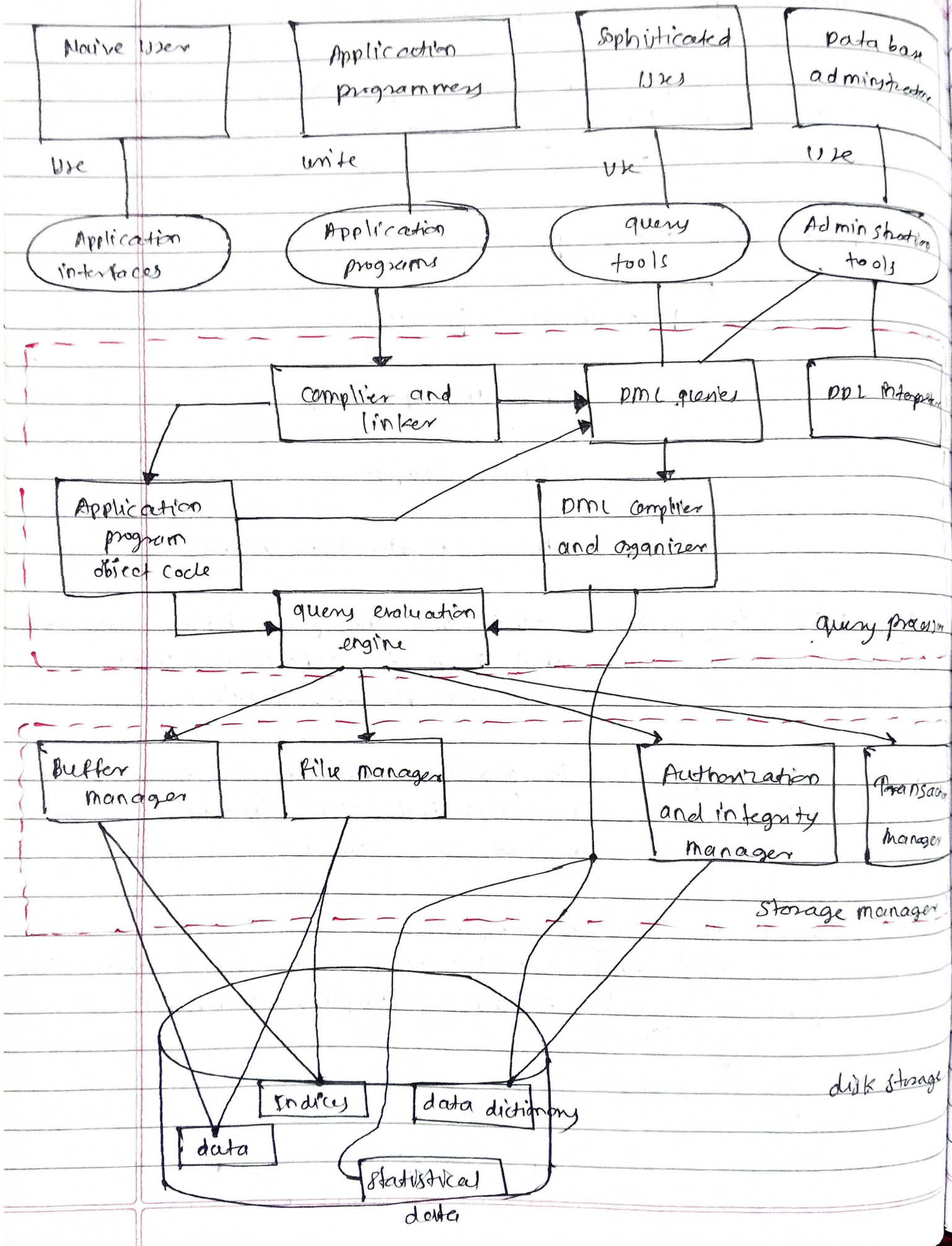


(1) DDL interpeter: This will interpeter DDL statements and return the definitions in the data director dictionary.

return definitions in the dictionary

(2) DML complier: it processes DML statements into low level instruction (machine language) so that they can be executed.

(3) Query evaluation engine:
This engine will execute low-level instruction generated by the DML complier or DBMS.


Naive User
Application programmers
Sophisticated users
Data base administrator
use
write
use
use
Application interfaces
Application programs
query tools
Administration tools
Complier and linker
DML queries
DDL Interpreter
Application program object code
DML complier and organizer
query evaluation engine
query processor
Buffer manager
File manager
Authorization and integrity manager
Transaction manager
Storage manager
Indices
data dictionary
data
Statistical data
disk storage

(2) Storage Mager / Storage Magements

- A storage manager is program module which acts like interface b/w the data stored in the database and the application programs and queries submitted to the system.
- the data is stored on the disk using the file system.
- The storage manager is programme which is responsible for the interaction with the file manager.
- The storage manager is translates the various database language statements into low level file system commands
- Thus, the storage manager is responsible for storing, retrieving and uploading data in the database.

(3) Transaction Maneagements:

- A transaction is a series of small database operations that together form a singe large operation
- A transaction is started by issuing a Begin Transaction command. once this command is executed the DBMS starts monitoring the transaction
- Transaction maneagements compones will ensure the ~~atomat~~ atomicity and durability properties.

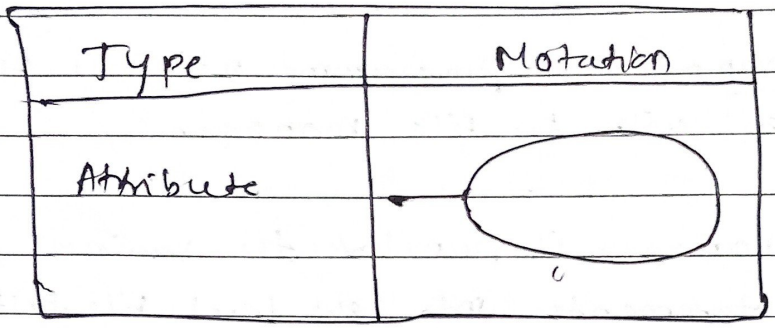
PAGE No.
DATE / /

Q) explain types of attributes.

Ans:

Attributes are the property which define entity type.

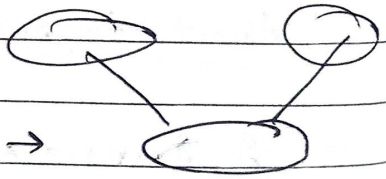
each entity has its own property which describe attributes.



- Types of Attributes

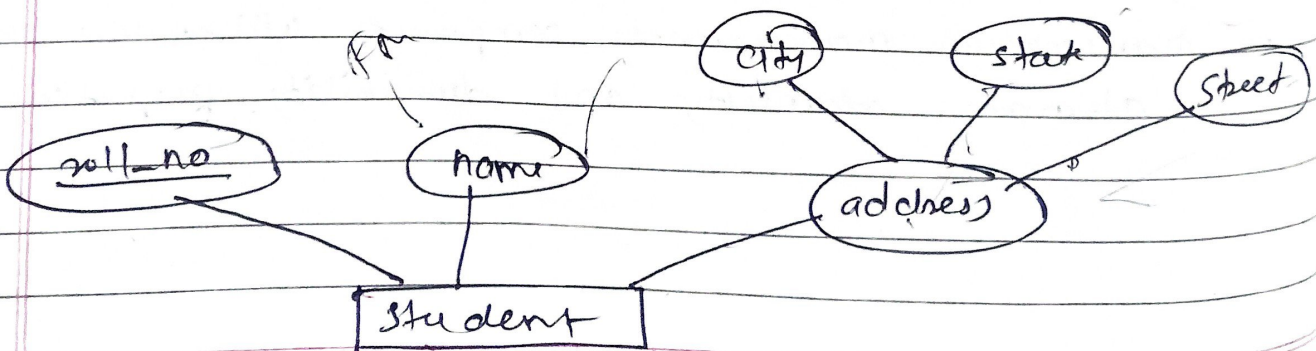
(1) composite Attribute
(2) multivalue -|-
(3) derived -|-
(4) key -|-

(1) composite Attribute : notation :→



- it can be divided into multiple sub parts

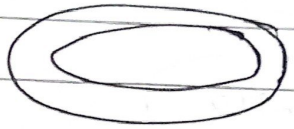
eg:



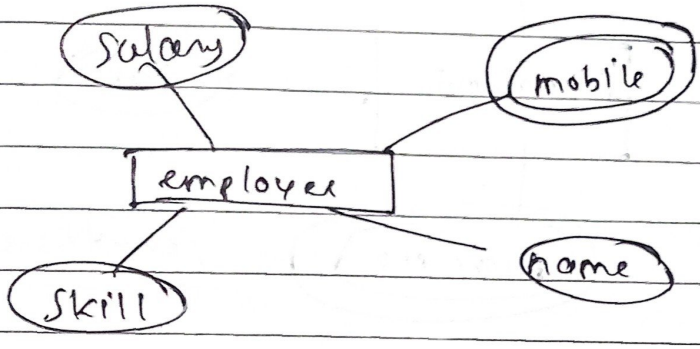
(2) Multivalue Attributes:

- A Attribute which have multiple values.

