- What is Data?

Data is a collection of raw, unorganized facts and details like text, observations, figures, symbols and descriptions of things, etc.

In other words, data does not carry any specific purpose and has no significance by itself.

Data is stored in computer memory in the form of bits and bytes.

The information regarding any image, text, etc all are stored in binary format in memory.

Data can be recorded and doesn't have meaning unless processed.

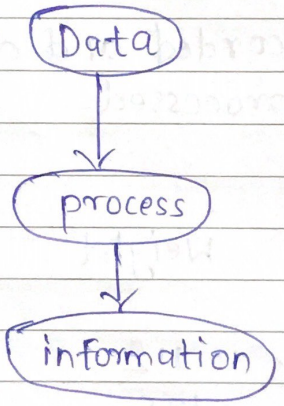
Ex:	BMI	Weight	height
	21.66	112	61
	21.79	135	66
	22.34	213	73

When, we are given above values without the information that what those values represent, so we cannot conclude anything out of that values.

But, as we get that what those values represent, so in the above example they represer BMI, height and weight of person, so we are able to conclude that there are 3 persons and with help of values of height and weight we can calculate the BMI value.

So, the above given data was not useful untill we process the data to gather some useful information out of that.

So, there is one step to gather information from data that is to process/interpret that data.



→ Types of Data:

1) Quantitative : i) Numerical form
ii) Weight, volume, cost of an item.

b) Qualitative : i) Descriptive, but not numerical.
ii) Name, gender, hair color of a person.

→ What is information ?

The data we get after processing is called as information.
So it is an organized and structured data. so it is an processed data and make some sense to us.

→ Ex:- Suppose, there is a new Business and the business owner wants to increase there influence on social media platforms to increase the sell and also people should be able to recognize them.
So, they told an advertising company and the advertising company creates post and publish it on social media platform like insta, fb and linkedin.
So, suppose they get 8 likes on their post on fb, 800 likes on insta and 2 likes on linkedin.
So, here the data can be called as the no.of counts of likes on the post on different social media platform.

so, after seeing the data we can conclude that the new business has got maximum like on insta for their business. so they should only focus on insta post to increase their sell and presence, rather than wasting time on other platforms.

So, after processing the data we get the information which is useful in decision making.

Ex: Suppose there are 2 persons and both of them starts a restaurant.

	Person A	Person B
→	10 dishes	10 dishes
→	No feedback	feedback
→	No Data	No Data

So both the person starts the restaurant with 10 dishes at start and Person A does not collect any feedback from customer and person B collects.

So, after collecting the data the person B would gather some kind of data and it can be called as review of customer on dishes. so, after processing the data the person B has came to know that it's only 5 dishes are doing well or the other way only 5 dishes most people like.

So person B would borrow more raw materials for that dishes and reduce it for the other 5 dishes.

But on the other hand person A does not take any feedback from customer so he would not generate any data that which dishes are most liked by d people, so he would borrow same quantity of raw materials for all dishes, which would increase his expenses and results in less profit as compared to person B, who manages the data about the dishes of his restaurant.

→

Data	information
1) Data is a collection of facts.	1) Information puts those facts into context.
2) Data is raw and unorganized.	2) Information is organized.
3) Data on it's own is meaningless.	3) When it's analyzed and interpreted it becomes a meaningfull information.
4) Data is not sufficient for decision making.	4) But information is useful in decision making.

→ What is database?

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently.

It is also used to organize the data in the form of table, schema, views and reports etc.

Ex: The college database organizes the data about the admin, staff, students and Faculty, etc.

using the database management system we can easily retrieve, insert and delete the information.

→ What is DBMS?

1) A DBMS is a collection of interrelated data and a set of programs to access those data. The collection of data, usually refered to as database, contains information relevant to an enterprise. The primary goal of ~~enterprise~~ DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

2) A DBMS is a software itself. It is used to perform different operations like CRUD on data.

→ Paradigm shift from file system to DBMS:

File system manages data using file on hard disk. Users are allowed to create, delete and update the files according to their requirement. Let us consider the example of file based university management system.

Data of students is available to their respective departments, Acadamics sections, result section, accounts sections, hostel office etc. Some of the data is common for all section, like name, roll no, contact no, etc. And some data is available to a particular section like hostel allotment number which is a part of hostel office. The issues with the system is,

1) Redundancy of data:

Data is said to be redundant if the same data is copied at many places. If the students wants to change their phone number, he has to get it updated in various section. Similarly old records must be deleted from all sections representing that student.

2) Inconsistency of data:

Data is said to be inconsistent if multiple copies of the same data do not match each other. If the phone number is different

in different sections it will be inconsistent.

3) Difficult data access:

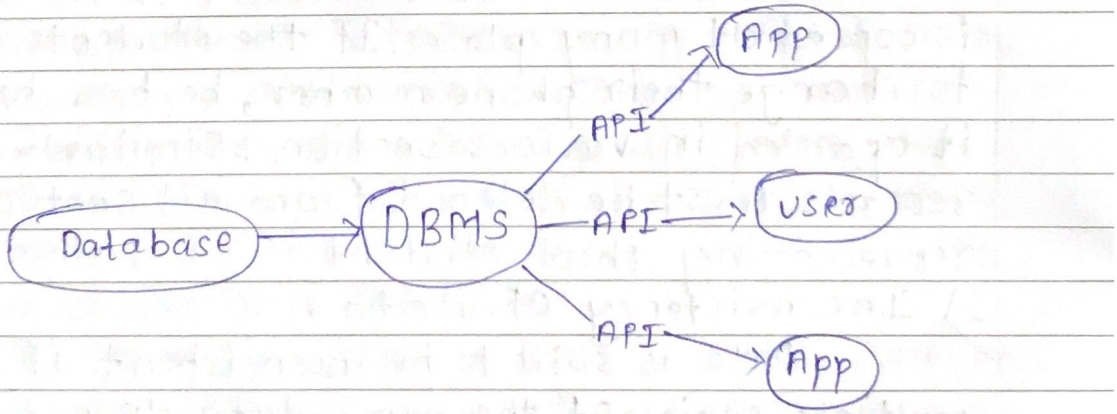
A user should know the exact location of the file to access data, so the process is very cumbersome and tedious. If the users wants to search the student hostel allotment number of a student from 1000 unsorted student's record how difficult it would be.

4) Unauthorized access.

5) No concurrent access:

The access of same data by multiple users at same time is called concurrency.

6) No Backup and Recovery.



→ Advantages of DBMS :-

1) controls database redundancy :-
It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in database.

2) Data sharing :-
In DBMS, the authorized users of an organization can share the data among multiple users.

3) Easily Maintainance :-
It can be easily maintainable due to the centralized nature of database system.

4) Reduce time :-
It reduces development time and maintainance speed.

5) multiple user interface :-
It provides different types of user interface like GUI, or application program interfaces.

→ Disadvantages of DBMS:-

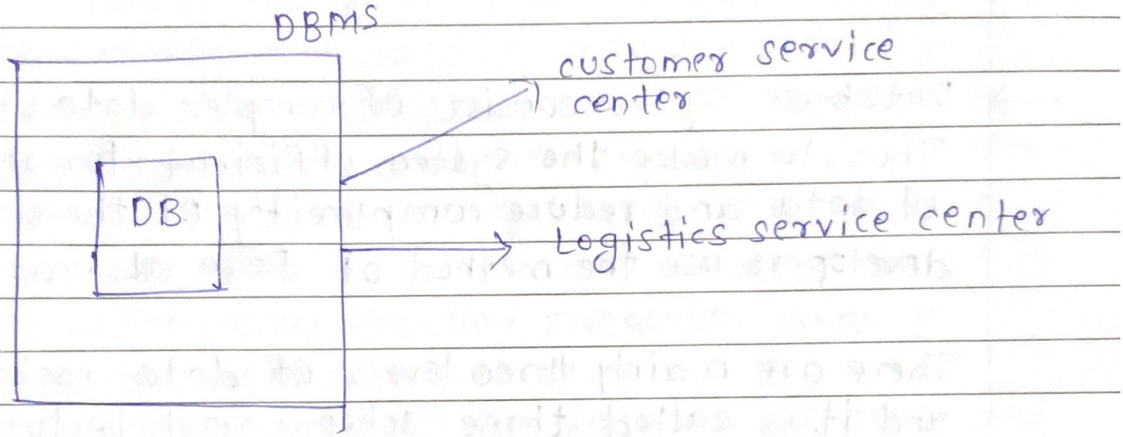
1) cost of hardware and software.
2) size : It occupies a large space of disks and large memory to run them efficiently.
3) complexity.
4) Higher impact of failure :-

→ Data Abstraction in DBMS :-

Data abstraction in DBMS refer to the hiding of unnecessary data from end-user. Database systems have complex data structures and relationships. These difficulties are masked so that users may readily access the data, and just the relavant section of the database is made accessible to them through data abstraction.

Ex: If we want to retrieve any email from Gmail, we don't know where that data is physically kept, such as in India or US, or what data model was utilized to store it. These things are not essential to us. Only our email is of interest to us.

Ex: let's take an example of a company like Amazon, so these company will have it's own database where it will store details of comstumer like their Name, address, phone no, age, credit card details for payment, UPI id, and etc.



And the company have different sections to manage all the things, ~~like~~ and also these sections would have access to their database. So, the customer service center would have access to the data like name, add, phone no, products boughts by customer, etc.

In the database all the details of the customer will be saved but these different service center would not need all the details so with the help of abstraction in DBMS we show the details that are of use to the specific user and hide all the other details.

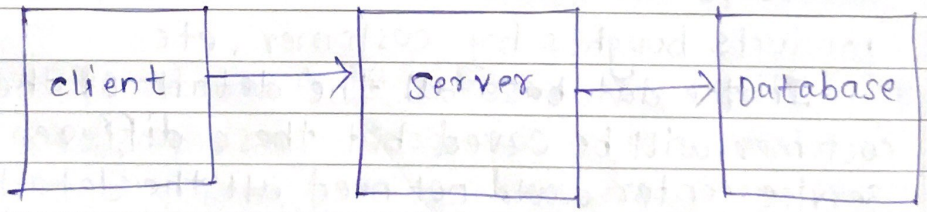
So, if we compare these system with File Management system, so we can see that in FMS we would have to create different files

for different section and all files will have some common information like name, add, phone no of the customer, but in DBMS all the details are saved once and using abstraction we show the specific details which are of use to that user.

→ Database system consists of complex data structures. Thus, to make the system efficient for retrieval of data and reduce complexity of the users, developers use the method of Data abstraction.

There are mainly three levels of data abstraction and it is called three schema architecture.

External level / View level | conceptual / logical level | Internal level



To achieve data independence we divide the data abstraction into three levels. Data independence means users and data should ~~be at~~ not directly interact with each other.

The user should be at different level and the data should be at different level. By doing so, data independence is achieved.

1) Physical level :-

This is the lowest level of abstraction. It tells us how the data is actually stored in memory. The access methods like sequential or random access and file organizations methods like B+ trees and hashings are used for the same. Actually it is decided by developers or database application programs how to store the data in database.

So overall all the database is described in this level.

for ex: customer information is stored in tables and data is stored in the form of blocks of storages such as bytes, gigabytes, etc.

2) Logical level :-

Logical level is the intermediate level. It describes what data is stored in the database and what relationship exist among data. It tries to describe the entire or whole data because it describes what tables to be created and what are the links among those tables that are created.

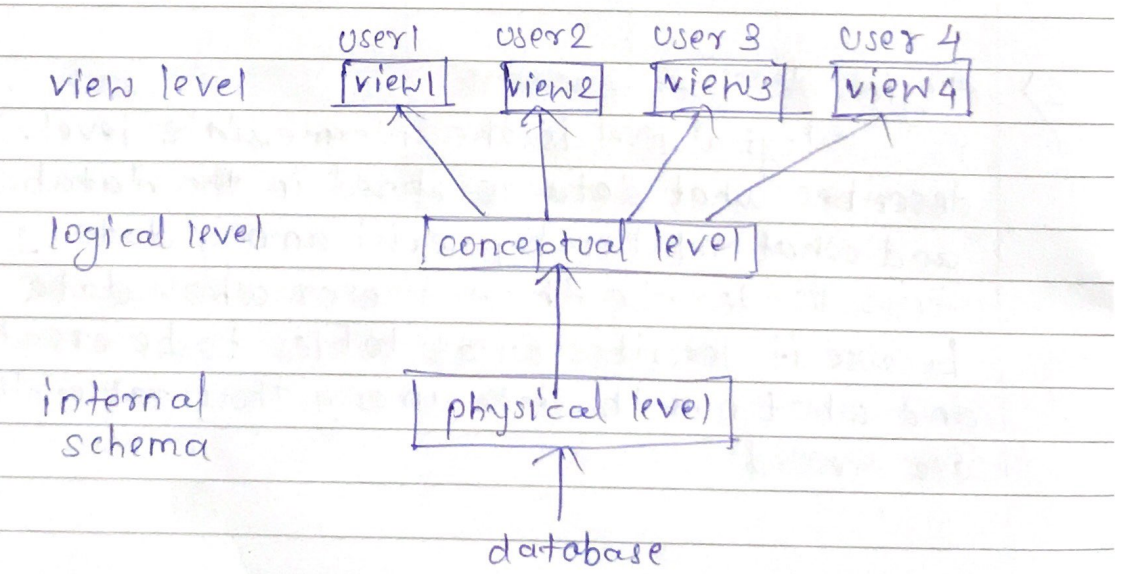
Logical level is used by developers or database administrators (DBA).

3) View or External level :-

It is the highest level. In view level, there are different levels of views and every view only defines a part of entire data. It also simplifies interaction with user and it provides many views or multiple views of the same database.

View level can only be used by all users. Each view schema, describes the database part that a particular group is interested and hides the remaining database from that user group.

At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe different view of database.



→ Ex:

Suppose we have to store information of students in the database, which includes name, phone no, address and ~~bach~~ batch no of students.

So in the physical layer or level all these information will be stored for ex, in file, in these way —

Name, ~~roll~~ phone no, address, batch,
Name, phone no, address, batch,
. . . . .
. . . . .

file

So, here the data is not organized and it is very difficult to understand.

So, in the next level i.e logical level a table is being created with the help of data which is present in physical layer. So, the mapping of data from physical layer to logical layer is done with the help of logical schema.

Student

Sr. no	Name	phone no	address	Batch no
1	—	—	—	—
2	—	—	—	—

we have to define all these at this level while we are creating a database. Though the data is stored in database but the structure of the tables like student table are defined here in this level.

Now lets assume we had to add one more column in previous table for cource selected by student and it requires a separate table which include all information regarding that course like course ID, name, duration, etc. so, as we can see that the two tables are now interrelated.

So, there's one more responsibility of logical level is that it defines how tables are related to each other.

And at the view level the user will be able to see the data which is of it's use.

→ Instance in DBMS:-

Instances are the collection of information stored at particular moment. The instance can be changed by certain CRUD operation as like addition, deletion of data.

Ex:-

Let's suppose we have a table of students in our database. And we have data of 50 students in the table. So the intance of database has 50 records now. And if we add 50 more students to our database then the instance would be 100 at that time.

→ Schema of Database:-

Schema is the overall description of database. The basic structure of how the data will be stored is called schema. It contains a list of attributes (attributes means the name of the column in the tables) that informs the database engine that how the data is organized and how the elements are related to each other.

In actual the data is physically stored in files that may be in unstructed form, but to retrieve it and use it, we need to put it in a structured form. To do this, a database schema is used. It provides knowledge about how the

data is organized in a database and how it is associated with other data.

The schema does not physically contain the data itself, instead it gives information about the shape of data and how it can be related to other tables or models.

→ A database object schema includes following

1) Consistent formatting for all data entries
2) Database objects and unique key for all data entries.
3) Tables with multiple columns, and each column contains its name and datatype.

→ The main type of database schema is logical schema :-

→ Logical database schema :-

The logical database schema specifies all the logical constraints that need to be applied to the stored data. It defines the views, integrity constraints and table. Here the term integrity constraint define the set of rules that are used by DBMS to maintain the quality for insertion and update of data.

The logical schema represents how the data is stored in the form of tables and how the attributes of a table are linked together.

→ The main purpose of data abstraction is to achieve data independence in order to save time and cost required when the database is modified or altered.

1) Physical level data independence:-
If refers to the characteristics that if we change or modify physical schema there would not be any change in logical schema. So changing from sequential to random access files is one such example.
The alteration or modifications to the physical structure may include
1) utilizing new storage device.
2) Modifying data structure used for storage.
3)

2) Logical level data independence:-
It refers to the characteristics that modifying logical schema would not affect view level schema.

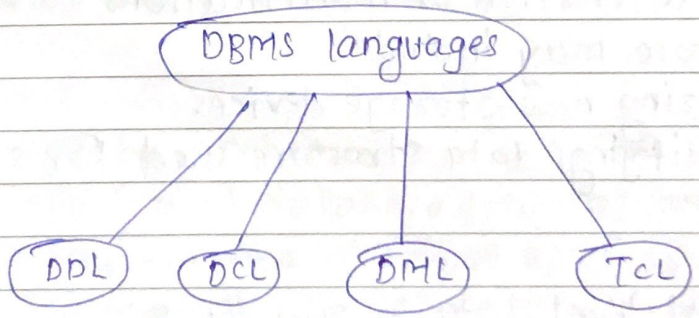
→ Data Model :-

Data models define how logical structure of a database is modelled. Data models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside system.

Ex: ER Model, Relational model, etc.

→ Database languages :-

1) A DBMS has appropriate languages and interfaces to express database queries and updates.
2) Database languages can be used to read, store and update the data in a database.



1) Data definition language :- (DDL)

1) It is used to define database structure or patter i.e schema.
2) Used to define schema, tables, indexes, constraints, etc in database.
3) Using DDL statements you can create skeleton of database.

Some of the tasks that come under DDL:
1) Create
2) Alter
3) Drop
4) Truncate
5) Rename
6) comment

2) Data Manipulation Language (DML) :-

It is used for accessing and manipulating data in a database. It handles user requests.

Some of the tasks that come
1) Retrieval of data stored in DB.
2) Insertion of new information in DB.
3) Deletion -/- .
4) updating -/- .

→ How is DB accessed from application programs:-

If our App is written in Java or C/c++ then it cannot directly interact with DB. Because both the languages are different. So to ensure soft communication between these two languages there is a package in Java called JDBC and ODBC for C/c++ which converts the Java statements to DB language and vice versa.

→ Database Administrator (DBA):-

1) A person who has central access of both the data and the programs that access those data.

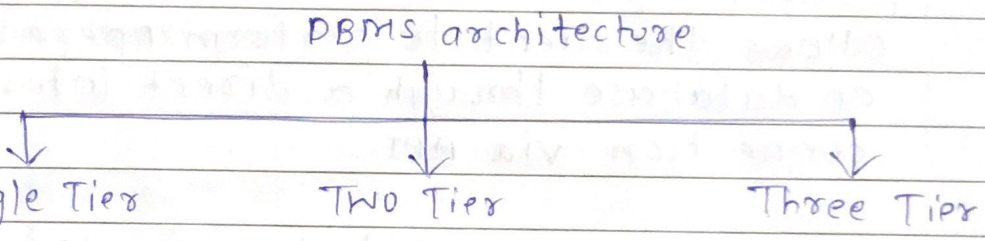
→ functions of DBA:-

1) Schema Definition.
2) Storage structure and access methods.
3) Schema and physical organizations modification
4) Authorization control.
5) Routine maintainance.

→ DBMS architecture :-

DBMS are divided into multiple levels of abstraction for proper functioning. These modules / layer describes the functioning and design of DBMS.

Since a DBMS is not always directly accessible by the user or an application, we can maintain it with the help various architectures based on how the user is connected to DB. These architecture follow a tier-based classification, i.e the DBMS architecture is classified depending upon how many layers are present in structure of DBMS.