- Notation:



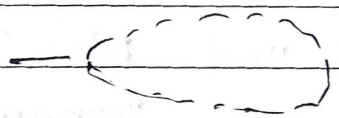
eg:

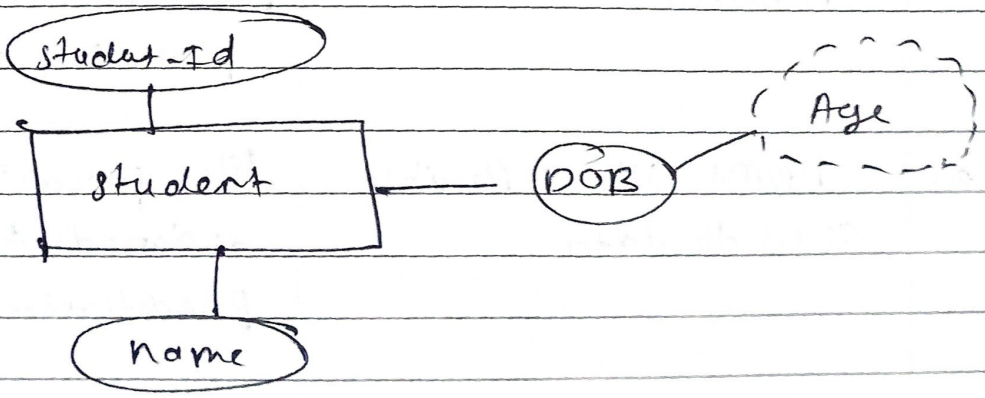


(3) Derived Attributes:

This are the types of attributes which are not exist physically in database but there values can be derived from other attributes present in database.

- Notation:

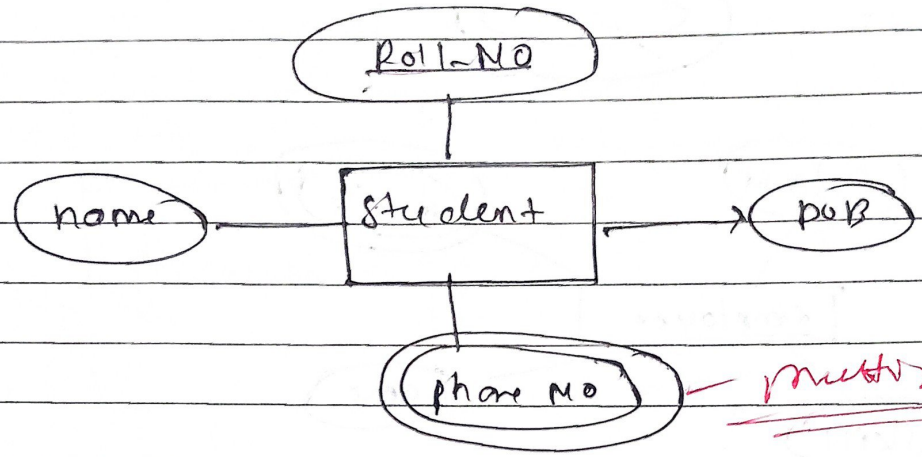




(4) Key Attributes:

To identify attributes uniquely we set key to attributes and it's can be denoted by underline

(_____)



(5) compare file processing system & DBMS

NO	DBMS	file processing system
(1)	Computerized record-keeping system is used in DBMS	Collection of individual files accessed by application programs is called file processing system
(2)	DBMS allow flexible access to data	File processing system is designed to allow predetermined access to data
(3)	It co-ordinates both physical and logical	It co-ordinate only the physical access to data

No.	DBMS	File processing System
(4)	DBMS provide multiple user interface	data is isolated in the file system
(5)	Unauthorized access is restricted in DBMS	Unauthorized access cannot be restricted
(6)	Redundancy can be controlled	Redundancy cannot be control.

Q) explain the features of EER model with example

- The enhanced entity Relationship (EER) model is an extension of the traditional entity Relationship (ER) model.

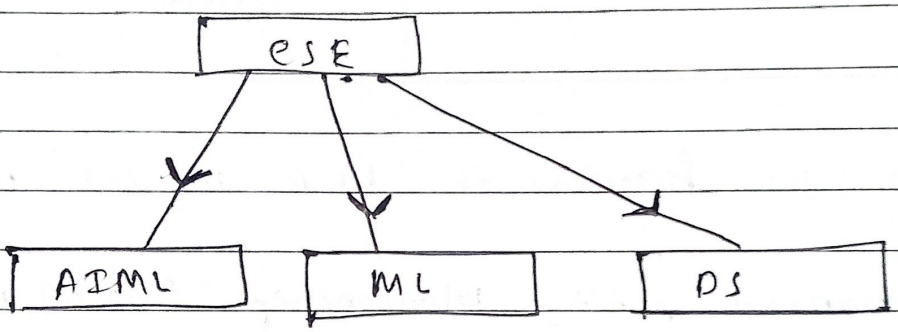
* It includes all the modeling concept of ER model

(1) specialization
(2) generalization
(3) Aggregation
(4) attributes & inheritances
(5) cardinality
(6) partial participation
(7) Integrity constraints.

(1) Specialization:

- it is a top down approach of super class / sub class relationship
- It is process of defining a set of superclass and subclass entities of same entity type.

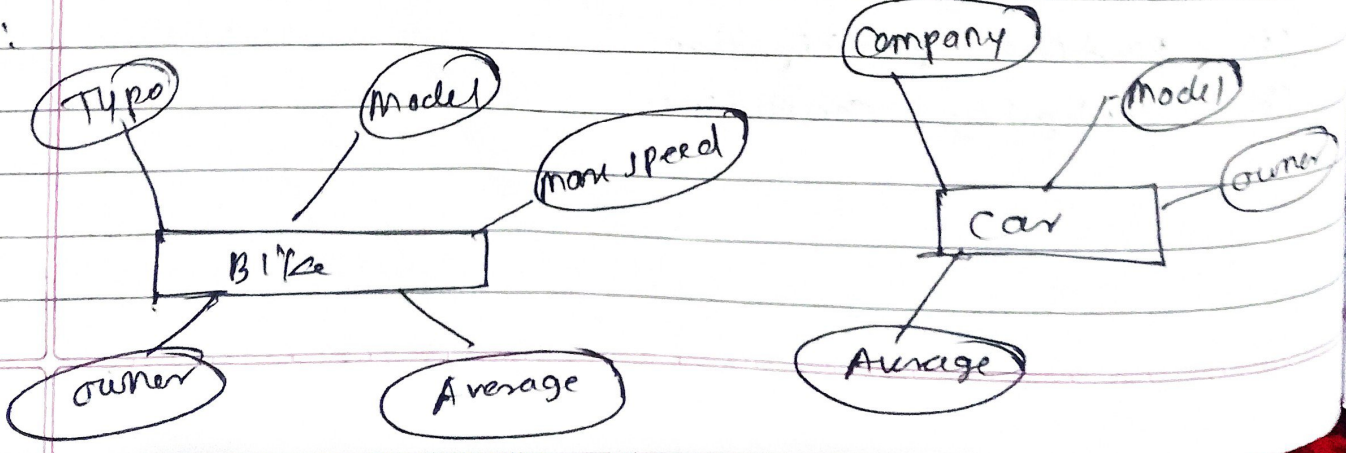
eg:



(2) Generalization:

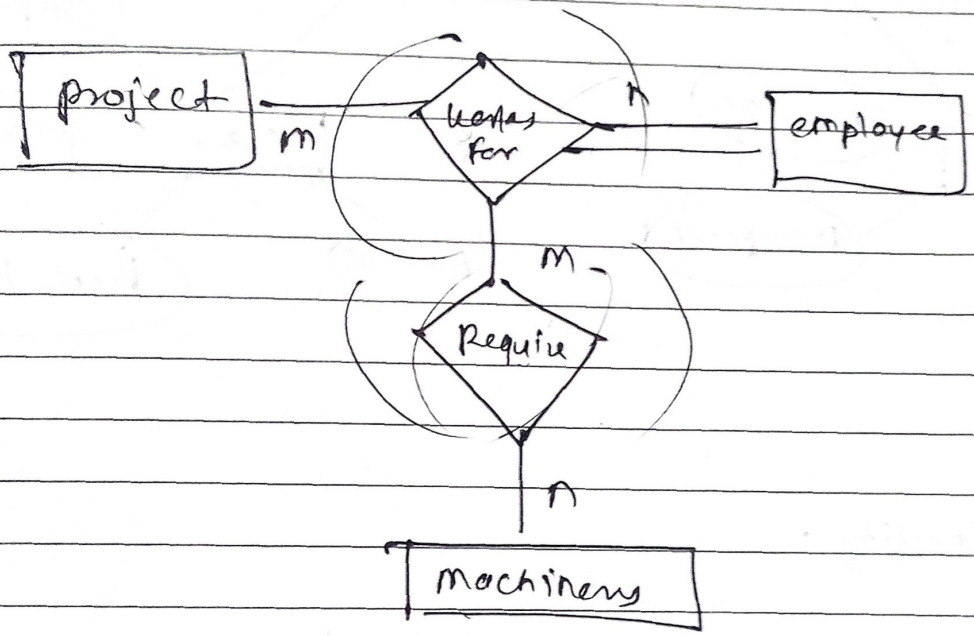
- It is a ~~bottom~~ top approach
- Generalization is the reverse of specialization, it combines two or more lower level entites into a higher level entity based on shared attributes

eg:



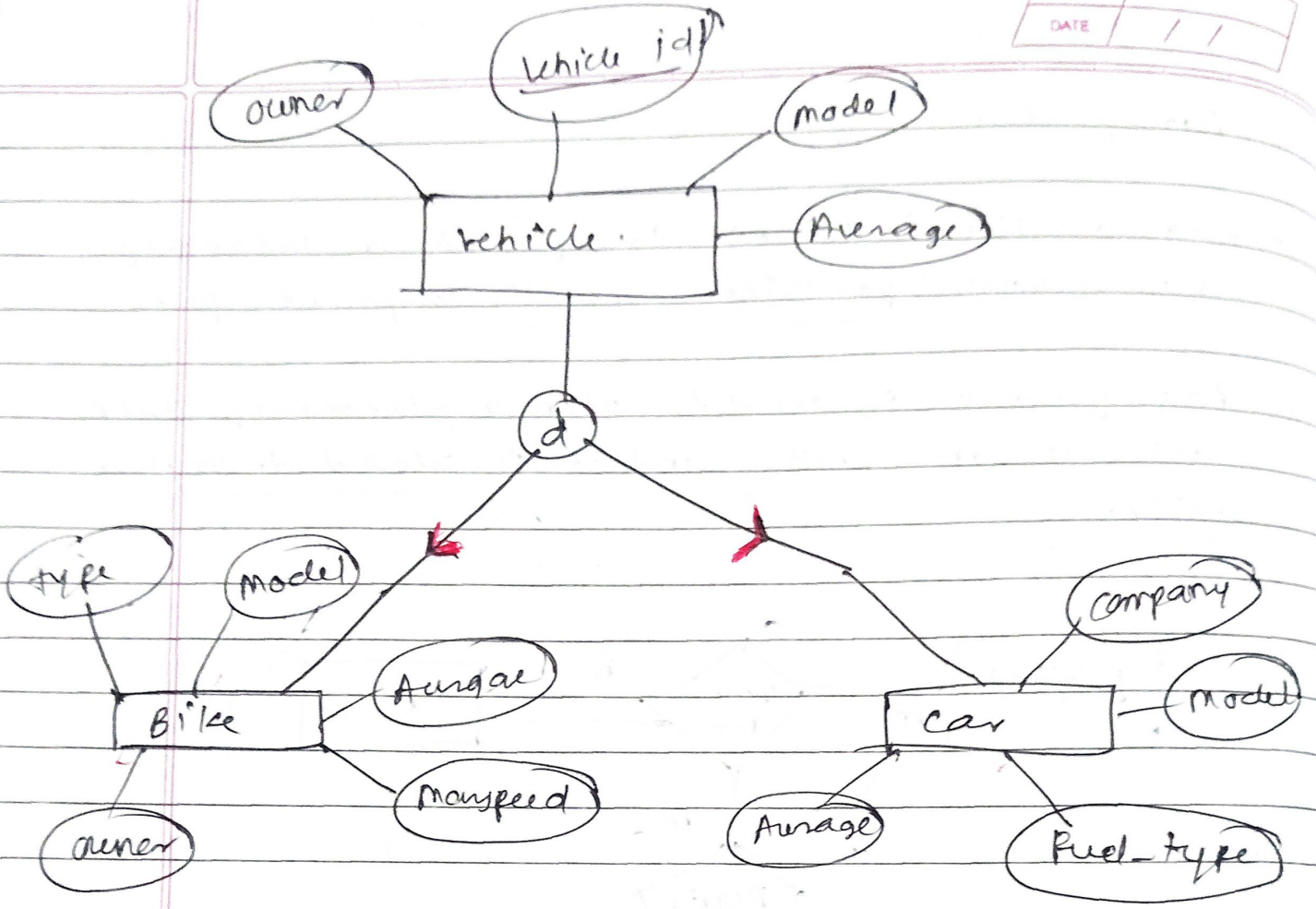
(3) Aggregation

- Aggregation is meant to represent a relationship b/w a whole ~~pr~~ object and its components parts
- Aggregation is used to when a relationship itself act as an entity and it is related to another entity.



(4) Attribute inheritances:

- The attributes of higher and lower level entities created by specialization and generalization are attributes inheritances.
- Here subclass automatically inherits attributes from superclass.

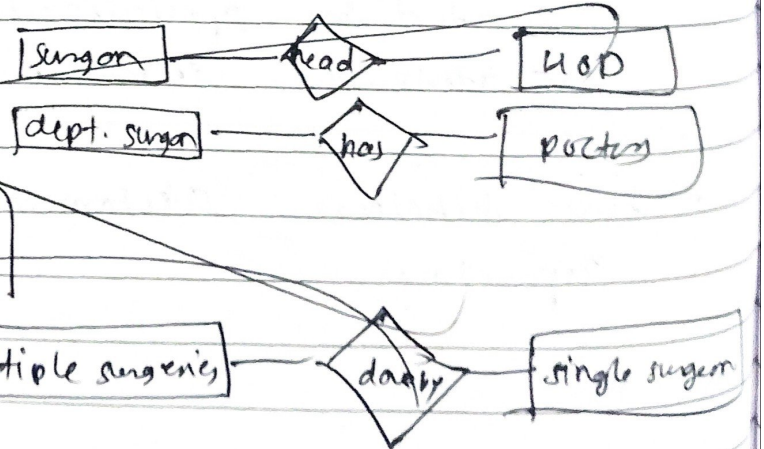


(5) Cardinality :

refers to no. of relationships blw entities in a database relationship.

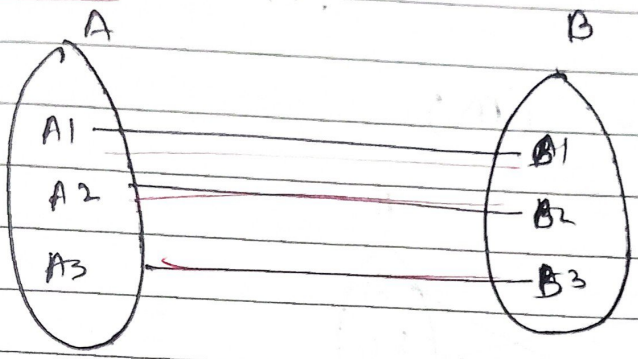
Types

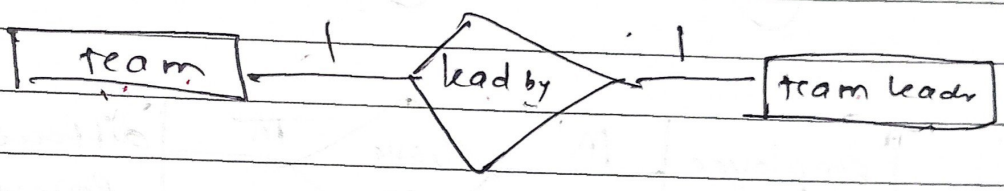
- one to one (1 : 1)
- one to many (1 : N)
- many to many (M : N)



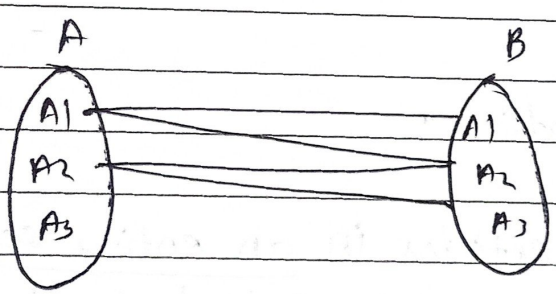
(5) Partial Participation:

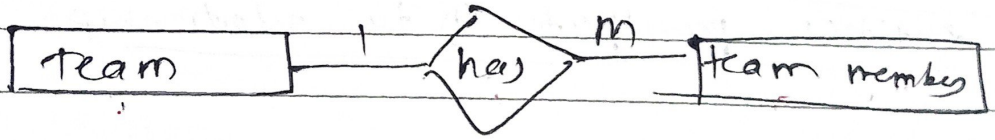
(1) one to one (1:1)



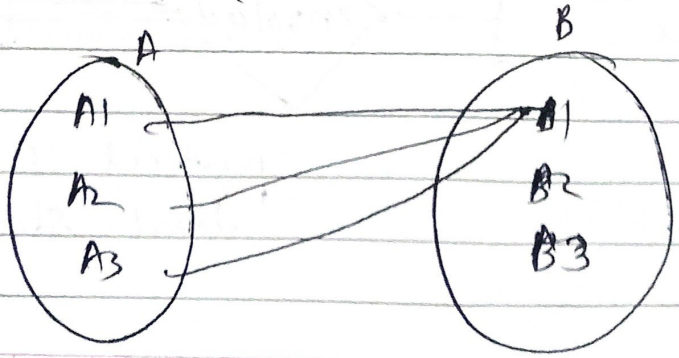


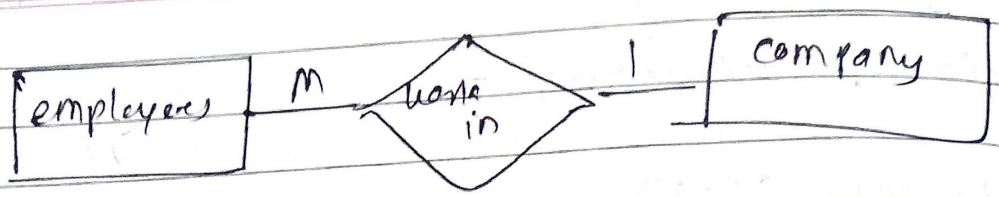
(2) one to many : (1:M)



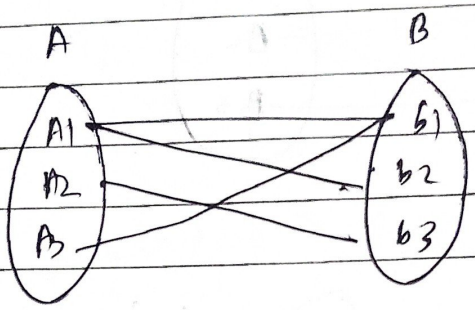


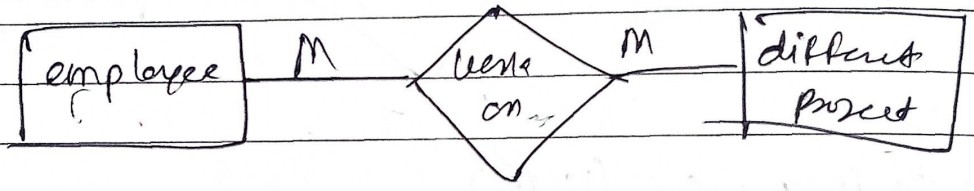
(3) many to one (M:1)





(4) many to many (M)

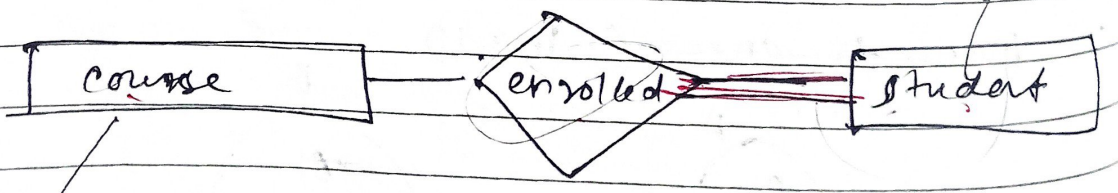




(6) Partical participation:

occurs when some entities in an entity set are related to another entity set, but not all entities participate in the relationship

eg:



partical course
total students

(7) Integrity constraints

Integrity constraints in a dbms are rules that ensure data is accurate & constraint. and secure. Secure.