↳ Single Tier :-

All the DBMS components reside on a single server or platform, i.e the database is directly accessible by the end-user. Because of this direct connection, the DBMS provides a rapid response, due to which programmers widely use this architecture to enhance the local application.

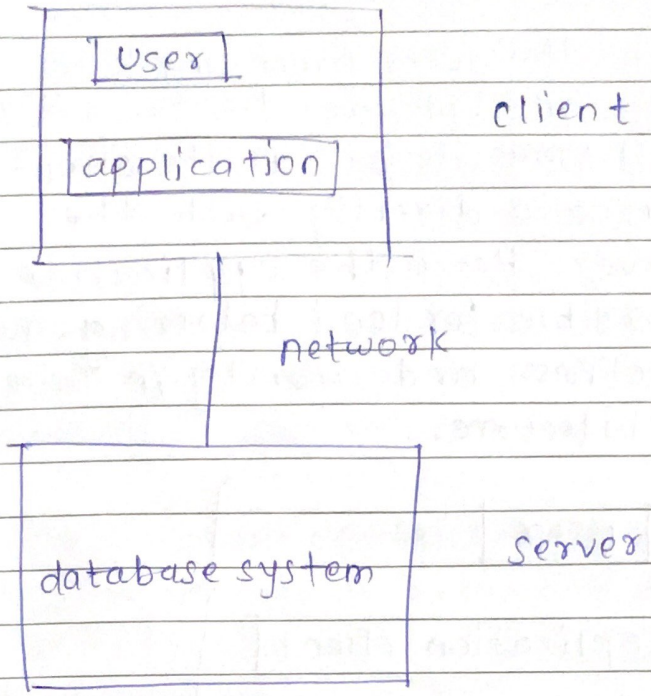
Ex:-

In order to learn SQL, we set up our SQL server and the database on our local system. This SQL server enables us to directly interact with the relational database and execute certain operations without requiring any network connection.

2) Two Tier Architecture :-

In two tier architecture, the server provides the database functionality and it allows the client to perform operations on database through a direct internet connection via API.

The two tier architecture is used when we wish to access the DBMS with the help of an application. client side applications can access database server directly with the help of API calls, making the application independent of database in terms of operation, design and programming.



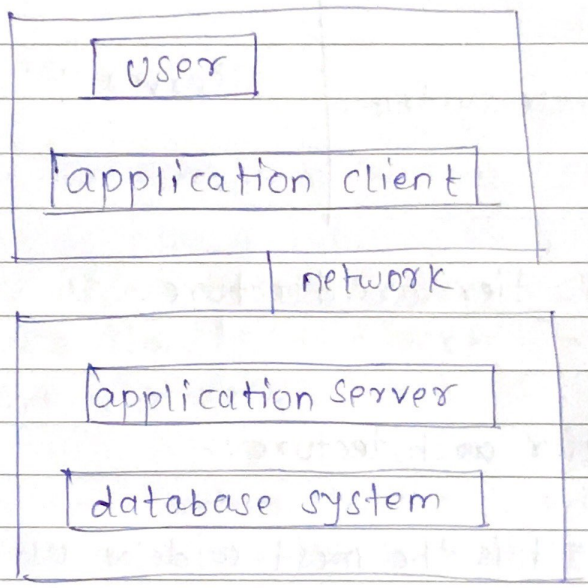
Two tier architecture.

3) Three tier architecture :-

It is the most widely used DBMS architecture in which another layer known as Intermediate or Application layer is added between server and client, system to reduce the query processing load of the server.

Since there is no direct connection between the client and server, all the user requests are handled by application layer, i.e, the requests send by users are checked and verified by intermediate layer before transforing them to server.

"This reduces the query processing load from the server and enhances the security of the overall DBMS design as the client can't communicate directly with the database server. Hence the application layer is responsible for load balancing, query request correctness, and security in Three-Tier DBMS architecture.



→ The main advantage of three-tier architecture:

1) scalability:- since the database server isn't aware of any users beyond application layer and application layer implements load balancing, there can be as many clients as you want.
2) Data integrity: Data corruption is less as app server acts as middleware.
3) security: client can't directly access, DB hence it is more secure.

→ Data Model :-

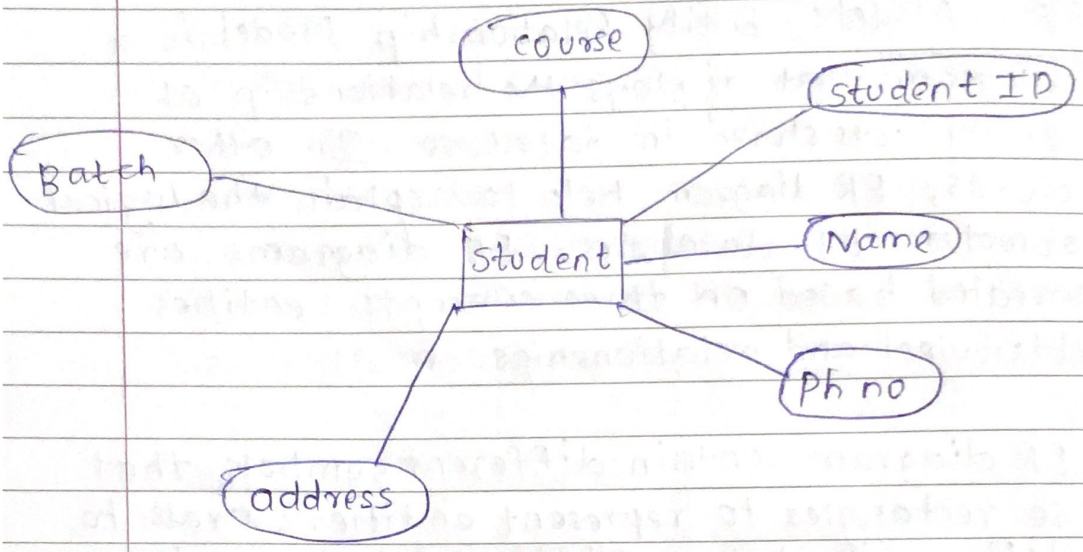
ER - Model: Entity Relationship Model is a diagram that displays the relationship of entity sets stored in database. In other words, ER diagram help to explain the logical structure of databases. ER diagrams are created based on three concepts : entities, attributes and relationships.

ER diagram contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

→ Entity :- An entity is anything in the real world, such as an object, class, person or place. Objects that physically exist and are logically constructed in the real world are called entities. Each entity consist of several characteristics or attributes that describe the entity.

for exc: If a person is an entity its attributes or characteristics are age, name, height, weigh, occupation, address, etc.

→ Entity set :- It is a group of entities of similar kinds. It can contain entities with attributes that ~~are~~ share similar values.
for ex: an car entity set, a bank account entity set.



Every Entity has a unique attribute to identify that entity. In the above Ex we can see that student ID is the unique attribute for student entity type to uniquely identify it.

unique attribute is also called as primary key.

→ Entities are of two types :-

1) strong entity :- A strong entity is an entity type that has a key attribute. It doesn't depend on other entities in the schema. A strong entity always has a primary key and it is represented by a single rectangle in ER diagram.

Ex:- Roll no. identifies each student of organization uniquely and hence, we can say that the student is a strong entity type.

2) Weak entity :- Weak entity type doesn't have a key attribute and we cannot uniquely identify them by their attributes alone. Therefore, a foreign key must be used in combination with its attributes to create a primary key. They are called weak entity because they can't be identified on their own. A weak entity is represented by a double outlined rectangle in ER diagrams.

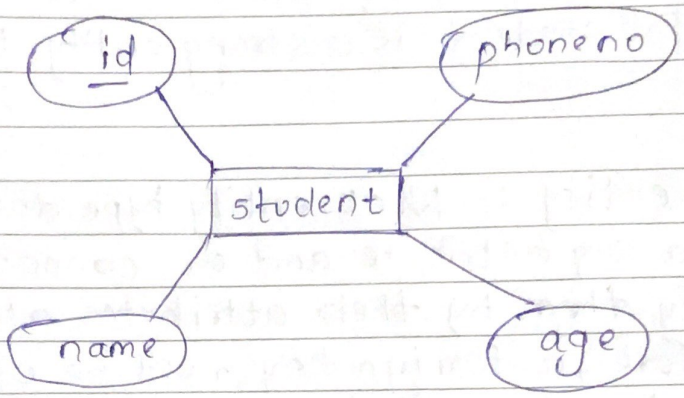
→ Attributes:-

Attributes are the characteristics or properties which define the entity type. In ER diagram the attributes is represented by an oval.

→ Types of attributes:-

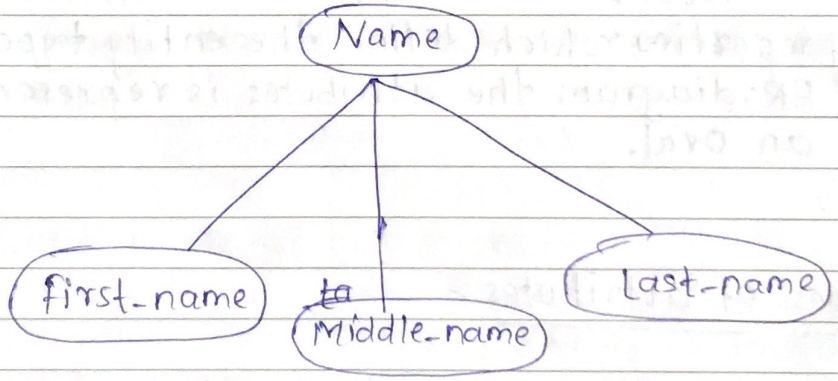
1) Simple attribute:- Attributes that cannot be further decomposed into sub-attributes are called simple attributes. It's an atomic value and is also known as key attribute. The simple attributes are represented by an oval shape in ER diagrams with the attribute

name underlined.



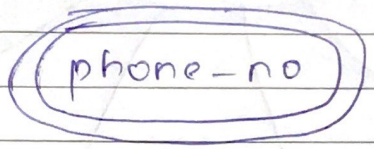
2) composite attribute :

At attribute that is composed of many other attribute and can be decomposed into simple attributes is known as composite attribute.

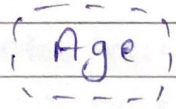


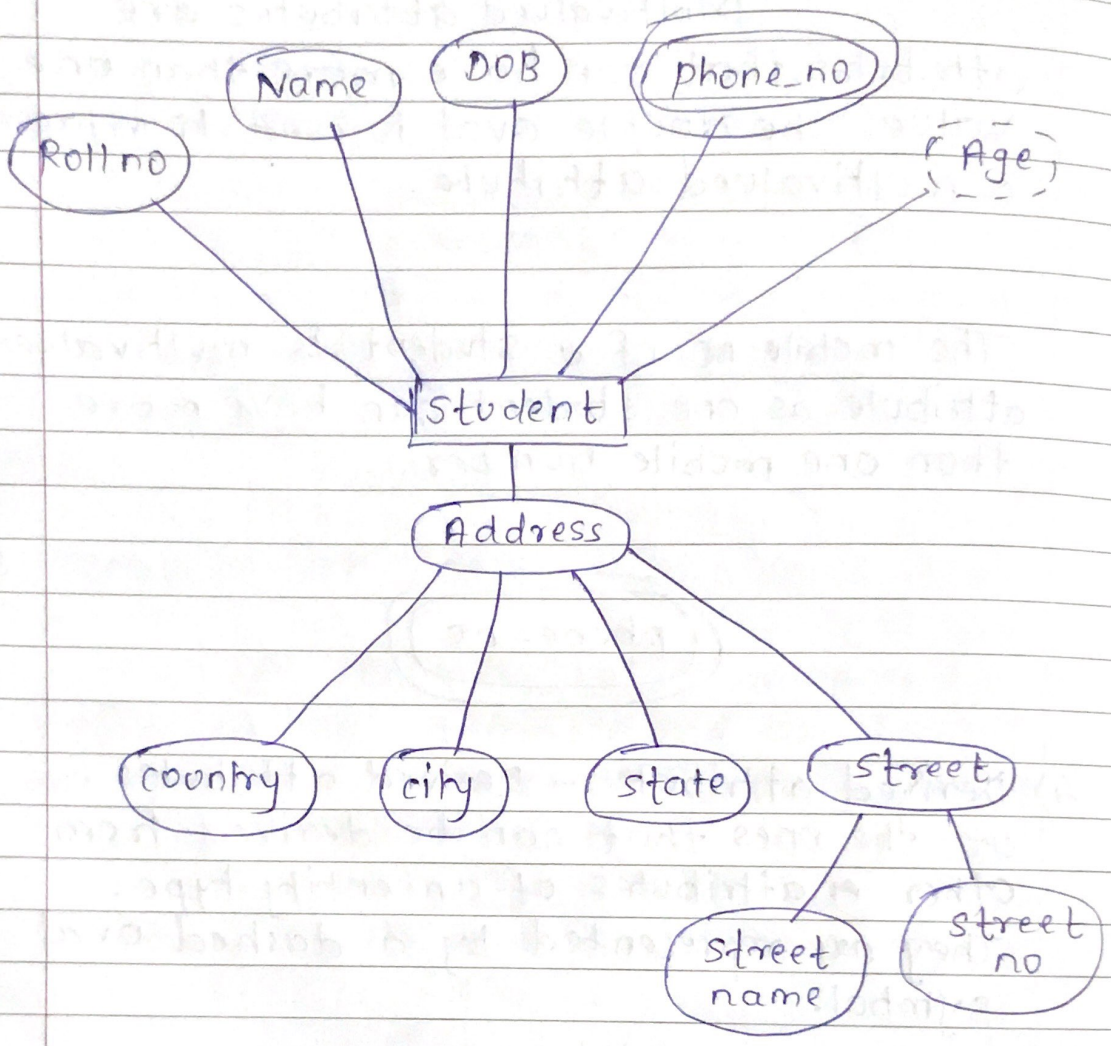
3) Multivalued attribute :-
Multivalued attributes are attributes that can have more than one value. The double oval is used to represent a multivalued attribute.

The mobile no. of a student is multivalued attribute as one student can have more than one mobile number.



4) Derived attribute :- Derived attributes are the ones that can be derived from other attributes of an entity type. They are represented by a dashed oval symbol.





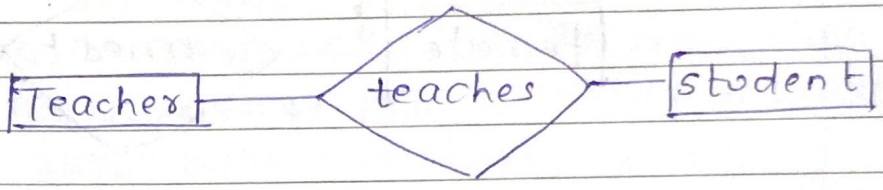
5) NULL Value :-

1) An attribute takes a null value when an entity does not have a value for it.
2) It may indicate 'unknown',
   1) unknown can indicate missing entry, eg, name value of a customer is NULL.

→ Relationship:-

The concept of relationship is used to describe the relationship between different entities. This is denoted by the diamond or a rhombus symbol.

Ex: The teacher entity type is related to student entity type and their relation is represented by diamond shape.



→ Degree of Relationship:-

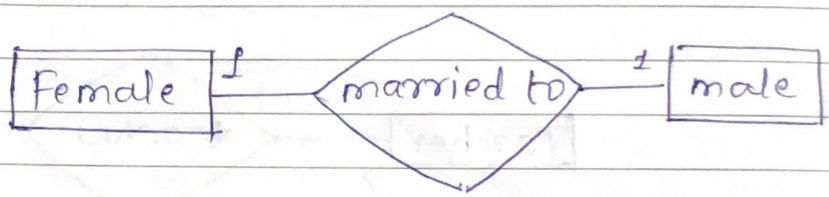
1) No. of entities participating in relationship.
2) Unary, only one entity participate, eg Employee manages employee.
2) Binary two entities participate, eg student takes course.
3) Ternary relationship, three entities participate, Eg employee works on branch, employee works on job.

1) Mapping cardianility

No. of entities to which another entity can be associated

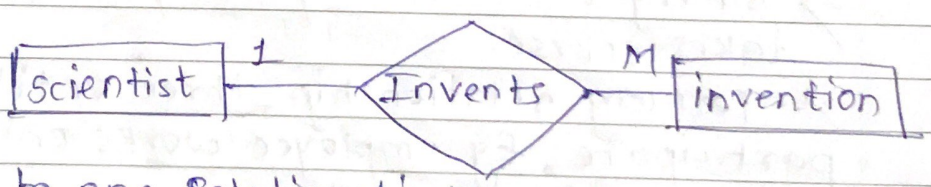
1) one-to-one Relationships :-

When only one instance of an entity is associated with the relationship to one instance of another entity, then it is known as one to one relationship. For ex, let us assume that a male carry One female and a female can carry one male. Therefore the relation is one-to-one.



2) one to many relationship :-

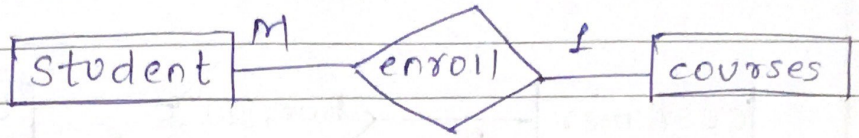
If only one instance of the entity on the left side is linked to multiple instances of entity on right side then this is considered a given one-to-many relationship.



3) Many to one Relationship :-

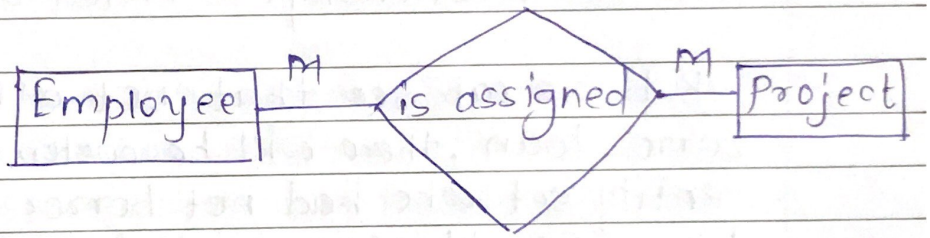
multiple

If ~~only one~~ instance of entity on left side of the relationship is linked to one ~~multiple~~ instances of entity on right side, then it is called Many to one.



4) Many to Many relationships :-

If multiple instances of the entity on the left are linked by relationships to multiple instances of entity on right, then is considered a many-to-one relationship.

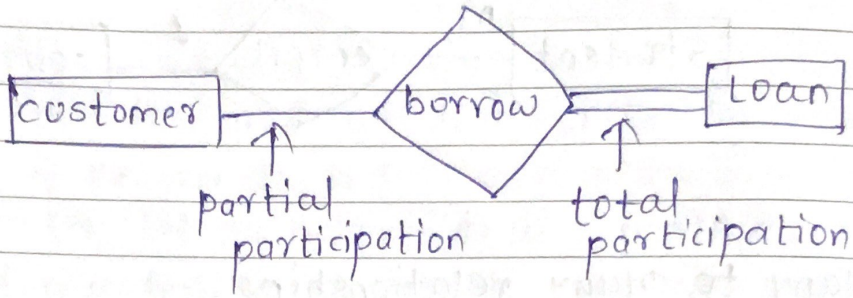


2) Participation constraints :-

Types, Partial and total participation

1) Partial participation :- Not all entities are involved in relationship instance.

2) Total participation :- each entity must be involved in atleast one relationship instance.



Here, the customer represents customer entity set and loan represents loan entity set. So, we can see that there is a relationship between customer and loan, and the relationship is called as borrow.

But, we can see that not all customers borrow loan, there will be customers in the entity set who had not borrowed any loan. So, there is partial participation of customer entity set.

And we can see that there will not be any loan which has not taken by customer. So, there is Total participation of loan entity set.

→ Extended ER features:-

1) Specialization:-

1) In ER Model we may require to subgroup an Entity set into other Entity set that are distinct in some way with other entity sets.