Types of integrity constraints

- Domain
- Entity integrity
- Referential
- Key

Types
- Key
- Domain
- Referential
- Check

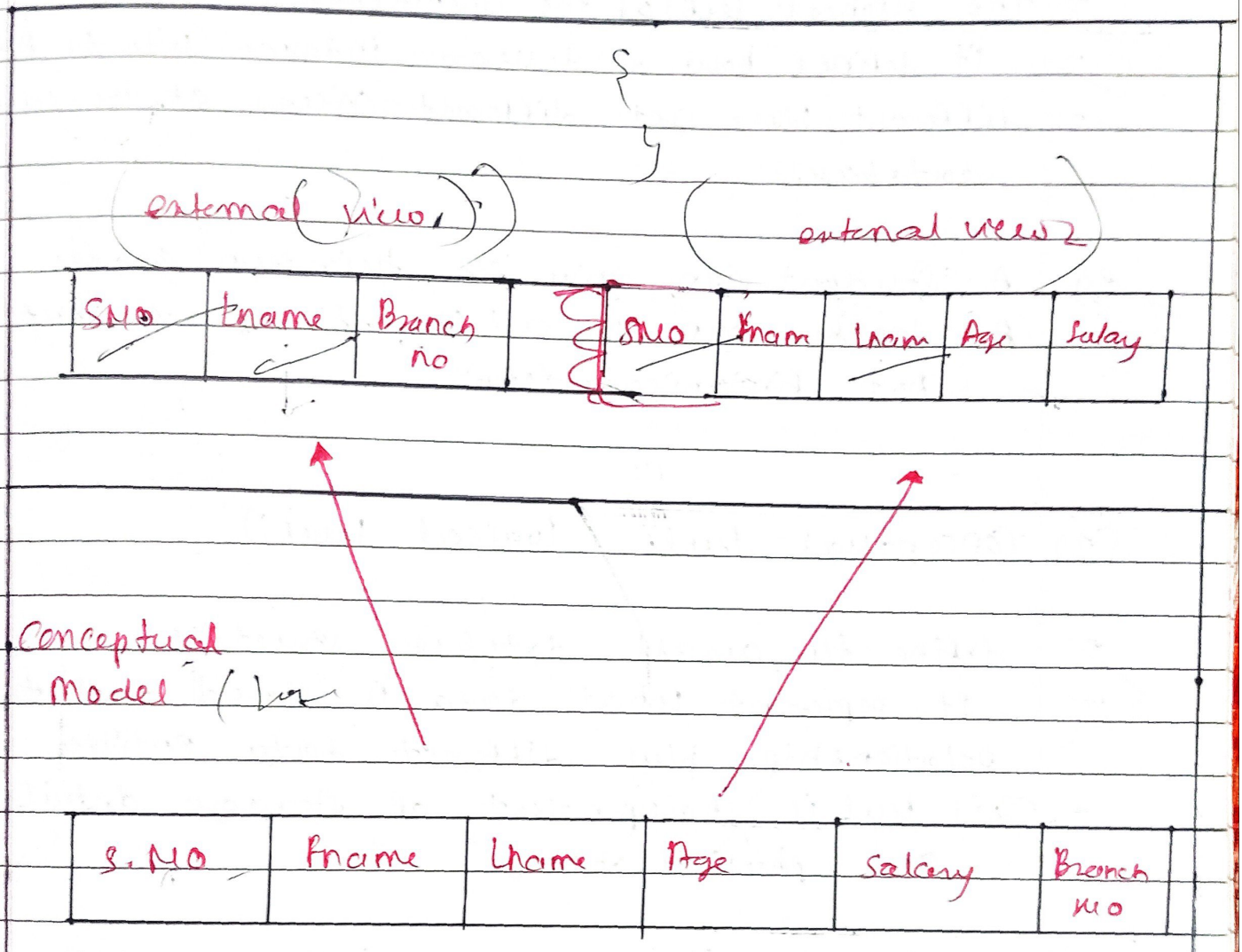
Q) explain 3 layer architecture of DBMS

Ans:

- The goal of three layer architecture to separate the front end and the back end
- The three schema architecture is a tool with which the user can visualize the schema levels in DBMS
- Many DBMS system do not separate the three level completely But support the three schema architecture to some extent.

- A description of data in terms of a data model is called schema.

- The description of database is called database schema,



Internal view

create table of employees
(
Sno number (30);
Fname varchar2 (50);
Lname varchar2 (50);
Age Number
Branch no Number (10);
Salary Number (10, 2)
);

(1) External level (view level)

- The highest level of the architecture
- it defines how end-users interact with the DB
- different user get different views of the same database.

eg: A student can only see their own grades
A teacher can see all student's grades but not financial details.

(2) Conceptual level (logical level)

- define the overall database structure
- It represent what data is stored and the relationship btw different data entities.
- This level is independent of storage details and physical storage

eg: A university DB stores info about students, course, and enrollements and their relationship.

(3) internal level (physical level)

- The lowest level of abstraction
- it defines how data is physically stored in memory (file, indexing and storage structures)

eg: Data is store in B-Tree, hash, index on hard disks or SSD

(1) external level (view level)

- defines how diff user see the database
- each user may see a diff part of the database

eg: A student can see only his marks.
An admin can see marks of all students.

(2) Conceptual Level (Logical level)

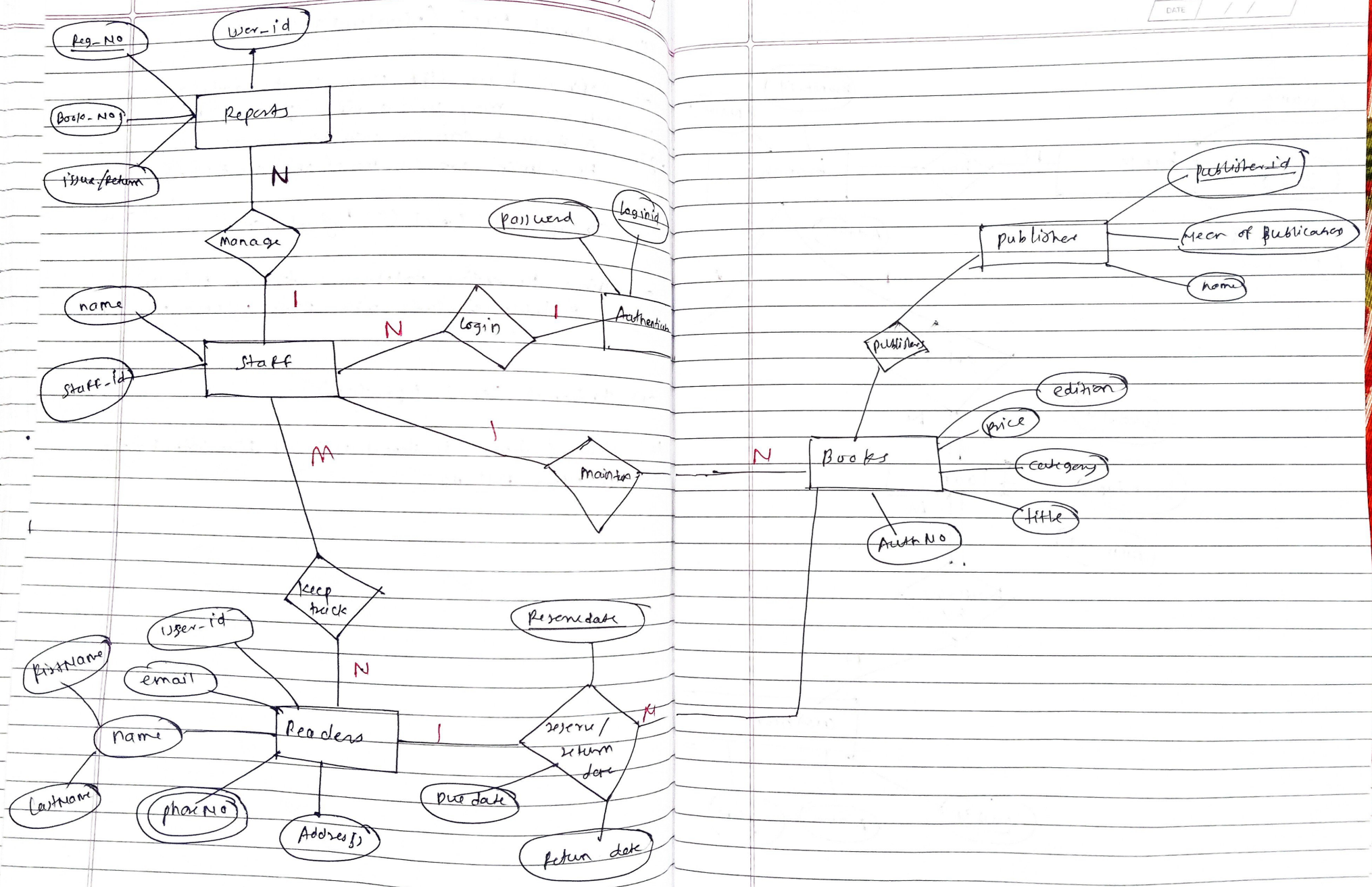
- defines the entire database structure logically

eg: A table students with fields Roll.No, Name, marks

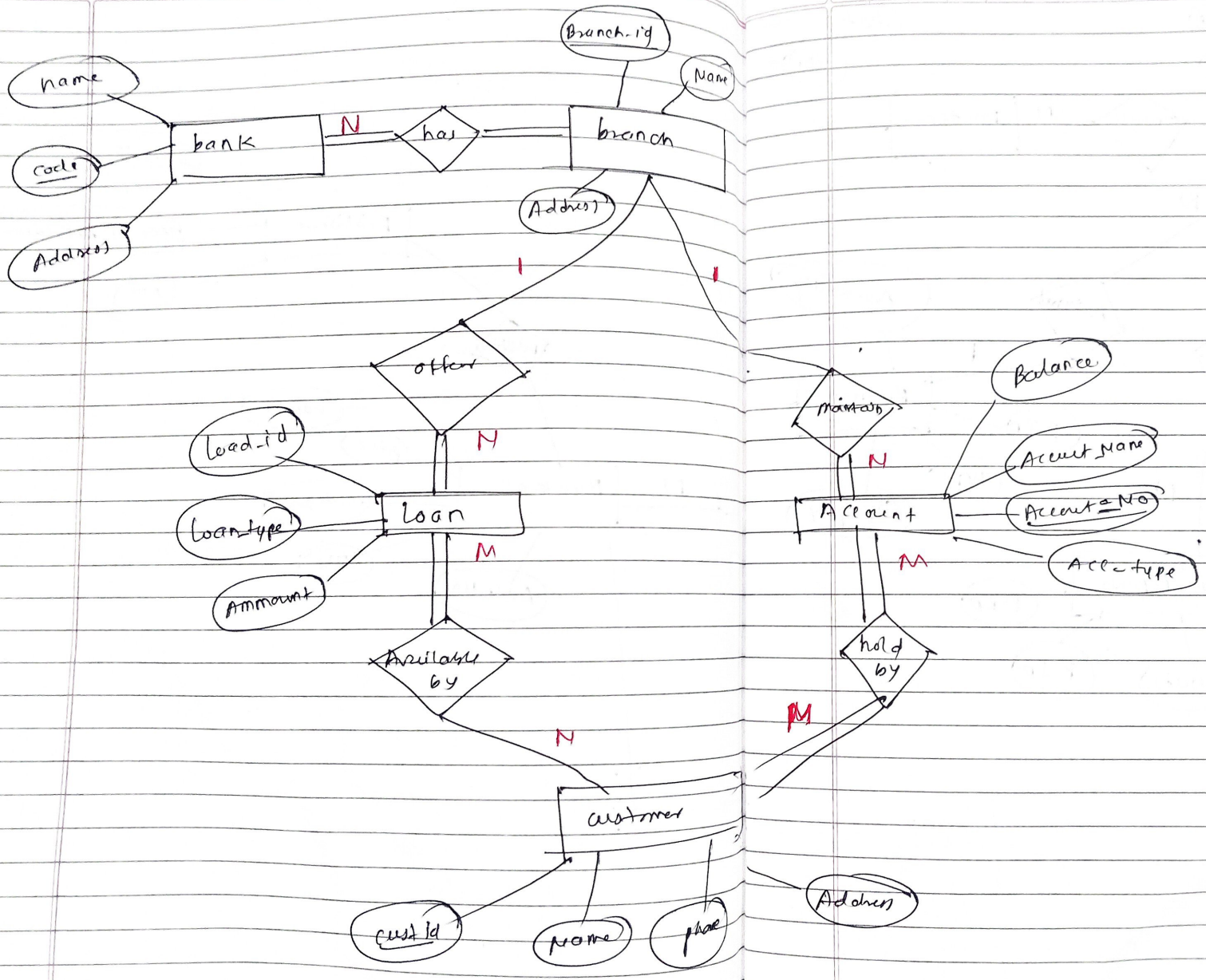
(3) Internal Level (physical)

eg: Student table is stored as sorted files with B+ tree indexing Roll No.

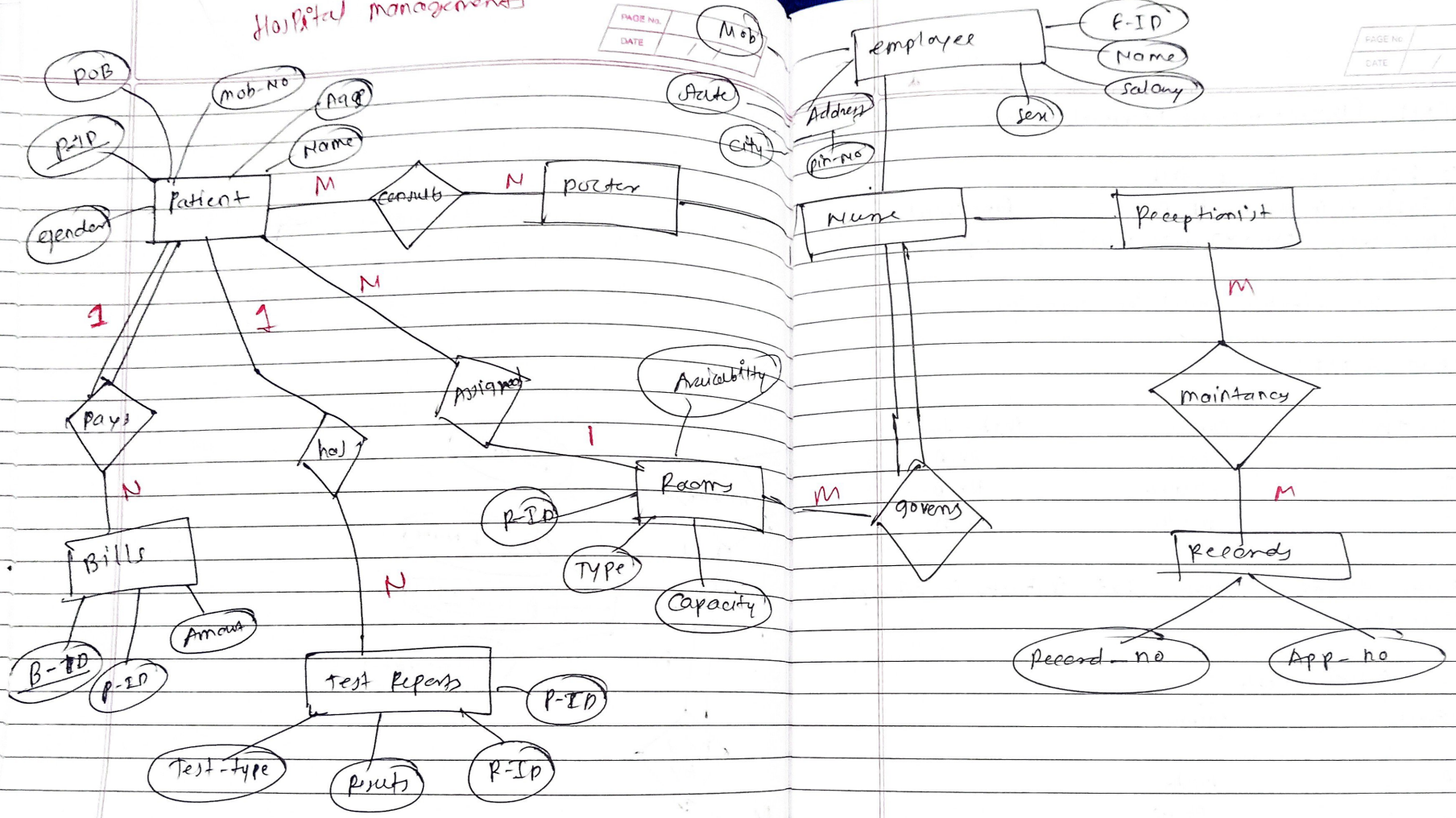
★ ER diagram
Library Management



Bank Management



Hospital management



PAGE No.
DATE / /

DBMS UT-2

Q) Why there is need of Normalization. explain 1NF, 2NF & 3NF

Ans:

Normalization is the process in DBMS to organizing data in database to reduce redundancy (Repetition) and avoid data anomalies.

⊙ Why is there a need of Normalization. ?

(1) eliminate data redundancy — No unnecessary duplicate data.

(2) Ensure data integrity : data stay accurate & consistent.

(3) Avoid anomalies : prevent
- insertion anomaly (trouble inserting data)
- Deletion — " — (losing valuable data)
- update — " — (inconsistent when updating)

(4) Better database design — clean and logical table structure

(1) 1NF (First Normal form)

A Relation in 1NF if:

- All attributes contain only atomic (indivisible) value
- each column contains values of a single type
- each record is unique.

eg: Not in 1NF:

Roll.No	Name	Subject
1	Megh	maths, science
2	Ash	English

Here 'subject' has multiple values (not atomic) so not in 1NF.

After 1NF:

Roll.No	Name	Subject
1	Megh	Maths
1	Megh	science
2	Ash	English

" if may introduce data redundancy "

no partial dependency : no non-prime attribute ~~dependent only on~~ ~~Part of a composite key~~ must depend on whole primary key, not just part of it

(2) 2NF (Second Normal Form):

A Relation in 2NF if:

- ✓ It is already in 1NF
- ✓ No partial dependency
- ✗ partial dependency means: ~~a non-prime attribute~~ dependent only on part ~~of the~~ primary key, not the whole

example: (1NF but not 2NF)
primary key : Roll.NO, Subject

Roll.NO	Subject	Student Name
1	Maths	Megh
1	science	Megh

Here student names depends only on Roll NO, not on full primary key ⟶ partial dependency ⟶ not in 2NF

After 2NF: Student table. (split into two table)

Roll.No	Student Name
1	Megh

Subject table:

Roll. No	Subject
1	Math
1	science

NO → Transitive →

a → b → c

a → non-prime attribut depending on another non-prime attribut

(3) 3NF (third Normal form)

A Relation in 3NF if:

- It is in 2NF
- NO transitive dependency exists.

example: (2NF but not 3NF)

emp ID	Emp Name	Dept ID	Dept Name
101	Ash	D1	IT
102	Megh	D2	CS

- Here Emp ID is the primary key
- Dept Name depends on Dept ID, which turn depends on Emp ID - transitive dependency.