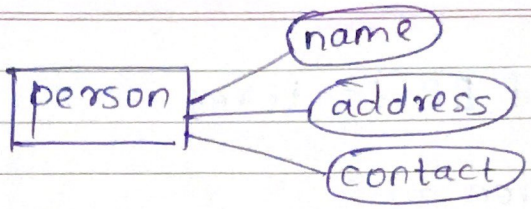
2) Specialization is splitting up the entity set into further sub entity set on the basis of their functionalities, specialities and features.

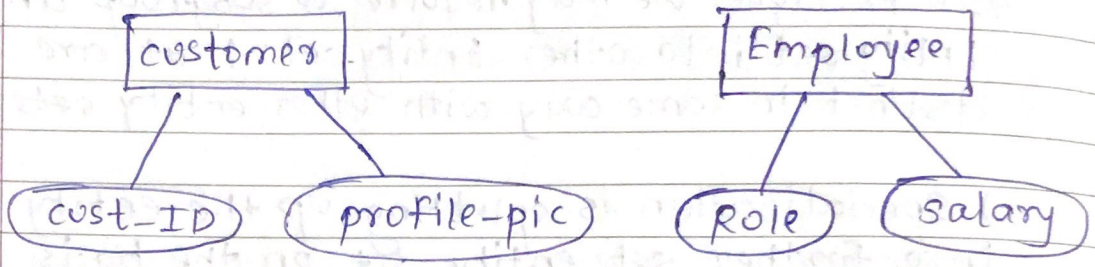
3) It is a Top down approach

Ex:- Person entity set can be divided into customer, student, employee. Person is a superclass and other specialised entity sets are subclasses.

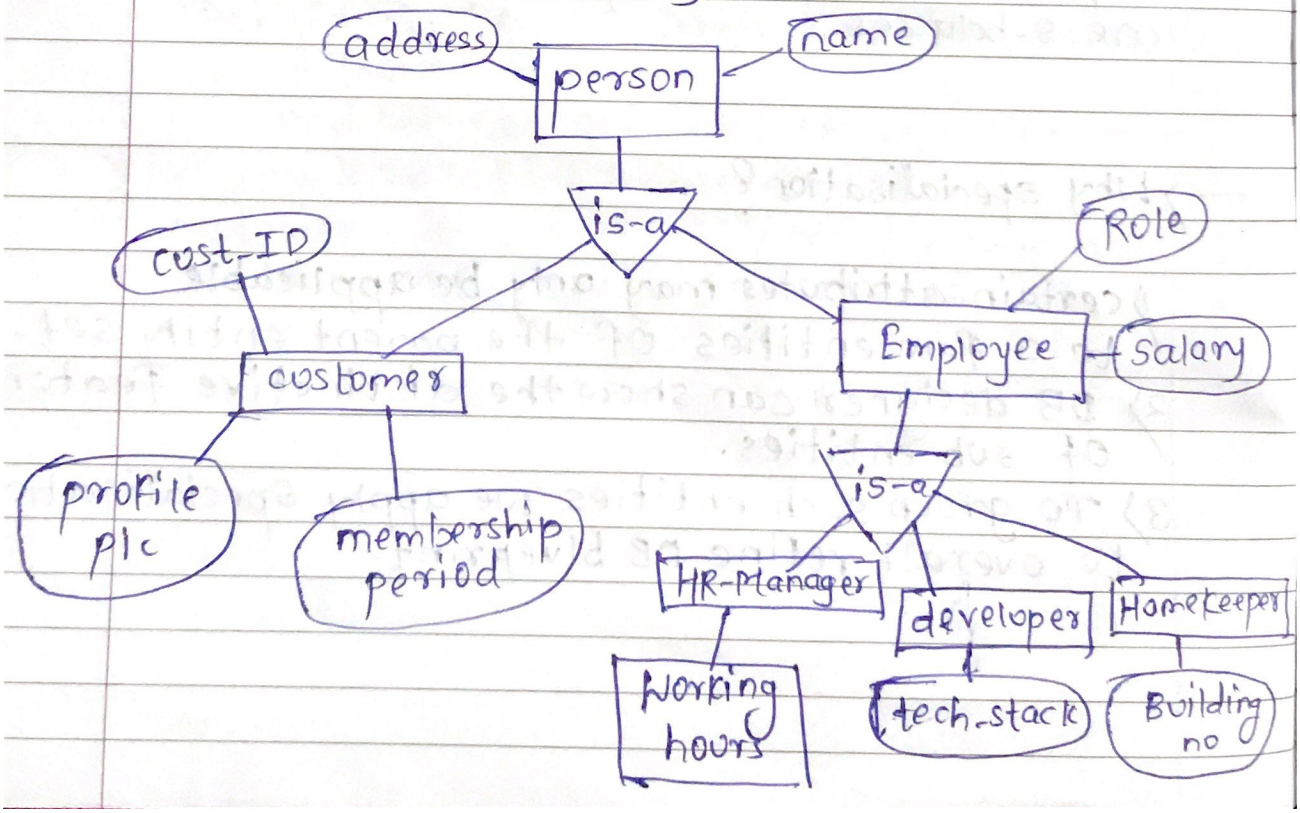
→ Why specialisation?

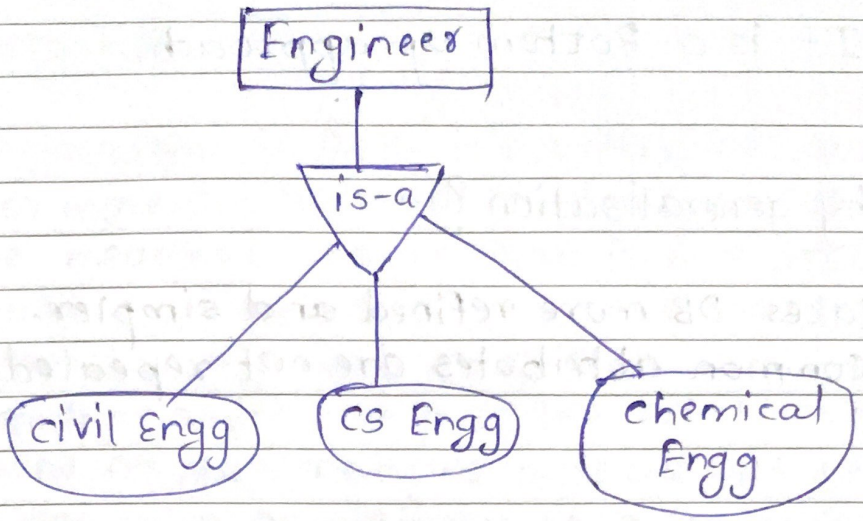
1) certain attributes may only be applicable to a few entities of the parent entity set.
2) DB designer can show the distinctive feature of sub entities.
3) To group such entities we apply specialisation, to overall refine DB blueprint.





→ A person is a customer and a Employee but at a certain time he can be any one. So to write all attributes of a person in a single Entity doesn't make sense. So, here we basically divide the entity set into lower entity set.





2) Generalisation:-

1) It is a just a reverse of specialisation.
2) DB designer, may encounter certain properties of two entities are overlapping. Designer may consider to make a new generalised entity set. That generalised entity set will be a super class.
3) "is-a" relationship is between sub class and super class.
4) Eg: Car, Jeep, bus all have some common attributes, to avoid data repetition for the common attributes. DB designer may consider to generalise to a new entity set, "vehicle".

5) It is a Bottom up approach.

6)

→ Why generalisation?

1) Makes DB more refined and simpler.
2) Common attributes are not repeated.

→ Attribute Inheritance:-

1) Both specialisation and generalisation has attribute inheritance.
2) The attributes of higher level entity set are inherited by lower level entity set.
3) Eg: customer and Employee inherit the attribute of a person.

→ Participation Inheritance:-

If a parent entity set participates in a relationship then its child entity set will also participate in relationship.

→ Aggregation:-

Aggregation in DBMS is a process of combining one or more entities into a single one that is more meaningful entity. Also it is a process in which a single entity does not make a sense in a relationship and one cannot infer results from that relationship so the relationship of one or more entities acts as one entity. It can also be defined as a procedure for combining multiple entities into one.

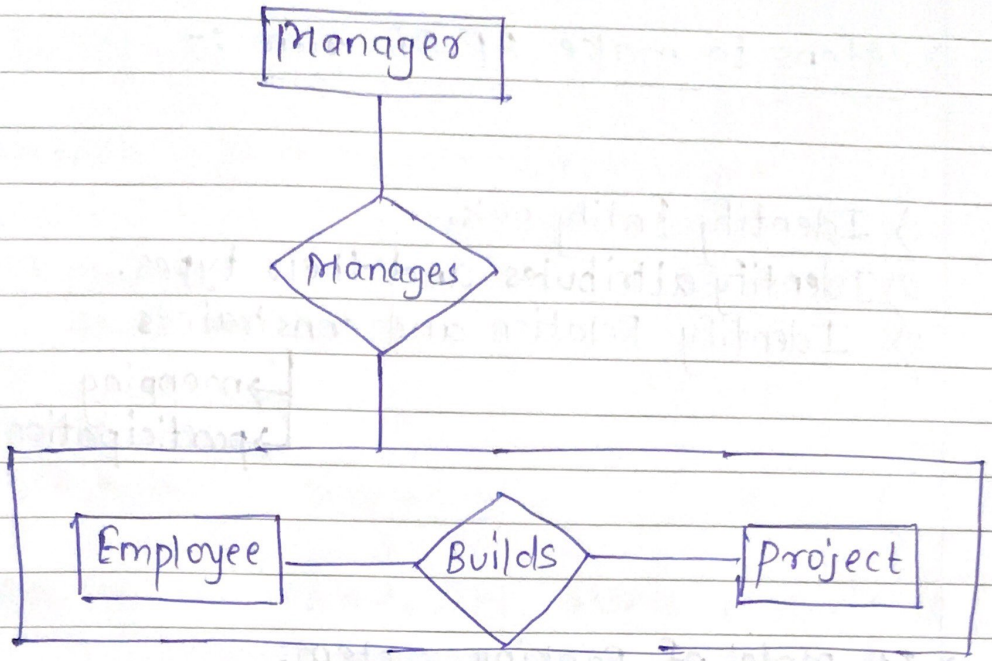
Ex:- let's take an example of an employee and his manager and the employee working on a project which is also managed by his manager. In a real scenario, you know that a manager not only manages employee under them but he has to manage the work undertaken by the employee and the project as well. As discussed if the manager makes a relationship "manages" with "Employee" or "project" entity alone then it will not make any sense because a manager has to manage both employee as well as project, similarly it can be said that if a single entity cannot derive results alone then it can be aggregated with another entity to form a meaningful relationship.

So, in this case a new relationship is formed between two entities, "Employee" and "project", the relationship between two entities acts as one entity.

→ Why use aggregation :-

1) At times the database might contain some entities which are less important, such as two or more entities that can be used to create a relationship with one another. This aggregation process helps to create more meaningful entities which can derive great results.

2) When the ER model is not able to represent the relationship between any of the present entities in the system then that particular entity can be used to create a new relationship with other entity. The resultant entity can be used to create a relationship in ER model.



In the above figure, the Employee and the project are connected to 'Builds' operation. The builds operation is connected to the "manages" operation that represents the aggregation function. The "manages" operation is then connected to the new entity set called as Manager entity.

→ steps to make ER Diagram :-

1) Identify Entity sets.
2) Identify attributes and their types.
3) Identify Relation and constraints
   - → mapping.
   - → participation.

→ ER-Model of Banking system:-

1) Banking system — Branches (Name)
2) Bank → customers.
3) customers → accouts, take loan
4) customer associated with some banker.
5) Bank has employees.
6) Accounts
   - → saving Acc
   - → current Acc
7) Loan originated by branch
   - Loan >/ 1 customers
   - → payment Schedules

→ Entity set :-

1) Branch 2) customer 3) Employee
4) Saving account 5) current account 6) Loan
7) payment (Loan payment/installments)
weak ~~en~~ entity.

→ Attributes :-

1) Branch :- name, city, assets, liabilities.

2) customer :- cust-id, name, address, contact-no
DOB, age.

3) Employee :- emp-id, name, contact-no,
dependent name, years of service, start-date

4) Savings account :- account-no, balance,
interest rate, daily withdraw limit.

5) current account :- account-no, balance,
transaction charge, overdraft amount.

6) Generalized Entity :-

Account :- account no, balance.

7) Loan :- loan number, account.

8) Weak Entity payment :- Payment no, date,
amount.

3) Relation and constraints:-

1) customer ←→ borrows ←→ Loan
M N

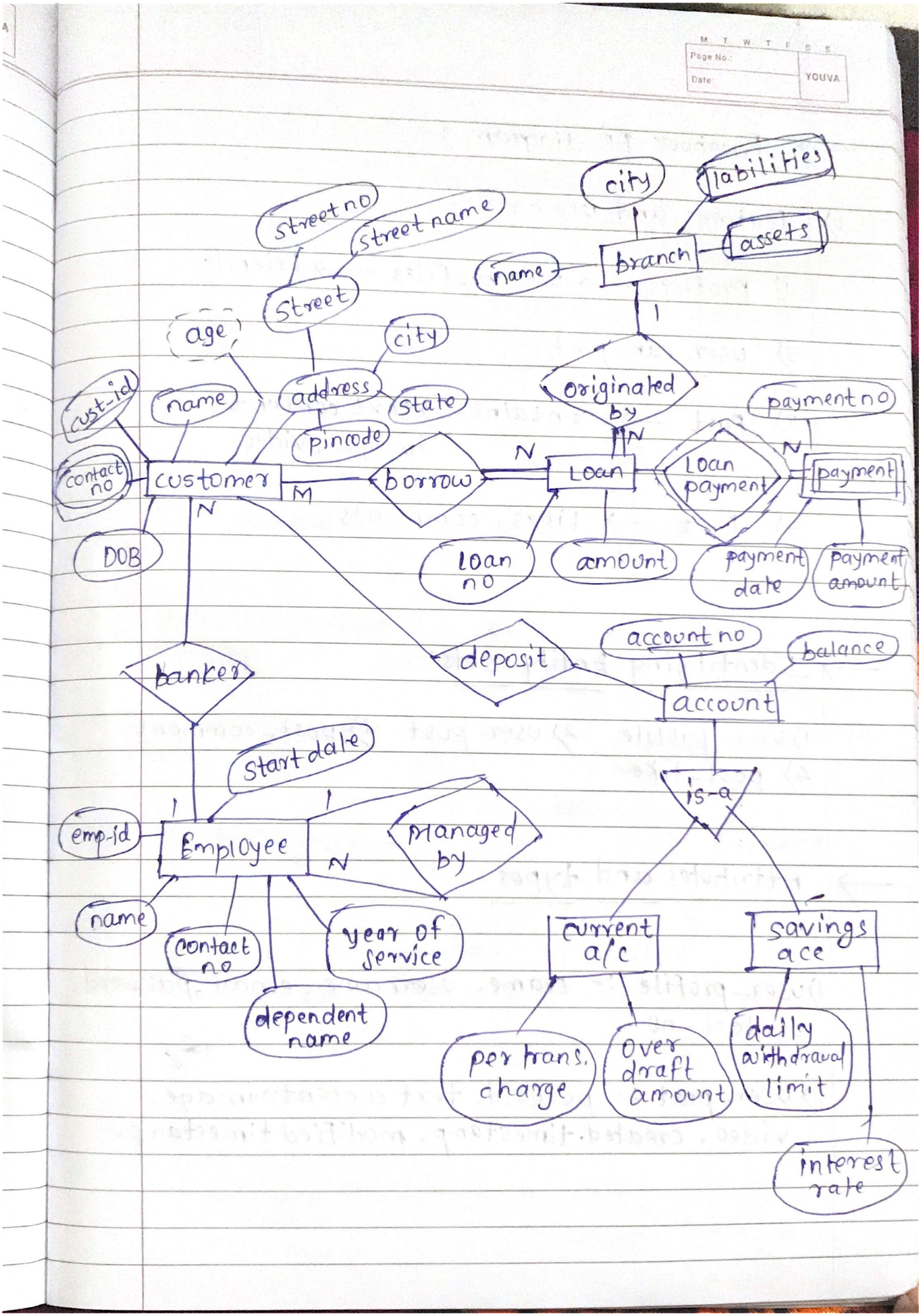
2) Loan N originated by 1 Branch

3) Loan 1 loan-payment N Payment

4) customer M deposit N account

5) customer N banker 1 employee

6) Employee N manage by 1 employee


street no
street name
street
age
city
cust-id
name
address
state
pincode
contact no
customer
DOB
banker
borrow
N
M
N
Loan
loan no
amount
originated by
1
N
branch
name
city
liabilities
assets
Loan payment
payment no
payment
payment date
payment amount
deposit
account no
balance
account
is-a
current a/c
savings ace
per trans. charge
over draft amount
daily withdrawal limit
interest rate
start date
emp-id
Employee
Managed by
name
contact no
year of service
dependent name

→ Facebook ER diagram :-

1) features and use case :-

1) profile → user profiles → friends

2) user can post.

3) Post → contains → text content, images, videos,

4) post → likes, comments.

→ Identifying Entity sets :-

1) user_profile 2) user_post 3) post_comment
4) post_like

→ Attributes and types :-

1) user_profile :- Name, username, email, password, contact-no

2) user_post :- post-id, text content, image, video, created.timestamp, modified timestamp.

member of one of the lower entity sets. Here do not create a table for higher level entity set. Instead, for each lower level entity set, create a table that includes a column for each of the attributes of that entity set plus a column for each attribute of higher level entity sets.

1) Table 1 :- savings-account (account-number, balance, interest-rate, daily-withdrawal-limit)

2) Table2 :- current-account(— / —)

8) Aggregation :-

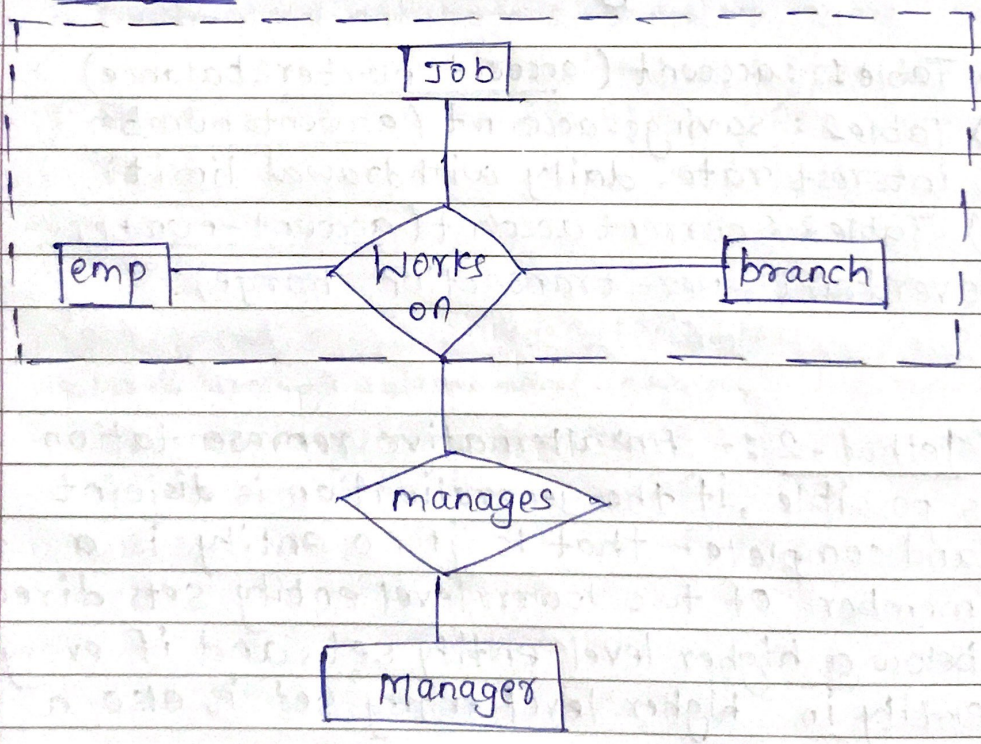


Table manages(mng-id, emp-id, job-id, branch-id)

SQL

SQL is a short form for structured query language.