After 3NF: Employee Table

emp ID	emp name	Dept ID
101	Ash	D1

Department table:

Dept ID	Dept Name
D1	IT
D2	CS

(4) BCNF (Boyce - Codd Normal Form)

- It's higher version of 3NF
- A table is in BCNF if for every functional dependency ($X \rightarrow Y$), X must be super key
- in simple words
  determinants must be super key

eg: Before BCNF:

Student ID	Course	Instructor
1	Maths	Dr. A
2	Physic	Dr. B
3	Maths	Dr. A

Instructor determines Course, but Instructor is not super-ridcate BCNF

After BCNF: Instructor table

Instructor	Course
Dr. A	Maths
Dr. B	Physic

- Student table

Student_id	Instructor
1	Dr. A
2	Dr. B
3	Dr. A

Explain Integrity constraint.

Ans:

- Integrity constraints are rules applied to database table to ensure that the data entered is valid, consistent and meaning ful.

- These constraint help maintain the integrity (correctness) of the data.

* There are 4 types:

(1) Key constraints:

- These ensure that each record (row) in the table can be uniquely identify.
- primary key is the most common key constraint.
- It must be unique and not null.

eg: In a Student table, Student id is a primary key.

(2) Domain constraints:

- These define the valid set of values a column can hold.
- Every attribute (column) in a table must have a defined data types and format

Example: - Age column should only accept integers
- gender column can only have 'male' or 'female' or 'others'.

(3) Referential Integrity:

- maintains consistency b/w two related tables
- A foreign key in one table must match a primary key in another table.
- It prevent deleting or changing data that is used elsewhere.

example:

Department_ID in the employee table must exist in the department table.

(4) Check constraints:

- It ensures that all values in a column meet a specific condition
- It can be used to apply custom rules on data

example:

CHECK (Age >= 18) ensures only adult ages are entered.

PAGE No.
DATE / /

what is concurrency control & timestamp based protocol.

Ans:

① concurrency control: Concurrency control is a technique used in DBMS to manage the execution of multiple transaction at the same time without interfering with each other.

It main goals is to ensure data consistency and isolation when transactions run concurrently.

② Why is concurrency control needed?

- to prevent conflict b/w simultaneous transactions.
- to avoid problem like:
  - Lost update
  - temporary update (dirty read)
  - uncommitted data
  - inconsistent retrieval.

③ Timestamp based concurrency control protocol.

The timestamp based protocol is a concurrency control method that uses timestamp to order transaction and ensure Serializability. Serializability (Seriality)

④ Concept of timestamp

- every transaction get unique timstamp (like a serial number)

When it's start

- older transaction (smaller timestamp) get priority over newer ones.

- each data item keep:

  - Read timestamp (RTS) - latest time it was read
  - Write timestamp (WTS) - " - written

Q) Explain lossless decomposition.

Ans:

Lossless decomposition is property of database normalization ; where a relation (table) is split into two or more smaller relations, and when these are joined back, they reproduce the original relation without any loss of information.

- Why is it important?

- ensures data is not lost while splitting tables during normalization!

- Maintains data integrity

- Helps to avoid inconsistent or incomplete data after decomposition.

example: Student table.

Roll.NO	Name	Course
101	Amit	BCA
102	Ayush	BBA
103	Ash	MCA
104	Megh	BE

- decomposit into two tables.

(R1) . Roll.NO & Name

Roll.NO	Name
101	Amit
102	Ayush
103	Ash
104	Megh

(R2) Roll.NO, course

Roll.NO	Course
101	BCA
102	BBA
103	MCA
104	BE

Natural join of R1 and R2 (R1 ⋈ R2)

→ draw as it original one.

PAGE No.
DATE / /

Q) Define triggers and explain its syntax with example.

Ans:

definition: A trigger is stored program in the database that automatically executes (fires) in response to certain event like INSERT, UPDATE or DELETE on a table.

think of it's like "condition - based automatic action". in the database.

★ Syntax of triggers

```
CREATE [OR Replace] Trigger trigger_name
AFTER | BEFORE INSERT | UPDATE | DELETE
ON table_name
FOR EACH ROW
BEGIN
---- trigger Logic here
END;
```

* ~~Syntati~~ explanation of syntax:

1. CREATE Trigger → start trigger creation
2. OR REPLACE → If trigger is already created then drop and recreate the trigger
3. trigger_name → Name you given to trigger
4. BEFORE or AFTER → when trigger should execute.
5. INSERT / UPDATE / DELETE → events that activates triggers
6. on table_name → the table on which the trigger is defined
7. FOR EACH ROW → execute for every row affected
8. BEGIN ... END → A block where you write the triggers logic

* example: trigger after inserting into a table.

- suppose we have table students (id, name) and we want to log every new insert into a logs table

```
CREATE TRIGGER log_insert
  AFTER INSERT
ON Students

FOR EACH ROW

BEGIN
    INSERT INTO logs (action)
      VALUES ("New student inserted");
END;
```

* what it does?

Whenever a new record is inserted into the Student table, this trigger automatically insert message into the logs table.

Q) Explain Acid properties

Ans:

ACID stands for:

- A → Atomicity
- C → consistency
- I → isolation
- D → durability

These are the 4 key properties of a transaction in DBMS to ensure data reliability and correctness, even in case of failure.

① Atomicity — "All or Nothing"

- Atomicity refers to principle of a database transaction is a single unit of work.
- This means that either all operations in a transaction are completed successfully or if any part fails, then the entire transaction is rolled back to it's original state.
- In simple term the transaction is "all or nothing".

example: If transaction involves transfering a money from Account A to Account B, both debit and credit operation must complete successfully. if one's fails, then money is roll back to user.

② Consistency : ~~"validate~~ valid state to valide state

Consistency ensure that a transaction brings the db from one valid state to another, preserving all the integrity constraints (Rules)

After the transaction, the db should still be in valid state, and all the predefined rules must be maintained.

example: if the system has rule that an account balance must always be positive, transaction that violates this rules would not be allowed to complete.

(3) Isolation "No Interference"

Isolation ensures that the operation of one transaction are invisible to other transaction until the transaction is completed.

Means No interference from other transaction while it's executing.

example: two user booking the last movie ticket at the same time, only one should succeed handling isolation.

(4) Durability "Changes are permanent"

durability guarantees that once a transaction has been comitted, its changes are permanently recorded in the db. even in the case of system failure or power failure.

- This ensures that no commited data is lost

example: If a transaction transferring money is successfully completed. even if in the system failure the transfer will be recorded and no data will be lost.

Q) Explain Aggregation Functions in SQL.

Ans:

Aggregation functions are used to perform calculation on group of values and return a single summarized results.

They are commonly used with Group by clause to group rows having similar values.

- List of Aggregation functions:

Functions		Meaning
① COUNT(*)	→	Counts all rows, including duplicates and NULLs
② COUNT(DISTINCT COL)	→	Counts only unique non-null values in the column.
③ COUNT(ALL COL)	→	Counts all non-null values in column (default behaviour)
④ SUM()	→	Returns the total sum of numeric values in column
⑤ AVG()	→	Return Average (mean) values
⑥ MIN()	→	Returns smallest Minimum values
⑦ MAX()	→	Returns Largest maximum value

(7) SELECT SUM (Amount) as total_amount from sales ⇒ 140

(8) SELECT MIN (Amount) as min_amt from sales
= 10

DATE / /

example: Sales table.

Product	Amount
pen	10
Book	50
Bag	80

* Sample queries:

(1) SELECT COUNT (*) from sales
↳ 3

(2) SELECT SUM (Amount) from sales
↳ 140

(3) SELECT MAX (Amount), MIN (Amount) from sales
↳ 80 ↳ 10

(4) SELECT COUNT (Amount) from sales table.
→ 3

(5) SELECT COUNT (DISTINCT Product) from sales table
→ 3

(6) SELECT AVG (Amount) as Avg_Amount from sales table
↳ (10+50+80)/3 = 46.67

PAGE No.
DATE / /

g) Explain 2PL (2 phase Locking) Concurrency control protocol.

Ans:

Two phase locking (2PL) is a concurrency control protocol that ensure serializability in db by managining the order of locking and unlocking of data items during a transaction

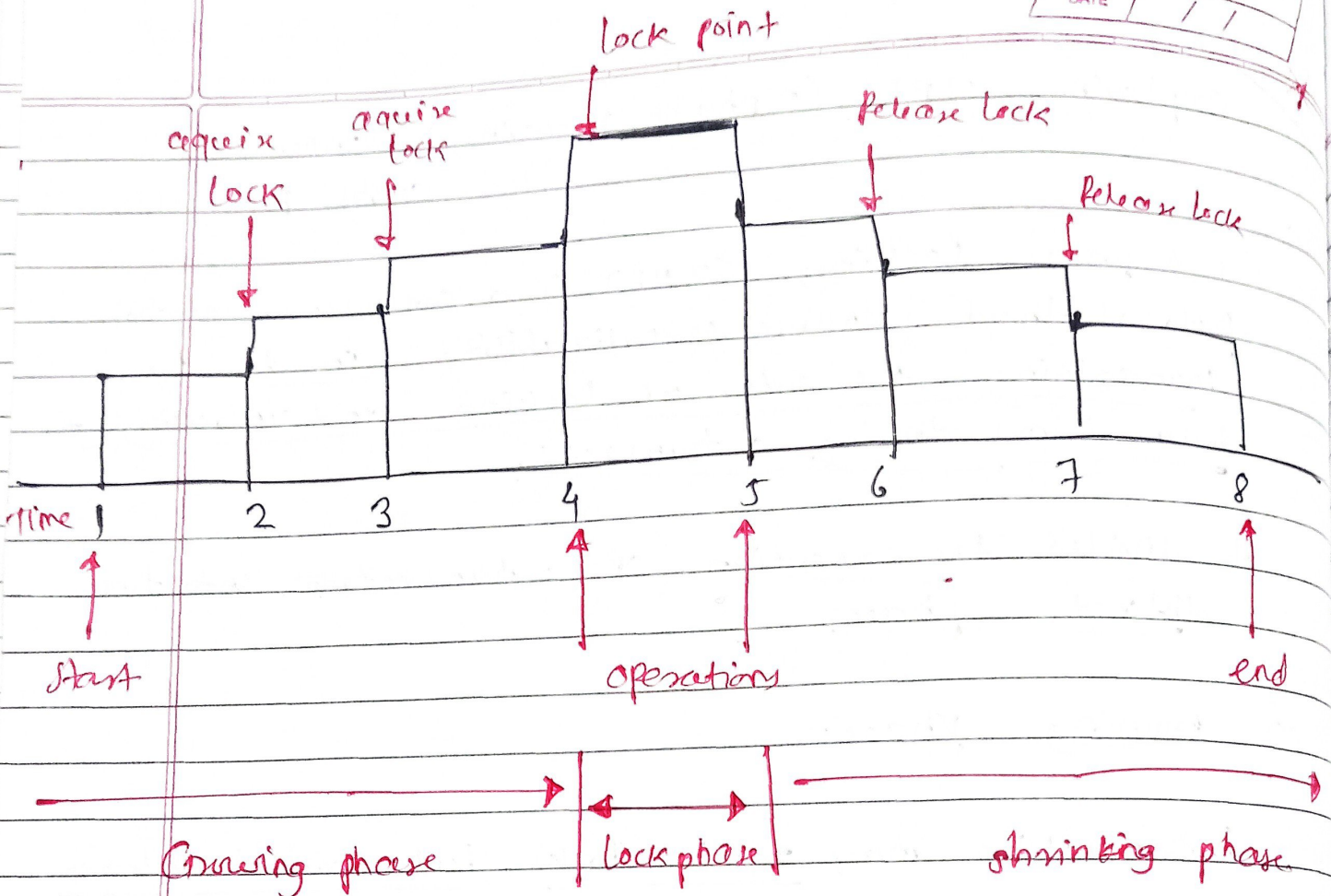
it divides the execution of a transaction into two distinct phase.

(1) Growing phase:

- A transaction may aquire locks, but cannot release any.
- It keeps locking data items as it needs them.

(2) Shrinking phase:

- A transaction may release locks but cannot aquire any new lock.
- once it release the first lock, it cannot request any more.



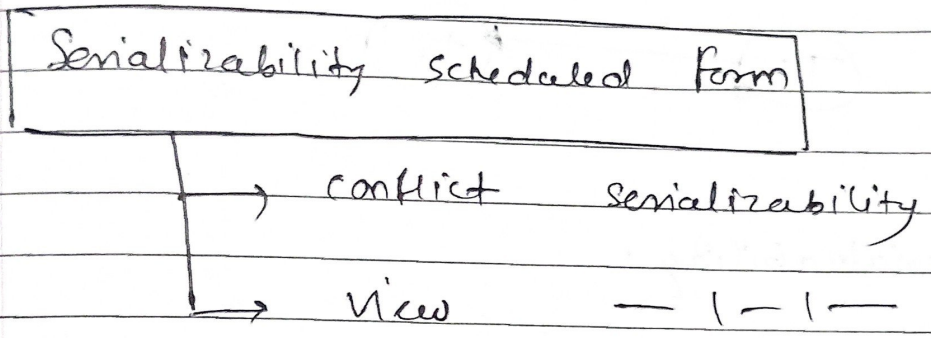
★ Advantage	★ disadvantage
① ensure serializability	① can cause Deadlock
② Avoids concurrency issues	② reduce concurrency
	③ blocking of other trans

Serializability → even if transaction run together & the final result must be same as some one by one execution

Q) explain View & Conflict Serializability. (optional)

→

first of all what is serializability?



① Conflict Serializability

A schedule is conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.

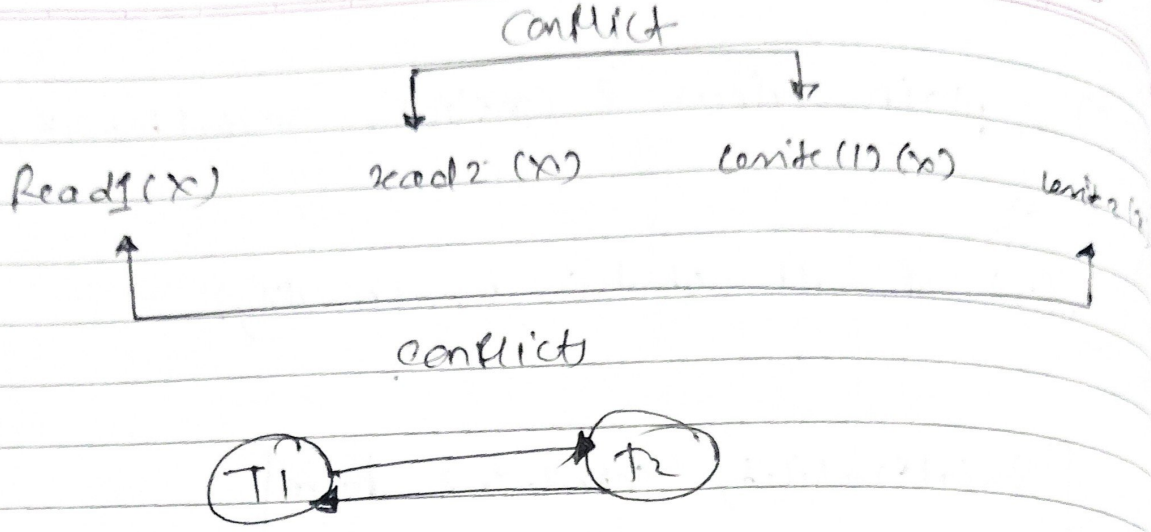
★ When occurs?

- two transactions access the same data item
- At least one of them perform a write
- And operation are in different order.

★ How to check?

- Using precedence (or conflict) graph
  If the graph has no cycle, the schedule is conflict Serializable.

example:



(2) View Serializability:

A schedule is a view Serializable if transaction procedure produces same final output as a Serial schedule even if execution order differs.

⊙ three cond^n for View Serializability

(1) Initial Read - first read of data item must be the same initial value as in the Serial schedule.

(2) Read from write - if a transaction read a value ~~from~~ written by another it must read from the same transaction in both schedules.

(3) Final write :- the last write on each data item must be by the same transaction in both schedules.

Q) Explain Functional dependency & it's types:

Ans:

A Functional dependency (FD) is a ~~relational~~ relationship b/w two attributes in a relational database. such that for a given value of one attribute (called determinant), there is exactly one associated value of another attribute.

in short, A Functional dependency occurs when one attribute uniquely determines another.

Symbolically written as:

$A \longrightarrow B$ (Read as: A Functionally determines B)

Example: Student table

Roll. NO	Name	Course
1	Ash	BCA
2	Steave	MCA
3	Megh	B. Tech

Roll. No $\longrightarrow$ Name. it means

knowing a student's roll. no is enough to know their name.

DATE / /

* Types of functional dependency:

(1) Full functional dependency
(2) Partial
(3) Transitive
(4) Trivial
(5) Multivalued

(1) Full Functional dependency

: A full functional dependency $X \longrightarrow Y$ is fully functional dependency if Y depends completely on the whole of X, not just part of it.

example:

Student_id	Course ID	Marks
101	CS101	90
102	IT101	85
103	CS101	88

full FD $\Longrightarrow$ (Student_id, Course_id) $\longrightarrow$ marks

marks depends on both student_id and course_id. Neither student or new course_id alone can uniquely determine marks

(2) Partial Dependency: A functional dependency $X \rightarrow Y$ is partial if Y depends on only part of composite key.

eg:

ROLL NO	Subject	Marks
101	Maths	90
102	physics	85
103	physics	88

Partial FD: ROLL NO → subject.
ROLL NO can determine subject independently of the subject column. So ROLL NO → subject is partial because subject doesn't on the entire composite key.

(3) Transitive Dependency: A dependency is transitive if $X \rightarrow Y$ and $Y \rightarrow Z$ and implying $X \rightarrow Z$ through an indirect relationship via Y.

eg:

ROLL NO	student Name	Department
101	Megh	CS
102	Ash	IT
103	spidey	CS

ROLL NO → Student_Name
Student Name → Department
ROLL NO → department

(4) Trival functional dependency:

A functional dependency $x \to y$ is trivial if y is a subset of x. In this, y is already present of x, so no new information is being determined.

eg:

Student ID	Name	Address
101	Megh	NY
102	Aish	LA
103	Spidey	SF

Trival FD: {Student ID, Name} → Name

The attribute Name is already part of {Student ID, ~~Address~~ Name}

so it's trivial

(5) Multivalued Dependency:

- A multivalued dependency exist when a single value X can determine multiple independent values of Y
- The values of Y are not dependent on each other but determine independently by X.

eg:

Employee ID	Skills
101	CPP
102	C
103	JS

An employee can have multiple skills and these are independent of each other. One can can have several skills like CPP, JS.

What is Serializability:

Ans:

Serializability is a concept in DBMS that ensures that the results of executing multiple transaction together is the same as if the transaction were executed one after another (serially, without overlapping)

eg: Suppose two transactions?