This database language is mainly designed for maintaining the data in relational database management system (RDBMS)

You can easily create and manipulate the database, access and modify the table rows and columns, etc.

→ SQL used CRUD operations to communicate with DB.

1) CREATE
2) READ
3) UPDATE
4) DELETE

→ What is RDBMS?

software that enables us to implement designed relational model.
Ex: MySQL, MS SQL, ORACLE, IBM, etc.

→ MySQL is open source RDBMS, and it uses SQL for all CRUD operations.

→ MySQL uses client-server model, where client is CLI or frontend that used a services provided by MySQL server.

→ Difference between SQL and MySQL:-

SQL is structured query language used to perform CRUD operations in R-DB, while MySQL is a RDBMS used to store, manage and administrate DB using SQL.

→ The SQL commands are mainly categorized into five categories:-

1) DDL -
2) DQL -
3) DML -
4) DCL -
5) TCL -

CREATE	INSERT	COMMIT	SELECT	GRANT
DROP	UPDATE	SAVEPOINT		REVOKE
ALTER	DELETE	ROLLBACK		
TRUNCATE	CALL			
	EXPLAIN CALL			
	LOCK			

→ DDL (Data Definition Language):-

DDL or data definition language acutually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. DDL is a set of SQL commands used to create, modify and delete database structures but not data.

1) CREATE DATABASE IF NOT EXISTS db-name;

2) USE db-name; (need to execute to choose on which DB CREAT TABLE etc commands will be executed.)

3) DROP DATABASE IF EXISTS db-name;

4) SHOW DATABASES;

5) SHOW TABLES;

6) ALTER : This is used to alter the structure of database.

7) TRUNCATE : This is used to remove all records from a table, including all spaces allocated for the records are removed.

→ DQL (Data Query Language):-

1) Syntax :- SELECT <set of column names> FROM <table-name>;

2) order of execution from Right to left.

3) Can we use SELECT keyword without using FROM clause ?

1) Yes, using DUAL Tables.
2) Dual tables are dummy tables created by MySQL, help users to do certain obvious actions without referring to user defined tables.
3) eg, SELECT 55+11;
SELECT now();
SELECT ucase(); etc

→ What is clause in SQL?

SQL is a query language that is used to query the given data and the desired or required data is returned from the database. SQL is widely used for multiple operations that are related to the data and to achieve that there are various methods or processes available in SQL.

A clause in SQL is a built in function that helps to fetch or access the required records from the database table.

A clause in SQL receives a conditional expression i-e, a column name or some terms that involve the columns and the functions or the clauses calculate the result based on the given statements in the expression.

1) WHERE :-

1) Reduce rows based on given conditions.
2) Eg, SELECT * FROM customer WHERE age>18.

2) BETWEEN:-

1) SELECT * FROM CUSTOMER WHERE age between 0 AND 100.

3) OR:-

SELECT * FROM Worker WHERE DEPARTMENT='HR' OR DEPARTMENT='Admin' OR DEPARTMENT='Account';

4) IN:-

SELECT *FROM Worker WHERE DEPARTMENT IN ('HR', 'Admin');

5) NOT:-

SELECT * FROM Worker WHERE DEPARTMENT NOT IN ('HR', 'Admin');

6) IS NULL:-

SELECT * FROM Customer WHERE Pincode is NULL;

7) Pattern searching / Wildcard ('%', '_')

1) '%' any number of character from 0 to n. similar to '*' asterisk in regex.
2) '_', only one character.

Ex:-

i) SELECT * FROM Worker WHERE first-name Like '%i%';

→ Returns all name which includes i.

ii) SELECT * FROM Worker WHERE first-name LIKE '_i%';

→ Now i, should come on 2nd position.

8) ORDER BY

1) sorting the data retrieved using WHERE clause.
2) ORDER BY <column_name> DESC;
3) DESC = Descending and ASC = Ascending.

Ex:- SELECT * FROM Worker order By salary;
gives all the columns in ascending order of salary.

SELECT salary FROM Worker ORDER BY salary DESC;

gives salary column in descending order.

9) DISTINCT

1) finds distinct values in the table.
2) SELECT DISTINCT (col-name) FROM table-name;

Ex:-

SELECT DISTINCT department from Worker;

10) GROUP By :-

1) GROUP By clause is used to collect data from multiple records and group the result by one or more column. It is generally used in SELECT statement.
2) Groups into category based on column given.
3) All the columns names mentioned after SELECT statement shall be repeated in GROUP BY, in order to successfully execute the query.

Ex:-

1) SELECT department, COUNT(department) FROM Worker group by department;

→ It gives the count of all the departments

department	COUNT(department)
HR	2
Manager	4
Accounts	3

2) SELECT department, AVG (salary) From Worker group by department.

3) Min and Max ~~also~~. and SUM functions.

11) GROUP BY Having:

1) out of the categories made by GROUP BY we would like to know only particular thing.
2) Similar to WHERE.
3) SELECT COUNT(cust-id), country from customer GROUP BY country HAVING COUNT(cust-id) > 50;

Ex:-

SELECT department, COUNT(department) from worker group by department having COUNT(department) > 2

→ WHERE Vs HAVING:-

1) Both have same function of filtering the row base on certain conditions.
2) WHERE clause is used to filter the rows from the table based on specified condition.
3) HAVING clause is used to filter the rows from the groups based on specified condition.
4) ~~to~~ HAVING is used after GROUP BY while WHERE is used before GROUP BY clause.

→ constraints in DDL:-

1) Primary key:-

PK is not null, unique and only one per table.

2) Foreign key:-

1) FK refers to PK of another table.
2) Each relation can have any number of FK.
3) Ex:-

```
CREAT TABLE customer(

id INT PRIMARY KEY,
branch_id INT,
Firstname CHAR(50),
Lastname CHAIR(50),
DOB DATE,
Gender CHAR(6));


CREATE TABLE ORDER (

    id INT Primary key,
    delivery_date DATE,
    order_placed_date DATE,
    cust_id INT,
    FOREIGN KEY(cust_id) REFERENCES customer(id));
```

3) UNIQUE

1) Unique can be null, table can have multiple unique attributes.

Ex:-

```
CRETE TABLE customer(
    :
    email Varchar(1024) UNIQUE,
    :
    );
```

4) CHECK

Using the check constraint we can specify a condition for a field, which should be satisfied at the time of entering values for this field.

Ex:-

```
CREATE TABLE student(
    ID int(6) NOT NULL,
    Name Varchar(10) NOT NULL,
    AGE int NOT NULL CHECK (AGE >=18)
    );
```

5) DEFAULT:-

i) Set default value of the column.
2) Ex:-

```
CREATE TABLE account(
....
....
Saving-rate DOUBLE NOT NULL DEFAULT 4.25
....
....
);
```

→ ALTER OPERATIONS:-

1) ADD
i) Add new column.
2) ALTER TABLE table-name ADD new-col-name datatype;

Ex:-
ALTER TABLE customer ADD age INT NOT NULL;

2) MODIFY:-
i) change datatype of an attribute.
ii) ALTER TABLE table-name MODIFY col-name col-datatype;

3) Eg: Varchar to char

ALTER TABLE customer MODIFY name char(1024);

3) CHANGE COLUMN

1) Rename column name.
2) ALTER TABLE table-name CHANGE COLUMN old-col-name new-col-name new-col-datatype;

Ex:- ALTER TABLE customer CHANGE COLUMN name customer-name char(1024);

4) DROP COLUMN

1) Drop a column completely.
2) ALTER TABLE table-name DROP column col-name.

Ex:- ALTER TABLE customer DROP COLUMN middle name.

5) RENAME

1) Rename table name itself.
2) ALTER table table-name RENAME to new-table-name;

Ex:- ALTER TABLE customer RENAME TO customer-details;

→ Data manipulation Language:-

1) INSERT INTO table_name (col1, col2, col3) VALUES (v1, v2, v3), (val1, val2, val3);

2) UPDATE:-

1) UPDATE table_name SET col1=1, col2='abc' WHERE id=1;

2) UPDATE all rows:-

UPDATE customer SET Pincode=110000;
it will give error, because we try to update ~~all~~ all values in a column.

To update all values in a column we have use one more command with it:-

→ SET SQL-SAFE-UPDATES=0;
UPDATE customer SET Pincode=110000;

3) ON UPDATE CASCADE

can be added to the table while creating constraints. Suppose there is a situation where we have two tables such that primary key of one table is the foreign key for another

table, if we update the primary key of the first table then using ON UPDATE cascade foreign key of second table automatically get updated.

3) DELETE :-

1) DELETE FROM table_name WHERE id=1;

2) DELETE FROM table_name;
All rows will be deleted

→ DELETE CASCADE - (to Overcome DELETE constraint of Referential constraint)

1) What would happen to child entry if parent table's entry is deleted ?

Ex :- CREATE TABLE ORDER (
order_id INT PRIMARY key,
delivery_date DATE,
cust_id INT,
FOREIGN KEY (cust_id) REFERENCES
customer(id) ON DELETE CASCADE);

⊖ using ON DELETE CASCADE will delete all the values # of the respective row in child table.

→ ON DELETE NULL

Ex: {

FOREIGN KEY (cust-id) REFERENCES customer (id) ON DELETE SET NULL);

Using ON DELETE SET NULL the FK value of that row will be set to NULL.

4) REPLACE :-

1) Data already present – Replace
2) Data not present – add

Ex:-

1) REPLACE INTO customer (id, city) VALUES (1251, 'colony');

2) REPLACE INTO customer SET id=1300, cname='', city='';

→ SQL Joins :-

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows

1) Inner Join
2) Outer Join

→ consider TWO tables below :-

→ Student :

ROLL_NO	Name	address	Phone	Age
1	Harsh	Delhi	XX	18
2	pratik	Bihar	xx	19
3	Priyanka	Siliguri	XX	20
4	Deep	Ramnagar	xx	18
5	Saptarhi	Kolkata	xx	19
6	Dhanraj	Barabajar	xx	20
7	Rohit	Balurghat	XX	18
8	Niraj	Alipur	XX	19

→ StudentCourse :

COURSE_ID	ROLL-NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

1) Inner Join :-

The inner Join keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.

Syntax :-

```
SELECT table1.column1, table1.column2, table2.column1,
.... FROM table1 INNER JOIN table2 on
table1.matching_column = table2.matching_
column.
```

Ex :-

```
SELECT studentCourse.course_ID, student.Name,
 student.AGE FROM student INNER JOIN
 studentCourse ON student.ROLL-NO = studentCourse.
ROLL-NO;
```

COURSE-ID	Name	Age
1	Harsh	18
2	pratik	19
2	Priyanka	20
3	Deep	18
1	Saptarhi	19

Ex:

```
SELECT C.*, O.* FROM Customer AS c INNER
JOIN Orders as O ON C.id = cust-id;
```

→ Alias in MYSQL:

1) Aliases in MYSQL is used to give temporary name to a table or a column in a table for the purpose of a particular query. It works as a nickname for expressing the tables or column names. It makes the query short and neat.

2) SELECT col-name AS alias-name FROM table-name.

3) SELECT col-name1, col-name2, ... FROM table-name AS alias-name;

2) Outer Join:-

1) LEFT JOIN:-

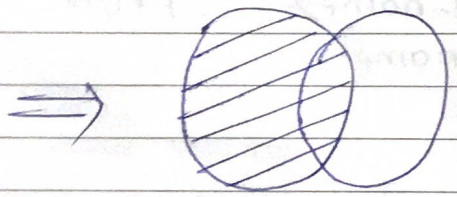
This Join returns all the rows of table on the left side of the join and matches rows for the table on the right side of the join.

for the rows for which there is no matching row on the right side, the result set will contain null.

Ex:-

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student LEFT JOIN StudentCourse ON
StudentCourse.ROLL-NO = Student.ROLL-NO;
```

Name	Course_ID
Harsh	1
Pratik	2
Priyanka	2
Deep	3
Saptarhi	1
Dhanraj	null
Rohit	null
niraj	null



TableA TableB

2) RIGHT JOIN:-

RIGHT JOIN is similar to LEFT JOIN. This JOINS returns all the rows of the table on the right side of Join and matching rows for the table on the left side of join. For the rows for which there is no matching row

on the left side, the result set ~~co~~ will contain null.

Ex:-

```
SELECT Student.NAME, StudentCourse.COURSE-ID
FROM Student RIGHT JOIN StudentCourse
ON StudentCourse.ROLL-NO = Student.ROLL-NO;
```

Name	COURSE-ID
Harsh	1
Pratik	2
priyanka	2
Deep	3
Saptarhi	1
NULL	4
NULL	5
NULL	4

3) FULL JOIN:

Full join creates the result-set by combining results of both left join and right join. The result set will contain all the rows from both tables.

Ex:-

```
SELECT Student.Name, studentCourse.COURSEID
FROM Student FULL JOIN StudentCourse ON
StudentCourse.ROLL-NO = Student.ROLL-NO;
```

4) CROSS JOIN:

1) This returns all the cartesian products of the data present in both tables. Hence, all possible variations are reflected in the O/p.
2) Used rarely in practical purpose.
3) Table-1 has 10 rows and table-2 has 5, then resultant would have 50 rows.
4) SELECT column list FROM table1 CROSS JOIN table2;

5) SELF JOIN:

1) It is used to get the output from a particular table when the same table is joined to itself.
2) Used very less.
3) Emulated using INNER JOIN.
4) SELECT columns FROM table as t1 INNER JOIN table as t2 ON t1.d = t2.d.

→ Join without using join Keywords :-

1) SELECT * FROM table1, table2 WHERE condition;
2) Ex :-

```
SELECT artist-name, album-name, year_recorded
FROM artist, album WHERE artist.id = album.
artist-id.
```

→ SET Operation in SQL :-

SQL supports set operators, which can be performed on the data. These operators are used to get desired results from the table data stored in the table. The set operators looks similar to SQL Joins, but there is a big difference, SQL Joins combine the columns from different tables, whereas SQL SQL operators combine rows from different queries.

→ There are some roles that you have to follow while applying set operators in SQL.

1) The no. of columns in the SELECT statement on which you want to apply SQL set operator must be same.

3) post_comment → post_comment_id, text_content, timestamp.

4) post_like → postlike_id, timestamp.

→ Relation and constraints :-

1) user_profile M friendship N user_profile

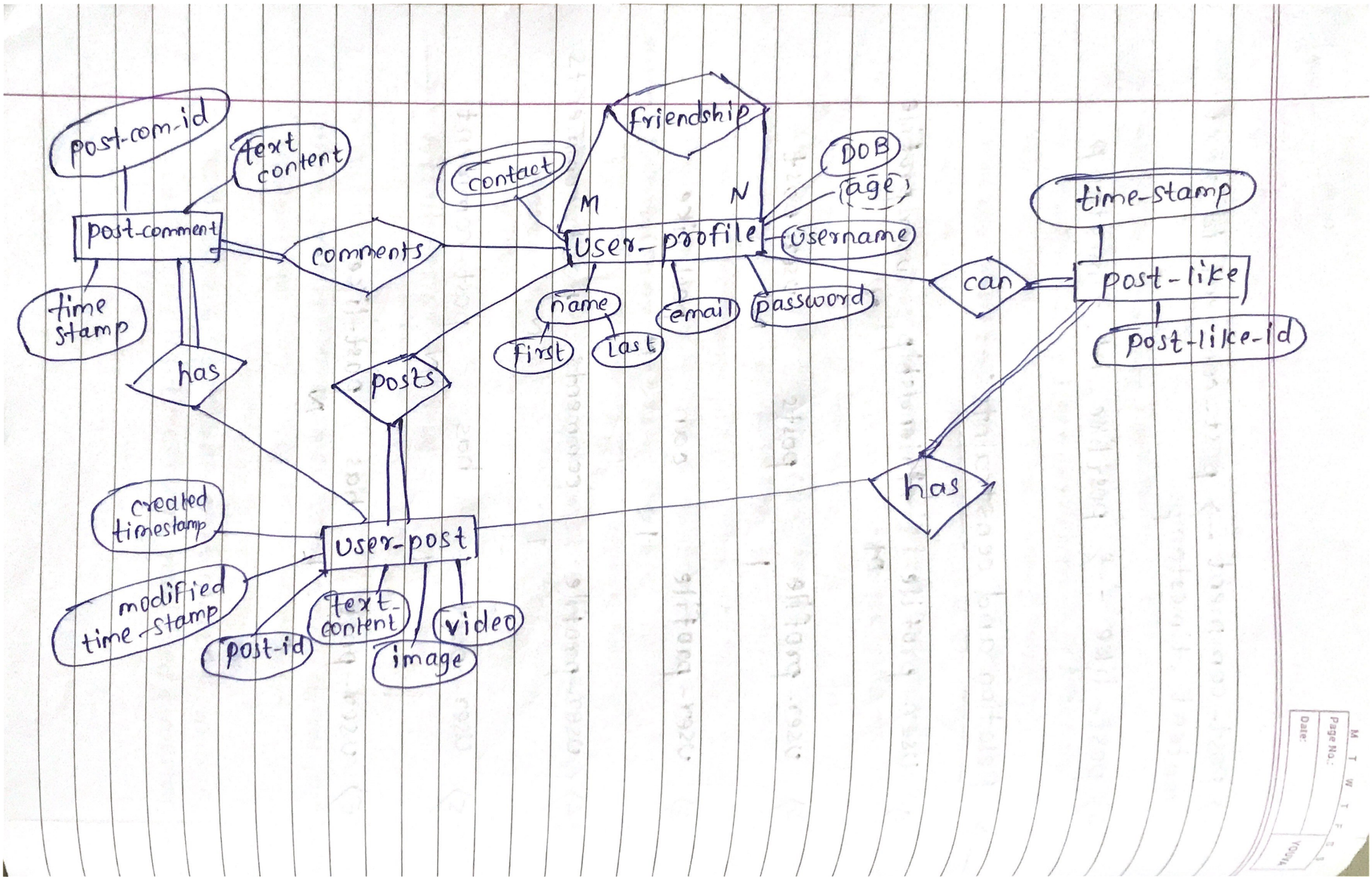
2) user_profile 1 posts N user_post

3) user_profile 1 can N post_like

4) user_profille 1 comments N post_comments

5) user_post 1 has N post_comment

6) user_post 1 has N post_like


post-com-id
text content
post-comment
comments
time stamp
has
posts
friendship
contact
M
N
User_profile
DOB
age
username
name
first
last
email
password
can
time-stamp
post-like
post-like-id
has
created timestamp
User-post
modified time-stamp
post-id
text-content
image
video

→ Relational Model in DBMS:-

The relational model for database management is an approach to logically represent and manage the data stored in database. In this model, the data is organized into collection of two dimensional inter related tables, also known as relations. Each relation is a collection of columns and rows, where the column represents the attributes of an entity and the rows represents the records.

The use of tables to store the data provided a straight forward, efficient, and flexible way to store and access structured information. Because of this simplicity, this data model provides data sorting and data access. Hence it is widely used around the world for data storage and processing.

→ Consider a case where you wish to store the name, the CGPA obtained, and the roll number of all the students of a particular class. This structured data can be easily stored in table as described.

Student Table (Relation)

Primary Key → Roll_Number	Name	CGPA
001	Vaibhav	9.1
002	Neha	9.5
003	Harsh	8.5
004	Shreya	9.3

Tuples (Rows)

Column (attributes)

Note: A database implemented and organized in terms of relational model is known as a relational database management system (RDBMS).

→ Highlights:-

1) Relational model stores the data into tables.

2) It makes data sorting and data access easier.

3) Provides a standard way to organize data in databases.

→ Relational Model constraints :-

1) Relation :- Two dimensional table used to store a collection of data elements.

2) Tuple :- Row of the relation, depicting a real world entity.

3) columns :- represents the attributes of the relations.

4) Attribute domain :- set of pre-defined atomic values that an attribute can take, i.e, it describes the legal values that an attribute can take.

5) Degree :- It is the total no. of attributes present in the relation.

6) cardinality :- It specifies the no. of entities involved in the relation i.e, it is the total no. of rows present in relation.

7) Relational Schema :- It is the logical blueprint of the relation, i.e, it describes the design and the structure of the relation. It contains table name, its attributes and their types.

Table-name(attribute 1 type 1, attribute 2 Type 2)

→ for our student relation example, the relational schema will be:

STUDENT(ROLL-NUMBER TYPE 1, NAME VARCHAR(20), CGPA FLOAT)

8) Relational instance :-
It is the collection of records present in the relation at a given time.

9) Important properties of a table in Relational Model.

i) The name of relation is distinct among all other relation.
ii) The values have to be atomic. can't be broken down further.
iii) The name of each attribute / column must be unique.
iv) Each tuple must be unique in a table.
v) The sequence of row and column has no significance.
vi) Tables must follow integrity constraints - it helps to maintain data consistency across the tables.

→ Relational model keys :-

It is an attribute or a group of attributes that can be used to uniquely identify an entity in a table or to determine the relationship between two tables.

1) Super key :-

A super key is just a key that can uniquely identify a set of attributes, all candidate keys come under the bracket of super keys

Ex :-

Customer

cust-ID	Name	contact	address	email

Super keys of above relation ~~are~~ is,

{Name, contact}, {Name, email}, {cust-ID}, {cust-ID, email}, {cust-ID, name, email, contact}

2) candidate Key :-

A candidate key is a part of the super key only. We can define a candidate key as a combination of attributes of a table that can uniquely identify other attributes of the table. A candidate key is also known as minimal super key.

As we know candidate key is chosen from set of super keys. Among the super keys set, the key which does not contain any redundant.

Suppose a table has 5 columns or attributes mainly — A, B, C, D, E.
And the super keys are : {ABC, AB, DE}

Now among the super keys, the candidate key can only be AB and DE. The reason why we not considered ABC to be a candidate key is that ABC contains redundant attributes. ABC has redundant attributes because only AB is capable of identifying all the other attributes of the table so we do not need the C attribute.

→ Note :- Candidate keys are chosen from super keys and one of these candidate keys will further become primary key. The primary key selection is done by DBA according to the frequencies of queries.

→ How is the candidate key different from primary key?

1) The most important and basic difference is that any attribute of a candidate key can have null values. But the attributes of a primary key cannot contain Null values
2) In a table, we can have one or more than one candidate key, but there is one and only one primary key in a relation.

→ 3) Primary key:-

We can define a primary key as an attribute that can uniquely identify each row of a database table. The primary key is a type of key having the combination of two constraints namely NOT NULL and unique.

4) → Alternate key

All candidate key except the primary key.

5) Foreign key :-

A foreign key is a column or a group of columns in a table whose values are preferred from a primary key in another table. A primary key in SQL uniquely identifies record in a table. However you must have a primary key for using a foreign key.

Department

EmpNO	EmpName	DepNO ← Foreign key
1001	Sahil	101
1004	Kavish	102
1006	Aditya	103
1005	Atul	104

Primary key →

DepNO	DName	Location
101	HR	Delhi
102	Sales	Bangalore
103	Marketing	Hyderabad
104	Technical	chennai

The table containing a parent key is known as parent table, whereas the table which contains the foreign key is known as child table. Therefore a foreign key relates two tables in database. Moreover, the primary key values in the parent table must match the values in the foreign key in another table.

→ 6) Composite key:-

Primary key formed using atleast 2 attributes.

→ 7) compound key:

Primary key which is formed using 2 foreign key.

→ 8) Surrogate key:-

In case we do not have a natural primary key in a table, then we need to artificially create in order to uniquely identify a row in the table, this key is called the surrogate key or synthetic primary key.

→ consider an example:-

suppose we have two tables of two different schools having the same column registration-no, name and percentage, each table having its own natural primary key, that is registration-no.

Table of school A-

registration-no	name	%
210101	Harry	90
210102	Maxwell	65
210103	Lee	87
210104	Chris	76

Table of school B

registration	name	%
CS107	-	49
CS108	-	86
CS109	-	96
CS110	-	58

suppose we have to merge both the tables:-

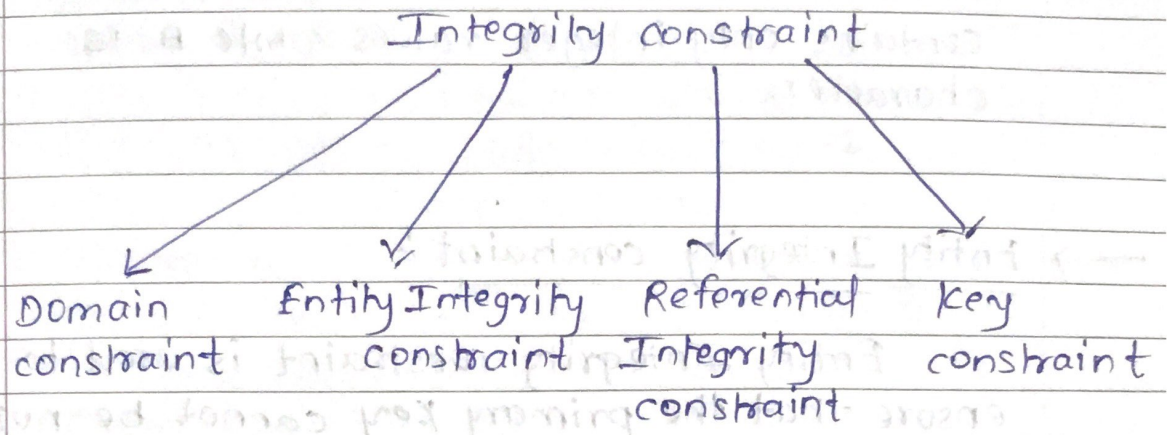
surr-no	registration-no	name	%
1	210101		
2	210102		
3	210103		
4	210104		
5	CS107		
6	CS108		
7	CS109		
8	CS110		

As we can observe the above table and see that the registration_no cannot be the primary key of the table as it does not match with all the records of the table though it is holding all unique values of table. Now, in this case, we have to artificially create one primary key for this table. We can do this by adding a column surr_no in the table that contains anonymous integers and has no direct relation with other column,

→ Integrity constraints:-

In DBMS, integrity constraints are the pre-defined set of rules that are applied on the table fields or relations to ensure that the overall validity, integrity, and consistency of the data present in the database table is maintained. Evaluation of all the conditions or rules maintained in the integrity constraints is done every time a table insert, update, delete or alter operation is performed. The data can be inserted, updated, deleted or altered only if the result of the constraints come out to be True.

Types of Integrity constraints:-



1) Domain constraint:-

Domain Integrity constraint contains a set of rules or conditions to restrict the kind of attributes or values a column can hold in database table. The data type of a ~~string can be~~ domain can be String, int, char, DateTime, currency, etc.

Consider a student table:-

Roll No	Name	Age	class
101	Adam	14	6
102	steve	16	8
103	David	8	4
105	Tim	6	A

In the above table, the value A in the last ~~row~~ column violates the domain integrity constraint because the class attribute contains only integer values while A is character.

→ Entity Integrity constraint :-

Entity Integrity constraint is used to ensure that the primary key cannot be null. A primary key is used to identify individual row in a table and if it is NULL, then we can't identify those records. There can be null values anywhere except primary key.

→ Referential Integrity constraint :-

Ensures that there must always exist a valid relationship between two relational database tables. This valid relationship between two tables confirms that a foreign key exists in a table. It should always reference a corresponding value or attribute in other table or be NULL.

→ Customer Table :-

Parent relation / Referenced table	cust-id	Name	add	contact-no
	1	A	x	12
	2	B	y	23
	3	C	z	45

→ ORder Table :-

child Relation / Referencing table	Order-id	Timestamp	dd	cust-id (FK)
	21	-	-	1
	22	-	-	2
	23	-	-	3

1) Insert constraint :- Value can't be inserted in child table if the ~~table~~ value is not lying in parent table.
2) Delete constraint :- value can't be delete from parent table if the value is lying in child table.

→ On delete cascade :-

Q) Can we delete value from parent table if the value is lying in child table w/o violating delete constraint?

→ delete value from parent table → delete corresponding entry from child table to.

Q) Can Foreign key have NULL value?

delete value from parent table → put corresponding foreign key value NULL.

→ Key Constraints :-

1) Not Null :-

1) Null represents a record where data may be missing data or data for that record may be optional.
2) Once not null is applied to a particular column, you cannot enter null values to that column and restricted to maintain only

some proper value other than null.

3) A not null constraint cannot be applied at table level.

Ex:-

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
);
```

In the above ex, orderID and OrderNumber is not null.

2) **Unique:-**

1) sometimes we need to maintain only unique data in the column of database table, this is possible by using a unique constraint.

2) Unique constraint ensures that all values in a column are unique.

```
CREATE TABLE Persons (
    ID int UNIQUE,
    LastName varchar(255) NOT NULL,
);
```

3) DEFAULT:-

1) Default clause in SQL is used to add default data to the columns.
2) When a column is specified as default with some value then all rows will use the same value i.e each and every time while entering the data we need not enter that value.
3) But default value can be customized, i.e can be overridden.

4) Check:-

1) suppose in real time if you want to give access to an application only if the age entered by users is greater than 18 this is done at the back-end by using check constraint.
2) Check constraint ensures that the data entered by the user for that column is within the range of value or possible values specified.

5) Primary key:

1) A primary key is a constraint in a table that uniquely identifies each row record in a database table by enabling one or more the columns in table as primary key.

6) Foreign Key:-

1) The foreign key a constraint is a column ~~of~~ or list of columns that points to the primary key column of another table.

2) The main purpose of foreign key is only those values are allowed in the present table that will match the primary key column of another table.

→ ER model to Relational Model:-

1) Both ER-Model and Relational Model are abstract logical representation of real world enterprises. Because the two model implies similar design principles, we can convert ER design into to Relational design.

2) converting a DB design from an ER diagram to a table format is the way we arrive at Relational DB design from an ER diagram.

3) ER diagram notations to relations:-

1) Strong Entity:-

1) Becomes an individual table with entity name, attributes becomes columns of the relation.
2) Entity's primary key (PK) is used as relations PK.
3) FK are added to establish relationship with other relations.

Ex:- Consider the Bank ER diagram, where loan is a strong entity.

PK → Loan

Loan number	amount

2) Weak Entity:-

1) A table is formed with all attributes of the Entity.
2) PK of its corresponding strong entity will be added as FK.
3) PK of the relation will be a composite PK, {FK + Partial discriminator key}.

Ex:- payment was the weak entity set in bank ER diagram.

FK ↘

Payment

Loan number	Payment-no	Payment-date	Pay-amou

PK = Loan number + Payment-no

3) single valued attributes:-

1) Represented as columns directly in the tables / relations.

4) composite attributes:-

1) Handled by creating a separate attribute itself in the original relation for each composite attribute.

2) eg: Address : {street-name, house-no} is a composite attribute in customer relation, we add address-street_name and address-house-name as new columns in the attribute and e ignore 'address' as an attribute.

Ex:- customer

customer_name	address_city	add_state	add-pin	add-street-no

5) Multivalued attribute:-

1) New tables (named as original attribute name) are created for each multivalued attribute.
2) PK of the entity is used as column FK in the new table.
3) Multivalued attribute's similar name is added as column to define multiple values.
4) PK of the new table would be {FK + multivalued-name}.
5) Ex:-
for strong entity Employee, dependent_name is a ~~weak~~ multivalued attribute.

new relation → dependent-name.

FK ← emp-id	d_name

PK ⇒ {emp-id, d-name}

6) Derived attributes:- not considered in table.

→ Generalisation:-

1) Method-1:- Create a table for the higher level entity set. For each lower level entity set, create a table that includes a column for each attribute of that entity set plus a column for each attribute of the primary key of the higher level entity set.

→ For eg:- Banking system generalisation of account - saving and current.

1) Table 1 : account (account_number, balance)
2) Table 2 : savings-account (account_number, interest rate, daily withdrawal limit).
3) Table 3 : current account (account_number, overdraft, per-transaction charge).

2) Method-2:- An alternative representation is possible, if the generalisation is disjoint and complete - that is, if no entity is a member of two lower level entity sets directly below a higher level entity set, and if every entity in higher level entity set is also a

2) The order of columns must be in same order

3) The selected columns must have same data type.

→ UNION:-

1) combines two or more select statements.

2) SELECT * FROM table1
UNION
SELECT * FROM table2

3) No. of column, order of column must be same for table1 and table2.

→ INTERSECT:-

1) Returns common values of the table.

2) SELECT DISTINCT column_list FROM table1 INNER JOIN table-2 USING (Join_cond);

3) Ex:-
SELECT DISTINCT * FROM table1 INNER JOIN table2 USING (id);

→ MINUS:-

1) This operator returns distinct row from the first table that does not occur in second table.

2) SELECT column_list FROM table1 LEFT JOIN table2

ON condition WHERE table2.column_name is NULL;

Ex:-

SELECT id from table-1 LEFT JOIN table-2
USING (id) WHERE table-2.id is NULL.

→ NORMALISATION:-

→ What is functional dependency?

Relational database is a collection of data stored in rows and columns. Columns represent the characteristics of data while each row in a table represents a set of related data, and every row in the table has the same structure.

→ functional dependency in dbms, as the name suggests it is the relationship between attributes of a table related to each other.

→ It helps in maintaining the quality of data in the database, and the core concepts behind database normalization are based on functional dependencies.

→ It is a relationship between primary key attribute of the relation to that of the other attribute of the relation.

→ How to define FD?

A functional dependency is denoted by an arrow "→". The functional dependency of A on B is represented by A → B.

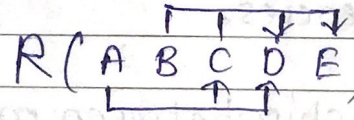
→ Consider a Relation :

R(ABCD)

1) A → BCD
2) B → CD

1) for the first functional dependency A → BCD attributes B, C, D are functionally dependent on attribute A.
2) functional dependency B → CD has two attributes C and D functionally depending upon attribute B.

→ sometimes everything on the left side of functional dependency is also referred to as determinant set, while everything on the right side is referred to as depending attributes.



A → CD
BC → DE

→ Armstrong's Axioms / Properties of functional dependency :-

1) Reflexivity :- If A is a set of attributes and B is a subset of A, then the functional dependency A → B holds true.

for ex : {Employee-ID, Name} → Name is valid.

2) Augmentation :- If a functional dependency A → B holds true, then appending any number of the attribute to both sides of dependency doesn't affect the dependency. It remains true.

ex :- X → Y holds true then, ZX → ZY also holds true.

3) Transitivity :- If two functional dependencies X → Y and Y → Z hold true, then X → Z also holds true by the rule of transitivity.

→ Types of FD:-

1) Trivial FD:-

In trivial FD, a dependent is always a subset of the determinant. In other words, a functional dependency is called trivial if the attributes on the right side are the subset of the attributes on the left side of the functional dependancy.

→ $X \rightarrow Y$ is called a trivial functional dependancy if Y is the subset of X.

→ Non Trivial FD:-

Dependent is not a subset of the determinant.

→ $X \rightarrow Y$ is called a non-trivial functional dependency if Y is not a subset of X. so, a functional dependency $X \rightarrow Y$ where X is a set of attributes and Y is also a set of attribute but Y is not a subset of X.

→ Normalization:-

Normalization is the process of organizing the data and the attributes of a database. It is performed to reduce the data redundancy in a database and to ensure that data is stored logically. Data redundancy means having the same data but at multiple places. It is necessary to remove data redundancy because it causes anomalies in a database which makes it very ~~eleor~~ hard for a DBA to maintain it.

→ Why do we need Normalization?

Normalization is used to reduce data redundancy. It provides a method to remove the following anomalies from the database and bring it to a more consistent state:-

A database anomaly is a flaw in the database that occurs because of poor planning and redundancy.

1) Insertion anomaly:- This occurs when we are not able to insert data into a database because some attribute may be missing at the time of insertion.

2) Updation anomalies:

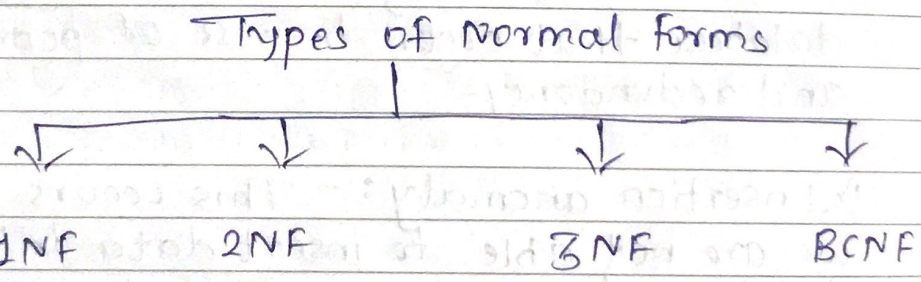
This occurs when the same data items are repeated with the same value and are not linked to each other.

→ The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

→ Due to updation to many places, may be data inconsistency arises, if one forgets to update data at all the places.

3) Deletion anomalies:

The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.



→ Ex of updation anomaly :-

id	name	age	Branch_code	Branch_name	HOD
1	A	18	1	CS	X
2	B	19	1	CS	X
3	C	20	1	CS	X
4	D	18	2	IT	Y
5	E	20	2	ENTC	Z
6	F	21	3	ME	W

Suppose, if we have to update the name of HOD of CS department then we have to look at all the rows in the database for a single value data and update it everywhere. But if we divide the table into two then,

id	Name	age	Branch_code
—	—	—	→ Table1
—	—	—	
—	—	—	

Branch_code	Branch_name	HOD
—	—	— → Table 2
—	—	—

Now, If we have to change the name of HOD, then we can directly look at Table 2 and update the name of HOD only once.

→ What we do in Normalization?

We decompose the table into multiple tables since the table is not divided any further, and ~~SPR~~ ≠ SRP is not achieved.

SRP → single Responsibility principle.

SRP means dividing the tables untill they show a single id.

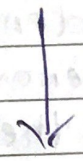
→ 1NF :-

If a relation contain composite or multi valued attribute, it violates the first normal form or a relation is in first normal form if it does not contain any composite or multi valued attribute.

Ex:-

Employee

id	Name	phone
1	A	88
2	B	12,99



id	Name	phone
1	A	88
2	B	12
2	B	99

→ 2NF:

The normalization of 1Nf relation to 2Nf involves the elimination of partial dependencies. A partial dependency exist cohen any non prime attributes, i-e, an attribute not a part of candidate key, is not fully functionally dependent on one of the candidate key.

Ex:- R{A B C D E}

{A B} ⟶ primary key

A, B ⟶ prime attributes
C, D ⟶ non-prime.

Functional dependency ⇒ B ⟶ C it is a partial dependency because C depends only upon one of the attributes of a primary key.

AB ⟶ C ✓✓
A ⟶ C ✗

Ex:

A	B
NULL	1
2	NULL
NULL	NULL
3	4

{A, B} ⟶ PK

FD ⇒ B ⟶ C,
then C ⇒ NULL

For, these kind of situations to avoid we generally convert the ~~#~~ table into 2NF.

→ For a relational table to be in a 2NF it most satisfy the conditions:-

1) Relation must be in 1NF.
2) There should ~~be~~ not be any partial dependency.
   1) All non prime attributes must be fully dependent on PK.
   2) Non prime attribute can not depend on the part of FK.