T1 → transfer 100rs from Account A to Accut B

T2 → Depoite 200rs to Account A.

initial values:

A = 1000rs

B = 500rs

If we run them serially

After T1 → A = 900 B = 600

After T2 → A = 900 + 200 = 1100

final values

A = 1100rs

B = 600rs

- Two types

1. conflicts
2. View

PAGE No.
DATE / /

* What is log-Based Recovery System in DBMS

Ans:

Defination: log-Based recovery is a technique in DBMS where all database transaction and their changes are recorded in log file.

if a failure occurs, the system uses the log to recover the database by redoing committed transaction and undoing uncommitted transaction.

Purpose: - To maintain data consistency after system crash
- to recover the database to the last consistent state
- to ensure durability

How it works?

There are two important operation:

(1) UNDO: Rollback changes of incomplete transaction

(2) REDO: Reapply changes of completed transaction after crash.

example: consider database.

Account	Balance
A	500
B	300

two transactions:

T1 → transfer 100 from A to B

T2 → deposite 200 to A

Log entries

Time	Log entry
1	start T1
2	T1 → A old = 500 new = 400
3	T1 → B old = 300 new 400
4	commit T1
5	start T2
6	T2 → A old = 400 new = 600
7	(system crash occurs)

Analysis: T1 was comitted → Redo it

T2 was not comitted → undo it

★ After Recovery

Account	Balance
A	400
B	400

Q) What is deadlock? explain wait-die and wound-wait methods with suitable example.

Ans:

A deadlock happens when two or more transactions are wait for each other forever and none of them can continue.

As a result, the system get stuck and no transaction can move forward.

eg: T1 → locks Resource A and wants Resource B
T2 → locks Resource B and wants Resource A

Now
T1 is waiting for T2
T2 is waiting for T1

Neither can proceed – deadlock happens.

★ cond^n for deadlock (when it is occur)

(1) Mutual execution
(2) Hold and wait
(3) No preemption
(4) circuit wait

* Methods to handle deadlock

(1) wait - die = older transaction may wait, younger die (rolled back)

(2) wound - wait = older transaction wounds younger

(1) wait - die method:

in wait-die method -

(i) If older transaction asks → let it wait
(ii) if younger transaction ask → kill it (die / rollback)

- older transaction is respected and allowed to wait
- younger transaction is forced to rollback

(2) wound - wait method:

in wound - wait method:

(i) if older transaction ask → wound (kill) younger one
(ii) if younger transaction asks → let it wait.

- older transaction is powerful and can force rollback of younger

- younger transaction must wait patiently if older is using the resources.

PAGE No.
DATE / /

* example of both

Suppose: T1 → older
T2 → younger

Situation: T1 (older) wants lock held by T2
T2 (younger) wants lock held by T1

Wait - die (Action)

T1 waits & T2 die (rollback)

Wound - wait (Action)

T2 is wounded (rolled back) & T2 waits.

PAGE No.
DATE / /

Q) Give example of serial schedule and equivalent to serial schedule with respect to conflict serializability discuss conflict serializability with example

Ans:

* What is serial schedule:

A serial schedule is a schedule in which transaction are executed one after another, without overlapping.

Eg: T1:
R(A), W(A)

T2:
R(B), W(B)

schedule S1 (Serial):

T1: R(A), W(A)
T2: R(B), W(B)

This is serial schedule because all operations of T1 are executed before T2 starts.

* equivalent to serial schedule (conflict serializable conflict)

Let's create a schedule S2 where operations are interleaved

T1: R(A)
T2: R(B)
T1: W(A)
T2: W(B)

- in this operations are interleaved.
- both operation of T1 and T2 act on different data items (A and B)
- There is no conflict between them

So, S2 is conflict equivalent to serial schedule S1 (T1 → T2)

Hence S2 is conflict Serializable.

* what is conflict Serializability:

A schedule (sequence of operations from multiple transaction) is conflict serializability if it can be transformed into a serial schedule by swapping non-conflict operations.

5) explain all relational algebra operation

Ans:

- Fundamental operations of Relational Algebra:

(1) Unary RO:

- Project operation (π)
- Select — 1 — (σ)
- Rename — 1 — (ρ)

(2) SET Theory operations:

- Union operation ($\cup$)
- Difference — 1 — (-)
- Intersection — 1 — ($\cap$)

(3) Binary operations

- Join operation ($\bowtie$)
- Cartesian product . ($\times$)
- Division . ($\div$)

Unary operators

table: employee

empID	Name	Dept	Salary
1	Alice	HR	50000
2	bob	IT	60000
3	carol	IT	70000
4	dave	Sales	40000

table department

Dept	Manager
HR	John
IT	Smith
Sales	Rick

(1) Selection (σ)

- Retrieves rows that satisfy a certain condn

(i) Works like WHERE clause in SQL
(ii) Does not affect column, only filters rows
(iii) This is unary relational operator

Notation: $\sigma_{<condn>}(Relation)$

eg: $\sigma_{Dept = 'IT'}(Employee)$

EmpID	Name	Dept	Salary
2	Bob	IT	60000
3	Carol	IT	70000

(2) Projection (Π):

- Retrieves specific columns from a relation, remove duplicates
- This is unary Relational operator

Notation: $\Pi_{<col1, col2, ...>}(Relation)$

eg: $\Pi_{Name}(Employee)$

Name
Alice
Bob
Carol
Dave

(3) Rename (ρ):

- We can give alternative name to any column or any table query expression using operator called ReName
- Rename operator denoted by the lowercase Greek letter rho (ρ)

Syntax: $\rho_{(New\_Name)}(Input\_table\_Name)$

eg: ρ(Emp, Employee)

Renames Employee table to Emp

SET Operations

(1) Union operation (∪):

- This operation combines rows of 2 two relation and remove duplicate values
- both relation must have same number of column and same data type.
- denoted by (Query Exp 1) ∪ (Query exp 2)

eg: Student table

Roll No	Name	Class
1	Alice	10
2	bob	10
3	Carol	9

Teacher table

TID	Name	subject
101	David	Eng
102	Bob	Math

eg: $\Pi_{Name}(Student) \cup \Pi_{Name}(teacher)$

Result: Name

Alice
bob
Carol
david

(2) Intersect operator ($\cap$):

Return the tuples (rows) are common to both relation

- both relation must be union-compatible
- denoted by (query exp1) $\cap$ (query exp2)

eg: draw both table student & teacher Here

Result: $\Pi_{Name}(Student) \cap \Pi_{Name}(teacher)$

Name
Bob

(3) Difference (−):

- Returns tuples that are in one relation but not in the other.

denoted by (query exp 1) − (query exp 2)

eg: ΠName(Student) − ΠName(Teacher)

draw both tuple

Result: Name

Alice
Carol.

Binary operations

(1) Cartesian product (×)

Combines each tuple of one relation with every tuple of another

- Results in large no. of tuple.
- denoted by R × S

eg: Relation A (Student)

Roll.No	Name
1	Alice
2	Bob

- Relation B (course)

CourseID	CourseName
C1	DBMS
C2	OS

~~Result~~ ~~A × B~~

~~Roll No~~

Result: $A \times B$

Roll No	Name	CourseID	Course Name
1	Alice	C1	DBMS
1	Alice	C2	OS
2	Bob	C1	DBMS
2	Bob	C2	OS

(2) Division ($\div$)

- it is mainly used in all queries
- It must have common column.

denoted by $A \div B$

eg:

Student_course table

Student	Course
Alice	DBMS
Alice	OS
Bob	DBMS
Carol	DBMS
Carol	OS

PAGE No.
DATE / /

Course table

DBMS
OS

Result: Student $(A \div B)$

Alice
Bob

(3) Join operations (⋈)

- it is used to combine two tables based on common attribute

- It allow us to combine rows of two table that have matching value in one or more table.

- In result we will get new table which has combine data from both the relation.

* Types of Joins

- Natural join
- Inner join / Theta Join
- Outer join
  - Left, Right, Full

(1) Natural Join (⋈)

- Natural join can join table based on the common columns in the tables being joined.
- It join two relations by matching column with the same name and removing duplicate attributes.

eg:

Student table

id	Name
1	Alice
2	Bob

course table

id	course
1	dbms
2	OS

A ⋈ B

id	Name	course
1	Alice	DBMS
2	Bob	OS

(2) Inner Join / Theta join (⋈θ)

- Theta join will combines tuples from multiple relations if they satisfy the specified join condition
- This join condition is also called as theta and denoted by the symbol θ

PAGE No.
DATE / /

eg:

A

ID	Name
1	Alice
2	Bob

B

I.D	Course
1	DBMS
2	OS

A ⋈ₒ A.ID = B.ID. B

ID	Name	Course
1	Alice	DBMS

(3) Outer Join: An outer join returns matching rows between two relation and also include non-matching rows, with NULL values filled for missing attributes.

(1) Left Outer Join:

- Returns all rows from the tables, and matched rows from the right table.
- if there is no match, Fills 'NULL' in right-side Column

eg: Student table.

Id	Name
1	Alice
2	Bob
3	Carol

Course table:

STD	Course
1	DBMS
2	OS

Result: ⋈

Id	Name	Course
1	Alice	DBMS
2	Bob	OS
3	Carol	NULL

(2) Right outer join (⟖)

- Returns all rows from the right table, and matched rows from the left
- If no match, fill NULL in left-side columns

eg: Students ⋈ course

ID	Name	Course
1	Alice	DBMS
2	Bob	OS
NULL	NULL	Maths

(3) Full outer Join (⋈)

- Returns all rows from both tables, matching where possible.
- Fill NULL if non-matching parts on both sides

eg: if student had "Carol" (not in course) and course had "Maths" (not in student)

ID	Name	course
1	Alice	DBMS
2	Bob	OS
3	Carol	NULL
NULL	NULL	Maths