Ex:-

Table (Student Project):

Student-id	project-id	Student-name	Project-name
S89	P09	—	—
S76	P08	—	—
S71	P06	—	—
S65	P07	—	—

PK: {Student-id, project-id}

FD → Student-id → Student-name
project-id → project-name

In the above table, the prime attributes of the table are student-id, project-id. We have partial dependencies in the table because student-name can be determined by student-id and project-name can be determined by project-id, the above table violates the rule of 2NF.

To remove partial dependencies we can decompose the table :-

→ **2NF form of the table:**

Student table:

Student-id	Project-id	student-name
S89	—	—
S76	—	—
S56	—	—
S92	—	—

Project table:

Project-id	project-name
—	—
—	—
—	—

→ 3NF:

The normalization of 2NF relations to 3NF involves the elimination of transitive dependencies.

→ A FD is said $X \rightarrow Z$ to be a transitive if the following three functional dependencies hold:-

1) $X \rightarrow Y$
2) Y does not $\rightarrow X$
3) $Y \rightarrow Z$

→ for a relational table to be in a 3NF, it most satisfy following rules:-

1) It should be in 2NF.
2) No non prime attribute is transitively dependend on primary key.

→ Ex:-

A	B	C
a	1	X
b	1	X
c	1	X
d	2	Y
e	2	Y
f	3	Z
g	3	Z

Prime attribute: {A}

$A \rightarrow B$

FD ⇒ $B \rightarrow C$

The above table is not in 3NF because A → C has transitive dependency.

To remove transitive dependency from this table and normalize it into the third normal form, we can decompose the table into two tables.

→ 3NF of the table:-

A	B
a	1
b	1
c	1
d	2
e	2
f	3
g	3

B	C
1	x
2	y
3	2

→ **BCNF:-**

Boyce-Codd Normal form is an advanced version of 3NF as it contains additional constraints compared to 3NF.

1) The table must be in third normal form
2) For every non-trivial functional dependency X → Y, X is the superkey of the table.

That means X cannot be a non-prime attribute, if Y is a prime attribute.

Ex

In this example we have a relation R with three columns : ID, Subject and Professor. We had to find the highest normalization form and also if it is not in BCNf, we have to decompose it to satisfy the conditions of BCNf.

ID	Subject	Professor
101	Java	Mayank
101	C++	Kartik
102	Java	Sarthak
103	C#	Lakshay
104	Java	Mayak

→ Interpreting the table :-

1) one student can enroll in more than one subject.

2) Professor is assigned to the student for a specified subject, and there is always a possibility that there can be multiple professors teaching a particular subject.

→ finding soln:-

1) Using ID and subject together we can find all unique records and also the other columns of the table.
2) The table is in INF because all the values inside a column are atomic and of same domain.
3) We can't uniquely identify a record solely with the help of either the id or subject name. As there is no partial dependency, the table is in 2NF.
4) The table is also in 3NF.

→ Why the table is not in BCNF?

As we know each professor teaches only one subject but one subject may be taught by multiple professors.

So, there is a dependancy between a professor and subject.

professor → subject

As we know professor is a non attribute column and subject is a prime attribute.

→ Student table:-

id	p-id
101	1
101	2
102	3
103	4
104	1

P_id	P_name	subject
1	x	a
2	y	b
3	z	c
4	w	d

→ Advantages of Normalization:-

1) Normalization helps to reduce data redundancy.
2) Greater overall database organization.
3) Data consistency within the database.
4) Much more Flexible database region.
5) E

→ Transaction in DBMS :-

Transactions are set of operations used to perform a logical set of work. A transaction usually means that the data in the database has changed. One of the major uses of DBMS is to protect the user's data from system failures. It is done by ensuring that all the data is restored to a consistent state when the computer is restarted after a crash. The transaction is any one execution of the user program in a DBMS.

But in the case of transaction in DBMS there are three major components that are used for a transaction to get executed in an efficient manner. These are :-

1) Read / Access data.
2) Write / change data.
3) Commit data.

Let's understand the above ~~steps~~ sets of operations in a transaction with a real-life example of transferring money from Account 1 to Account 2.

Initial bank balance :-
Account 1 => 5000Rs.
Account 2 => 2000Rs.

This data before the start of transaction is stored in secondary memory which once initiated is bought to the primary memory (RAM) of the system for faster and better access.

Now for a transfer of Rs.500 from account 1 to account 2, the following set of operations takes place,

→ for account 1
Read(account1)
account1 = account1 - 500
write(account1)

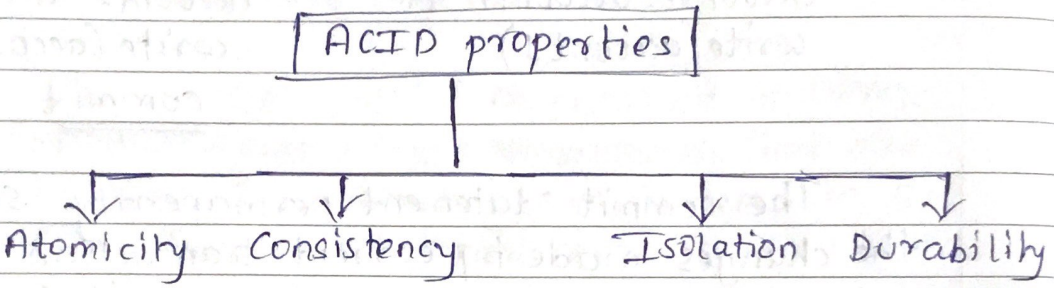
→ for account 2
Read(account 2)
account2 = account2 + 500
write(account 2)
commit

The commit statement permanently saves the changes made by current transaction. When a transaction is successful, commit is applied. If the system fails before a commit is applied, the transaction reaches its previous state after rollback.

After commit operation the transaction ends and updated values of account1 and account 2. Every single operations that occurs before commit is said to be in a partially committed state and is stored in the primary memory.
After transaction is committed the updated data is accepted and updated in the secondary memory.

→ Acid properties in DBMS:-

DBMS is the management of data that should remain integrated when any change are done in it. It is because if the integrity of the data is affected, whole data will get disturbed and corrupted. Therefore to maintain the integrity of data, there are four properties described in the database management system which are known as ACID properties.



1) Atomicity (A):-

1) It states that all the operations of the transaction take place at once if not, the transaction is aborted.
2) There is no midway, i.e, the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.

→ Atomicity involves two operations :-

Abort :- If a transaction aborts then all the changes made are not visible.

commit :- If a transaction commits then all the changes made are visible.

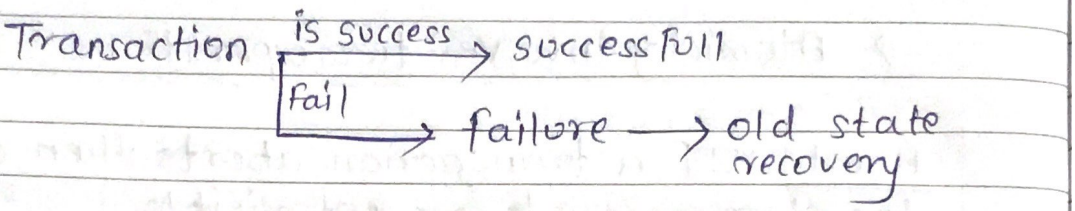
Ex :-

Let's assume the following transaction T consisting of T1 and T2. 'A' consist of Rs 600 and B consist of Rs 300. Transfer of Rs 100 from 'A' to account 'B'.

T_1	T_2
Read(A)	Read(B)
A = A - 100	B = B + 100
write(A)	write(B)

After completion :- A = 500 and B = 400

If the transaction 'T' fails after the completion of T1 and before completion of T2, then the amount will be deducted from 'A' but not added to 'B'. This shows the inconsistent database state.



The database maintains the old state (i.e, the initial value of account) as well as intermediate state, so in case if transaction ~~is~~ from one side is completed and it fails before completion of other side then ~~the data~~ we rollback to the old state, that is the initial state.

2) Consistency :-

1) Integrity constraints must be maintained before and after transaction.
2) DB must be consistent after transaction happens.

Ex :- The total amount must be maintained before and after transaction.

3) Isolation :-

1) It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.

2) In Isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until transaction T1 ends.

3) The concurrency control subsystem of the DBMS enforced the isolation property.

4) Durability:

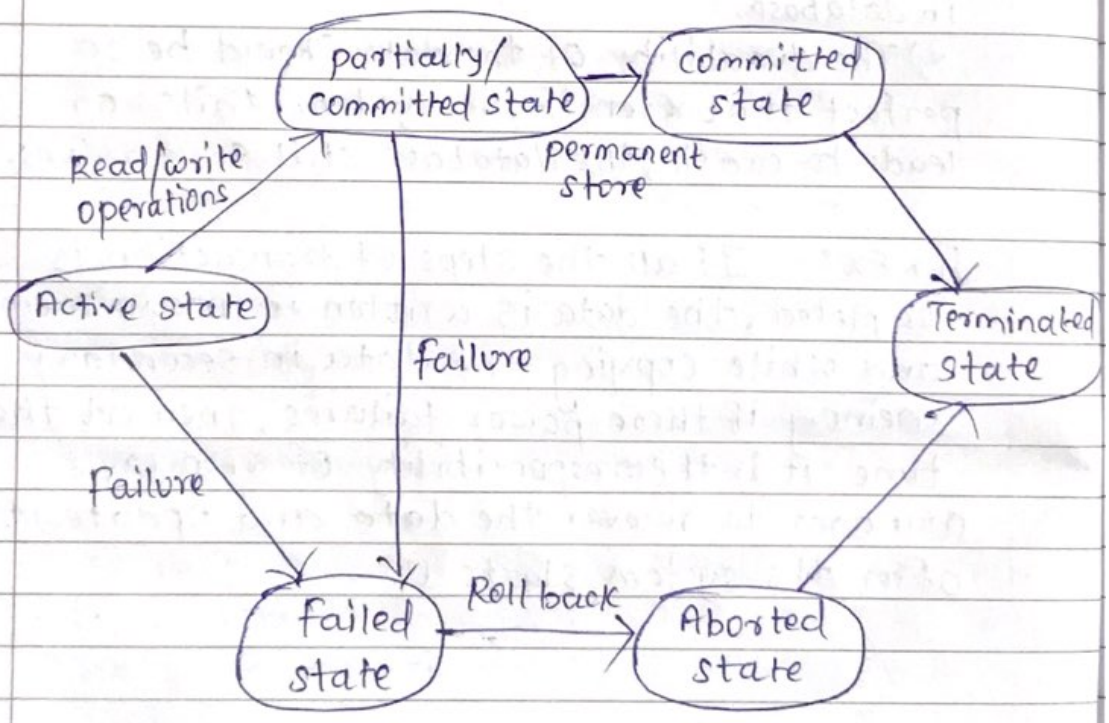
1) Durability ensures the permanency of something. In DBMS the term durability ensures that the data after the successful execution of the operation becomes permanent in database.
2) The durability of the data should be so perfect that even if the system fails or leads to crash, the database still survives.

For ex:- If all the steps of transaction is completed, the data is written in main memory and while copying the data in secondary memory if there power failures, then at that time it is the responsibility of recovery manager to recover the data and update it after the system starts up.

→ Transaction states in DBMS:-

States through which a transaction goes during its lifetime. These are the states which tell about the current state of the transaction and also tell how we will further do the processing in the transactions.

They also use Transaction log. Transaction log is a file maintain by recovery management component to record all the activities of the transaction.



1) Active state:-

The very first state of the transaction life cycle is all the read and write operations are being performed. If they execute without any error T comes to partially committed state. Although if any error occurs it comes to failed state.

2) Partially committed:-

After transaction is executed the changes are saved in the buffer in main memory. If the changes made are permanent on the DB then the state will transfer to the committed state and if there is any failure, the T will go to failed state.

3) committed state:-

When updates are made permanent on the DB. Then the T is said to be in the committed state. Rollback can't be done from committed states.

4) failed state:-

When T is being executed and some failure occurs. Due to this it is impossible to continue the execution of the T.

5) Aborted state :-

When T reaches failed state, all the changes made in Buffer are reversed. T reaches abort state after roll back.

6) Terminated state :-

A transaction is said to have terminated if has either committed or aborted.

→ Methods to implement Atomicity and Durability :-

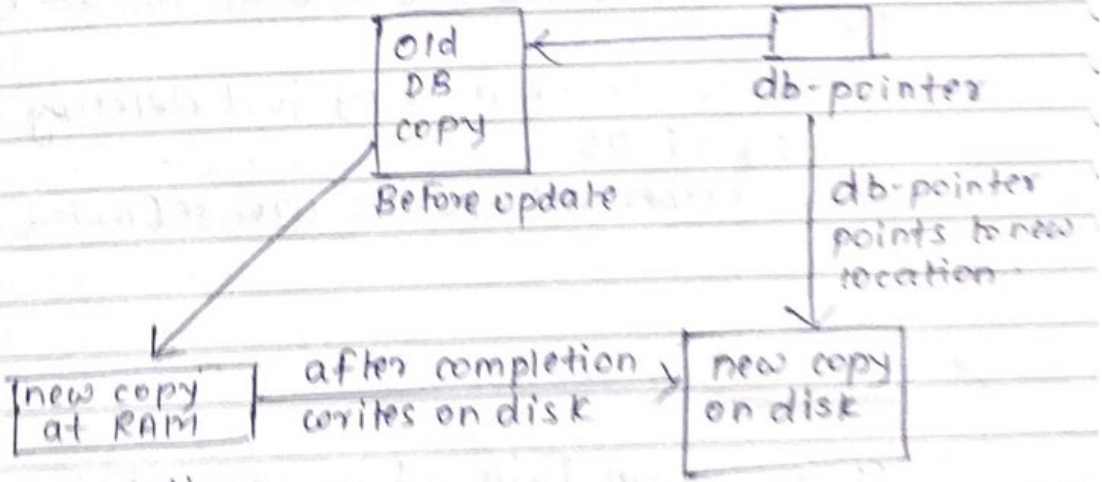
The recovery management component of a database system can support atomicity and durability by a variety of schemes.

1) Shadow copy :-

1) In shadow copy scheme, a transaction that wants to update the database first creates a copy of the database first in the ram. All updates are done on the new copy in RAM, leaving the original copy untouched.

If at any point the ~~system~~ transaction has to be aborted, the system merely deletes the new copy. The old copy of the database has not been affected.

2) This scheme is based on making copies of the database, called shadow copies, assumes that only one transaction is active at a time. ~~This~~ The scheme also assumes that the database is simply a file on a disk. A pointer called db-pointer is maintained on disk, it points to the current copy of the database.



all operations perform on new copy

After dp db-pointer points to new location it notifies that transaction is committed.

→ If Transaction success, it is committed as,

1) Os makes sure all changes of the new copy of DB written on disk.
2) DB system updates the db-pointer to point to the new copy of DB.
3) New copy is now the current copy of DB.
4) The old copy is deleted.
5) The 'T' is said to have been committed at the point where the updated db-pointer is written to disk.

→ Atomicity:-

1) If T fails at any time before db-pointer is updated the old content of DB are not affected.
2) T abort can be done by just deleting the new copy of DB.
3) Hence, either all updates are reflected or none.

→ Durability:-

1) Suppose system fails at any time before ~~the~~ db-pointer is updated, the old-content of DB are not affected.
2) When the system restarts, it will read db-pointer and will thus see the original content of DB and none of the effects of T will be visible.

3) T is assumed to be successfull only when db-pointer is updated.
4) If system fails after db-pointer has been updated, Before that all the pages of new copy were written to disk. Hence when system restarts it will read new DB copy.

→ Log Based Recovery :-

Log based recovery in DBMS provides the ability to maintain or recover data in case of system failure. DBMS keeps a record of every transaction on some stable storage device to provide easy access to data when the system fails.

A log is a series of records. The logs for each transaction are kept in log file to allow recovery in case of failure. A log is kept for each operation performed on the database. It is important to store the log before the actual transactions are applied to the database.

→ Stable storage is a classification of computer data storage technology that guarantees atomicity for any given write operations and allows software to be written that is robust against some hardware and power failures.

→ Deferred DB Modifications:-

1) Ensuring atomicity by recording all the DB modifications in the log but we are deferring the execution of all write operations until the final action of T has been executed.
2) When the transaction is completed then the log information is used to execute the write operations on the database.
3) If the system crashes before T completes, i.e before T is committed or if T is aborted the information in the logs are ignored.
4) If failure occurs while we update database using transaction log, we perform redo operation.

Ex:- So, if we are updating database using log and system crash, then after system starts up it sees the log and see if transaction is completed or not, if it is completed then it checks the values are updated or not if not then it performs redo.

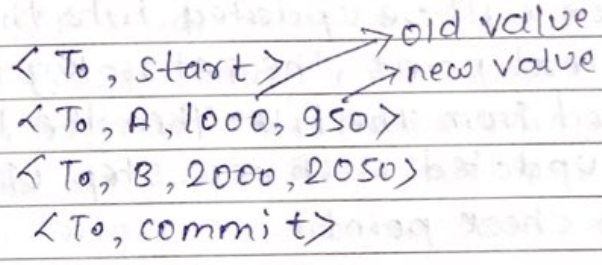
→ Log table

<T0, start>
<T0, A, 950>
<T0, B, 2050>
<T0, commit> ← Transaction is completed

Immediate DB Modification:-

1) DB modification to be output to the DB while T is still ~~a~~ in active state.
So, as soon as the transaction starts, each step of transaction is recorded in transaction log first and then it is written to the database directly.
2) DB modifications written by active T are called uncommitted modifications.
3) In the event of crash or T failure, system uses old value field of the log records to restore modified values.

Log table



4) updates takes place only after log in a stable storage.

5) failure handling:-

1) system failure before T completes, or if ~~it~~ T is aborted, then old value field ~~of the~~ is used to undo the T.

2) If T completes and system crashes, then new value field is used to redo T having commit logs in the logs.

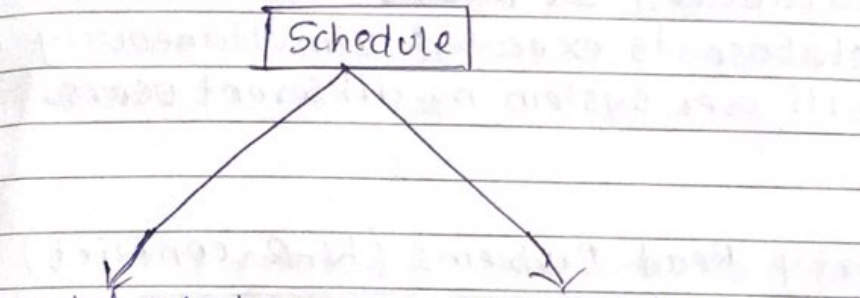
→ Checkpoint :-

1) The checkpoint is a type of mechanism where all previous logs are removed from system and permanently stored in storage disk.
2) The checkpoint is like a bookmark. while the execution of transaction, such checkpoints are marked and the transaction is executed then using the steps of transaction, log files will be created.
3) When it reaches to the checkpoint, then the transaction will be updated into the database, and till that point, the entire log file will be removed from the file. Then the log file will be updated with new step of transaction till next check point

4) The checkpoint is used to declare a point before which the dbms was in consistent state and all transactions were committed.

→ Types of schedules in DBMS:-

Schedule as the name suggest, is a process of lining the transactions and executing them one by one.



1) Serial Schedule:-

The serial schedule is a type of schedule where one operation is executed completely before starting another transaction. In the serial schedule when first transaction completes its cycle, then the next transaction is executed.

2) Non-serial schedule:-

1) In non-serial schedule multiple transactions are executed at once.
2) It contains many possible orders in which the system can execute the individual operations of the transactions.

→ Concurrent Execution in DBMS :-

1) In a multi-user system users can access and use the same database at one time, which is known as concurrent execution of database. It means that the same database is executed simultaneously on a multi user system by different users.

→ Dirty Read Problems (W-R conflict) :-

The dirty Read problem occurs when one transaction updates an item of the database and somehow the transaction fails, and before the data gets rollback, the updated database is accessed by another transaction. There comes the Read-write conflict between both transaction.

→ A = 70

	T_1	T_2
70	R(A)	
20	A = A - 50	
20	W(A)	
	⋮	R(A) = 20
	⋮	A = A * 2 = 40
	⋮	W(A) = 40
Transaction → fails at these step	R(B)	
	W(B)	

So, when transaction fails there will be rollback and A will gets it's initial value.

Now, we can see that T2 has performed operation on 20 ~~when~~, and after transaction fails A has got it's initial value, so there is no value 20, so it can be called as Dirty-Read.

→ Read-write conflict (Unreapeatable Read):-

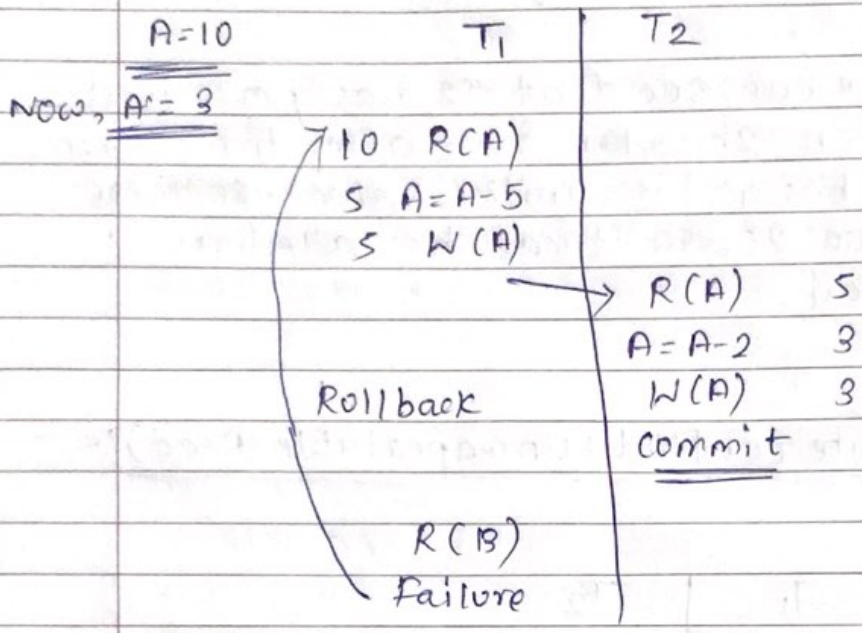
A = 2

T_1	T_2
2 = R(A)	
	R(A) = 2
	W(A) = 2-2 = 0
	commit
0 R(A)	
W(A)	
commit	

Initially, T_1 Reads value 2, and then T_2 starts execution and it also reads value 2 and then he performs write operation and make value as 0 and commit it. Then after that T_1 again reads value and see it as 0, then there is a conflict that T_1 has not performed any operation and previously it has read value as 2 and now 0. So, the T_1 transaction is aborted and started again.

→ Non-serializability in DBMS:-

→ Irrecoverable schedule:-



After failure on T1 side there is rollback and all the changes made are disrupted and the database in initialized again to it's original value.

Consider the following schedule involving two transaction T1 and T2. T2 read the value of A written by T1, and committed. T1 might later abort/commit; therefore the value read by T2 is wrong, but since T2 committed, this schedule is non recoverable.

→ Recoverable Schedule :-

A schedule is recoverable if each transaction commits only after all the transactions from which it has read have committed. In other words, if some Transaction T_2 reads a values that has been updated by some other transaction T_1, then the commit of T_2 must occur after the commit of T_1.

Transaction (T_1)	Transaction (T_2)
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
Abort	
	commit

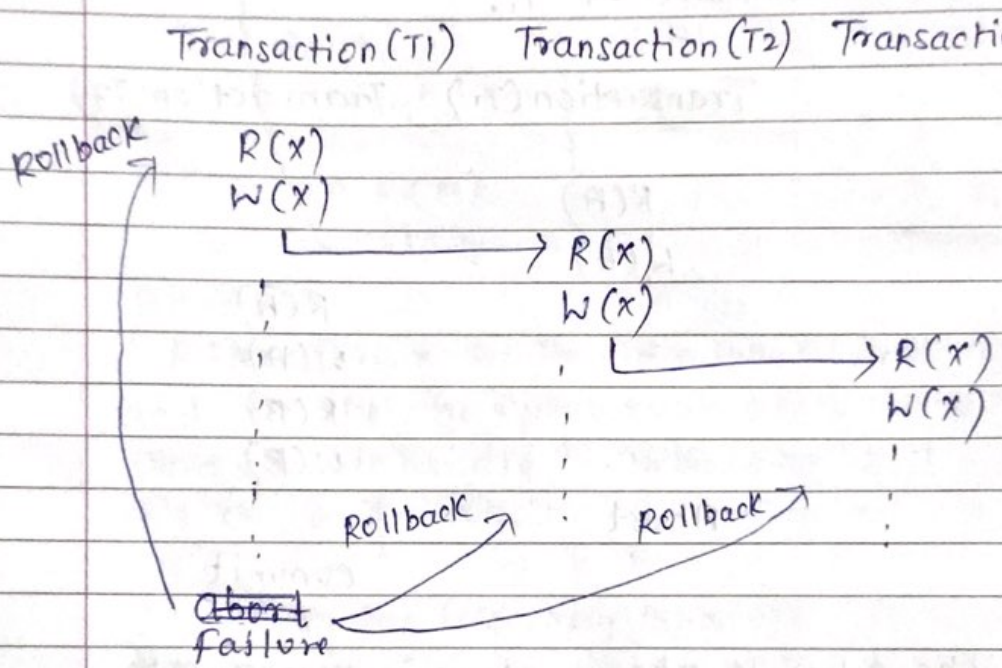
The schedule shown above is recoverable since T1 commits before T2, which makes all value read by T2 correct.

→ Recoverable schedules are further categorized into 3 types:-

1) Cascading schedule.
2) Cascadless schedule.
3) Strict schedule.

1) Cascading Schedule.

If in a schedule, several other dependent transactions are forced to rollback abort because of the failure of one transactio, then such a schedule is called as cascading schedule. It simply leads to the wastage of CPU time.

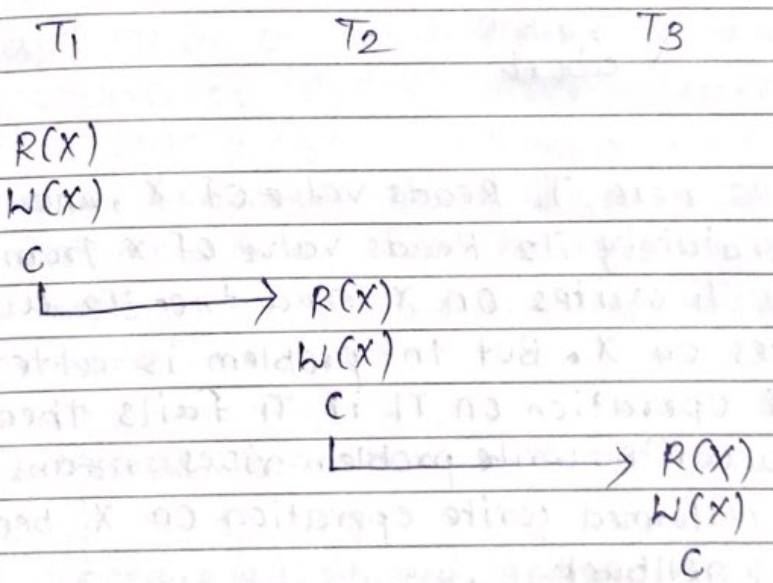


Here transaction T2 depends on T1 and T3 depends on T2. Thus, in this schedule the failure of transaction T1 will cause T2 to roll back and a similar case for transaction T3. Therefore it is a cascading schedule. If the T2 and T3 has been committed before failure of T1, then should would have been irrecoverable.

2) Cascadless Schedule :-

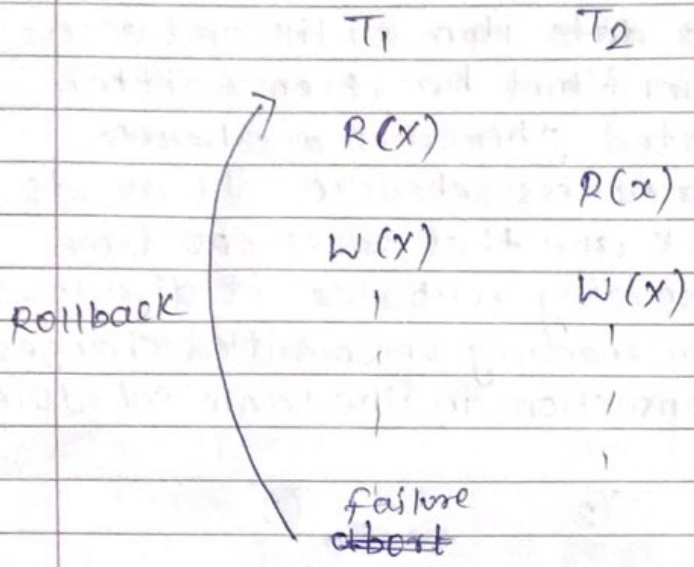
If in a schedule, a transaction is not allowed to read a data item untill and unless the last transaction that has been written is committed/aborted, then such a schedule is called a Cascadless schedule. It avoids cascading rollback and thus saves CPU time.

To prevent cascading rollbacks, it disallows a transaction from reading uncommitted changes from another transaction in the same schedule.



Here the updated value of X is ~~red~~ read by T2.

But, there is one problem in cascadless schedule:-



So, here T_1 Reads value of X, and then immediately T2 Reads value of X from T_1 and then T_1 writes on X and then T2 also writes on X. But the problem is after the write operation on T1 if T_1 fails then there is a write-write problem rises, means T2 has performed write operation on X because of T1 rollback.

→ Serializibility in DBMS:-

Serializibility in DBMS is a concept that helps to identify which non-serial schedules are correct and will maintain the consistency of the database. A serializable schedule always leaves the database in a consistent state. A serial schedule is always a serializable schedule because in a serial schedule, a transaction only starts when the other transaction has finished execution. A non-serial schedule of n transactions is said to be a serializable schedule, if it is equivalent to the serial schedule of those n transaction. A serial schedule does not allow concurrency; only one transaction executes at a time, and the other starts when the already ~~runny~~ ≠ running transaction is finished.

Serial schedule	serializable schedule
1) No concurrency is allowed.	1) Concurrency is allowed
2) serial schedules leads to less resource utilization and CPU throughput.	2) Serializable schedule improve both resource utilization and CPU throughput.
3) less efficient.	3) More better than serial.

1) Conflict Serializibility:-

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

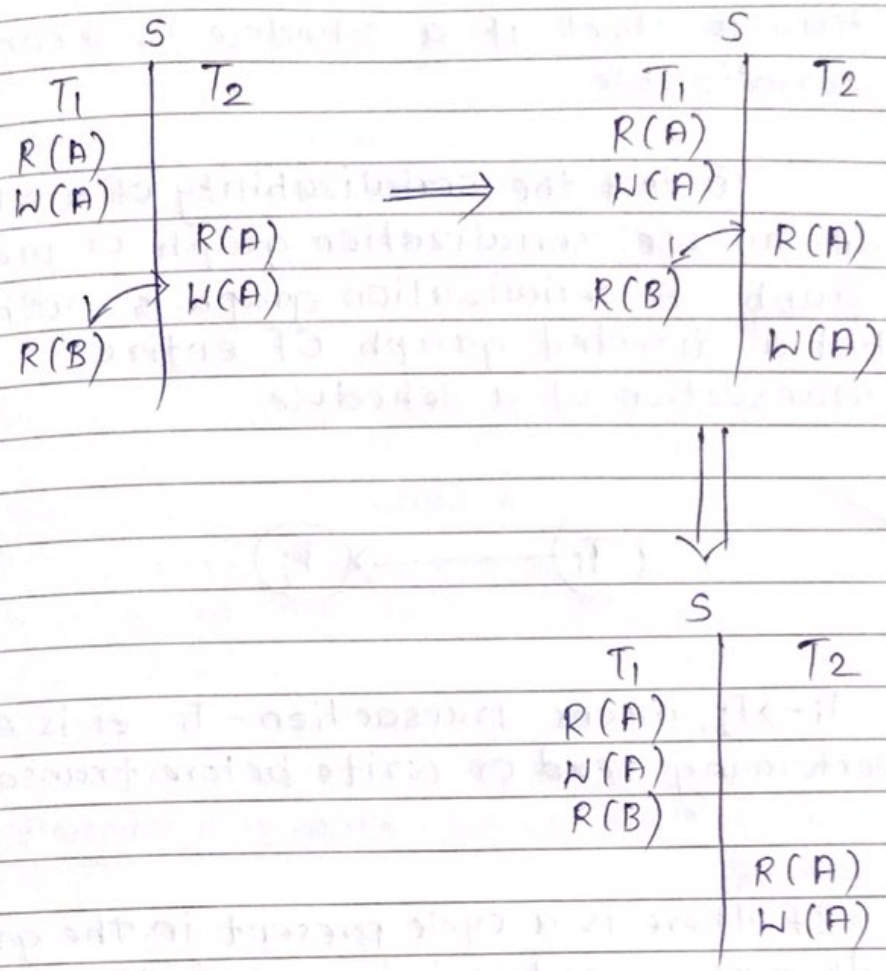
R(A) R(A) } non-conflict pairs

R(A) W(A) }
W(A) R(A) } conflict pairs
W(A) W(A) }

R(B) W(A) }
W(B) R(A) }
R(B) ≠R(A) } Non-conflict pairs
W(B) W(A) }

S == S'

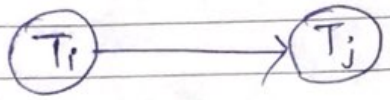
S		S'	
T_1	T_2	T_1	T_2
R(A)		R(A)	
W(A)		W(A)	
	R(A)	R(B)	
	W(A)		R(A)
R(B)			W(A)



Hence proved, $S == S'$

→ How to check if a schedule is conflict serializable

To test the serializability of a schedule we can use serialization graph or precedence graph. A serialization graph is nothing but a directed graph of entire transaction of a schedule.



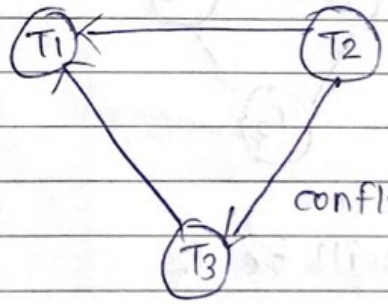
$T_i \rightarrow T_j$, means Transaction - T_i ~~ei~~ is either performing read or write before transaction T_j.

NOTE:- If there is a cycle present in the graph then the schedule is non-serializable because the cycle resembles that one transaction is dependent on the other ~~thra~~ transaction and vice versa. On the other hand, no-cycle means that the non-serial schedule is serializable.

Ex:-

T_1	T_2	T_3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)		
W(x)		
W(z)		

Precedence graph:-



No cycle.

conflict Serializable

For, R(x) in T_1 we have to check if that if W(x) is present in T_2 and T_3, if not then continue, if it is then draw an edge from T_1 to in which transaction it is present.

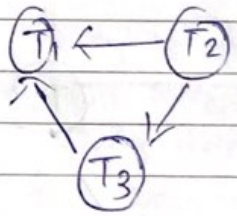
for W(Y) we have to check for R(Y) as well as W(Y) in other transactions.

Now, to make a schedule which will be serial, we have to look for the possible way which will lead to serial schedule.

So, All possible transactions are :-

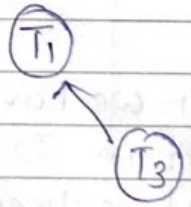
$T_1 \rightarrow T_2 \rightarrow T_3$
$T_1 \rightarrow T_3 \rightarrow T_2$
$T_2 \rightarrow T_1 \rightarrow T_3$
✓ $T_2 \rightarrow T_3 \rightarrow T_1$
$T_3 \rightarrow T_1 \rightarrow T_2$
$T_3 \rightarrow T_2 \rightarrow T_1$

Now, we have to look for indegree in graph. Look for the vertex which has no edge directed.



So, here it will be T_2.

T_2



$T_2 \rightarrow T_3$

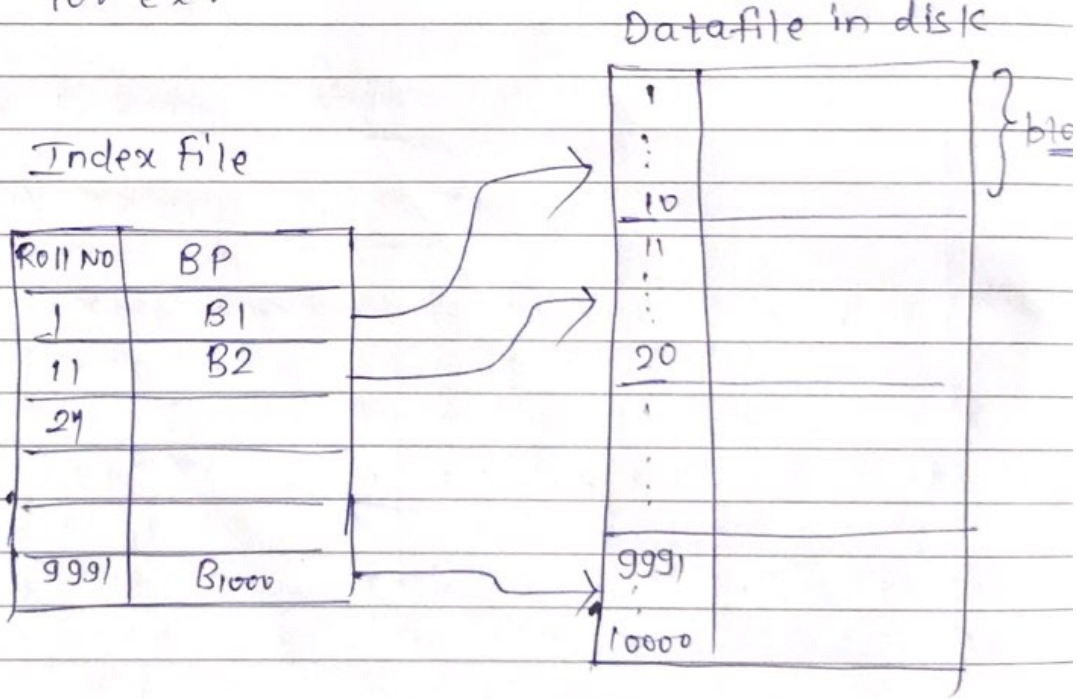
$T_2 \rightarrow T_3 \rightarrow T_1$

Indexing in DBMS:-

For searching a record in database we corite an SQL query to fetch that record. But, if we have a large database stored in file system in the disk then the retrieval time of the query will be large. In order to, reduce this retrieval time we use an Inde Table.

Index table basically stores the addres of the data along with a search key. The Next time we wish to search for a data then the database looks at the index table and then moves to the address with the help of addresses stored in table.

for ex:-



Each block has 10 Records

Page No.
Date:
YOUVA

Data is stored in blocks in datafile in disk.

Now, if we have to search for a ROLL NO then we can go the index file and search for a block and then can get the address of the block and search for the d ROLL NO in database.

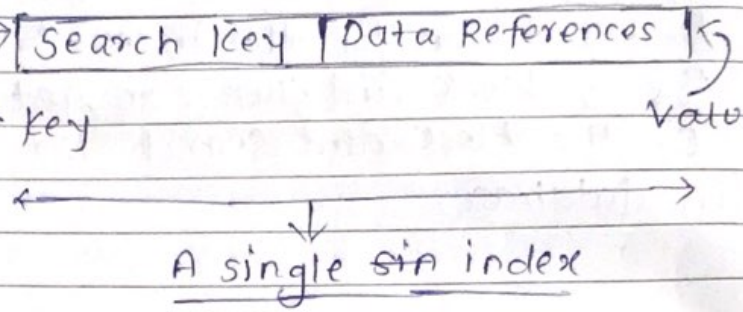
→ Indexing is a way to optimize the performance of a database by minimizing the no. of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in database.

→ Indexes are created using a few database columns:-

1) The first column is the search key that contains a copy of primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.

2) The second column is the data Reference or pointer which contains a set of pointers holding the address of the disk block where that particular key value can be found.

→ Structure of an Index in Database :-



→ Indexing is optional, but increases access speed. It is not the primary mean to access tuple, it is the secondary mean.

→ Index file is always sorted.

→ Indexing Methods :-

1) Primary Index :-

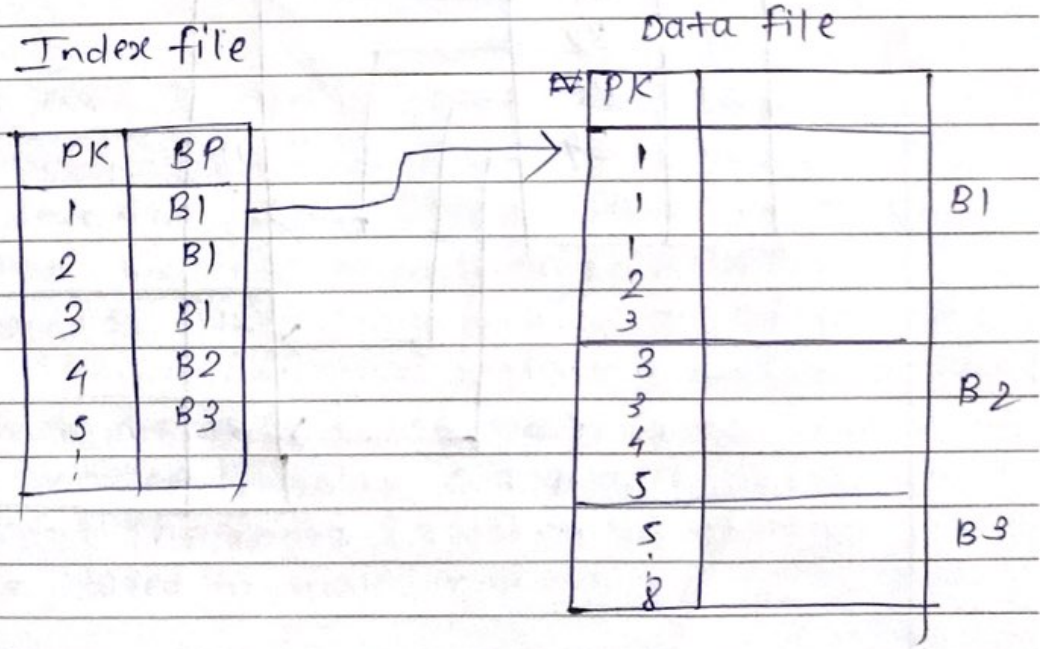
1) Primary index is an ordered file which has fixed length size with two fields.
2) The first field is the same as primary key and second field is pointed to that specific data block.
3) In primary indexing there is always one to one relationship between the entries in the index table.

→ Primary indexing is further divided into two types:-

1) Dense index.
2) sparse index.

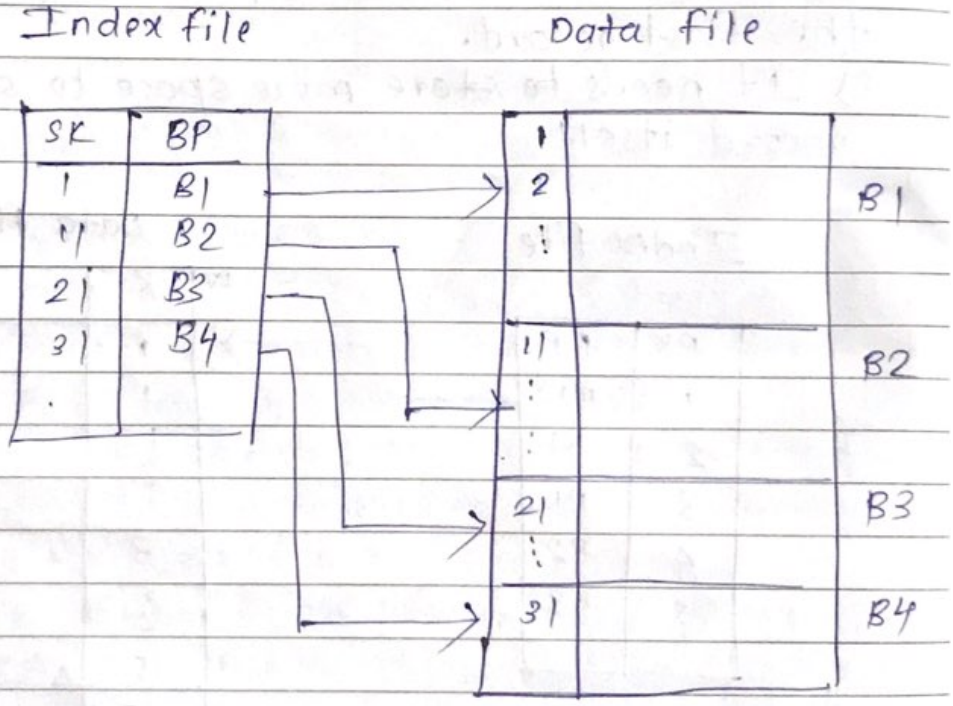
→ Dense Index:-

1) In dense Index, a record is created for every search key valued in the database.
2) The index record contains the search key value and a pointer to the first data record with that search key value. The rest of the records with the same search key value would be stored sequentially after the first record.
3) It needs ~~to store~~ more space to store record itself.



→ Sparse Index:

1) The sparse Index is an index record that appears for only some of the values in the file.
2) Sparse Index helps you to resolve the issue of dense Indexing.
3) In sparse indexing technique, a range of index columns stores the same data block address, and when data needs to be retrieved, the block address will be fetched.
4) Sparse Indexing method stores index records for only some search values.



→ Primary indexing can be based on Data file is sorted w.r.t Primary key or non-key attributes.

1) Based on key attributes:-

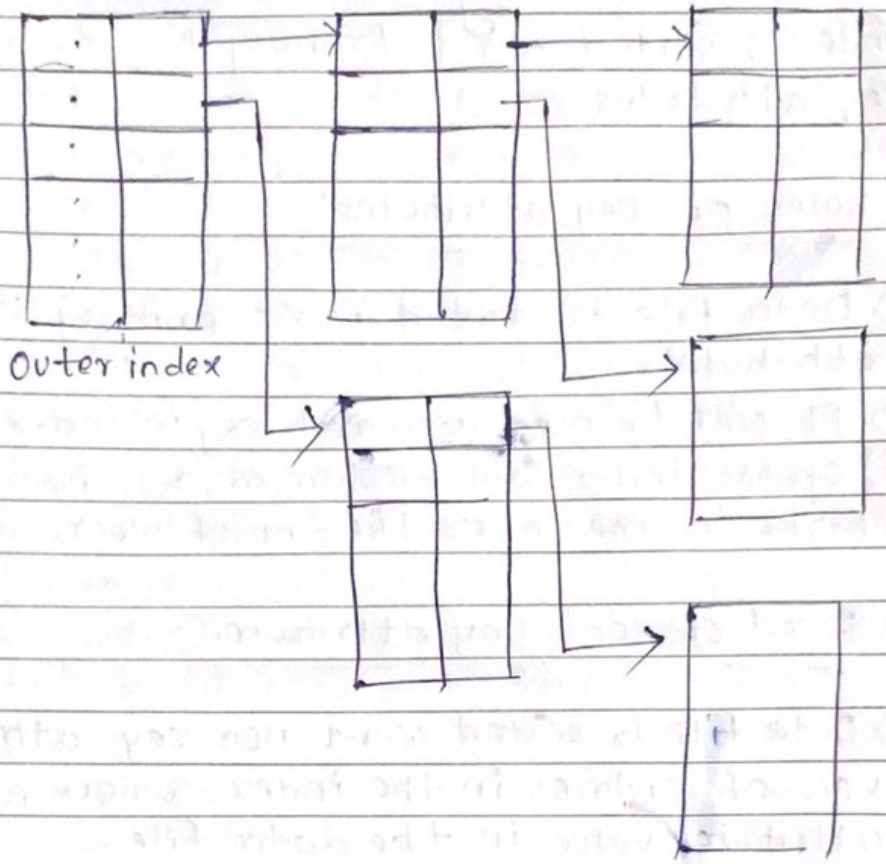
1) Data file is sorted w.r.t primary key attribute.
2) PK will be used as search key in index.
3) sparse index will be formed, i.e, no. of entries in the index file = no. of blocks in datafile.

2) Based on non-key attribute:-

1) Data file is sorted w.r.t non key attribute.
2) No. of entries in the index = unique non key attribute value in the data file.
3) The is dense index, all the unique values have an entry in the index file.

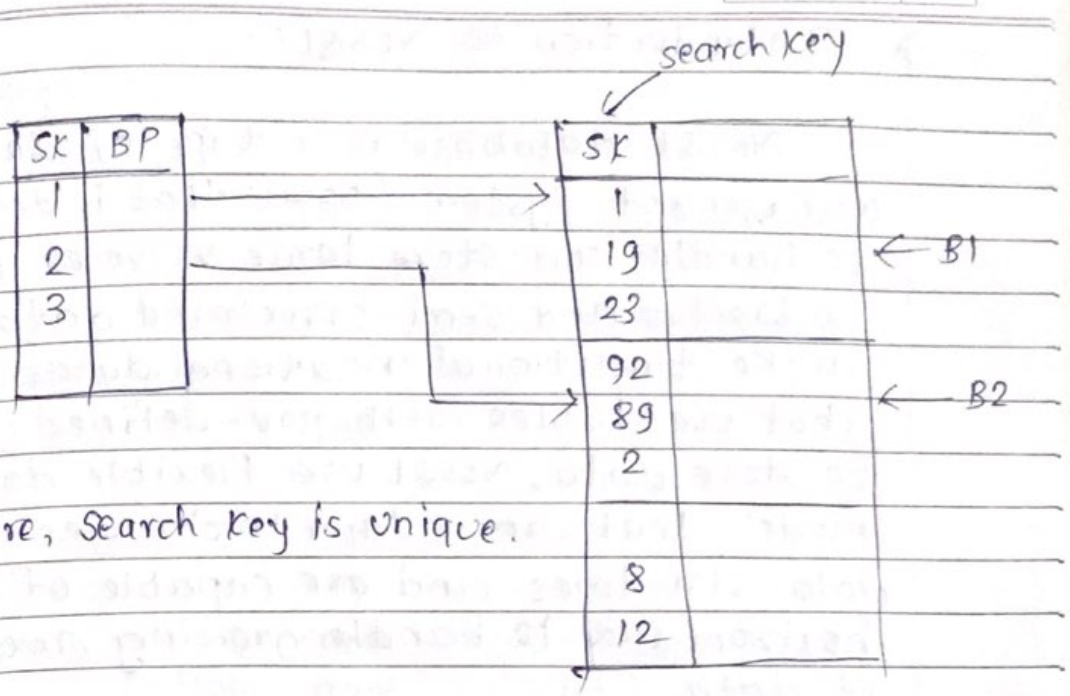
→ Multi level Indexing:-

Since the Index table is stored in main memory, single level indexing for a huge amount of data requires a lot of memory space. Hence, multilevel indexing was introduced in which we divide the main data block into smaller blocks. This makes the outer block of the index table small enough to be stored in main memory.



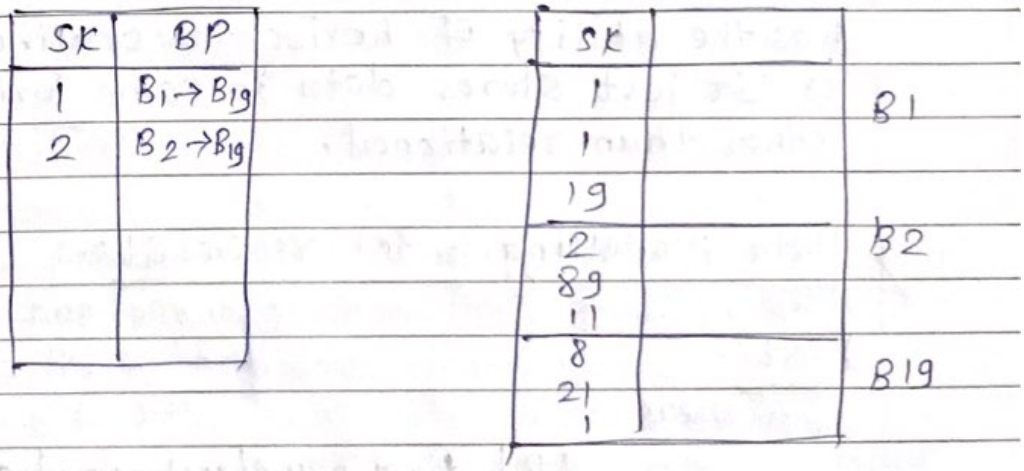
→ Secondary Index:-

1) Datafile is unsorted. Hence primary indexing is not possible.
2) Can be done on key or non key attribute.
3) called secondary indexing because normall one indexing is already applied.
4) No. of entries in the index file = no. of records in the data file.
5) It's an example of dense index.



Here, search key is unique.

→ If search key is not unique and unsorted then we use linked list data structure to store multiple addresses of that search key in a single block



→ Introduction to NoSQL:-

NoSQL database is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructed and semi-structured data. Unlike traditional relational databases that use tables with pre-defined schema to store data, NoSQL use flexible data models that can adapt to changes in data structures and are capable of scaling horizontally to handle growing amounts of data.

1) They are schema free.
2) Data structures used are not tabular, they are more flexible, has the ability to adjust dynamically.
3) Can handle huge amount of data.
4) Most of the NoSQL are open sources and has the ability of horizontal scaling.
5) It just stores data in some format other than relational.

→ Data Modelling in SQL Vs NoSQL:-

1) SQL:-

Users

ID	first_name	last_name	cell	city
1	Tata	Salt	811	Mumb

Hobbies

ID	user-id	hobby
10	1	Scrapbooking
11	1	Games
12	1	Biking

1) NoSQL:-

```
{
    "id" : 1,
    "first-name" : "Tata",
    "last-name" : "salt",
    "cell" : "811",
    "city" : "Mumbai",
    "hobbies" : ["scrapbooking", "Games",
                 "Biking"]
}
```

Advantages of NoSQL:-

1) flexibility:-

NoSQL databases are designed to handle unstructured or semi-structured data, which means that they can accomodate dynamic changes to data model. This makes NoSQL databases a good fit for applications that need to handle changing data requirements.

for ex:-

In SQL we can see that the tables are defined with schema and if the corresponding data for a column is not defined, then in that place we are corithir NULL, but in case of NoSQL we are defining a JSON object with a key-value pair to hold data, and for some key if the value is not defined then we are not placing it in the database. So, we are saving space by not placing NULL.

2) Horizontal scaling:-

Horizontal scaling also known as scale out, refers to bring on additional nodes to share load.

→ Scaling:-

We are storing our data on some device and if the device storage is filled up, then we have to scale our device to meet the requirement. So basically there are two types of scaling vertical and horizontal scaling.

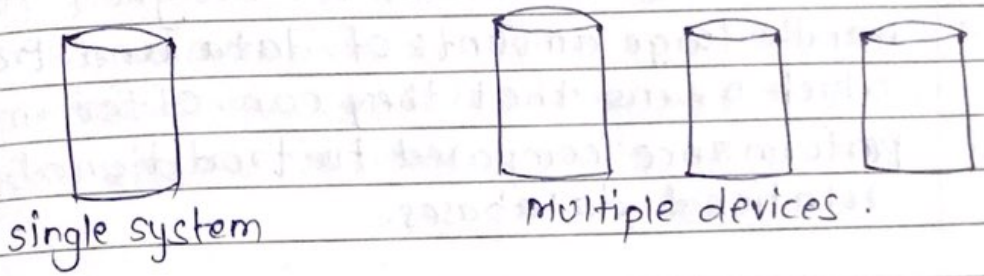
Scaling

Vertical

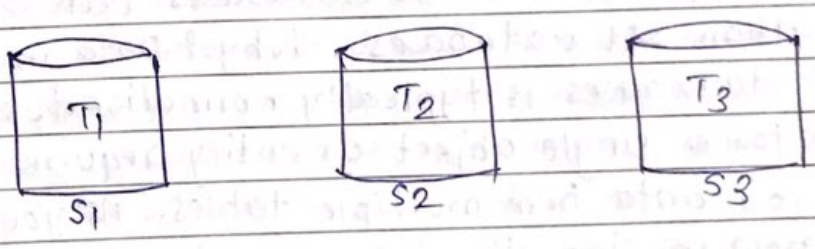
1) Hardware upgrade.
2) RAM.
3) CPU.

horizontal

1) additional node.
2) load share.



Q) SQL → Why no horizontal scaling?



Suppose we have 3 servers and in each server we have our data stored and if we have to fetch our data from these servers which are located at different places. And if we use Joins to retrieve the data we want then it will take much more time to fetch the data.

3) High Availability:-

Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state

4) Performance:-

NoSQL databases are designed to handle large amounts of data and traffic which means that they can offer improved performance compared to traditional relational databases.

5) Easy insert and read operations:-

Queries in NoSQL databases can be faster than SQL databases. Why? Data in SQL databases is typically normalised, so queries for a single object or entity require you to join data from multiple tables. As your tables grow in size, the joins can become expensive. However, data in NoSQL databases is typically stored in a way that is optimised for queries.

→ When to use NoSQL ?

1) Fast paced agile development
2) storage of structured and semi-structured data.
3) Huge volumes of data.
4) Requirements for scale out architecture.

→ Types of NoSQL databases ?

A database is a collection of structured data or information which is stored in a computer system and can be accessed easily.

1) Document based databases :-

1) The document based databases is a nonrelational database. Instead of storing the data in row and columns, it uses the documents to store the data in a database. A document database stores data in JSON, BSON, XML.
2) Documents can be stored and retrieved in a form that is much closer to the data objects used in applications which means less translation is required to use these data in the applications.
3) Each document contains pairs of fields and values. The value can typically be a variety of types including strings, numbers, boolean, arrays or objects.
4) MongoDB, couchDB, ← ex

2) Key-value stores:-

1) The simplest type of NoSQL database is a key value store. Every data element in the database is stored as a key value pair consisting of an attribute name and a value.
2) Ex:- Oracle NoSQL, MongoDB also supports key-value store, Redis.
3) The data can be retrieved by using a unique key allocated to each element in the database. The values can be simple data types like strings and numbers or complex object.

→ There are several use cases where choosing a key value store approach is an optimal soln.

1) Real time Random data access, e.g. user session attributes in an online application such as gaming or finance.
2) Caching mechanism for frequently accessed data or configuration based on keys.
3) Application is based on simple key based queries.

3) column-based :-

Name	city	age
matt	delhi	27
dave	Jaipur	30

Row-wise :-

matt	delhi	27	→	dave	Jaipur	30

Column-wise :-

matt	dave	delhi	Jaipur	27	30

1) The data is stored such that each row of a column will be next to other rows from that same column.
2) A column oriented database is a non-relational database that stores the data in column instead of rows. That means when we want to run analytics on a small number of columns, you can read those columns directly without consuming memory with unwanted data.

4) Graph based stores:-

1) A graph based ~~stores~~ database focuses on the relationship between data elements. Each element is stored as a node (such as a person in a social media graph). The connection between elements are called links or relationships. In a graph database, connections are first-class elements of the database, stored directly.

2) A graph database is optimised to capture and search the connections between data elements, overcoming the overhead associated with joining multiple tables in SQL.